Bright Star Institute Catalog

Bright Star Institute is an equal opportunity employer/program.
Auxiliary aids and services are available upon request to individuals with disabilities.

Bright Star Institute Mission Statement

Bright Star Institute is committed to providing cost effective paths for students to obtain new skills. Combining online training with mentored hands-on practice, our students achieve competency at the lowest cost possible. Our task-oriented curriculum is continually updated to provide the greatest return on time investment and provide the greatest possible relevance to real-world tasks.

Table of Contents

Bright Star Institute Mission Statement 1
Organizational Chart 3
Admissions 4
Distance Education 4
Operating Calendar 4
Tuition 5
Cancellation and Refund Policy 5
Financing Options 5
Attendance 6
Leave of Absence 6
Withdrawals 6
Academic Advising and Student Evaluation 6
Student Records 6
Policy on Student Conduct and Dismissal 7
Student Complaint/Appeal Process 7
Informal Conflict Resolution Procedure 7
Formal Conflict Resolution Procedure 8
Reasonable Accommodations Policy – Individuals With Disabilities 8
Notification of Rights Under FERPA with Respect to Student Records 9
Unlawful Harassment Policy 12
Names of Instructional Staff and Qualifications 12
Course Descriptions 16
- Analyzing and Presenting Data with Power BI 17
- Analyzing Data Using DAX and M Power Query 21
- ASP.NET MVC Programming for Experienced C# Programmers 23
- C Programming 25
- C++ Programming for Non-C Programmers 28
- C++ Programming for C Programmers 32
- Customer Support and Relations Management 35
- Cyber Security Cloud Analyst 37
- Developing Web Applications Using Angular 41
- Effectively Using Java Packages and Features 45

Introduction to ASP.NET Core MVC 48
Introduction to Programming Concepts 52
Introduction to Spring 5, Spring MVC, and Spring REST 54
Java Programming 57
JavaScript Programming 62
Learning to Program with C# 65
Learning to Program with Java 68
Microsoft SQL Server 2017 Administration 71
Microsoft Transact-SQL Programming 73
Object Oriented Analysis & Design with UML 76
Python Programming 79
SQL Programming 82
SQL Server 2019 Business Intelligence: Integration Services and Analysis Services 85
SQL Server 2019 Reporting Services 88
Test Driven Development (TDD), and Refactoring Legacy Code Using C# 90
Test Driven Development (TDD), and Refactoring Legacy Code Using Java 93
Website Development with HTML5, CSS and Bootstrap 96
Windows Forms Programming Using C# 101

Organizational Chart

Board of Directors

Patrick Spada
Raymond Van Liew
Roland Van Liew

Administrative Personnel

Patrick Spada, Executive Director
Roland Van Liew, CFO, CIO
Raymond Van Liew, CMO

Faculty

Dana Wyatt, Courseware Developer, Instructor - .NET Framework and C/C++

Dominick A. Pante, Jr., Software Developer, Instructor – Java and UNIX

Edgar Machado, Courseware Developer, Instructor – Business Intelligence and Power BI

John Skelton, Web Designer and Programmer, Instructor - HTML/JavaScript, Python, and SQL

Ken Hart, Software Developer, Instructor - .NET Framework, C/C++, and Test Driven Development

Kirk Smith, Web Developer, Instructor - HTML/JavaScript, Python, and SQL

Robert Broadhead, Software Developer, Instructor – Java, JavaScript, Node.js, Perl, PHP, and React

Student Services Administrative Staff

Erica Sims, Registrar/Bursar
Raymond Van Liew, Student Support

Our designated EO Officer is **Patrick Spada**. You may reach him directly for any compliance-related inquiries or to request assistance with accessibility and rights at patricks@brightstarinstitute.com or (302) 335-8868.

Admissions

Bright Star Institute programs are open to all persons regardless of race, creed, gender, or national origin. Applicants must be at least 18 years of age, possess a high school diploma, GED, or an equivalent qualification, have access to an internet-connected computer and be able to benefit from the program

Corporate-sponsored applicants are exempt from the above minimum-education requirement.

Application for admission to Bright Star Institute may be completed online at www.brightstarinstitute.com. Admissions personnel may also be contacted by phone for any questions applicants may have about Bright Star or the admissions process.

Distance Education

At Bright Star Institute online learning gives you the flexibility to plan study time around the rest of your day. Our programs combine video, hands-on lab exercises and one-on-one facilitation to enable you to learn as your schedule allows. Labs are performed via remote access to a dedicated desktop PC lab machine and monitor (unless a student requests to utilize their own equipment).

Our unique commitment to student success begins with a low student-to-instructor ratio of 15 to 1 and includes a personal instructor to assure that your courses are completed on time and with full understanding of both the lecture content and the lab exercises.

If students are unable to complete a course within a reasonable time, the Institute's policy is to be flexible and permit the student to complete the course at a later date. If a student is enrolled in a program consisting of multiple courses, all courses in the program must be successfully completed before a certificate is granted.

Operating Calendar

The Bright Star Institute operates year-round. Because Bright Star is a fully online institution, our students reside in many locations. All services are provided online and staff are available via email and chat. The main office of the Institute operates on Eastern Time and is open to students by appointment, except on the following holidays: News Year's Day, Memorial Day, Independence Day, Labor Day, Thanksgiving Day and Christmas Day.

Tuition

The tuition cost for each program includes all fees needed to complete the course. There are no additional fees other than the cost of the program. Costs for all programs are listed on our website at https://www.brightstarinstitute.com/it-certificate-training.htm.

Cancellation and Refund Policy

If a student wishes to withdraw from a program or course, the student must notify the school in writing and mail or e-mail the letter to the school. A student is eligible for a full refund if they withdraw prior to class start. Once the student starts class, the refund of the tuition will be, per course:

A. After 0.001 percent enrollment time to 14.9 percent enrollment time of the course, 100 percent of the tuition will be refunded
B. Between 15 percent to 24.9 percent enrollment time of the course, 55 percent of the tuition will be refunded
C. Between 25 percent to 49.9 percent enrollment time of the course, 30 percent of the tuition will be refunded
D. After 50 percent or more enrollment time of the course, the student may be obligated for the full tuition

Financing Options

Prospective students applying to the programs are provided the opportunity to utilize a number of financing options to assist with tuition payments. Detailed information can be obtained by calling Bright Star Institute.

Attendance

Courses begin every Monday except for weeks where observed holidays occur on a Monday – for a complete list of observed holidays, please see the Operating Calendar section earlier in this document. Classes will start on Tuesday of those exception weeks and will extend to the following Monday. Students may access the online content 24 hours a day. An instructor is assigned to assist you in completing lab exercises and to help with any questions about the course content.

Leave of Absence

If a student does not complete the coursework within the specified timeframe of the course, the student may receive a grade of Incomplete. All incomplete coursework should be resolved within eight (8) weeks of the end of the course. Students who require a longer time are required to apply for a leave of absence.

Withdrawals

If a student withdraws from a program or course, the student must notify the school in writing and mail or e-mail the letter to the school. Once withdrawn, refunds are based on completion percentage per course, as stated in the Refund Policy. It is the student's responsibility to withdraw officially from the school. Failure to withdraw formally may result in additional financial obligations.

Academic Advising and Student Evaluation

The focus of Bright Star Institute is on the success of each student. Advisors are available to assist students, whether beginner or advanced, with selecting the appropriate courses to achieve their desired certificate.

For each class in their program, students are assigned a personal facilitator to assure successful completion and verify solid understanding of the course content through monitoring of lab exercises and dialogue with each student. Program certificates are awarded when all courses in the program have been completed satisfactorily.

Student Records

All student records and transcripts are kept on file permanently and are protected from fire, theft and other perils.

Students may request a copy of their program/course completions by contacting Bright Star Institute.

Policy on Student Conduct and Dismissal

There are typically very few problems with student misconduct in an online classroom. The instructor and administration do have the ability to delete offensive messages from any Discussion Board. The Institute also reserves the right to expel students for failure to observe acceptable online and Internet etiquette including, but not limited to, inappropriate behavior such as repeated posting of obscene or offensive messages. The Institute reserves the right to revoke a student's password to the Online Classroom environment in order to prevent such behavior.

Students may be disciplined for acts of academic misconduct.

Student Complaint/Appeal Process

Bright Star Institute is committed to providing all of our students a quality educational experience that meets their career training needs. Students with complaints about any aspect of their experience with Bright Star Institute can have their concerns addressed. Bright Star Institute has both an informal and formal conflict resolution process as outlined below.

Informal Conflict Resolution Procedure

When a student believes that he/she has been treated unfairly by a Bright Star Institute staff member, the student shall communicate with the individual in a good faith effort to resolve the dispute promptly and fairly. Such communication shall be conducted via correspondence, email or phone communication at the option of the student. This is an informal process that promotes constructive dialogue and understanding, requiring no paperwork or forms.

Prior to pursuing the formal conflict resolution procedure, the student must use the following informal procedure:

- The student shall discuss the issue informally with the Bright Star Institute faculty or staff member. Both parties should openly discuss the matter in an attempt to understand varying perspectives, explore alternatives, and attempt to arrive at a satisfactory resolution to the issue within fifteen (15) instructional days from the date the student first makes the issue known.
- In the event of an absence from the Institute by the faculty or staff member for reasons such as vacation or termination of employment, the student shall contact the faculty or staff member's supervisor for advice on how to proceed with the process. If the student feels that s/he cannot communicate directly with the faculty or staff member, s/he may contact the faculty or staff member's supervisor directly.

- The faculty or staff member's supervisor may act as a mediator to resolve the conflict in a prompt and fair manner.

Formal Conflict Resolution Procedure

1. In the event resolution is not achieved through the informal procedure, the student may initiate the formal procedure by writing a letter to the appropriate administrator or designee of the faculty or staff member. The letter must be submitted within ten (10) days of the conclusion of the unresolved informal process. The letter must include the following: A detailed description of the issue, including dates and time A summary of the actions taken by the student to resolve the issue A proposed solution
2. The appropriate administrator designee shall determine if the issue has merit. An issue without merit will be dismissed without further review and the student will be notified in writing of this determination.
3. If the issue is found to have merit, the administrator or designee shall attempt to resolve the issue by the following:
    a. Serving as an intermediary between the student and the faculty or staff member
    b. Considering all facts of the situation and talking with all parties involved
    c. Determining how to resolve the issue promptly and fairly
4. The administrator or designee handling the case will notify all parties in writing of the decision.
5. The student may appeal the decision to the Director.
6. The decision of the Director shall be final.

In the event a student has proceeded through the formal complaint process at Bright Star Institute and is unable to reach a resolution with the Institution, the student may file a complaint with the State of Delaware Department of Education, 35 Commerce Way, Suite 1, Dover, DE 19901.

Reasonable Accommodations Policy – Individuals With Disabilities

Bright Star Institute does not discriminate against individuals on the basis of physical or mental disability and is fully committed to providing reasonable accommodations to qualified individuals with a disability, unless providing such accommodations would result in an undue burden on the institution or fundamentally alter the nature of the relevant program, benefit, or service provided by Bright Star Institute. Students may request an accommodation by contacting the school. Individuals requesting an accommodation will need to complete a Request for Accommodation at least three weeks before beginning a program, or as soon as feasible. Disagreements regarding an appropriate accommodation or alleged violations of this policy may be addressed pursuant to Bright Star Institute's grievance procedures.

Notification of Rights Under FERPA with Respect to Student Records

The Family Educational Rights and Privacy Act (FERPA) afford eligible students certain rights with respect to their education records. An "eligible student" under FERPA is a student who is 18 years of age or older or one who attends a postsecondary institution. These rights include:

1. The right to inspect and review the student's education records within 45 days after the day Bright Star Institute receives a request for access. A student should obtain a Request to Inspect and Review Education Records form from the portal and submit to the Director, a written request that identifies the record(s) the student wishes to inspect.
The school official will make arrangements for access and notify the student of the time and place where the records may be inspected. Students are not entitled to inspect and review financial records of their parents. If the records are not maintained by the school official to whom the request was submitted, that official shall advise the student of the correct official to whom the request should be addressed.

2. The right to request an amendment of the student's education records that the student believes is inaccurate, misleading, or otherwise in violation of the student's privacy rights under FERPA. A student who wishes to ask the school to amend a record should write to the Director, clearly identify the part of the record the student wants changed, and specify why it should be changed. If the school decides not to amend the record as requested, the school will notify the student in writing of the decision and the student's right to a hearing regarding the request for amendment. Additional information
regarding the hearing procedures will be provided to the student when notified of the right to a hearing.

3. The right to provide written consent before Bright Star Institute discloses personally identifiable information from the student's education records, except to the extent that FERPA authorizes disclosure without consent. The school discloses education records without a student's prior written consent under the FERPA exception for disclosure to school officials with legitimate educational interests. A school official is a person employed by the institution in an administrative, supervisory, academic, research, or support staff position (including law enforcement unit personnel and health staff); a person serving on the board of directors; or a student serving on an official committee, such as a disciplinary or grievance committee. A school official also may include a
volunteer or contractor outside of Bright Star Institute who performs an institutional service of function for which the school would otherwise use its own employees and who is under the direct control of the school with respect to the use and maintenance of the education records, such as an

attorney, auditor, collection agent, or a student
volunteering to assist another school official in performing his or her tasks. A school official has a legitimate educational interest if the official needs to review an education record in order to fulfill his or her professional responsibilities for the institution.

Parental access to a student's record will be allowed by Bright Star Institute without prior consent if: (a) the student has violated a law or the institution's rules or policies governing alcohol or substance abuse, if the student is under 21 years old; or (b) the information is needed to protect the health or safety of the student or other individuals in an emergency.

4. The right to file a complaint with the U.S. Department of Education concerning alleged failures by Bright Star Institute to comply with the requirements of FERPA. The name and address of the Office that administers FERPA is:
Family Policy Compliance Office
U.S. Department of Education
400 Maryland Avenue, SW
Washington, DC 20202
Below is a listing of the disclosures that postsecondary institutions may make without consent:

FERPA permits the disclosure of education records, without consent of the student, if the disclosure meets certain conditions found in the FERPA regulations. Except for disclosures to school officials, disclosures related to some judicial orders or lawfully issued subpoenas, disclosures of directory information, and disclosures to the student, FERPA regulations requires the institution to record the disclosure. Eligible students have a right to inspect and review the record of disclosures. A postsecondary institution may disclose education records without obtaining prior written consent of the student in the following instances:

- To other school officials, including teachers, within Bright Star Institute whom the school has determined to have legitimate educational interests. This includes contractors, consultants, volunteers, or other parties to whom the school has outsourced institutional services or functions.
- To officials of another school where the student seeks or intends to enroll, or where the student is already enrolled if the disclosure is for purposes related to the student's enrollment or transfer.
- To authorized representatives of the U. S. Comptroller General, the U. S. Attorney General, the U.S. Secretary of Education, or State and local educational authorities, such as a State postsecondary authority that is responsible for supervising the institution's State-supported

education programs. Disclosures under this provision may be made, in connection with an audit or evaluation of Federal- or State-supported education programs, or for the enforcement of or compliance with Federal legal requirements that relate to those programs. These entities may make further disclosures to outside entities that are designated by them as their authorized representatives to conduct any audit, evaluation, or enforcement or compliance activity on their behalf. In connection with financial aid for which the student has applied or which the student has received, if the information is necessary to determine eligibility for the aid, determine the amount of the aid, determine the conditions of the aid, or enforce the terms and conditions of the aid.

- To organizations conducting studies for, or on behalf of, the school, in order to: (a) develop, validate, or administer predictive tests; (b) administer student aid programs; or (c) improve instruction.
- To accrediting organizations to carry out their accrediting functions. To comply with a judicial order or lawfully issued subpoena.
- To appropriate officials in connection with a health or safety emergency.
- Information the school has designated as "directory information" may be released at the school's discretion. Bright Star Institute have defined directory information as the student's name, address (es), telephone number(s), e-mail address, birth date and place, program undertaken, dates of attendance, honors and awards, photographs and credential awarded. If a student does not want his or her directory information to be released to third parties without the student's consent, the student must present such a request in writing to the Director within 45 days of the student's enrollment or by such later date as the institution may specify. Under no circumstance may the student use the right to opt out to prevent the institution from disclosing that student's name, electronic identifier, or institutional e-mail address in a class in which the student is enrolled.
- To a victim of an alleged perpetrator of a crime of violence or a non-forcible sex offense. The disclosure may only include the final results of the disciplinary proceeding with respect to that alleged crime or offense, regardless of the finding.
- To the general public, the final results of a disciplinary proceeding if the school determines the student is an alleged perpetrator of a crime of violence or non-forcible sex offense and the student has committed a violation of the school's rules or policies with respect to the allegation made against him or her.
- To parents of a student regarding the student's violation of any Federal, State, or local law, or of any rule or policy of the school, governing the use or possession of alcohol or a controlled substance if the school determines the student committed a disciplinary violation and the student is under the age of twenty-one.

Unlawful Harassment Policy

Bright Star Institute is committed to the policy that all members of the school's community, including its faculty, students, and staff, have the right to be free from sexual harassment by any other member of the school's community. Should a student feel that he/she has been sexually harassed, the student should immediately inform the Director. Sexual harassment refers to, among other things, sexual conduct that is unwelcome, offensive, or undesirable to the recipient, including unwanted sexual advances.

Names of Instructional Staff and Qualifications

Dana Wyatt

Dana Wyatt holds a Ph.D. in Computer Science from Texas A&M University and has been a Principal Consultant with Hands On Technology Transfer since 1995. She currently spends much of her time designing courseware and teaching about a wide variety of topics, including: programming desktop, web and backend applications with the .NET Framework, programming devices using iOS and Swift and C/C++ programming.

Ms. Wyatt has written or co-written over 50 technical courses. She has also co-authored two textbooks on Visual Basic.NET for Prentice-Hall.

Dr. Wyatt has served on Computer Science faculties at the University of North Texas, Texas Christian University and Texas A&M University, where she taught classes in computer architecture, programming languages, database systems, simulation techniques, operating systems and problem solving. She received an ASEE fellowship and worked on distributed database problems at NASA Goddard Space Flight Center.

Dominick A. Pante, Jr.

Dom Pante has over thirty (30) years of experience in the data processing field. He has managed technical professional staff and projects, advised corporate executives on new technology, and designed and implemented computer systems. He has developed and implemented corporate data processing, quality assurance, testing, and training standards, and has developed and taught numerous data processing, open system, programming language, object oriented, and client/server related courses.

Prior to working with Bright Star, Mr. Pante has directed a corporate technical and professional development Educational Services organization with an operating budget of over $3 million, directing a staff of over forty-five (45) management, administrative, and technical personnel. Previous to that he managed a staff of over twenty (20) UNIX technical consultants and instructors. He has extensive software development experience using C, C++, XML and Java.

Mr. Pante has designed and taught courses in areas related to data processing standards, structured testing techniques, the C, C++, and Java Programming Languages, and the use of the UNIX and Linux operating systems. Specific training courses developed and taught include Structured Test Plans, Introduction to UNIX/Linux, UNIX Shell Programming, UNIX Pattern Tools (awk, sed), Introduction to C Language Programming, Advanced C Language Programming, C/UNIX Interface(system calls), Programming in Object Oriented C++, Java Programming, Java Packages and Features, and UNIX System Internals Overview.

Mr. Pante has developed and presented various UNIX Tutorials at multiple Local and National UNIX conferences including Tek21, UniForum, UNIX Expo, and UNIX Systems Expo.

Mr. Pante holds a B.S. in Industrial Engineering and a M.S. in Engineering Management from New Jersey Institute of Technology.

Edgar Machado

Edgar Machado has been a technical instructor delivering software programming classes for over 20 years. He currently delivers a wide array of classes using both Microsoft Official Curriculum and proprietary courseware. Classes range from Windows Active Directory to SQL Server Database Design and programming with MS Visual Studio and Visual Basic for Applications. Mr. Machado provides consulting services including integration of client/server applications, database development and application development. His current area of specialization is MS SQL Server design and implementation ranging from relational database design and devotement to complex Business Intelligence solutions that integrate with SharePoint, integration with Microsoft Azure, and PowerBI.

Mr. Machado has extensive experience working with Microsoft server products including integration between database, email, system management, web, B2B and Active Directory environments. He has assisted clients performing integration with client front end applications and programming using Visual Basic, Structured Query Language, VBScript and ADO, C, XML, ADO.Net, and ASP.Net.

Mr. Machado holds several professional certifications MCSA, MCITP, and MCT (Microsoft Certified Trainer). He holds a B. S. in Industrial Distribution, College of Engineering, Texas A&M University.

John Skelton

John Skelton has been a Certified Technical Trainer (CompTIA CTT+) for over 25 years, delivering instruction on many different technologies, including SQL programming, Python programming, Microsoft SQL Server, HTML, JavaScript, CSS, XML, Adobe FrameMaker, Relational Database Design, and ReactJS.

In addition to authoring and revising courseware for many of these courses, Mr. Skelton has developed several industry certification examinations as a Subject Matter Expert.

Mr. Skelton's client list includes many Fortune 500 companies and government agencies, including Apple, Adobe, Microsoft, The US Department of Energy, The United States Air Force, Los Alamos National Laboratories, NBC/Universal, and many more.

The recipient of a Bachelors Degree (BA) from the University of New Mexico, Mr. Skelton also earned a Master of Science in Education from Indiana University.

Ken Hart

Ken Hart has been an independent consultant since 2002. Most recently for Hands On Technology Transfer, he has delivered Advanced MVC Programming, Advanced VB.NET, Angular Web Development, ASP.NET, Advanced C#, C++ Programming, Object Oriented Analysis and Design, Java Spring MVC, and Test Driven Development. Mr. Hart has also directed software development projects for industrial clients including KPMG, AMD and IBM. In India during 2016 Mr. Hart acted as chief architect for a full stack Java-based application involving an Oracle database, Spring/MVC server-side processing, SOAP and REST-ful web services, and front ends written in JavaScript and C#.

Mr. Hart holds MCSD certification from Microsoft. He holds top secret clearance from the NSA, where he worked as a Russian linguist for six years in the 1990s.

Kirk Smith

Kirk Smith has over 20 years of experience in UNIX/Linux computer system programming and administration, Web development, TCP/IP networking support, and Windows system support. Kirk has been a full time technical instructor for the past ten years, delivering courses in UNIX/Shell programming, Linux System Administration, HTML/CSS/JavaScript programming, AngularJS and Angular Programming, jQuery, PHP, Perl, C and SQL Programming. Prior to that he worked for ten years as an independent consultant and web developer subsequent to a 7-year stint as the Director of Purdue University Ag Computer Network. Mr. Smith holds a Bachelor's Degree in Computer and Electrical Engineering from Purdue University and a Master's Degree in Computer and Electrical Engineering, also from Purdue University.

Robert Broadhead

Robert Broadhead has over 30 years of experience in Software development. Roles range from developer to fractional CTO and many point in between. Experience in various industries, including E-Commerce, Non-Profits, Call Centers, Help Desks, Healthcare, CRM, Credit Card Processing, Document Management, music, finance, education, social networking, logistics, and commercial retail software. Expert in Java and Python programming, as well as web technologies including React, Node.js, JavaScript, Perl, PHP, and the AWS cloud computing framework. Rob has managed software development teams in Eastern Europe, the Mid-East, and India as well as the U.S. He is currently a Lead Consultant at RB Consulting Inc. where he has led multiple software development projects across multiple industries. He has authored several books and white papers on software development.

Rob holds a B.S. in Computer Science from Rose-Hulman 1993 and an MBA – E-Commerce from University of Phoenix 2006.

Course Descriptions

Analyzing and Presenting Data with Power BI

Duration: 60 hours

Prerequisites: Familiarity with database concepts and experience with spreadsheet software such as Microsoft Excel. Basic familiarity with relational databases as well as data warehousing is helpful but not required.

Description: This course provides a robust and in-depth introduction to Microsoft's *Power BI* suite of products. Students will learn to use the Power BI Desktop and Power BI online service to import, analyze, and visualize business data, and to share business intelligence.

The course starts with an exploration of the evolution of relational databases, data warehouse and business intelligence. Afterwards, students learn to do tasks such as importing and combining data from diverse data sources, shaping and transforming data, modeling data, developing useful and insightful visualizations, creating and sharing reports and dashboards in the Power BI service, and collaborating in the Power BI service. As the course nears its end, there is also an introduction to Data Analysis Expressions (DAX).

Course Overview

Introduction to Business Analysis

- The Evolution of Data Analysis
  - Databases
  - Normal Forms
  - Data Modeling
  - Data Processing
  - Beyond the Database
  - Data Warehouses
  - Fact and Dimension Tables
  - Data Warehouse Schema
  - Star Schema
  - Snowflake Schema
  - Big Data
- Self-Service Business Intelligence
  - Enterprise Intelligence
  - Self-Service BI
  - The BI Team
  - The Data Steward
- The Tasks
  - Accessing Data
  - Loading Data
  - Shaping Data
  - Reporting

Getting Data with Power BI Desktop

- The Power BI Workflow
  - Terminology
  - Opening the Power BI Desktop
  - The Navigation and Filter Panes
  - The Visualizations Pane
  - The Fields Pane
  - Power BI Desktop Settings
- Data Sources
  - On-Premises Data Files
  - Databases
  - Online Databases
  - Software as a Service (SaaS) Providers
  - Internet Data
  - Data Access Issues
  - Data Reliability Issues
- Getting Data
  - Import vs. Direct Query
  - Advantages of Direct Query
  - Limitations of Direct Query
  - Data Limits
  - Cleaning Data
  - Selecting Data

- Visualizing Data
- Collaborating
- The Tools
- The Power BI Suite
- Related Tools

- Fundamentals of Getting Data
- Importing from Excel
- Preparing Excel Data
- Importing from Excel
- Importing from a Database
- Selecting Data with SQL
- Importing from a Web Page
- Revising Data

Transforming and Shaping Data

- Getting the Right Data
  - Shaping Data
  - How and Where to Transform Data
  - Formatting Currency
  - Columns and Measures
  - Filters
- The Power Query Editor
  - How Power BI Accesses and Stores Data
  - Query Editor Layout
  - The Transform Tab
  - The Add Column Tab
  - The View Tab
  - Manipulating Columns
  - Applied Steps
  - The Advanced Query Editor

Visualizing Data

- Visual Data Processing
  - Why We Visualize
  - Choosing the Right Visualization
  - Report Design
- Creating Visualizations
  - Basic Layout
  - Selecting Data
  - Starting with Data
  - Starting with Visualization Type
  - Sorting
  - Adding Data, Changing Visualizations
  - The Slicer
  - Fields, Format and Analytics in the Visualizations Pane
- Formatting Visualizations
  - General Formatting
  - Formatting Groups
  - Title
  - Report Themes
  - Data Colors
  - Data Labels
  - Data Labels Display Units
  - Axes
  - Text Boxes, Images, Shapes
- Visualization Mechanisms
  - The Slicer
  - Q&A
  - Custom Visuals

Types of Visualizations

- Basic Visualizations
  - Tables
  - Matrixes
- Bar/Column Charts

Modeling Data

- Data Models
  - The Utility of Data Models
  - Data Types
  - Cardinality

        - Clustered
        - Stacked
        - 100%
        - Ribbon Charts
        - Waterfall Charts
    - Line/Scatter/Area Charts
        - Line Chart
        - Scatter Chart
        - Area Charts
        - Line-and-Column Charts
    - Pies and Donuts
        - Pie Charts
        - Donut charts
    - Map Charts
    - Specialized Charts
        - Funnel Chart
        - Gauge Chart
        - Card Chart
        - Multi-Row Card
        - KPI
        - Key Influencers
        - Tree Maps

The Power BI Service

- Overview
    - Functions by Application
    - Limitations
- Connecting to the Power BI Service
    - Publishing a Report to the Power BI Service
    - Direct Connectivity from the Power BI Service
    - Refreshing Data
    - Gateways
- The Power BI Service Interface
    - Layout
    - My Workspace
    - Report Edit Mode
- Dashboards
    - Dashboard Foundations
    - Dashboards vs. Reports
    - Building Dashboards
    - Editing a Dashboard
    - Filtering a Tile
- Working the Power BI Service

        - Cross Filter Direction
    - Relationships
        - Understanding Relationships
        - Relationships and Keys
    - Hierarchies
        - The Utility of Hierarchies
        - Creating Hierarchies
    - The Model View
        - Changing Relationships
        - Using the Model View

Power BI and Programming Languages

- Survey of Power BI Languages
    - Why?
    - The Power BI REST API
    - R
    - M
    - Python
- DAX
    - Data Analysis Expressions
    - DAX Syntax
    - Logical Operators
    - DAX Examples

        - Reports
        - Exporting Reports and Data
        - Workbooks
        - Datasets
        - Apps
        - Sharing Reports and Dashboards
        - Q&A
        - Featured Questions
        - Power BI Service Settings
- Administration
    - Power BI Administration
    - Admin Portal
- Power BI Mobile
    - The App
    - To Set Up Your Smartphone Layout
    - Power BI Mobile Features and Limitations

Analyzing Data Using DAX and M Power Query

Duration: 40 hours

Prerequisites: Familiarity with database concepts, and prior completion of the **Analyzing and Presenting Data with Power BI** course.

Description: This course is an extension of the **Analyzing and Presenting Data with Power BI** course, and focuses on using DAX and M Power Query to increase the flexibility and data analytics capabilities of Power BI reports.

Students will learn beginner and intermediate techniques for adding calculations to their Power BI Data models using DAX. Our hands-on labs provide practical exercises that cover some of the common data modeling problems that users face, and how DAX can be used to solve those. Students will experience flattening out an OLTP database into a star schema by using DAX, resolving common granularity issues with data models, and learn how Measures can be used to solve advanced calculation problems that languages like SQL are unable to tackle.

Students will also learn about the M language that Power Query uses behind its graphical user interface, and how writing M directly can enable advanced queries against the data-sources that populate a model with its data.

Course Overview

DAX Fundamentals

- DAX Capabilities and Common DAX Use Cases
- DAX Formula Syntax Fundamentals
- Formula Evaluation Order
- DAX Operators
- DAX Operator Precedence
- DAX Data Types
- DAX Type Conversions
- DAX Functions
- Nested Function Execution Order
- Naming Requirements

Basic DAX Calculations

- Calculated Columns
- Evaluation Context and Row Context
- Working with Relationships
- Using RELATED() and RELATEDTABLE()
- Using Ad-hoc Relationships with USERELATIONSHIP()
- Flattening Models with Calculated Columns
- Calculated Tables

Advanced DAX Calculations

- Measure Use Cases
- Implicit Measures
- Calculated Measures
- Evaluation Context and Filter Context

DAX and Modeling

- Resolving Model Granularity Issues
- Flattening Models
- Row Level Security with Row Filters

- Implicit Filter Context
- Explicit Filter Context
- Filtering with Calculate() and Filter()
- Undoing Filters with All()
- Iterator Fundamentals
- Working with built-in Iterator Functions
- Using Nested Iterators
- Avoiding Ambiguous Relationships with Role Playing Tables

M and Power Query

- Power Query Basics
- Using the Formula Bar in Power Query
- Using the Advanced Editor in Power Query
- PowerQuery's Standard Function Library
- M Language
- Case Sensitivity in Power Query and M
- Single Literal Values
- Intrinsic Values
- Structured Values: Lists, Records, and Tables
- M Language Operators
- Commenting Code in M
- Using the Let Statement
- Variable Naming in M
- M's Each Statement
- Creating Columns in M
- Conditional Constructs in M
- M Functions

ASP.NET MVC Programming for Experienced C# Programmers

Duration: 60 hours

Prerequisites: Knowledge of fundamental HTML syntax is helpful, but not required. Prior experience with C# is required.

Description: This course provides students with hands on experience using Visual Studio to create dynamic web applications using ASP.NET MVC and C#. Students should already have a working knowledge of C#.

This course teaches students how to leverage the power of the Model-View-Controller design pattern with the ASP.NET MVC Framework to separate the layers of a web application. Students will use the ASPX and Razor view engines to design a user interface. Students will learn how to build models to manage an application's data layer using both the Entity Framework and LINQ to SQL. Students will also learn how to build controllers containing action methods to manage communication between views and models.

Other topics include data scaffolding; URL routing; implementing security; using MVC and Web Forms in the same application; unit testing; and deploying ASP.NET MVC applications. Comprehensive labs provide the students with experience creating, debugging, testing and deploying dynamic ASP.NET MVC applications.

Course Overview

Overview of ASP.NET MVC

- Overview of Model-View-Controller Design Pattern
- ASP.NET MVC Application Architecture
- Understanding the MVC Execution Process
- Building an ASP.NET MVC Application Using Visual Studio
- Visual Studio MVC Project Templates
- Using a web.config File

Developing Controllers

- Creating Controllers
- Defining Action Methods
- Mapping URLs to Action Methods
- Understanding ActionResult Types
- Working with ViewData and ViewBag

Developing Views

- Creating Views
- Understanding View Engines
- Using the ASMX View Engine
- Using the Razor View Engines
- Using HTML Helpers
- Adding Validation

Developing Models

- Creating Model Classes
- Working with the Entity Framework
- Working with LINQ to SQL
- Using Scaffolding

- Working with Strongly-Typed Views

Routing Control

- Understanding Routing in ASP.NET MVC
- Defining URL Routes
- Registering Routes
- Adding Constraints to Routes
- Debugging Routes

MVC Unit Testing

- Test-Driven Development
- Designing Test Cases
- Creating Unit Tests
- Using MS Test

Deploying ASP.NET MVC Applications

- Understanding Deployment Issues
- Required MVC Assemblies
- Server Requirements
- Configuring an ASP.NET Application for Deployment
- Using XCOPY Deployment
- Using WebDeploy

Integrating ASP.NET MVC and Web Forms

- What are Web Forms
- Using Web Forms in an MVC Application
- Using MVC in a Web Form Application
- Linking to MVC Actions from Web Forms

Securing MVC Applications

- ASP.NET Security
- Windows vs Forms Authentication
- Configuring Authentication
- Configuring Authorization
- Building a Secure Web Site
- Defending against Attacks
    - Cross-site Scripting
    - Session Hijacking
    - SQL Injection
    - Input Forgery

C Programming

Duration: 90 hours

Prerequisites: Understanding of fundamental programming concepts.

Description: This hands on C programming course provides a comprehensive introduction to the ANSI C language, emphasizing portability and structured design. Students are introduced to all major language elements including fundamental data types, flow control, and standard function libraries. Thorough treatment is given to the topics of string and character manipulation, dynamic memory allocation, standard I/O, macro definition, and the C runtime library. The course explains the use of aggregate structures, unions, and pointers early on so the students can practice extensively in the hands on labs. Structured programming constructs and varargs functions are also covered. Emphasis is given to the processing of command line arguments and environment variables so students will be able to write flexible, user-friendly programs. The course also includes coverage of portability tips drawn from experienced programmers working in production environments. Comprehensive hands on exercises are integrated throughout to reinforce learning and develop real competency.

Course Overview

Overview of C

- Operating System Independence
- Design Goals and Capabilities
- Flavors of C

Fundamental Data Types, Storage Classes, and Scope

- Fundamental Data Types and Qualifiers
- Constants and Strings
- Storage Classes
- Scope and Block Structure
- Scope and Data Hiding
- Data Initialization

Macros

- Functions vs. Inlining
- Purpose of Macros
- Use of Macros
    - Making Code More Readable
    - Auto Adjustment of Compile Time Values
    - Conditional Compilation

Compiler Directives and the C Preprocessor

- Compile-Time Directives
- Use of typedef
- C Preprocessor Syntax

Pointers and Dynamic Allocation

- Advantages of Pointers
- User of Pointers
- Pointer and Address Arithmetic
- Dynamic Storage Allocation
- sizeof Operator
- Double Indirection

Arrays

- Purpose of Arrays
- Declaring an Array
- Initializing an Array
- Addressing Elements
- Stepping Through an Array
- Variable Size Arrays

    - Making Code Portable
    - Simplifying Complex Access Calculations
- Advanced Micro Design Tips
- Using Macros to Help Write Portable Programs
- When to Use a Macro instead of a Function
- Using Macros for Debugging

Basic Formatted I/O

- Standard I/O Library
- Character Set Encoding
- Standard Input and Output
- Character I/O Functions
- Formatted I/O Functions
- String Constants

Operators and Expressions

- Arithmetic, Logical, and Bit Operators
- Precedence and Associativity
- Assignment and Casting
- The Conditional Operator

Functions (Subroutines)

- Purpose of Functions
- Functions vs. Inlining
- Automatic Variables
- The Argument Stack
- Passing By Value
- Passing By Reference
- Declaring External Functions
- Function Prototyping
- ANSI Prototyping
- The _NO_PROTO Compiler Symbol
- Varargs Functions
- Passing a Function as an Argument
- Designing Functions for Reusability
- Calling a Function from Another Language
- Returning a Dynamically Allocated Value Using Double Indirection
- Casting the Return Value of a Function
- Recursion and Reentrancy

Advanced Structures and Unions

- Arrays of Pointers
- Arrays of Strings
- Passing an Array to a Function
- Dynamic Memory Allocation
- Multidimensional Arrays

Program Debugging

- Problem Analysis
- Instrumenting with printf
- Instrumenting with ctrace
- The Purpose of Debuggers
- How Not to Use Debuggers
- Symbolic Debuggers

Flow Control Constructs

- Conditional Constructs: if, switch
- Looping Constructs: while, do, for
- Programming Style

Structures

- Purpose of Structures
- Defining and Declaring Structures
- Accessing Members
- Pointers to Structures
- Dynamic Memory Allocation
- Passing a Structure to a Function
    - As a Pointer
    - Passing the Actual Structure

C Runtime Library Standard Functions

- Nested Structures
- Arrays of Structures
- Bit Fields
- Unions
- Linked Lists

Strings and Character Manipulation

- Strings as Character Arrays
- String Library Functions
- Reading and Writing Strings

Structured Programming

- Structuring Code for Quality, Reliability, Maintainability
- Designing for Modularity and Reusability

Overview of C

- Operating System Independence
- Design Goals and Capabilities
- Flavors of C

- Character I/O
- Unformatted File I/O
- Formatted File I/O
- Math Functions
- Miscellaneous Functions

Accessing Command Line Arguments and Environment Symbols

- argc and argv
- Parsing Command Line Options
- Accessing the Environment Array

Advanced Programming Consideration

- Writing Portable Code
- Use of Macros
- ANSI C Limits
- Feature Test Macros
- Client/Server Design
- Performance Considerations

Compiler Directives and the C Preprocessor

- Compile-Time Directives
- Use of typedef
- C Preprocessor Syntax

C++ Programming for Non-C Programmers

Duration: 90 hours

Prerequisites: Prior programming experience in any programming language.

Description: This hands on C++ programming course provides an accelerated introduction to the most essential syntactical components of the C and C++ languages for the first briefer portion of class, and the majority remainder of the course focuses on object-oriented programming with C++. The C++ programming training begins by introducing the built in data types, fundamental control constructs, and rich expression operator repertoire common to both C and C++.

Later, the central concepts of C++ syntax and style are taught in the context of using object-oriented methods to achieve reusability, reliability, and adaptability. Emphasis is placed on the features of C++ that support abstract data types, inheritance, and polymorphism. Students will learn to apply the process of data abstraction and class design in their C++ code development. Extensive code examples and programming exercises are provided, with approximately half of course time spent performing hands on programming labs. Practical aspects of C++ programming including efficiency, performance, testing, and reliability considerations are emphasized throughout the course.

Course Overview

ANSI C++ Fundamentals

- Block Structure of C and C++ Programs
- Fundamentals of Syntax
- Built in Data Types
- The Preprocessor and Macros
- Standard Runtime Libraries and Header Files

Data Types, Storage, Classes, and Scope

- Data Types and Qualifiers
- Constants and String Literals
- Static versus Automatic Storage
- Scope and Variables
- Initialization Rules

Operators and Expressions

- Arithmetic, Logical, and Bit Operators
- Precedence and Associativity
- Assignment
- Type Conversion Rules
- Type Casting

Flow Control Constructs

- Conditional Constructs: if, switch
- Looping Constructs: while, do, for
- Programming Style

Pointers

- Advantages of Pointers
- Uses of Pointers
- Declaring Pointers

Functions

- Purpose of Functions
- Functions versus Inlining
- The Argument Stack

- Pointer and Address Arithmetic
- Initializing and Dereferencing Pointers
- Pointers vs. Arrays

- Passing by Value
- Passing by Reference
- Declaring External Functions
- Function Prototyping

Moving from C to C++

- New Compiler Directives
- Stream Console I/O
- Explicit Operators
- Standard Libraries
- Data Control Capabilities

Handling Data

- New Declaration Features
- Initialization and Assignment
- Enumerated Types
- The bool Type
- Constant Storage
- Pointers to Constant Storage
- Constant Pointers
- References
- Constant Reference Arguments
- Volatile Data
- Global Data

Functions

- Function Prototypes and Type Checking
- Default Function Data Types
- Function Overloading
- Problems with Function Overloading
- Name Resolution
- Promotions and Conversions
- Call by Value
- Reference Declarations
- Call-by-Reference and Reference Types
- References in Function Return
- Constant Argument Types
- Conversion of Parameters Using Default Initializers
- Providing Default Arguments
- Inline Functions

Creating and Using Objects

- Creating Automatic Objects
- Creating Dynamic Objects
- Calling Object Methods
- Constructors
- Initializing Member consts
- Initializer List Syntax
- Allocating Resources in Constructor
- Destructors
- Block and Function Scope
- File and Global Scope
- Class Scope
- Scope Resolution Operator ::
- Using Objects as Arguments
- Objects as Function Return Values
- Constant Methods
- Containment Relationships

Dynamic Memory Management

- Advantages of Dynamic Memory Allocation
- Static, Automatic, and Heap Memory
- Free Store Allocation with new and delete
- Handling Memory Allocation Errors

Controlling Object Creation

- Object Copying and Copy Constructor
- Automatic Copy Constructor
- Conversion Constructor

Inheritance

Streaming I/O

- Inheritance and Reuse
- Composition vs. Inheritance
- Inheritance: Centralized Code
- Inheritance: Maintenance and Revision
  - Public, Private and Protected Members
  - Redefining Behavior in Derived Classes
  - Designing Extensible Software Systems
- Syntax for Public Inheritance
- Use of Common Pointers
- Constructors and Initialization
- Inherited Copy Constructors
- Destructors and Inheritance
- Public, Protected, Private Inheritance

Introduction to Object Concepts

- The Object Programming Paradigm
- Object-Orientated Programming Definitions
- Information Hiding and Encapsulation
- Separating Interface and Implementation
- Classes and Instances of Objects
- Overloaded Objects and Polymorphism

Strings in C++

- Character Strings
- The String Class
- Operators on Strings
- Member Functions of the String Class

C++ Program Structure

- Organizing C++ Source Files
- Integrating C and C++ Projects
- Using C in C++

Polymorphism in C++

- Definition of Polymorphism
- Calling Overridden Methods
- Upcasting
- Accessing Overridden Methods
- Virtual Methods and Dynamic Binding
- Virtual Destructors

- Streams and the iostream Library
- Built-in Stream Objects
- Stream Manipulators
- Stream Methods
- Input/Output Operators
- Character Input
- String Streams
- Formatted I/O
- File Stream I/O
- Overloading Stream Operators
- Persistent Objects

Templates

- Purpose of Template Classes
- Constants in Templates
- Templates and Inheritance
- Container Classes
- Use of Libraries

Exceptions

- Types of Exceptions
- Trapping and Handling Exceptions
- Triggering Exceptions
- Handling Memory Allocation Errors

Reliability Considerations in C++ Projects

- Function Prototypes
- Strong Type Checking
- Constant Types
- C++ Access Control Techniques

Multiple Inheritance

- Derivation from Multiple Base Classes
- Base Class Ambiguities
- Virtual Inheritance
  - Virtual Base Classes
  - Virtual Base Class Information

- Abstract Base Classes and Pure Virtual Methods

Declaring and Defining Classes

- Components of a Class
- Class Structure
- Class Declaration Syntax
- Member Data
- Built-in Operations
- Constructors and Initialization
- Initialization vs. Assignment
- Class Type Members
- Member Functions and Member Accessibility
- Inline Member Functions
- Friend Functions
- Static Members
- Modifying Access with a Friend Class

The Standard Template Library

- STL Containers
- Parameters Used in Container Classes
- The Vector Class
- STL Algorithms
- Use of Libraries

Operator Overloading

- Advantages and Pitfalls of Overloading
- Member Operator Syntax and Examples
- Class Assignment Operators
- Class Equality Operators
- Non-Member Operator Overloading
- Member and Non-Member Operator Functions
- Operator Precedence
- The this Pointer
- Overloading the Assignment Operator
- Overloading Caveats

C++ Programming for C Programmers

Duration: 75 hours

Prerequisites: Prior programming experience with C.

Description: This C++ course presents a thorough introduction to object-oriented programming in C++ for experienced C programmers. The central concepts of C++ syntax and style are taught in the context of using object-oriented methods to achieve reusability, adaptability and reliability. Emphasis is placed on the features of C++ that support abstract data types, inheritance, and polymorphism. Students will learn to apply the process of data abstraction and class design. Extensive programming examples and exercises are provided, with approximately half of course time spent performing hands on programming labs. Practical aspects of C++ programming including efficiency, performance, testing, and reliability considerations are stressed throughout.

Course Overview

Moving from C to C++

- New Compiler Directives
- Stream Console I/O
- Explicit Operators
- Standard Libraries
- Data Control Capabilities

Handling Data

- New Declaration Features
- Initialization and Assignment
- Enumerated Types
- The bool Type
- Constant Storage
- Pointers to Constant Storage
- Constant Pointers
- References
- Constant Reference Arguments
- Volatile Data
- Global Data

Functions

- Function Prototypes and Type Checking
- Default Function Data Types
- Function Overloading
- Problems with Function Overloading
- Name Resolution
- Promotions and Conversions
- Call by Value
- Reference Declarations
- Call-by-Reference and Reference Types
- References in Function Return
- Constant Argument Types

Creating and Using Objects

- Creating Automatic Objects
- Creating Dynamic Objects
- Calling Object Methods
- Constructors
- Initializing Member consts
- Initializer List Syntax
- Allocating Resources in Constructor
- Destructors
- Block and Function Scope
- File and Global Scope
- Class Scope

- Conversion of Parameters Using Default Initializers
- Providing Default Arguments
- Inline Functions
- Scope Resolution Operator ::
- Using Objects as Arguments
- Objects as Function Return Values
- Constant Methods
- Containment Relationships

Dynamic Memory Management

- Advantages of Dynamic Memory Allocation
- Static, Automatic, and Heap Memory
- Free Store Allocation with new and delete
- Handling Memory Allocation Errors

Controlling Object Creation

- Object Copying and Copy Constructor
- Automatic Copy Constructor
- Conversion Constructor

Inheritance

- Inheritance and Reuse
- Composition vs. Inheritance
- Inheritance: Centralized Code
- Inheritance: Maintenance and Revision
  - Public, Private and Protected Members
  - Redefining Behavior in Derived Classes
  - Designing Extensible Software Systems
- Syntax for Public Inheritance
- Use of Common Pointers
- Constructors and Initialization
- Inherited Copy Constructors
- Destructors and Inheritance
- Public, Protected, Private Inheritance

Streaming I/O

- Streams and the iostream Library
- Built-in Stream Objects
- Stream Manipulators
- Stream Methods
- Input/Output Operators
- Character Input
- String Streams
- Formatted I/O
- File Stream I/O
- Overloading Stream Operators
- Persistent Objects

Introduction to Object Concepts

- The Object Programming Paradigm
- Object-Orientated Programming Definitions
- Information Hiding and Encapsulation
- Separating Interface and Implementation
- Classes and Instances of Objects
- Overloaded Objects and Polymorphism

Templates

- Purpose of Template Classes
- Constants in Templates
- Templates and Inheritance
- Container Classes
- Use of Libraries

Strings in C++

- Character Strings
- The String Class
- Operators on Strings
- Member Functions of the String Class

Exceptions

- Types of Exceptions
- Trapping and Handling Exceptions
- Triggering Exceptions
- Handling Memory Allocation Errors

C++ Program Structure

Reliability Considerations in C++ Projects

- Organizing C++ Source Files
- Integrating C and C++ Projects
- Using C in C++
- Function Prototypes
- Strong Type Checking
- Constant Types
- C++ Access Control Techniques

Polymorphism in C++

- Definition of Polymorphism
- Calling Overridden Methods
- Upcasting
- Accessing Overridden Methods
- Virtual Methods and Dynamic Binding
- Virtual Destructors
- Abstract Base Classes and Pure Virtual Methods

Multiple Inheritance

- Derivation from Multiple Base Classes
- Base Class Ambiguities
- Virtual Inheritance
    - Virtual Base Classes
    - Virtual Base Class Information

Declaring and Defining Classes

- Components of a Class
- Class Structure
- Class Declaration Syntax
- Member Data
- Built-in Operations
- Constructors and Initialization
- Initialization vs. Assignment
- Class Type Members
- Member Functions and Member Accessibility
- Inline Member Functions
- Friend Functions
- Static Members
- Modifying Access with a Friend Class

Operator Overloading

- Advantages and Pitfalls of Overloading
- Member Operator Syntax and Examples
- Class Assignment Operators
- Class Equality Operators
- Non-Member Operator Overloading
- Member and Non-Member Operator Functions
- Operator Precedence
- The this Pointer
- Overloading the Assignment Operator
- Overloading Caveats

The Standard Template Library

- STL Containers
- Parameters Used in Container Classes
- The Vector Class
- STL Algorithms
- Use of Libraries

Customer Support and Relations Management

Duration: 102 hours (including 40 hours of field work)

Prerequisites: Basic personal computer skills and basic Internet knowledge.

Description: Exceptional customer support can mean different things to different people, but at the heart of it, it's solving for your customer. This course will expose students to many different concepts and skills to ensure that students have a well-rounded understanding of customer support and other aspects of business, so that effective problem solving and customer service can be provided in many types of job environments.

In addition to covering "soft skills" such as critical thinking, time management, and general understanding of the sales process, during the course students will also spend a significant amount of course time learning about and working with Customer Relationship Management (CRM) software, to gain practical experience with real customer management software so that they will be more confident when they need to use similar software in their future jobs. CRM allows users to navigate many aspects of customer support, such as technical support case management, providing dashboards to let an employee know if certain tasks still need to be completed, viewing purchase histories, and internal communications. This particular course will utilize HubSpot CRM software, and having prior experience with many of the tasks performed in class, such as support ticket creation, will be useful no matter what brand of CRM software a student will wind up using in their future workplace. Many positions in the workforce also use CRM, even if they don't have phrases like "Customer Support" or "Customer Service" in the job title. For example, advanced technical support positions often use CRM software as they manage progression and resolution of customer technical support issues and bug tracking. Sales positions also use CRM in some form to manage various stages of the sales process with a customer.

The course also includes 40 hours of practical field work, in which the student can observe and participate in actual support situations. Students will be given the opportunity to experience real world business protocol training at corporations with different CRM systems. This will include training in real world business environments, while learning different types of help desk and customer service communication procedures. Students will learn how to analyze and identify problems, and then will be taught how to propose one or more solutions in a variety of industries. Putting concept into practice is always a valuable component in the learning process, and being able to apply what they've studied during class and encountering unexpected situations and resolving them in a managed environment will better prepare students for future scenarios in their ensuing workplaces.

Course Overview

Introduction to Customer Support

Handling Data

- Understanding Customer Support Competencies

- Course Roadmap

- Framework of a Support Case
  - Issue Discovery
  - Troubleshooting and Consultation
  - Resolution and Impact
- Managing Your Time as a Support Rep

CRM Software and Configuration

- Understanding CRM and Help Desk
- Setting Up Conversations & Messages
- Setting Up Tickets
- Setting Up a Knowledgebase
- Setting Up a Customer Portal
- Setting Up Customer Feedback
- Setting Up Reports and Analytics

Inbound Service Fundamentals

- Guiding Customers to Good Outcomes
- The Customer Experience and Meeting Expectations

Understanding the Sales Process

- Inbound Sales Fundamentals
- Prioritizing Active Buyers over Passive Buyers
- Earning the Attention of Todays Empowered Buyer
- Understanding the Buyers Context
- Delivering Personalized Sales Presentations
- Prospecting and Closing Sales
  - Identifying Good Fit Leads
  - Connecting With Your Contacts
  - Closing Sales

Critical Thinking and Problem Solving in the Workplace

- Critical Thinking Foundations
- Critical Thinking in Action
- Effective Problem Solving
- Strategic Decision Making

Cyber Security Cloud Analyst

Duration: 480 hours

Prerequisites: Rudimentary knowledge of computer systems. Students should know how to basically operate a computer, browse the world wide web, send and receive email messages, and write and print memos..

Description: The Cyber Security Cloud Analyst course is tailored for professionals aiming to excel in securing cloud environments. The course first educates students on fundamental principles and topics in cloud computing, and provides them with hands on experience in a cloud environment. Afterwards, they will learn general IT security concepts and fundamentals, and then will take a deeper dive into applying those principles in a cloud software environment.

Course attendees will begin their training by learning how cloud computing is implemented, and how to handle basic design and management issues when utilizing cloud computing services. Attendees will obtain the skills required to determine when a cloud solution is appropriate for an organization, develop cloud infrastructure, implement the cloud development life cycle, manage cloud operations, and understand cloud resource governance. Upon completion of this portion of the course, attendees will have a working-level proficiency with cloud computing.

Next, students will learn about the important task of securing cloud environments. After an introduction to fundamental security concepts, students will dive into the complexities of cloud security, covering key areas such as risk management and compliance in the cloud, data protection and encryption, cloud architecture, cloud network security, and more. Upon completion of this part of the course, the student will be prepared to become a certified IT Specialist in Cloud Computing and will be able to perform the basic responsibilities of an entry-level professional in the field.

Featuring a mix of detailed video lessons, practical lab exercises, and case studies, the course provides hands-on experience with the latest cloud security tools and techniques. The course offers flexibility through a self-paced format, allowing you to learn on your own schedule. Expert instructors are on hand to provide personalized support and ensure a deep comprehension of advanced cloud security concepts. This cloud security analyst course is ideal for aspiring cloud security professionals or those looking to enhance their expertise in cloud protection. This course will also prepare students for the Certificate of Cloud Security Knowledge (CCSK) exam, so they can graduate from the course with an added certificate to boost their resume.

Course Overview

Fundamentals of Cloud Computing

- Essential Concepts in Cloud Computing
- Planning a Cloud Computing Project

Introduction to Cybersecurity

- Overview of Cybersecurity
  - Definition and Scope of Cybersecurity

- Selecting Cloud Solutions
- Defining Cloud Solution Requirements
- Determining Hardware and Storage Requirements for a Cloud Project
- Designing Cloud Apps and Solutions
- Developing and Testing Cloud Apps
- Integrating and Deploying a Cloud Solution
- Managing Cloud Operations
- Evaluating Cloud Governance
    - Historical Evolution
    - Current Trends and Technologies
- The Importance of Cybersecurity
    - Impact on Personal and Financial Information
    - Consequences of Cyberattacks
    - Role in National Security
- Cybersecurity Roles and Career Paths
    - Key Cybersecurity Roles
    - Required Skills and Qualifications
    - Career Development and Growth

Understanding Cyber Threats

- Types of Cyber Threats
    - Malware and Ransomware
    - Phishing and Social Engineering
    - Advanced Persistent Threats (APTs)
- Common Attack Vectors
    - Email Attachements and Links
    - Unpatched Software and Vulnerabilities
    - Insider Threats
- Case Studies
    - Notable Cybersecurity Breaches
    - Successful Cybersecurity Defenses
    - Emerging Threats and Responses

Basic Cybersecurity Concepts

- The CIA Triad
    - Confidentiality, Integrity, Availability
- Security Policies and Procedures
    - Developing Effective Security Policies
    - Incident Response Procedures
    - Compliance and Regulatory Requirements
- Introduction to Risk Management
    - Risk Assement Processes
    - Risk Mitigation Strategies
    - Risk Communication
- Understanding Risks
    - Types of Risks
    - Risk Management Frameworks
    - Risk Monitoring and Review

Fundamentals of Network Security

- Introduction to Networks
    - Basic Networking Concepts
    - Types of Networks
    - Network Devices and Protocols
- Network Security Basics
    - Firewalls and Intrusion Detection Systems (IDS)
    - Network Segmentation and VLANs
    - Encryption and Secure Communication
- Wi-Fi Security
    - Common Wi-Fi Vulnerabilities
    - Securing Wi-Fi Networks
    - Public Wi-Fi Risks and Precautions
    -

Introduction to Cloud Security

- Understanding Cloud Computing
    - Overview of Cloud Computing Models (IaaS, PaaS, SaaS)
    - Key Cloud Service Providers (AWS, Azure, Google Cloud)
    - Cloud Deployment Models (Public, Private, Hybrid)
- Cloud Security Fundamentals
    - Basic Principles of Cloud Security
    - Shared Responsibility Model
    - Key Security Concepts
- Cloud Security Architecture
    - Cloud Security Architecture Components
    - Network Security in the Cloud (VPC, Subnets, Security Groups)

        - Intro to Identity and Access Management (IAM)
    - Cloud Compliance and Governance
        - Understanding Compliance Frameworks (GDPR, HIPAA, PCI-DSS)
        - Cloud Governance Practices
        - Data Protection and Privacy Considerations

Cloud Network Security

- Network Security Fundamentals
    - Basic Networking Concepts (IP, DNS, TCP/IP)
    - Network Security Principles
    - Cloud-Specific Networking Components
- Securing Cloud Networks
    - Configuring Virtual Private Cloud (VPC) and Subnets
    - Implementing Security Groups and Network ACLs
    - Setting Up VPNs and Private Connections
- Advanced Network Security
    - Intrusion Detection and Prevention Systems (IDPS)
    - Network Segmentation and Micro-Segmentation
    - Security in Serverless Architectures

Identity and Access Management (IAM)

- IAM Concepts and Architecture
    - Identity Management and Authentication
    - Authorization Models (RBAC, ABAC)
    - Federation and Single Sign-On (SSO)
- Implementing IAM in Cloud Environments
    - Configuring IAM Roles, Policies, and Permissions
    - Managing User and Group Access
    - Implementing Multi-Factor Authentication (MFA)
- Security and Best Practices
    - Least Privilege Principle
    - IAM Policy Management Best Practices
    - Auditing and Monitoring IAM Activities

Security Operations and Incident Response

- Security Monitoring and Logging
    - Cloud-Native Logging and Monitoring Tools
    - Setting Up Alerts and Dashboards
    - Analyzing and Interpreting Security Logs
- Incident Response
    - Incident Response Lifecycle
    - Cloud-Specific Incident Response Strategies
    - Case Studies and Lessons Learned
- Vulnerability Management and Penetration Testing
    - Identifying and Managing Vulnerabilities
    - Cloud-Specific Penetration Testing Tools and Techniques

- Best Practices for Securing Cloud Applications

Developing Web Applications Using Angular

Duration: 90 hours

Prerequisites: Knowledge of HTML, CSS and JavaScript equivalent to attending the **Website Development with HTML5, CSS and Bootstrap** and **JavaScript Programming** courses. Knowledge of jQuery is helpful, but not required.

Description: This hands on programming course provides an in-depth introduction to the Angular Framework for the purpose of web application development. Attendees will learn the fundamental skills necessary to build Web Applications using Angular and the MVVM (Model-View-ViewModel) design pattern. Topics include using TypeScript and ECMAScript to create object-oriented Angular applications, extending HTML by creating reusable UI components, implementing data-binding, designing and using custom structural and attribute directives, as well as creating and using Angular pipes for formatting and transforming data in the UI. Students will explore creating UX's (User Experiences) by designing Web animations and implementing both template-driven and reactive style forms. Students will learn to use Angular routing to create SPA's (Single Page Applications). The course includes coverage of using DI (Dependency Injection) and Angular services to provide business and data-access logic to the application, both locally as well as communicating with RESTful web services to provide CRUD database operations.

Comprehensive hands on exercises are integrated throughout to reinforce learning and develop real competency.

Course Overview

Overview of Angular and the MVVM Design Pattern

- Features and Benefits of Angular
- Angular Architecture
- MVVM Design Pattern Overview
- Downloading Angular
- Choosing an IDE
- Creating a Simple Application with Angular

Using Visual Studio Code

- Downloading and Installing Visual Studio Code
- Generating Angular Projects with the CLI (Angular Command Line Interface)
- Angular Project Structure, Files and Configuration
- Debugging Applications in VS Code and the Browser
- Using the Terminal Window

Working with ECMAScript

- Enhancements to Legacy JavaScript
- New let to Declare Variables
- Block Scoping
- Using for of Loops
- Literals and Strings
    - Extended Literal Support

Working with TypeScript

- Types
    - Working with Built-In Types
    - Custom Types
- Setting Up and Using Node.js
- Transpiling TypeScript into JavaScript
- TypeScript Compiler Configuration

    - Template Literals
    - Object Literal Changes
- Function Enhancements
    - Default Parameters
    - Rest and Spread Operators
    - Arrow Functions/Lambdas

Angular Components

- Component LifeCycle
- Component Templates to Define Views
- Using Decorators to Define MetaData
- Styling
    - Per-Component Styling
    - Defining Global Styles in angular.json
    - Adding Bootstrap Framework to an Angular App
- View Encapsulation
    - ViewEncapsulation.ShadowDom
    - ViewEncapsulation.ShadowDom
    - ViewEncapsulation.ShadowDom
- Referencing DOM elements with ElementRef
- Lifecycle Hooks
    - OnInit, OnDestroy, OnChanges, DoCheck
    - AfterContentInit, AfterViewInit
    - AfterContentChecked, AfterViewChecked
- Change Detection
- Passing Data to Components

Angular Directives

- Built-In Directives
- TypeScript Functions
    - Default and Optional Parameters
    - Function Overloading
    - Parameter Types and Return Types

Angular Modules

- Using Modules to Create an Application
- Default Modules
- Bootstrapping an Application
- Exporting Classes, Functions and Values
- Limiting Scope
- Grouping Modules
- Specifying Module Dependencies
- Organizing Code Files
- Module Testing
- Best Practices

Component Templates and Data Binding

- Basic Data Binding Concepts
- Interpolation
- One-Way Property Binding
- Two-Way Property Binding
- Event Binding
- Custom Binding
    - Exposing Properties and Events to Parent Controls
    - Custom Property Binding
    - Custom Event Binding

Working with Pipes

- Text Casing Pipes

        - NgIf, NgFor, NgClass, NgStyle, NgSwitch, etc.
    - Building Custom Directives
        - Using the Renderer2 Service
        - ElementRef
        - Attribute Directives
        - Structural Directives

Defining and Consuming Services

- Dependency Injection
    - Registering Providers with the Injector
    - Changes to Injection in Angular 6
- Creating a Service
- Consuming a Service

Asynchronous Programming in Angular

- Reactive Programming Model
- The RxJs Library
    - Observables
    - Observers
    - Subjects
    - Subscriptions
    - Operators
- Using Operators
    - Creation Operators
    - Filtering of Data Operators
    - Conversion of Data Operators
    - Math and Aggregate Operators
    - Utility Operators
    - Pipeable Operators
- Using the async Pipe
- Changes to RxJs in Version 6

Angular Routing and Navigation

- Formatting Numbers and Dates
- Internationalization and Cultures
- Restricting Data Collections with slice
- Custom Pipes
    - Implementing the PipeTransform Interface
    - Pure vs Impure Pipes
    - Passing Parameters to Pipes

Working with Web Services

- Using HttpClient
- Importing the HTTP Module
- Creating Requests
- Processing Responses
- Web API
- Using PostMan to Test the Server-Side Service
- Interacting with a RESTFul Service
    - POST Requests
    - DELETE Requests
    - PUT Requests
    - HEAD Requests
- Dealing with CORS (Cross Origin Resource Sharing)

Angular Forms

- The ngNativeValidate Directive
- HTML novalidate Attribute
- Template Driven Forms
    - ngForm and ngModel in Forms
    - Input and Output Properties
- Reactive-Driven/Model-Driven Forms
    - FormGroup
    - FormControl
    - FormArray
    - Validators Class
- Using the FormBuilder Factory
- Working with Form State
- Client-Side Forms Validation

Unit Testing and TDD with Angular

- Overview of Routing
- Implementing Single Page Applications (SPAs)
- Location Strategies
- Client-Side vs Server-Side Routing
- Working with the Component Router
- Adding Router Imports
- Performing Router Configuration
- Using Router State
- Redirects
- Routing Parameters
- Router Lifecycle Events
- Nesting Routes
- Routing Guards
    - Defining a Guard
    - Registering Guards
    - Securing Routes
    - CanActivate Guard
    - CanActivateChild Guard
    - CanDeactivate Guard
    - CanLoad Guard

The Angular Animation System

- The Web Animations API
- States and Transitions
- Entering and Leaving
- Animating Properties
- KeyFrames
- Parallel Animation Groups

- TDD vs End-to-End Testing
- Jasmine Testing Framework
    - Defining Expectations
- Running Tests in Karma
- Using the Angular Unit Test Framework
    - Fulfilling Dependencies
    - Mocking Out Data
    - Creating Testing Fixtures
- Testing Services and HTTP
- Using Test-Doubles (Mocks, Stubs and Spies)
- Testing Components
- Testing Forms

Building and Deploying Applications

- Developing a Deployment Strategy
- Managing Dependencies
- Tree Shaking
- Transpiling
- Linting

Effectively Using Java Packages and Features

Duration: 88 hours

Prerequisites: Java SE programming experience and an understanding of object-oriented design principles. Fundamental knowledge of XML is helpful but not required. The Java Programming course or equivalent knowledge provides a solid foundation.

Description: This intermediate level course is intended for programmers who already have a fundamental understanding of Java programming and some experience writing code. It provides additional insights and details regarding some of the more advanced and useful capabilities contained in the Java Programming Language and its associated packages. Topics include reflection and JavaBeans, Java type safety enhancements, the Java Collections Framework, Java Database Connectivity (JDBC), multithreading, inner classes, lambda expressions and networking.

Comprehensive hands on exercises are integrated throughout to reinforce learning and develop real competency.

Course Overview

JavaBeans, Reflection, and Introspection

- JavaBean Requirements
- Determining Type of an Object
- Reflection Overview and Uses
- Reflection Issues
- Reflection Using the Class Object
- Invoking Methods Using Reflection
- Methods Available on the Class Object
- Creating a New Instance
- Introspection
- Customizing `BeanInfo`
- Invoking Methods Using Introspection
- Using Reflection to Access Properties
- Review of `equals` Method

Type Safety Enhancements

- Annotations
  - Standard Annotations
  - User Defined Annotations
  - Reflection Annotation Information
- The `enum` Data Type
- Generics
- Autoboxing
- Methods Having Variable Parameters
- Assertions

Collections Framework

- What is the Collections Framework
- Simple Arrays and Arrays of Objects
- Legacy Container Classes
- Collections Framework Overview
- Collections Interfaces

JDBC

- What is JDBC
- JDBC Drivers
- Accessing the Database
  - Loading Driver
  - Connecting to Data Source

    - Lists
    - Sets
    - Queues
- Map Interfaces
- Interface Implementations
    - Lists
    - Sets
    - Queues
    - Maps
    - Iterators

Threads

- Definition of a Thread
- Creating Threads
- Naming Threads
- Data Sharing Among Threads
    - Local Data
    - Instance Data
    - Class Data
    - `volatile` Keyword
- Thread States
- Thread Priority
    - Minimum. Maximum, Normal Priorities
    - Preemptive Thread Priority
    - `setPriority` Method
    - `getPriority` Method
- Piping Data Between Threads
- Coordination and Controlling Threads
- Synchronizing Threads
    - Why Synchronization is Needed
    - Producer/Consumer Example
    - `synchronized` Keyword
    - Thread Synchronization Methods

Inner and Nested Classes

- What Are Inner Classes
    - Benefits of Inner Classes
    - Terminology
    - Creating Statements
    - Executing Statements
- Processing Result Sets
    - Cursor Positioning
    - Column Retrieval
    - Updating Result Sets
- Connection Pooling
- Processing Errors and Warnings
- Using Prepared Statements
- Using Stored Procedures
- Metadata
    - Result Set Metadata
    - Database Metadata
- Transaction Processing
- Isolation Levels
- SQL Batches

Networking

- Overview of `java.net` Package
- Format of URL
- `URL` Class
- `URLConnection` Class
- `InetAddress` Class
- Definitions
    - Client
    - Server
    - Port
    - Socket
- TCP/IP Protocols
- Socket Class
    - Writing Client Side Applications
    - Read From Socket
    - Write to Socket
- `ServerSocket` Class
    - Writing Server Side Applications
    - Daemon Servers
    - Multi-Threaded Servers

        - Restrictions
        - Syntax
- Types of Inner Classes
    - Member Inner Class
    - Local Inner Class
    - Anonymous Inner Class
    - Nested Top Level Class
- Lambda Expressions
    - Functional Interfaces
    - Default Interface Methods
    - Static Interfaces Methods
    - Lambda Expression Uses
    - Lambda Expression Syntax
    - Method and Constructor References

Introduction to ASP.NET Core MVC

Duration: 75 hours

Prerequisites: C# programming experience.

Description: This ASP.NET Core MVC training course is designed to provide an introduction to .NET Core for programmers who already are familiar with the C# language. The course focuses on core portions of the .NET Framework that are common across many application areas. It starts with an introduction to the architecture and key concepts of .NET. This .NET Core course then discusses class libraries, packages, metapackages and frameworks. Coverage includes working with delegates and events, I/O and serialization, memory management, processes and threads as well as an introduction to the Task Parallel Library (TPL).

This course also provides a practical hands-on introduction to developing Web applications using ASP.NET Core MVC and C#. This Web development framework from Microsoft emphasizes the separation of concerns in the architecture and testability of applications. This .NET Core course covers the fundamentals of the Model-View-Controller design pattern and its implementation in ASP.NET Core MVC. For lab work, Visual Studio 2019 with ASP.NET Core is used as a productive platform for creating MVC Web applications.

After presenting the fundamentals of the technology with several examples, the main components of Model, Controller and View are covered in detail. The discussion of the Model incorporates Microsoft technologies for persisting data, which includes XML Serialization and ADO.NET with SQL Server. The routing mechanism of ASP.NET MVC is also covered, and the course includes an introduction to ASP.NET Web API as well.

Comprehensive hands on exercises are integrated throughout the course to reinforce learning and develop real competency.

Course Overview

Overview of ASP.NET Web Forms

- What is Microsoft .NET?
- Common Language Runtime
- Framework Class Library
- Language Interoperability
- Managed Code
- .NET Core and Cross-Platform Development

Class Libraries

- Components in .NET
- Comparing Components in COM and .NET
- Creating Class Libraries
- Using Components in Client Programs by Obtaining References to Class Libraries
- Using References

Packages and Frameworks

I/O and Serialization

- Overview of NuGet Packages and the NuGet Gallery
- Explaining the Role of Packages, Metapackages and Frameworks in .NET
- How Packages and Metapackages are Used in .NET Core
- Using the Visual Studio Package Manager to Manage Packages in Solutions
- Installing Packages from the NuGet Gallery
- Creating and Using Your Own NuGet Packages
- Porting from .NET 4.6 to .NET Core

- Using .NET Framework Classes for Working with Directories and Files
- Explaining and Using Streams for Performing File I/O
- Explaining the Role of Serialization in Persisting and Transporting Objects
- Serialization Mechanisms Available in .NET Core Contrasted with Classical .NET
- Using XML Serialization in .NET Core Programs

Delegates and Events

- Using Delegate Objects to Implement Callbacks
- Using the Random Class to Generate Random Test Data
- Using Aggregations of Delegate Objects
- Using Delegate Objects to Implement and Handle Event Notifications

.NET Programming Model

- Garbage Collection in .NET
- Implementing the Finalize/Dispose Pattern
- Using the Process and Thread Model for .NET Applications
    - Process and Thread Isolation
- Using the Process Class to Manage Processes in .NET Core Applications
- Working with Command-Line Arguments

.NET Threading

- Using the Thread Class to Implement Multithreading in .NET Applications
- Using the Monitor Class to Program Safe Concurrent Access to Shared Data
- Using the ThreadPool Class to Obtain Threads from a Pool that is Managed by the System
- Difference Between Foreground and Background Threads
- Overview of Different Classes that Can be Used for Synchronizing Threads
- Using the Task Parallel Library to Implement Task Parallelism and Data Parallelism in .NET Applications

Overview of ASP.NET MVC

- Advantages and Disadvantages of ASP.NET Web Forms
- Understanding the Model-View-Controller (MVC) Pattern
- Outlining the Parts of an ASP.NET MVC Application
- Advantages and Disadvantages of ASP.NET MVC
- Overview of ASP.NET Core
- Understanding the Use of Unit Testing in Creating ASP.NET MVC Applications

Introduction to ASP.NET Core MVC

- Understanding How ASP.NET Core MVC is Used Within Visual Studio
- Creating Several Versions of a Simple ASP.NET Core MVC Application

ASP.NET MVC Architecture

- Understanding the Controller in ASP.NET MVC and its Role
- Understanding the View in ASP.NET MVC and its Role

- Understanding How Views are Rendered
- Using the Razor View Engine in ASP.NET Core MVC
- Understanding How Dynamic Output Works
- Passing Input Data to an MVC Application in a Query String
- Rendering Views

The Model

- Working with More Complex Models in MVC Programs
- Overview of Various Microsoft Technologies Available to Use and Persist the Model
  - ADO.NET
  - LINQ
  - ADO.NET Entity Framework and LINQ to Entities
  - XML
- Using XML Serialization Technique to Persist a Complex Model
- Using the NuGet Package Manager and Use it to Install a New Package
- Persisting a Model Using ADO.NET

The View

- Responsibility of the View
- Using ViewBag as a Dynamic Type for Passing Data from a Controller to a View
- How Dynamic Objects Can be Used in ASP.NET Core MVC
- Using HTML Helpers
  - String Helpers
  - Link-Building Helpers
  - Form Helpers
  - Validation and Templated Helpers
- Using Validation Attributes in the Model

ASP.NET Core Web API

- Overview of the ASP.NET Web API
- Overview of RESTful Web Services
- Implementing HTTP Services Using the Web API with ASP.NET Core MVC

- Understanding the Model in ASP.NET MVC and its Role
- Use of Helper Methods for HTML in ASP.NET MVC
- How Form Submission, Model Binding and Input Validation Work in ASP.NET MVC

The Controller

- Facilities Provided by the Controller Base Class
  - Action Methods
  - Action Results
  - Filters
- Overview of How Controllers Can Receive Input
- Principal Types of Output from a Controller
- Using Attributes to Control How Actions are Invoked
- How Asynchronous Controllers Work and How they Can be Used

Routing

- Overview of the Use of Routing in ASP.NET Core MVC
- Understanding the Properties of Routes in ASP.NET Core MVC
- Understanding the Use of Parameters in Routing
- Understanding How to Register Routes and the Importance of Order
- Using Attribute Routing to Map Actions Directly to Route Templates by Means of Attributes

ASP.NET Core and Azure

- What Is Windows Azure?
- A Windows Azure Testbed
- Deploying an Application to Azure
- Updating an Application on Azure

- Using Postman to Exercise HTTP Requests while Developing a Web API Service
- Implementing Web API clients

Introduction to Programming Concepts

Duration: 11 hours

Prerequisites: Rudimentary knowledge of computer systems. Students should know how to basically operate a computer, browse the world wide web, send and receive email messages, and write and print memos.

Description: Courses that teach programming languages tend to focus on the particulars of each language (as they should), expending only minimal time discussing basic universal programming concepts. Students who are new to programming may find themselves overwhelmed in discussions of the details of matters such as control flow operators, executable images, third generation languages, and logical operators before they really understand the concepts. This situation can leave the student lost or frustrated or both.

This course is offered for the aspiring programmer to provide them the foundational knowledge that is necessary to learn and advance in a programming career. Students will be exposed to fundamental concepts that are utilized by all programming languages in this course.

Course Overview

The Essence of Programming

- The language of computers
- Computer components
- Hardware and software
- Operating systems
- Programs

The Tasks of Programmers

- Analysis and requirements gathering
- Design and planning
- Coding
- Testing
- Deployment/Documentation
- Maintenance
- Your job

Types of Programming

- Computer languages
- Machine code
- Early programming
- The evolution of programming languages
- First Generation Languages
- Second Generation Languages
- Third Generation Languages
- Fourth Generation Languages
- Fifth Generation Languages
- Compilers

Statements and Storage

- The organization of programs
- Libraries
- Declaring and initializing variables and constants
- Data types: integers, floats, Booleans, characters, arrays, structures, objects, pointers
- Proper data typing
- Allocation
- Static and dynamic data typing
- Strong and weak data typing
- Scope

- The process of compiling
- Practical compiling
- Interpreted languages
- Pros and cons or compiled and interpreted languages
- Imperative programming
- Structured programming
- Procedural programming
- Object-oriented programming
- Declarative programming
- Functional programming
- Stand-alone programs
- Distributed programming
- Client-server programming
- Web-enabled programming
- Textual programming
- IDEs
- Visual programming

Operators and Control Flow

- Assignment operators
- Arithmetic operators
- Relational operators
- Conditional operators
- Logical operators
- Bitwise operators
- Special operators
- *if*, *if-then*, *if-then-else* statements
- *case* or *switch-case* statements
- *for* or *foreach* statements
- *while* statements
- *do-while* statements

The Real World

- Working with the *Command Prompt*
- Customizing your environment
- Online documentation
- Batch files and scripts
- Installing and using a compiler
- Man pages
- Writing a C program
- Debugging C Programs

- Inheritance

The Language of Programming

- Algorithms
- Strings
- Regular expressions
- Input/Output
- Graphical user interfaces
- Events
- Frameworks
- Debugging
- Compile time vs. run time errors
- Database programming
- Technical communities
- Benchmarks
- Protocols

Introduction to Spring 5, Spring MVC, and Spring REST

Duration: 90 hours

Prerequisites: Java SE programming experience and an understanding of object-oriented design principles. Fundamental knowledge of XML is helpful but not required. The **Java Programming** course or equivalent knowledge provides a solid foundation.

Description: Spring 5 provides an major advance of Spring's powerful capabilities. This course introduces those capabilities, as well as providing guidelines on when and how to use them. It includes coverage of the three main configuration styles: Java-based (@Configuration), annotation-based (@Component), and the traditional XML-based configuration that may still play an important role in existing and new projects.

The course starts with in-depth coverage of Spring's Core module to reduce coupling and increase the flexibility, ease of maintenance, and testing of your applications. It goes on to cover many of the most important capabilities of Spring, including easing configuration with Spring Boot, integrating Hibernate and JPA persistence layers with Spring and Spring Data, and using Spring's declarative transaction capabilities.

The course includes integration of Spring with Java EE Web applications, a solid introduction to Spring MVC, and coverage of building RESTful resources with Spring MVC. It also provides an overview of Springs reactive programming model for repositories and Web resources.

This course is hands on with labs to reinforce all the important concepts. It will enable you to build working Spring applications and give you an understanding of the important concepts and technology in a short timeframe.

The course does all labs with the Eclipse IDE and the lab instructions include detailed directions for setting up and using it.

Course Overview

Introduction to Spring

- Overview of Spring Technology
  - Motivation for Spring, Spring Architecture
  - The Spring Framework
  - maven and Spring
- Spring Introduction
  - Declaring and Managing Beans
  - ApplicationContexts - The Spring Container
  - XML and `@Component`/`@Named` Config
- Dependencies and Dependency Injection (DI)

Configuration in Depth

- Java Based Configuration (@Configuration)
  - Overview, `@Configuration`, and `@Bean`
  - Dependency Injection
  - Resolving Dependencies
- Integrating Configuration Types
  - XML and `@Component` Pros/Cons
  - `@Configuration` Pros/Cons
  - Choosing a Configuration Style

        - Examining Dependencies
        - Dependency Inversion / Dependency Injection (DI)
        - DI in Spring - XML and `@Autowired`
- Spring Boot Quickstart

Spring Boot Overview

- Maven and Spring
- Spring Boot Structure
- Spring POMs with Boot Parents
- Spring Boot Starters

Spring and Spring Data with Hibernate/JPA

- Overview of Spring Database Support
- Configuring a DataSource
- Using Spring with Hibernate
- Using Spring with JPA
- Spring Data Introduction
    - Overview and Architecture
    - Configuring Spring Data
    - Repositories and JPA Repositories
    - Using CrudRepository
- Spring Data Querying
    - Naming Conventions for Querying
    - Creating more Complex Queries
    - Query Configruation

Spring Web Integration and Introduction to Spring MVC

- Java EE Web App Integration
- ContextLoaderListener and WebApplicationContext
- Web MVC Overview
- Spring MVC Basics

RESTful Services with Spring

        - Integrating with `@Import` and `<import>`
- Bean Scope and Lifecycle
    - Singleton, Prototype, and Other Scopes
    - Configuring Scope
    - Bean Lifecycle / Callbacks
- Externalizing Properties
- Profiles

Spring Testing

- Testing and Junit Overview
    - Writing Tests - Test Classes, asserts, Naming Conventions
    - Running Tests - IDE, Maven, ...
    - Test Fixtures - Setup and Teardown
- Spring TestContext Framework
    - Overview
    - Configuration
    - Running Tests

Spring Transaction (TX) Management

- Overview
- Declarative TX Management (REQUIRED, etc.)
- TX Scope and Propagation
- Configuration and Boot Auto-Configuration
- Pointcut-Based Configuration of Transactions

Additional Spring MVC Capabilities

- `@ModelAttribute` and Reference Data
- Forms and Binding, Spring Form Tags
- Sessions and `@SessionAttributes`
- Validation / JSR-303

Java Clients for RESTful Services

- REST Overview, URI Templates
- REST and Spring MVC
- Generating JSON
  - JSON Overview
  - JSON Representations for Resources
  - Message Converters
- Generating XML
  - JAXB and Jackson Message Converters for XML
  - JAXB / @XmlRootElement
- Content Negotiation
- Client Requirements and Spring's RestTemplate
- `getForObject()` / `getForEntity()`
- Other RestTemplate Methods
- Accessing Headers / `exchange()`

Common REST Patterns

- GET: Read
- POST: Create
- PUT: Update
- DELETE: Delete
- Programming on server side and client side (with RestTemplate)

What's New in Spring 5

- Updates to Spring Core
- WebFlux
- Reactive Web Framework

XML Specific Configuration

- Collections: Lists, Sets, etc.
- Additional Capabilities
  - Factory Classes and Factory Methods
  - Definition Inheritance (Parent Beans)
  - AutoWiring with XML
  - Inner Beans and Compound Names

Java Programming

Duration: 90 hours

Prerequisites: Basic programming skills in a structured language. Knowledge and experience with Object-Oriented Design (OOD) is helpful, but not required.

Description: This hands on course introduces experienced programmers to Java™ technology and Java programming techniques. The Java platform provides an object-oriented, portable and robust framework for application development. Included are core language concepts including fundamental data types, flow control, and standard function libraries. The course emphasizes object oriented programming and modular design to support distributed development environments. Included are the design of classes and objects, inheritance and polymorphism, and the details about creating programs for use on a distributed network, with emphasis on JSP, Servlets, and JDBC.The course also includes coverage of the Java Collections API, fundamental I/O, exceptions, and exception handling.

The course is designed to leverage the participants' existing programming skills and to highlight the new and extended features of the Java programming framework as compared to other common languages. Comprehensive hands on exercises are integrated throughout to reinforce learning and develop real competency.

Course Overview

Introduction to Java

- Cornerstones of the Java Platform
- Java Advantages
- The Java Programming Language
- The Java Virtual Machine (JVM)
- Core Java Libraries
- Extension Libraries

Developing Software Using Java

- Applications, Applets, Web Components
- Java SE, Java EE, Java ME
- Installing the JDK
- Compiling and Running Java from the Command Line
- The main() Method
- package and import Statements
- JAR Files
- Class Loading and CLASSPATH
- Online API Documentation
- JDK Tools
- Java Integrated Development Environments (IDEs)

Java Syntax Fundamentals

- Comments
- Identifiers
- Reserved Words

Data Types and Operators

- Primitive Types
- Boolean, Integer, Floating-Point and Character Types

- Classes
- Statements and Blocks
- Variables, Constants, Literals
- Scope of Variables
- Methods
- Method Overloading
- static Members
- Static Import (Java SE 5+)
- Naming Conventions

Flow of Control

- if/else Statement
- Combining ifs
- while and do/while Loops
- for Loop and Loop Counters
- break and continue
- Break to Labeled Loops
- switch Statement
- return Statement
- Exit Status

Strings

- String Manipulation
- StringBuffer and StringBuilder
- Simple Number/String Conversion

Developing Java Classes

- Object-Oriented (OO) Concepts
- Methods, Member Variables
- Accessing Members
- Tight Encapsulation
- Access Control Modifiers
- Constructors and Finalizer

- Unicode Characters and Strings
- Type Conversion and Casting
- Expressions and Operators
- Arithmetic Operators
- Increment/Decrement Operators
- Division and Remainder Operators
- Assignment Operators
- Relational Comparison and Logical Operators
- Conditional Operator
- Bitwise Operators
- Order of Evaluation
- Operator Precedence and Associativity

Using Java Classes and Objects

- Classes as Data Types
- Objects and References
- Memory in the JVM
- Object Initialization
- Objects as Arguments to Methods
- Objects as Return Values
- Garbage Collection
- Primitive Wrapper Classes - Integer, Double, etc.
- Autoboxing and Unboxing (Java SE 5+)

Arrays

- Declaring and Allocating Arrays
- Multi-Dimensional Array
- Array Literals
- The java.util.Arrays Class
- Command-Line Arguments
- Enhanced for Loop (Java SE 5+)
- Arrays as Method Arguments
- Variable-Length Arglists (varargs) (Java SE 5+)
- Autoboxing and varargs

Inheritance

- Extending Java Classes
- Accessing Superclass Constructors and Members
- Overriding Methods
- Abstract CLasses and Methods
- Polymorphism
- Overriding Methods of java.lang.Object

- Using this
- Class Variables - Static Members and Static Blocks
- Instance Variables
- Local Variables
- Variables and Initialization
- Inner Classes
- Anonymous Classes
- JavaBeans
- Driver Classes
- equals(), toString(), hashCode()
- Final Classes and Methods
- Multiple Inheritance
- Interfaces
- Casting Object References
- Documenting Classes with the javadoc Utility
- Unit Testing

Type Safety

- Annotations (Java SE 5+)
- Java SE Built-In Annotations
- Defining New Annotations
- Enumerated Types (Java SE 5+)
- Constants and Constrained Values
- Defining and Declaring enums
- enum Values
- enums and switch Statements
- values() and valueOf()
- Generic Classes (Java SE 5+)
- Generic Type Parameters
- Using Type Parameters inClass, Variable and Method Declarations
- Using a Generic Class
- Bounded Type Parameters

The Collections Framework

- The java.util Package
- Container Objects
- Arrays as Containers
- Legacy Container Classes - Vector, Hashtable, Enumeration
- Legacy Container Generic Forms (Java SE 5+)
- Collections Interfaces - Collection<E>, List<E>, Set<E>, SortedSet<E>
- Map Interfaces - Map<K,V>
- Coding to the Interface
- List<E>, Set<E>, Queue<E> and Map<K,V> implementations
- Iterating Collections with the Iterator<E> Interface
- Collections and the Enhanced for Loop
- Choosing the Correct Implementation and Interface
- The java.util.Collections Utility Class
- Sorting Using the Comparable Interface

Exceptions and Exception Handling

- The Throwable Hierarchy: Error, RuntimeException and Checked Exception
- Methods that Throw Exceptions
- Handling Exceptions with try-catch-finally Blocks
- Application-Defined Exceptions
- Throwing an Exception
- Assertions (Java 1.4+)
- Enabling Assertions at Run-Time

Basic Input and Output (I/O)

- The java.io Package
- Using Stream Classes
- Combining Streams
- flush() and close()
- Console Input and Output
- Navigating the File System
- File Streams
- Character File Input and Output
- Reader and Writer Interfaces
- BufferedReader and BufferedWriter

- Binary File I/O - DataOutputStream and DataInputStream
- Object Streams - ObjectInputStream and ObjectOutputStream
- Serialization and Versioning
- Random Access Files
- Formatted Input and Output
- Formatter (Java SE 5+)
- Format specifiers, printf() and format()
- java.text Classes for Formatting Dates, Numbers, Currencies
- Input with Scanner (Java SE 5+)

Network Programming

- The java.net Package
- IP Addresses and Port Numbers
- Client/Server Socket Programming
- URL and URLConnection Classes
- Communicating with Web Servers
- HTTP GET and POST Operations
- Posting to a Server-Side Program

Threads

- Life and States of a Thread
- Creating and Starting a Thread
- java.lang.Runnable and java.lang.Thread
- Stopping a Thread
- Inter-Thread Communication
- Thread-Safe Access to Shared Objects and Variables
- Synchronized Code
- Sleeping
- Interrupting a Blocked Thread
- wait(), notify(), notifyAll() Example
- Thread Scheduling
- Thread Groups
- Writing a Multithreaded Server

Java Database Connectivity

- The java.sql Package
- JDBC Architecture and Drivers
- SQL Exceptions
- DriverManager, Connection, Statement and ResultSet Interfaces
- Examining Database MetaData
- Basic Query and Update
- Improving Performance with PreparedStatement and CallableStatement Interfaces
- JDBC Transaction Management

Java Web Applications

- Java Enterprise Edition
- Java EE Application Servers
- Web Application Directory and WAR files
- Deploying a Web Application - The web.xml File
- Servlet Architecture
- The javax.servlet Package
- Servlet Classes and Interfaces
- Writing a Servlet
- HttpServletRequest and HttpServletResponse
- Handling HTML Forms
- Retrieving Request Parameters

JavaServer Pages (JSPs)

- JSP Lifecycle
- Elements of a JSP
- Directives, Declarative, Scriptlets
- Writing a JSP
- Objects Available in a JSP
- Repeated content in JSPs
- Translation-Time and Request-Time Includes
- Using JavaBeans in a JSP
- Session Management
- Mixing JSPs and Servlets
- Installing and Using Tag Libraries
- The JSP taglib Directive
- The Tag Library Descriptor

JavaScript Programming

Duration: 90 hours

Prerequisites: Basic computer skills and knowledge of HTML fundamentals including use of Cascading Style Sheets. The **Website Development with HTML5, CSS and Bootstrap** provides this foundational knowledge, if needed.

Description: JavaScript is a scripting language that is commonly used to create and control dynamic Website content along with its use in the Node.js runtime. This hands on JavaScript training course provides the fundamental knowledge necessary to design and develop dynamic Web pages using JavaScript and to be able to grasp JavaScript libraries and frameworks such as jQuery, React and Angular. Students will learn the syntax of the JavaScript language and how to use JavaScript in a Web browser.

Topics include ways to declare variables, use of intrinsic JavaScript objects such as Math, Date and Array, declaring and calling functions, defining custom objects, error handling, working with arrow functions, and using conditional logic. Students will learn how to include JavaScript in a Web page and how to use browser-based APIs such as the Document Object Model (DOM), Geolocation and Web Storage. Upon completion students will be able to utilize event handling, form validation, JSON and Ajax.

Comprehensive hands on exercises are integrated throughout to reinforce learning and develop real competency.

Course Overview

Overview of the JavaScript Language

- Typical Uses of JavaScript
- Evolution of the JavaScript Language
- ECMAScript Version Support
- What is Core JavaScript?
- What are JavaScript APIs?
- Writing JavaScript in Visual Studio Code

JavaScript Fundamentals

- Using Strict Mode
- JavaScript and Whitespace
- Adding Comments
- Declaring Variables
- var, let and const Keywords
- Local vs. Global Variables
- Declaring Block-Scoped Variables
- JavaScript DataTypes
  - Primitive Types
  - Objects
- Performing Data Conversions
- Working with JavaScript Operators
  - Assignment Operators
  - Type Operators
  - Bitwise Operators
  - Conditional Operator

- Working with Dates

Working with Numeric Data

- Solving Precision Problems with Numbers
- Rounding Numbers
- Creating Random Numbers
- Arithmetic Operation
- Working with Exponents
- Converting to Numbers
    - parseInt, parseFloat and Number Functions

Dealing with Errors

- Types of JavaScript Errors
- Error Handling
    - try
    - catch
    - finally
- Throwing Custom Errors with throw

Controlling Flow

- Creating Boolean Expressions
    - Using Standard and Strict Equality Operators
    - Using Logical and Comparison Operators
- Conditional Constructs
    - if else
    - switch
- Looping Constructs
    - do while
    - while
    - for
    - for in
    - for...of
- Branching Statements
    - break
    - continue
    - Using Labels

Working with Arrays

- Creating Arrays
    - Using the Array Constructor
    - Using Literal Notation
    - String.split()
    - Array.of()
- Accessing Array Elements
- Common Array Methods
    - forEach()
    - includes()
    - find() and findIndex()
- Adding and Removing Elements
- Sorting Arrays
    - Creating a Custom Sort Function
- Looping through an Array

Working with Strings

- Creating String Objects
- Using String Methods
- Searching within a String
- Returning a Substring
    - Converting Character Case
- Template Literals
- Escape Sequences

Implementing Functions

- Declaring Functions
- Function Parameters
    - Default Parameters
    - Rest Parameters
- Invoking Functions
    - Passing Arguments
    - Spread Operator
- Named and Anonymous Functions
- Local vs. Global Variables
- Returning Values from a Function
- The this Keyword

- IIFEs
- Closures
- Callback Functions
- Arrow Functions

Using JavaScript in the Browser

- Embedding JavaScript in a Page
- Referencing an External Script File
- Using the JavaScript Console
    - Logging Messages in the Console
    - Debugging with Console
- Deferring Script Loading
- Where to Include JavaScript
- DOMContentLoaded vs Load Events

Browser Object Model

- Overview of the Browser Object Model
- Using the window Object
    - Opening and Closing Windows
    - Using Timers
    - Interacting with the User with Dialogs
- Using the document Object
    - Writing to the Web Page
    - Useful Properties of the Document
- navigator Object
- Feature detection
- location Object
- screen Object

Learning to Program with C#

Duration: 90 hours

Prerequisites: Familiarity with computers.

Description: This hands on C# programming course provides an introduction to programming using the C# language. Students are introduced to the application development cycle, structure of programs, and specific language syntax. The course also contains "Thinking Like a Programmer" sections that provide students insight on how to develop common algorithms. The course covers console and file I/O, string and character manipulation, managing data using collections and fundamental object-oriented programming concepts. Error handling techniques are also emphasized. The course also introduces how to access databases using ADO.NET and illustrates how to build user interfaces using Windows Forms. Comprehensive hands on exercises are integrated throughout to reinforce learning and develop real competency.

This class is intended for non-programmers.

Course Overview

Application Development Fundamentals

- Overview of Programming Concepts
- Understanding the Structure of a C# Program
- Understanding Data Types
- Working with Variables
- Reading From and Writing to the Console
- Overview of the .NET Framework
- Using Visual Studio
- Thinking Like a Programmer
    - Designing Algorithms
    - Finding Patterns in Code

Working with Data

- Using Integer Data Types
- Using Floating Point Data Types
- Using Characters and Strings
- Using Dates
- Using Booleans
- Working with Constants and Literals

Managing the Flow of an Application

- Conditional Constructs
    - Working with if/else Constructs
    - Working with switch
- Looping Constructs
    - Working with while Constructs
    - Working with for and foreach
- Thinking Like a Programmer
    - Making Decisions
    - Designing Loops

Using Procedures to Modularize Code

- Defining and Calling Subroutines
- Defining and Calling Functions
- Understanding Variable Scope
- Overloading Procedures
- Passing Parameters
- Understanding by value vs by reference Parameters
- Understanding the Call Stack

- Thinking Like a Programmer
  - Finding the Procedures
  - Refactoring
  - Improving Productivity with Snippets

Object-Oriented Programming

- Understanding Object-Oriented Concepts
- Designing Classes
- Coding Properties and Methods
- Initializing Objects with Constructors
- Overloading Constructors
- Declaring and Instantiating Objects
- Calling Properties and Methods
- Understanding Value Types vs. Reference Types
- Working with Shared Data Members and Methods
- Thinking Like a Programmer
  - Finding your Classes

Understanding Namespaces

- Understanding the Role of Namespaces
- Understanding .NET Namespaces
- Defining Custom Namespaces
- Referencing Members in a Namespace
- Using the using Statement

Working with Data Collections

- Understanding Arrays
- Declaring and Instantiating Arrays
- Iterating through Arrays
- Working with System.Array Methods
  - Copying, Sorting, Searching and Resizing
- Passing Arrays to Methods
- Working with param array Parameters
- Working with Command-line Arguments
- Understanding .NET Collections
- Managing Data Using List
- Managing Data Using Dictionary<TKey, TValue>
- Working with LINQ
- Making LINQ Queries
- Enumerating LINQ Query Results
- Working with Anonymous Types
- Using Extension Methods with LINQ

Building Inheritance Hierarchies

- Understanding Inheritance
- Building Derived Classes
- Understanding Constructors in Derived Classes
- Defining and Using Protected Class Members
- Understanding Polymorphism
- Defining Overridable Methods
- Overriding Methods
- Understanding Abstract Classes
- Defining Abstract Classes
- Inheriting from Abstract Classes

Building GUIs with Windows Forms

- Overview of Windows Forms
- Designing Forms
- Working with Controls
  - Using Labels and Textboxes
  - Using Buttons

Accessing Databases Using ADO.NET

- Understanding the ADO.NET Object Model
- Opening Connections
- Executing Queries Using Commands
- Iterating Through Results Using DataReaders
- Calling Stored Procedures

    - Using Checkboxes and Radio Buttons
    - Using Menus
    - Using List Controls
- Handling Events

- Passing Parameters to Stored Procedures
- Working with DataSets
- Binding Data to Controls

Exception Handling

- Understanding Exception Handling
- Using try/catch to Handle Exceptions
- Working with the Exception Class
- Understanding Exception Propagation
- Using finally to Manage Cleanup Processing
- Throwing Exceptions

Working with Files, Directories and Streams

- Using the System.IO Namespace
- Discovering Drives
- Working with Directories
- Working with Files
- Parsing a File Path
- Understanding Streams
- Working with FileStream
- Reading and Writing Text Files
- Understanding other Types of Streams

Working with Strings

- Working with the String Class
- Working with String Literals and Escape Sequences
- Understanding String Manipulation Performance Issues
- Working with the StringBuilder Class
- Formatting Output with String.Format

Building N-Tier Applications

- Building Large Scale Applications
- Designing N-Tier Applications
- Building .NET Assemblies
- Referencing Assemblies

Learning to Program with Java

Duration: 90 hours

Prerequisites: Familiarity with computers.

Description: This hands on Java Programming course provides an introduction to programming using the Java language. Students are introduced to the application development cycle, structure of programs, and specific language syntax. The course introduces important algorithmic constructs, string and character manipulation, dynamic memory allocation, standard I/O, and fundamental object-oriented programming concepts. The course explains the use of inheritance and polymorphism early on so the students can practice extensively in the hands on labs. Structured programming techniques and error handling are emphasized. The course includes the processing of command line arguments and environment variables so students will be able to write flexible, user-friendly programs. Comprehensive hands on exercises are integrated throughout to reinforce learning and develop real competency.

Course Overview

Fundamentals of the Program Development Cycle

- Computer Architecture
- The Notion of Algorithms
- Source Code vs. Machine Code
- Compile-Time vs. Run-Time
- Software Program Architecture
  - Standalone
  - Client/Server
  - Distributed
  - Web-Enabled
- IDE (Interactive Development Environment) Concepts

Application Development Fundamentals

- Structure of a Java Program
- Memory Concepts
- Fundamental Data Type Declarations
- Fundamental I/O Concepts
- Fundamental Operators
  - Arithmetic Operators
  - Logical Operators
  - Precedence and Associativity
- Building and Deploying a Java Program

Introduction to Classes and Objects

- Classes, Objects and Methods
- Object Instances
- Declaring and Instantiating a Java Object
- Declaring Methods
- set and get Methods
- Initiating Objects with Constructors
- Primitive Types vs. Reference Types

Flow Control

- Conditional Constructs
- Looping Constructs
  - Counter-Controlled Repetition
  - Sentinel-Controlled Repetition
- Nested Control Constructs
- break and continue Statements
- Structured Programming Best Practices

Writing Methods (Functions)

Arrays

- Static vs. Dynamic Allocation
- Declaring Methods
- Declaring Methods with Multiple Parameters
- Method-Call Stack
- Scope of Declarations
- Argument Promotion and Casting
- Designing Methods for Reusability
- Method Overloading

Deeper Into Classes and Objects

- Controlling Access to Class Members
- Referencing the Current Object Using this
- Overloading Constructors
- Default and No-Argument Constructors
- Composition of Classes
- Garbage Collection and Destructors
- The finalize Method
- Static Class Members

Increasing Convenience by Using Polymorphism

- Purpose of Polymorphic Behavior
- The Concept of a Signature
- Abstract Classes and Methods
- final Methods and Classes
- Purpose of Interfaces
- Using and Creating Interfaces
- Common Interfaces of the Java API

Fundamental Searching and Sorting

- Introduction to Searching Algorithms
  - Linear Search
  - Binary Search
- Introduction to Sorting Algorithms
  - Selection Sort
  - Insertion Sort
  - Merge Sort

Exception Handling

- Types of Exceptions
- Exception Handling Overview
- Exception Class Hierarchy

- Purpose of Arrays
- Declaring and Instantiating Arrays
- Passing Arrays to Methods
- Multidimensional Arrays
- Variable-Length Argument Lists
- Using Command-Line Arguments
- Using Environment Variables

Defining Classes Using Inheritance

- Superclasses and Subclasses
- Advantages of Using Inheritance
- protected Class Members
- Constructors in Subclasses

Files and Streams

- Concept of a Stream
- Class File
- Sequential Access
- Object Serialization to/from Sequential Access Files
- Additional java.io Classes

Fundamental Data Structures

- Dynamic Memory Allocation
- Linked Lists
- Stacks
- Queues
- Trees

Formatted Output

- printf Syntax
- Conversion Characters
- Specifying Field Width and Precision

- Extending Exception Classes
- When to Throw or Assert Exceptions

Strings, Characters and Regular Expressions

- Fundamentals of Characters and Strings
- String Class
- String Operations
- StringBuilder Class
- Character Class
- StringTokenizer Class
- Regular Expressions
    - Regular Expression Syntax
    - Pattern Class
    - Matcher Class

- Using Flags to Alter Appearance
- Printing Literals and Escape Sequences
- Formatting Output with Class Formatter

Fundamental GUI Programming Concepts

- Overview of Swing Components
- Displaying Text and Graphics in a Window
- Event Handling with Nested Classes
- GUI Event Types and Listener Interfaces
- Mouse Event Handling
- Layout Managers

Microsoft SQL Server 2017 Administration

Duration: 90 hours

Prerequisites: Familiarity with database concepts, Windows desktop navigation and Transact-SQL. Attendance at the **SQL Programming** course or **Microsoft Transact-SQL Programming** course is highly recommended although not required.

Description: This hands-on course provides students with the knowledge and skills to administer a SQL Server 2017 database infrastructure. Students will learn how to install, configure, manage, secure, automate, monitor, and optimize SQL Server 2017. Students will also learn how to create, manage, back up, and restore individual databases, transfer and replicate data, configure for high availability, and plan disaster recovery.

Comprehensive hands on exercises are integrated throughout to reinforce learning and develop real competency.

Course Overview

Installation and Configuration

- Installing Multiple Instances of SQL Server 2017
- Applying a Service Pack
- Creating Aliases
- Setting up a Central Management Server

Authenticating and Authorizing Users

- Authenticating Connections to SQL Server
- Authorizing Logins to Access Databases
- Authorization Across Servers
- Partially Contained Databases

Assigning Server and Database Roles

- Working with Server Roles
- Working with Fixed Database Roles
- Creating User-Defined Database Roles

Authorizing Users to Access Resources

- Authorizing User Access to Objects
- Authorizing Users to Execute Code
- Configuring Permissions at the Schema Level

Protecting Data with Encryption and Auditing

- Options for Auditing Data Access in SQL Server
- Implementing SQL Server Audit
- Managing SQL Server Audit
- Protecting Data with Encryption

SQL Server Recovery Models

- Backup Strategies
- Understanding SQL Server Transaction Logging
- Planning a SQL Server Backup Strategy

Backup of SQL Server Databases

- Backing up Databases and Transaction Logs
- Managing Database Backups

Restoring SQL Server Databases

- Understanding the Restore Process
- Restoring Databases

- Working with Backup Options
- Working with Point-in-time Recovery
- Restoring System Databases and Individual Files

Tracing Access to SQL Server

- Capturing Activity Using SQL Server profiler
- Improving Performance with the Database Engine Tuning Advisor
- Working with Tracing Options

Monitoring SQL Server

- Monitoring Activity
- Capturing and Managing Performance Data
- Analyzing Collected Performance Data

Troubleshooting SQL Server

- Resolving Service Related Issues
- Resolving Login and Connectivity Issues
- Troubleshooting Common Issues

Importing and Exporting Data

- Transferring Data to/from SQL Server
- Importing and Exporting Table Data
- Using BCP and BULK INSERT to Import Data

Maintaining High Availability of Data

- Windows Clustering
- AlwaysOn Availability Groups
- Implementing AlwaysOn
- Implementing Log Shipping

Microsoft Transact-SQL Programming

Duration: 90 hours

Prerequisites: General computer knowledge.

Description: This Transact-SQL programming course teaches students relational database fundamentals and SQL programming skills in the Microsoft SQL Server environment. Topics covered include relational database architecture, database design techniques, and simple and complex query skills. The course also covers Microsoft-specific T-SQL programming constructs, creation and use of stored procedures and user-defined functions, use of cursors and updateable views.

This class is intended for analysts, developers, designers, administrators, and managers new to the SQL programming language. Upon completion, participants will understand SQL functions, join techniques, database objects and constraints, and will be able to write useful stored procedures and views as well as complex queries and updates. Comprehensive hands on exercises are integrated throughout to reinforce learning and develop real competency.

Course Overview

Relational Database Fundamentals

- Overview of Relational Database Concepts
- Relational Databases and Relational Database Management Systems
- Data Normalization
- DDL Syntax

Writing Basic SQL Queries

- Displaying Table Structures
- Retrieving Column Data From a Table or View
- Selecting Unique Values
- Filtering Rows Using the WHERE Clause
- Sorting Results Using ORDER BY
- Joining Multiple Tables
- Using Column and Table Aliases

Creating a Database

- Database Development Methodology Overview
- Building a Logical Data Model
  - Identifying Entities and Attributes
  - Isolating Keys
  - Relationships Between Entities
  - Creating Entity-Relationship Diagrams
- Transforming to Physical Design
  - Migrating Entities to Tables
  - Selecting Primary Keys
  - Defining Columns
  - Enforcing Relationships with Foreign Keys

Manipulating Query Results

- Using Row Functions
  - Character
  - Numeric
  - Date and Time
  - Data Conversion (CAST and CONVERT)
- Using the CASE Function
- Handling Null Values

- Constructing the Database Using DDL
  - Creating Tables, Indexes, Constraints and Views
  - Dropping Tables, Indexes, Constraints and Views
  - Modifying Tables, Indexes, Constraints and Views

Advanced Query Techniques

- Inner Joins
- Outer Joins (Left, Right, Full)
- Performing Self-Joins
- Subqueries
  - Simple
  - Correlated
- Using the EXISTS Operator
- Tips for Developing Complex SQL Queries
- Using Aggregate Functions
  - AVG
  - COUNT
  - SUM
  - MIN
  - MAX
- Performing Set Operations
  - UNION
  - INTERSECT
  - EXCEPT/MINUS
- Aggregating Results Using GROUP BY
- Restricting Groups with the HAVING Clause
- Creating Temporary Tables

Manipulating Table Data Using SQL's Data Manipulation Language (DML)

- Inserting Data into Tables
- Updating Existing Data
- Deleting Records
- Truncating Tables
- Performing Bulk Inserts
- Using the OUTPUT Clause
- Merging Data
- Working with Identity Columns and Sequences

User-Defined Functions

- Definition and Benefits of Use
- CREATE FUNCTION
  - Syntax
  - RETURN Clause and the RETURNS Statement
  - Scalar vs. Table Functions
- Comparison with Stored Procedures
- Returning Scalar Values and Tables
- ALTER and DROP FUNCTION

Stored Procedures

- Definition and Benefits of Use
- CREATE PROCEDURE
  - Syntax
  - Defining Input Parameters
  - Defining Output Parameters
  - Defining Optional Parameters
- ALTER and DROP PROCEDURE
- Implementation Differences

Triggers

Complex Queries

- Definition and Benefits of Use
- Alternatives (e.g., Constraints)
- CREATE TRIGGER
  - Syntax
  - Trigger Types
- "Inserted" (or "NEW") and "Deleted" (or "OLD") Tables
- Event Handling and Trigger Execution
- ALTER and DROP TRIGGER

T-SQL Code Constructs

- Exploiting the Programming Features of T-SQL
  - Conditional Constructs
  - Looping Constructs
- Building Multi-Batch Scripts
- Invoking System Functions
- Using Variables in Scripts
- Using Temporary Tables in Scripts
- Handling Errors
  - Using TRY...CATCH Blocks
  - Using System Variables and Functions

Using T-SQL Cursors

- Overview of Cursors
- Declaring a Cursor
- Using OPEN CURSOR, CLOSE CURSOR, DEALLOCATE CURSOR Statements
- FETCHing Results
- Testing @@FETCH_STATUS and @@CURSOR_ROWS
- Updating Records with Cursors

- Using Wildcard Characters with Like
- Allowing Users to Build SQL Queries Dynamically
- Pivoting Data
- Summarizing Data with ROLLUP and CUBE
- Using Partitioned Aggregates

Working with Data Types and Functions

- Effective Use of Data Types in SQL
  - String
  - Numeric
  - Time/Date
  - Other
- Substitution of Non-null Values with the COALESCE and ISNULL Functions
- Analyzing Data Points Using Ranking Functions

Working with Table Expressions

- Overview of Table Expressions
- Working with Views
- Using Derived Tables
- Common Table Expressions
- Table-Valued Functions

Object Oriented Analysis & Design with UML

Duration: 75 hours

Prerequisites: Knowledge of structured programming concepts.

Description: This OOA&D training course presents the key concepts and methodologies required to perform quality object-oriented software engineering, with particular attention to practical techniques such as use-case and CRC analysis, UML diagramming, and patterns. Students practice applying object oriented analysis during the course to improve software designs and to see how software objects can be altered to build software systems that are more robust and less expensive. Students use several methods for analyzing software systems, finding and refining useful classes and relationships between objects. Care is taken not to focus on any one language so that all students can participate in the design exercises without relying on specific programming skills. The course emphasizes the most practical analysis and design methods, including the application of use case analysis, CRC analysis, problem domain analysis, activity diagramming, interaction diagramming, and class diagramming.

The Unified Modeling Language (UML) is presented in detail and is used in the exercises and case studies. Practical aspects of project management and implementation are presented from the perspective of experienced object system designers. Special emphasis is given to the use of object patterns in developing software systems. The students apply their skills in labs that are mini design sessions, during which the instructor helps the students identify and overcome common obstacles that occur during group sessions.

Course Overview

The Object Paradigm

- Objects and Classes
- Abstraction and Encapsulation
- Methods and Messages
- Interfaces, Inheritance, and Polymorphism
- Access Control
- The Business Case for OO Development

Managing and Participating in the OOA&D Approach

- Information Gathering Techniques
- Group Orientated Problem Solving
- Brainstorming, Role-Playing
- Managing Complexity via the "Iterative and Incremental" Approach
- Managing Design Sessions
- Design vs. Implementation
- Quick Prototyping
- Validation and Quality

Diagramming & Notational Techniques Using the UML

- Overview of Analysis and Design Phases
- UML Notation
- Analysis Diagramming Techniques
- Design Diagramming Techniques

Requirements and Analysis Phase

- System Functions, Features and Constraints
- Behavioral Analysis
- Domain Analysis
- Identifying Use Cases

- Generalization/Specialization
- Aggregation and Composition
- Association, Cardinality, Navigability
- Package and Deployment Diagrams
- Icons, Relationships, and Adornments

- Use Case Descriptions
- Using CRC Cards
- Containment and Composition
- Referential Aggregation
- Inheritance, SubTypes and Is-A Hierarchies
- Association and Link Relationships
- Diagramming System Events
- State Transition Diagramming

Design Phase

- Translating Analysis Concepts into Software Classes
- Optimizing Classes and Objects: The Multi-Tiered Architecture View
- Mapping System Functions to Objects
- Object to Object Visibility
- Collaboration Diagrams
- Sequence Diagrams
- Specifying Object Interfaces
- Specification Class Diagrams

Patterns

- Benefits of Patterns
- Using Patterns During Analysis
- Using Patterns During Design
- Design Patterns (Gang-of-Four Format)
- GRASP Patterns
- Model-View-Controller Pattern
- Persistence Patterns
- Patterns as Internal Documentation

Design Refinement

- Designing for Extensibility
- Designing for Reusability
- Partitioning the Class Space
- Checking Completeness and Correctness
- Testing Business Processes
- Design Metrics
- Discovering Reusable Patterns

Project Management and Implementation Issues

- Planning for Reusability
- Transition Strategies and Planning Legacy System Integration
- Managing the Development Cycle
- Partitioning Work
- Source Code Organization
- Choosing Tools and Languages
- Software Quality Metrics

OO Languages and Tools

- Survey of OO Languages
- The Role of Class Libraries
- The Role of OOA&D Tools

Advanced Design Concepts

- Expanding Inheritance Hierarchies
- Abstract Classes and Virtual Methods
- Overriding and Overloading
- Multiple Inheritance
- Interface versus Implementation Inheritance

Persistent Object and Database Issues

- The Coad Data Management Domain
- Object Persistence

- Object-Orientated Database Management Systems (ODBMS)
- Object Orientated versus Relational Databases
- Mapping Objects to Relational Data Structures

Python Programming

Duration: 75 hours

Prerequisites: Prior scripting experience or knowledge of fundamental programming concepts.

Description: This hands on Python programming course shows how to rapidly develop and maintain effective Python programs. The course includes thorough coverage of Python syntax, built in data types and control constructs. The course takes a practical approach to creating and organizing Python programs using functions, packages, modules and classes as part of Python's object-oriented paradigm. Attendees will use regular expressions to rapidly process data captured from users and from the file system.

Attendees will learn how to use Python to create scripts that manipulate data, automate tasks, perform error handling and store and retrieve data by using relational databases. Students will be able to create Python scripts that assist with system administration.

Comprehensive hands on exercises are integrated throughout to reinforce learning and develop real competency.

Course Overview

Introduction to Python 3

- Origin and Goals of Python
- Overview of Python Features
- Getting and Installing Python
- Accessing Python Documentation: Python Enhancement Proposals (PEP)
- Python's Strengths
- Using Python with Other Programming Languages

Using Python

- Executing Python Programs from the Command Line
- Python Command Line Options
- Environment Variables that Influence Python
- Creating Python GUI Applications
  - Standalone vs. Web-Enabled Interfaces
- The Python Standard Library

Language Fundamentals

- Python's Lexical Analyzer
- Using Whitespace to Structure Programs
- Identifiers and Keywords
- Python's Execution Model
  - Naming Objects and Binding
- Python's Data Model
  - Immutable and Mutable Objects
  - Values
  - Types
- Creating and Using Variables

Flow Control Constructs

- if/elif/else Statements
- Creating Loops with while and for
- Understanding Iterators
- Returning Values with return Statements
- Loop Modification with break and continue
- Returning Generator Iterators with the yield Statement
- Retrieving Iterators with next()

Expressions

- Unary and Binary Arithmetic Operations
- Comparison and Boolean Operations
- Conditional Expressions
- Lambda Expressions
- Order of Operations and Operator Evaluation
- Expression Lists
- Assignment Operations

Using the String Object

- Using ASCII and Unicode Strings
- Manipulating Strings with String Methods
- Using the format() Function to Format Strings
- Using Escape Sequences
- Working with Raw Strings

Arrays, Collections and Dictionaries

- Sequenced Data Structures
  - Arrays
  - Collections
  - Dictionaries
- Creating and Accessing Lists
- Manipulating Lists
- Creating and Accessing Tuples
- Understanding the Differences Between Lists and Tuples
- Using Dictionaries to Create Data Records
- Manipulating Dictionaries Using Dictionary Methods
- Creating Sets
- Performing Set Operations
  - Union
  - Intersect
  - Difference

Exception Handling

- Types of Python Exceptions
- Handling Exceptions with try/except/finally
- Triggering Exceptions with raise
- Defining New Exception Types
- Implementing Exception Handling in Functions, Methods and Classes
- Working with the Regular Expression Error Exception

Organizing Code

- Defining Functions
- Calling Functions
- Creating Anonymous Functions
- Altering Function Functionality with Decorator Functions
- Creating Classes with the class Statement
- Creating Objects as Class Instances
- Using Preexisting Classes as the Basis of a New Class
- Using Modules to Group Related Functions, Classes and Variables
- Locating and Importing Modules
- Using Packages to Group Modules Together

Working with Arguments

- Passing Arguments to Functions by Reference and by Value
- Defining Functions with Required Arguments
- Defining Functions with Default Arguments
- Defining Flexible Functions that Take Variable Length Arguments

Regular Expressions

- Regular Expression Syntax
- Using Regular Expressions in Python
- Altering Regular Expression Processing with Regular Expression Modifiers
- Using Regular Expression Operators
- Scanning Through Strings Using the search() and match() Methods

- Differences Between Sets and Dictionaries
- Using Generators to Return Iterators

Object Oriented Programming Concepts

- The Object Oriented Programming Paradigm
- Encapsulating Information
- Classes vs. Instances of Objects
- Built-in Class Attributes
- Implementing Class Inheritance
- Using Objects in Code

Data Management

- Embedding SQLite Databases in Applications
- Best Practices for Data Management
- Storing Data in Local Databases
- Discussing and Understanding the DB API
- Understanding and Using Common SQL Statements
- Connecting to a SQLite Database
- Using Cursors to interact with Data from a Database
- Implementing Error Handling with Database Connections
- Creating Reusable Patterns by Using the compile() Method

I/O Handling

- Sending Output to STDOUT Using the print() Method
- Reading Input with the input() Method
- Creating File Objects with the open() Method
- Controlling File Access Modes
- Working with File Object Attributes
- Closing File Objects with the close() Method
- Reading and Writing to File Objects with read() and write()
- Using File Processing Functions from the OS Module

SQL Programming

Duration: 60 hours

Prerequisites: General computer knowledge.

Description: This SQL programming course teaches students relational database fundamentals and SQL programming skills. Topics covered include relational database architecture, database design techniques, and simple and complex query skills. This class is intended for analysts, developers, designers, administrators, and managers who need to harness the power of the SQL programming language. Upon completion, participants will understand SQL functions, join techniques, database objects and constraints, and will be able to write advanced SELECT, INSERT, UPDATE and DELETE statements. Comprehensive hands on exercises are integrated throughout to reinforce learning and develop real competency.

Course Overview

Relational Database Fundamentals

- Overview of Relational Database Concepts
- Relational Databases and Relational Database Management Systems
- Data Normalization
- DDL Syntax

Writing Basic SQL Queries

- Displaying Table Structures
- Retrieving Column Data From a Table or View
- Selecting Unique Values
- Filtering Rows Using the WHERE Clause
- Sorting Results Using ORDER BY
- Joining Multiple Tables
- Using Column and Table Aliases

Creating a Database

- Database Development Methodology Overview
- Building a Logical Data Model
  - Identifying Entities and Attributes
  - Isolating Keys
  - Relationships Between Entities
  - Creating Entity-Relationship Diagrams
- Transforming to Physical Design
  - Migrating Entities to Tables
  - Selecting Primary Keys
  - Defining Columns
  - Enforcing Relationships with Foreign Keys
- Constructing the Database Using DDL
  - Creating Tables, Indexes, Constraints and Views
  - Dropping Tables, Indexes, Constraints and Views

Manipulating Query Results

- Using Row Functions
  - Character
  - Numeric
  - Date and Time
  - Data Conversion (CAST and CONVERT)
- Using the CASE Function
- Handling Null Values

- Modifying Tables, Indexes, Constraints and Views

Advanced Query Techniques

- Inner Joins
- Outer Joins (Left, Right, Full)
- Performing Self-Joins
- Subqueries
  - Simple
  - Correlated
- Using the EXISTS Operator
- Tips for Developing Complex SQL Queries
- Using Aggregate Functions
  - AVG
  - COUNT
  - SUM
  - MIN
  - MAX
- Performing Set Operations
  - UNION
  - INTERSECT
  - EXCEPT/MINUS
- Aggregating Results Using GROUP BY
- Restricting Groups with the HAVING Clause
- Creating Temporary Tables

Manipulating Table Data Using SQL's Data Manipulation Language (DML)

- Inserting Data into Tables
- Updating Existing Data
- Deleting Records
- Truncating Tables
- Implementing Data Integrity with Transactions
  - Beginning Explicit Transactions
  - Committing Transactions
  - Rolling Back Transactions

User-Defined Functions

- Definition and Benefits of Use
- CREATE FUNCTION
  - Syntax
  - RETURN Clause and the RETURNS Statement
  - Scalar vs. Table Functions
- Comparison with Stored Procedures
- Returning Scalar Values and Tables
- ALTER and DROP FUNCTION

Stored Procedures

- Definition and Benefits of Use
- CREATE PROCEDURE
  - Syntax
  - Variables and Parameters
- Control of Program Flow
- ALTER and DROP PROCEDURE
- Implementation Differences

Triggers

- Definition and Benefits of Use
- Alternatives (e.g., Constraints)
- CREATE TRIGGER
  - Syntax

Working with Table Expressions

- Overview of Table Expressions
- Working with Views
- Using Derived Tables
- Common Table Expressions

    - Trigger Types
- "Inserted" (or "NEW") and "Deleted" (or "OLD") Tables
- Event Handling and Trigger Execution
- ALTER and DROP TRIGGER
- Table-Valued Functions

SQL Server 2019 Business Intelligence: Integration Services and Analysis Services

Duration: 90 hours

Prerequisites: Familiarity with database concepts, Windows desktop navigation and software installation techniques. Prior attendance at the **SQL Programming** course or **Microsoft Transact-SQL Programming** course is highly recommended although not required.

Description: SQL Server 2019 provides a rich environment for business intelligence development. The focus of this course is to familiarize developers with the SQL Server Engine and the use of SQL Server Integration Services (SSIS) and SQL Server Analysis Services (SSAS) for Business Intelligence purposes. Students will create and populate data warehouses through ETL processing, and will build Multidimensional and Tabular models which will be used as data sources for reports.

Students will learn how to design and build data warehouses and marts using SQL Server Management Studio. In a series of exercises, students develop SSIS packages designed to maintain a data warehouse using the Data Flow control flow task. Also demonstrated are other control flow tasks that can interact with an NTFS file system, FTP server, execute Win32 processes, send emails, and run .NET scripts.

Based on the populated data warehouse they have created, students will then learn how to develop both Multidimensional and Tabular SSAS models using the languages Multidimensional Expressions (MDX) and Data Analysis Expressions (DAX). Cubes will be customized to include Key Performance Indicators (KPIs), Calculated Members, Named Sets, Navigational Hierarchies, and Perspectives.

Course Overview

Business Intelligence Framework Overview

- SQL Server Data Tools Overview
- Installation and Configuration
- Components of a BI Solution
- Introduction to the BI Semantic Model

Integration Services Architecture

- Architecture of the SSIS Data Engine
- Using Data Transformation Tasks
- Managing Connections to Sources and Destinations
- ADO.NET Data Source and Destination
- Understanding Data Buffers
- Control Flow Tasks and Containers

Common SSIS Tasks

- Executing SQL Statements
- Connecting to FTP Servers
- Sending E-mail

Data Transformations

- Converting Data Types
- Merging Data from Multiple Sources
- Splitting Data to Multiple Destinations

- Notifying Administrators of Errors
- Completing Bulk Inserts
- Copying, Moving and Deleting Files and Folders
- Counting Rows
- Sampling and Sorting Records
- Copying Columns

Advanced SSIS Tasks

- Executing .NET Scripts and Win32 Processes
- Using the Windows Management Instrumentation (WMI) Tasks
- Performing Database Maintenance and Backups During SSIS Routines
- Using Variables and Input Parameters
- Profiling Database Tables
- Comparing XML Files Against Schemas

Advanced Data Transformations

- Filling in Missing Data with Lookups
- Locating Near Duplicate Rows with Fuzzy Grouping
- Adding Audit Information to Results
- Counting the Occurrence of Keywords
- Sending Rows that Process Correctly and Incorrectly to Different Destinations
- Responding to Truncation Errors

SSIS Administration and Automation

- Deploying SSIS Projects
- Manually Running SSIS Tasks
- Automating SSIS Package Execution
- Configuring Notifications for Execution Success, Failure or Both
- SSIS Security
- Troubleshooting Techniques

Data Warehouse Design

- Understanding Fact and Dimension Tables
- Modeling Slowly Changing and Rapidly Changing Dimensions
- Modeling Fact Tables
- Using Star and Snowflake Schemas for Dimension Tables
- Implementing Surrogate Keys
- Defining Business Keys

Creating and Populating Data Warehouses

- Creating Data Warehouses (OLAP Databases)
- Adding Fact Tables
- Adding Dimension Tables and Joining Them to Fact Tables
- Loading Data into Fact and Dimension Tables
- Validation Techniques for Data Loads

Creating and Managing Cubes

- Creating Data Sources to Connect to Data Warehouses
- Using SSAS to Create Cubes
- Applying Friendly Names to Measures and Attributes
- Customizing Dimensions and Measures
- Setting up Navigational Hierarchies
- Optimizing Cubes with Attribute Relationships

Multidimensional (MDX) Essentials

- Using MDX Queries to Pull Data from Cubes
- Understanding Tuples and Sets
- MDX Expressions vs. Queries
- Grouping Attribute Values into Named Sets
- Adding Custom Calculations for Cubes Using MDX

MDX Functions

- Using MDX Aggregate Functions
- Using Navigations Functions to Move Though Hierarchies
- Grouping, Filtering and Sorting Functions
- Time-Based MDX Functions

Customizing Cubes

Cube Deployment and Administration

- Adding Key Performance Indicators (KPIs)
- Customizing Dimensions and Attributes
- Adding Translations to Support Multiple Languages
- Adding Custom Calculations
- Subdividing Cubes Using Perspectives
- Cube Storage Calculations
- Configuring Desired Aggregation
- Configuring Caching
- Deploying and Processing Cubes
- Connecting to Cubes from Excel and Other Clients
- Partitioning and Processing Cubes
- Backing Up and Restoring Options
- Securing Cubes

Creating and Customizing Tabular Models

- Creating Tabular Modules in SSDT
- Introducing DAX
- Customizing Tabular Models
- Refreshing Data in Tabular Models

Understanding the Data Mining Process

- Types of Business Analysis Supported by Data Mining
- Data Mining Process Explained
- Understanding the Key Components of Data Mining
- Using Built-In Data Mining Algorithms
- Matching Mining Algorithms to Business Needs

Working with Data Mining Structures

- Adding Data Mining Structures
- Mining for Hidden Information
- Discovering Patterns in Data
- Creating Predictive Models
- Using the Data Mining Wizard
- Modifying Mining Structures with the Data Mining Designer

Using the Semantic Models in the Presentation Layer

- Using SSAS Data Sources in Excel and Power View
- Using SSAS Data Sources in SSRS
- Using SSAS Data Sources in Power BI
- Using SSAS Data Sources in SharePoint Performance Point Services

SQL Server 2019 Reporting Services

Duration: 60 hours

Prerequisites: Familiarity with database concepts, Windows desktop navigation and software installation techniques. Prior attendance of the **SQL Programming** course or **Microsoft Transact-SQL Programming** course is highly recommended although not required.

Description: This course introduces Microsoft's SQL Server Reporting Services 2019 (SSRS) utility. All reports are developed utilizing SQL Server Data Tools (SSDT), however Report Designer will also be introduced. Students will complete hands-on exercises creating a number of reports including table-based, cross tabular and forms based designs. Reports utilizing charts, gauges, KPIs, data bars, spark lines and tree maps will be built. Exercises will incorporate the use of report parameters and features such as drill down, interactive sorting, hyperlinks, book marks and report maps.

Data from relational, multi-dimensional and tabular data sources will be incorporated into report designs. Students will become familiar with report deployment to the new Reporting Service Web Portal and learn how to use the portal the management tools to configure report caches, snapshots and subscriptions. Instruction and practice in designing report security to ensure protection of enterprise data will be provided.

The student will gain experience with the new Report Services Web portal, which enables publishing content other than paginated reports, such as Excel and Power BI visualizations. Students will also learn how to include KPIs (Key Performance Indicators) directly from shared data sets.

Course Overview

Reporting Services Architecture

- Defining Report Server Components
- Reviewing the Report Server Databases
- Managing the Report Server Web Portal
- Creating Mobile Reports and Dashboards
- Report Lifecycle Step by Step

Creating Basic Reports

- Creating Table Reports
- Formatting Options
- Adding Headers and Footers
- Using Subtotals and Formulas
- Adding Charts and Gauges
- Using Lists for Free-Form Reports
- Paging and Sorting
- Managing Data Sources
- Adding Key Performance Indicators (KPIs)

Advanced Report Features

- Setting Up Input Parameters
- Adding Custom Report Navigation
- Adding Drill-Down for Master-Detail Reports
- Linking Reports

Matrix Reports

- Pivoting Data with Matrix Controls
- Adding Multiple Data Regions
- Grouping and Sorting
- Adding Subtotals

- Using Sub-Reports
- Adding Conditional Formatting

Implementing BI Semantic Model

- Understanding Multi-Dimensional Models
- Using Tabular Models
- Creating MDX Data Sources and Data Sets

Reporting Services Security

- Organizing Reports in Folders
- Setting Permissions on Reports and Folders
- Managing Security Roles
- Securing Data Sources and Report Models
- Logging Activity on the Report Server

Adding Custom Code

- Querying the Report Server
- Creating Custom Data Extensions
- Creating Command Objects
- Creating Data Reader Objects
- Generating Report Definition Language
- Adding Custom Aggregate Formulas
- Using the Tablix object

Report Designer

- Creating Reports with Report Designer
- Adding Charts and Gauges in Report Designer
- Differences Between SQL Server Data Tools and Report Builder

Automating Report Creation and Delivery

- Creating Snapshot Reports
- Storing Previous Versions of Reports
- Caching Reports to Improve Performance
- Defining Subscriptions to Automate Report Delivery
- Rending Reports in Excel, MS Word or as a PDF
- Hiding Reports from Users

.NET Integration with Reporting Services

- Writing User-Defined Functions in .NET
- Using Custom Functions
- Referencing .NET Libraries
- Managing Report Servers from .NET Applications
    - Accessing SSRS from .NET Applications
    - Retrieving Report Items
    - Listing Report Metadata

Test Driven Development (TDD), and Refactoring Legacy Code Using C#

Duration: 75 hours

Prerequisites: C# programming experience and an understanding of object-oriented design principles

Description: This course provides students with hands on experience learning Test Driven Development (TDD) using NUnit and Microsoft's Visual Studio. Students will build unit tests using mocks, fakes, stubs and drivers, and address issues working with databases and other systems. Students will create tests and code that will be more likely to meet and exceed requirements. Code that receives "test coverage" will not break existing systems, because tests are passed before code is checked in.

Students will spend time working with the issues involved in refactoring legacy code, safely cutting into an already deployed system. Students will work on looking for, or creating "seams" to more safely improve code or add features, and work on identifying "code smells" that need attention in a productive system.

Finally, students will explore dependency issues as well as techniques to better understand and improve complex systems.

Comprehensive C# labs throughout the course provide facilitated hands on practice crucial to developing competence and confidence with the new skills being learned.

Course Overview

Why TDD? Think Twice, Write Production Code Once

- Utilizing a Safety Net of Automated Testing
- Agile Development Concepts
- Eliminating Bugs Early
- Smoothing Out Production Rollouts
- Writing Code Faster via Testing
- Reducing Technical Debt
- Practicing Emergent Design
- Making Changes More Safe
- The Importance of Regression Testing

Basic Unit Testing

- NUnit and Visual Studio
- Adding Complexity to Initial Simple Tests
- Making Tests Easy to Run
- The TDD Pattern: Red, Green Refactor
- Using Methods of the Assert Class
- Boundary Testing
- Unit Test Limitations

Comprehensive Unit Testing Concepts

- Using Declarative-Style Attributes
- Using Hamcrest Matchers for More Complex Scenarios
- Using Test Categories

Mocks, Fakes, Stubs and Drivers

- TDD Development Patterns
- Naming Conventions for Better Code
- Using Mock Objects
- Using Fakes

- Exception Handling in Tests
- NUnit Test Initialization and Clean Up Methods
- Writing Clean and Dirty Tests
- Testing with Collections, Generics and Arrays
- Negative Testing

Database Unit Testing

- Mocking the Data Layer
- Identifying what Should Be Tested in Databases
- Stored Procedure Tests
- Schema Testing
- Using NDbUnit to Set Up the DB Test Environment

Patterns and Anti-Patterns in TDD

- The SOLID Principles
- Factory Methods
- Coding to Interface References
- Checking Parameters for Validity Test
- Open/Closed Principle: Open to Extension, Closed to Change
- Breaking Out Method/Object
- Extract and Override Call
- Extract and Override Factory Method
- Singleton Pattern
- Decorator Pattern
- Facade Pattern
- State Pattern
- MVP, MVC and MVVM Patterns
- Finding and Removing Code Smells/Antipatterns

- Using Stubs
- Test Doubles
- Manual Mocking
- Mocking with a Mock Framework
- Self-Shunt Pattern

Refactoring Basics

- Refactoring Existing Code
- Restructuring
- Extracting Methods
- Removing Duplication
- Reducing Coupling
- Division of Responsibilities
- Improving Clarity and Maintainability
- Test First - then Refactor
- More Complex Refactoring Considerations

Refactoring Legacy Code

- Reducing Risk of Change
  - Eliminating Dependencies
  - Characterization Tests as a Safety Net
  - Introducing Abstractions to Break Dependencies
- Analyzing Legacy Code
  - Identifying Pinch Points with Effect Analysis
  - Identifying Seams for Expansion and Testing
  - Listing Markup
- Minimizing Risk of Adding New Behavior
  - Sprout Method
  - Sprout Class
  - Wrap Method
  - Wrap Class
- Dealing with Code that's Difficult to Test
  - Globals and Singletons in Tests
  - Inaccessible Methods and Fields
- Using Smells to Identify What to Refactor
  - Dealing with Monster Methods
  - Dealing with Excessively Complex, Large Classes
  - Identifying and Eliminating Duplication
  - Other Smells
- Dealing with Large Legacy Systems

- Preserving Signatures

Code Coverage

- White Box vs Black Box Testing
- Planning to Increase Code Coverage Over Time
  - Goal 80% or More Test Coverage
  - Statement Coverage
  - Condition Coverage
  - Path Coverage

Risks Changing Legacy/Production Systems

- Refactoring
- Coupling and Cohesion Issues
- Taking Small Tested Steps

Test Driven Development (TDD), and Refactoring Legacy Code Using Java

Duration: 75 hours

Prerequisites: Java programming experience and an understanding of object-oriented design principles.

Description: This course provides students with hands on experience learning Test Driven Development (TDD) using Java and JUnit. Students will build unit tests using mocks, fakes, stubs and drivers, and address issues working with databases and other systems. Students will create tests and code that will be more likely to meet and exceed requirements. Code that receives "test coverage" will not break existing systems, because tests are passed before code is checked in.

Students will spend time working with the issues involved in refactoring legacy code, safely cutting into an already deployed system. Students will work on looking for, or creating "seams" to more safely improve code or add features, and work on identifying "code smells" that need attention in a productive system.

Finally, students will explore dependency issues as well as techniques to better understand and improve complex systems.

Comprehensive Java labs provide facilitated hands on practice during the course, allowing students to develop competence and confidence with the new skills being learned.

Course Overview

Why TDD? Think Twice, Write Production Code Once

- Utilizing a Safety Net of Automated Testing
- Agile Development Concepts
- Eliminating Bugs Early
- Smoothing Out Production Rollouts
- Writing Code Faster via Testing
- Reducing Technical Debt
- Practicing Emergent Design
- Making Changes More Safe
- The Importance of Regression Testing

Basic Unit Testing

- JUnit
- JUnit Testing
- Adding Complexity to Initial Simple Tests
- Making Tests Easy to Run
- The TDD Pattern: Red, Green Refactor
- Using Methods of the Assert Class
- Boundary Testing
- Unit Test Limitations

Comprehensive Unit Testing Concepts

- Using Declarative-Style Attributes
- Using Hamcrest Matchers for More Complex Scenarios
- Using Test Categories

Mocks, Fakes, Stubs and Drivers

- TDD Development Patterns
- Naming Conventions for Better Code
- Using Mock Objects
- Using Fakes

- Exception Handling in Tests
- JUnit Test Initialization and Clean Up Methods
- Writing Clean and Dirty Tests
- Testing with Collections, Generics and Arrays
- Negative Testing

- Using Stubs
- Test Doubles
- Manual Mocking
- Mocking with a Mock Framework
- Self-Shunt Pattern

Database Unit Testing

- Mocking the Data Layer
- Identifying what Should Be Tested in Databases
- Stored Procedure Tests
- Schema Testing
- Using NDbUnit to Set Up the DB Test Environment

Refactoring Basics

- Refactoring Existing Code
- Restructuring
- Extracting Methods
- Removing Duplication
- Reducing Coupling
- Division of Responsibilities
- Improving Clarity and Maintainability
- Test First - then Refactor
- More Complex Refactoring Considerations

Patterns and Anti-Patterns in TDD

- The SOLID Principles
- Factory Methods
- Coding to Interface References
- Checking Parameters for Validity Test
- Open/Closed Principle: Open to Extension, Closed to Change
- Breaking Out Method/Object
- Extract and Override Call
- Extract and Override Factory Method
- Singleton Pattern
- Decorator Pattern
- Facade Pattern
- State Pattern
- MVP, MVC and MVVM Patterns
- Finding and Removing Code Smells/Antipatterns

Refactoring Legacy Code

- Reducing Risk of Change
  - Eliminating Dependencies
  - Characterization Tests as a Safety Net
  - Introducing Abstractions to Break Dependencies
- Analyzing Legacy Code
  - Identifying Pinch Points with Effect Analysis
  - Identifying Seams for Expansion and Testing
  - Listing Markup
- Minimizing Risk of Adding New Behavior
  - Sprout Method
  - Sprout Class
  - Wrap Method
  - Wrap Class
- Dealing with Code that's Difficult to Test
  - Globals and Singletons in Tests
  - Inaccessible Methods and Fields
- Using Smells to Identify What to Refactor
  - Dealing with Monster Methods
  - Dealing with Excessively Complex, Large Classes
  - Identifying and Eliminating Duplication
  - Other Smells
- Dealing with Large Legacy Systems

- Preserving Signatures

Code Coverage

- White Box vs Black Box Testing
- Planning to Increase Code Coverage Over Time
  - Goal 80% or More Test Coverage
  - Statement Coverage
  - Condition Coverage
  - Path Coverage

Risks Changing Legacy/Production Systems

- Refactoring
- Coupling and Cohesion Issues
- Taking Small Tested Steps

Website Development with HTML5, CSS and Bootstrap

Duration: 90 hours

Prerequisites: Basic personal computer skills and basic Internet knowledge.

Description: This hands on course provides a thorough introduction into the creation of a Website using HTML, CSS and Bootstrap. The course starts with thorough coverage of HTML and Cascading Style Sheets (CSS) and progresses to using the Bootstrap framework to create mobile-friendly websites.

Topics include use of HTML5 semantic tags, block-level and inline elements, creating links, ordered and unordered lists, creation of tables and forms. Students will learn to attach CSS to a page using several different techniques, CSS selectors and pseudo-classes, CSS box model, and a variety of CSS properties.

Students will learn how to create a Bootstrap page utilizing the grid system, implement commonly used components (such as dropdowns and navigation bars), use Bootstraps CSS classes to format page elements and use Bootstrap's plugins to add tabs, modals and accordions to a page.

Comprehensive hands on exercises are integrated throughout to reinforce learning and develop real competency.

Course Overview

HTML Fundamentals

- HTTP Requests and Responses
- Static vs. Dynamic Pages
- Tag and Attribute Fundamentals
- Document Structure
  - Understanding <DOCTYPE>
  - Creating a Page Title
  - Indicating the Character Set
  - <head> and <body> Tags
- Browser Support
  - HTML5 Shiv
  - CSS Reset
  - Conditional Comments
- Using Data Attributes

Formatting Text

- Using Inline Tags
  - <b>, <i>, ,

Structuring an HTML Document

- Adding Headings
- Adding Paragraphs
- Working with the Tag
- Adding Quotes to a Page
- HTML5 Semantic Tags
  - <header>, <footer>, <nav>, <article>, <section>
- Adding Meters and Progress Bars
- Defining Hyperlinks
- Creating Anchors within a Page
- Creating an E-mail Link

Introduction to Cascading Style Sheets (CSS)

- Overview of Cascading Style Sheets (CSS)
- Creating CSS Rules

- Using the and <mark> Tags
- Defining Inline Styles
  - Changing Text Color
  - Changing Font Characteristics
  - Changing Horizontal Alignment
- Using Character Entities
- Setting CSS Properties
  - Working with Colors
  - Units of Measurement (px, em, %)
  - Working with Fonts
- Defining Type Selectors
- Defining Class and ID Selectors
- Using Attribute Selectors
- Grouping and Combining Selectors
- Linking to External Style Sheets
- Embedded Style Sheets
- Overriding Styles with !important

CSS Properties

- Text-Related Properties
  - text-decoration, color, line-height
- Font-Related Properties
  - font-family, font-size, font-weight
- Background-Related Properties
  - background-color, background-image, background-position
- Using @font-face to Add Custom Fonts
- Changing the Opacity of Text and Images
- Creating Gradients
  - Linear Gradients
  - Repeating Linear Gradients
  - Radial Gradients
  - Repeating Radial Gradients
- Creating Rounded Corners, Adding Shadows to Boxes and Using Images as Borders

CSS Box Model

- Positioning Elements
  - Static, Absolute and Relative
  - Controlling z-index
  - Floating and Clearing Elements
- Visual Effects
  - Setting visibility and display
  - Using overflow and clip Properties
  - Setting Vertical Alignment
- Box-Related Properties
  - margin, padding, border
- Setting the Size of an Element

CSS Selectors and Pseudo-Classes

- Relational Selectors
  - Descendant
  - Direct-Child
  - General Sibling
  - Adjacent Sibling
- Pseudo-Classes and Elements
  - Styling Links with :link, :visited, :active, :hover
  - Adding Content with :before and :after
- Attribute Selectors
  - "Starts With" Selector
  - "Ends With" Selector

CSS Transforms, Transitions and Animations

- Using Transforms
  - Rotating 2D Elements
  - Scaling 2D Elements
  - Translating 2D Elements
  - Skewing 2D Elements
- Configuring a Transition
  - Delaying the Start of a Transition
  - Changing the Speed of a Transition
- Creating an Animated Menu
- Animating Buttons with Transitions
- Using Animations
  - Controlling the Direction of an Animation
  - Controlling the State of an Animation

        - "Contains" Selector
    - Selecting the nth Element
    - Selecting the nth of a Type

Working with Lists

- Types of Lists
    - Ordered Lists
    - Unordered Lists
    - Description Lists
- Creating List Items Using the <li> Tag
- Creating Nested Lists
- Using CSS to Style a List
- Using Lists to Create a Navigation Menu
- Using CSS Counters
    - Incrementing a Counter
    - Creating Nested Counters

Displaying Information in Tables

- Adding Tables to a Page
- Working with <table>, <tr>, <td>, <th> and <caption> Elements
- Creating Nested Tables
- Grouping Table Content
- Using CSS to Style a Table
    - Controlling Table Borders
    - border-collapse and border-spacing Properties
- Using rowspan and colspan Attributes

Form Validation

- Form Validation Overview
    - Client-Side Validation
    - Server-Side Validation
- Overview of Form Security
- Validating Input Length Using minlength and maxlength Attributes
- Using the pattern Attribute to Specify Regular Expressions
        - Changing the Speed of an Animation
    - Using the animation Property

Adding Images to a Page

- Discussion of Common Image Formats
- Displaying Images on a Web Page
    - Using src and alt Attributes
    - Sizing an Image Using width and height Attributes
- Common Uses of Images
    - Image Thumbnails
    - Favicons
    - Image Rollovers
- Working with Client-Side Image Maps
- Using Icon Fonts

Working with Forms

- Defining a Form
- Using Common Form Attributes
    - action, method, enctype, autocomplete, novalidate
- Input Elements
    - email, url, tel, search
- Understanding Form Submission
- Adding <label> and <fieldset> Elements
- Single-line and Multi-line Text Fields
- Radio Buttons and Checkboxes
- Dropdown and Selection Lists
- Submit, Reset and Push Buttons
- Using Form-Related Pseudo Classes and Elements
- Controlling Form Layout Using CSS and Tables

Using CSS for Page Layout

- Using CSS for Page Layout
    - 2 Column Fixed Layout
    - 3 Column Fixed Layout
    - 2 Column Fluid Layout
- Using CSS Grid Layout
    - Defining Grid Columns
    - Defining Grid Row
    - Positioning Rows and Columns
    - Using Grid Areas

- Validation-Related Pseudo-Classes

- Adding Spacing Between Rows and Columns
- Aligning the Grid Container
- Aligning Grid Items
- Defining a Nested Grid

Audio and Video

- Browser Support for Audio/Video Formats
  - Supplying Fallbacks
- Creating Audio Effects with HTML5 <audio> Elements and Attributes
- Adding Video with HTML5 <video> Elements and Attributes
- <source> Elements

Bootstrap Overview

- Overview of Mobile First Design
- CSS3 Media Queries
- Downloading Bootstrap
- Customizing Bootstrap
- Adding Custom CSS to Bootstrap

Bootstrap Grid System

- Working with Grids
- Defining Rows, Containers and Columns
- Offsetting Columns
- Creating Nested Columns
- Changing Column Order

Bootstrap Base CSS

- Overview of Bootstrap's Base CSS
- Using Helper Classes and Responsive Utilities
- Working with Typography Classes
- Formatting Lists
- Formatting Tables
  - Creating a Striped Table
  - Creating a Responsive Table
- Enhancing the Appearance of Forms
  - Displaying Inline Forms
  - Displaying Horizontal Forms
- Formatting Images
  - Creating Responsive Images
- Styling Links and Buttons

Bootstrap Components

- Overview of Bootstrap Components
- Using Glyphicons to Add Symbols to Buttons and Links
- Creating Dropdowns
- Creating Navigation
  - Tabbed Navigation
  - Pill Navigation
  - Navigation Bars
  - Collapsible Navigation Bars
  - Breadcrumbs
- Grouping Inputs, Buttons and Lists
- Formatting Page Headers

Bootstrap Plugins

- Plugin Overview
- Creating Tabbed Navigation
- Creating a Slideshow Using the Carousel Plugin
- Using Accordions to Display a Large Amount of Content
- Adding a Dialog Box Using the Modal Plugin
- Displaying Secondary Information Using Popovers or Tooltips

- Creating Panels
    - Grouping Panels
    - Creating a Thumbnail Gallery
- Creating Progress Bars and Alerts

Windows Forms Programming Using C#

Duration: 90 hours

Prerequisites: Prior programming experience is required.

Description: This course provides students with hands on experience using Visual Studio to create desktop applications using Windows Forms and the .NET Framework using C#. The course provides a thorough introduction to the C# programming language, including coverage of the essentials of the C# programming language, built in data types, operators, control structures, classes and methods, collections and exception handling.

Students then learn how to leverage the power of the .NET Framework to build desktop applications. Students learn how to build Windows Forms applications and use with a variety of controls to create sophisticated user interfaces. Students also learn how to use the BackgroundWorker to perform asynchronous operations.

Students also learn how to use ADO.NET to interact with databases and XML files. Students learn how Windows Forms uses data binding to display data in controls such as the DataGridView and Chart. Students also learn how to build and interact with simple WCF SOAP Web Services.

Other topics include: debugging techniques; using a .config file to control application configuration; building menus, toolbars and status bars; reading and writing files; interacting with the file system; and deploying desktop applications.

Comprehensive labs provide the students with extensive experience creating and deploying Windows Forms-based desktop applications.

Course Overview

Introduction to .NET

- Overview of the .NET Framework
- How .NET is Different from Traditional Programming
- Common Language Runtime (CLR)
- Common Language Specification (CLS)
- Common Type System (CTS)
- .NET Assemblies
- Microsoft Intermediate Language (CIL)
- .NET Namespaces
- .NET Framework Class Library

Introduction to Visual Studio

- Creating a Project
- Using the Code Editor
- Correcting Syntax Errors
- Setting Project Properties
- Adding References
- Compiling a Program
- Running a Program
- Debugging a Program
- Using the MSDN (Help)

Language Fundamentals

- C# Program Structure
- Defining Namespaces
- Understanding C# Data Types
- Defining Variables and Constants
- Comparing Value Types vs. Reference Types
- Working with Operators and Expressions
- Performing Type Conversions
- Using Console I/O
- Formatting Numbers, Date and Times

Conditionals and Looping

- if/else
- switch
- while and do/while
- for
- foreach

Methods and Parameters

- Defining Static and Instance Methods
- Passing Parameters by value and by reference
- Overloading Methods
- Using Variable Length Parameter Lists

Exception Handling

- What are Exceptions?
- .NET Exception Hierarchy
- Catching Exceptions
- Throwing Exceptions
- Managing Resources with Finally

Collections

- Defining and Using Arrays
- Understanding System.Array
- .NET Collections vs Generic Collections
- Working with Lists
- Working with Dictionaries
- Using LINQ to Objects

Object-Oriented Programming

- Overview of Object-Oriented Programming
- Building Classes
- Defining Properties
- Using Auto-Implemented Properties
- Defining Methods
- Understanding Constructors
- Extending .NET Classes via Inheritance
- Defining and Implementing Interfaces
- Understanding the Role of Interfaces in .NET

Windows Forms Applications

- Windows Forms Applications
- Setting Form Properties
- Understanding the Life-cycle of a Form
- Using the Windows Forms Designer
- Using the MessageBox Class
- Using a .config File

Using Controls

- Working with Windows Forms Controls
- Using Text Controls
- Using Button Controls
- Using Selection Controls
- Using List Controls
- Using Container Controls
- Using Image Controls
- Using Up/Down Controls
- Using the ErrorProvider and ToolTipProvider Controls

Handling Events

- Understanding the Event-Driven Programming Model
- Writing Event Handlers
- Sharing Event Handlers

Performing Asynchronous Activities

- Understanding Threading
- Working with Delegates
- Using the Background Worker
- Updating Controls from Other Threads

ADO.NET

- Understanding the ADO.NET Object Model
- Connected vs. Disconnected Access
- Using a Connection to Connect to a Data Source
- Using a Command to Execute Queries and Stored Procedures
- Using a DataReader to Work with Cursors
- Using the DataSet with Disconnected Data
- Using DataAdapters with DataSets

Using XML

- Understanding XML and XML Schemas
- Reading XML Data with a DataSet
- Writing XML Data with a DataSet

Data Binding

- Understanding ADO.NET Data Binding
- Binding to Simple and Complex Controls
- Manually Binding Controls
- Using the BindingSource Control
- Using the BindingNavigator Control
- Using the DataGridView Control
- Using the Chart Control

Working with Forms

- Understanding Modal vs Modeless Forms
- Displaying Modal Forms
- Working with DialogResult
- Retrieving Data from Modal Forms
- Displaying Modeless Forms
- Working with Data in Modeless Forms
- Using the Common Dialogs

Working with Menus, Toolbars and Status Bars

- Working with Menus
- Working with Toolbars
- Working with Status Bars

Working with Files and Directories

- Working with the Windows File System
- Discovering Drives
- Discovering Directories
- Discovering Files
- Examining File, Directory and Drive Attributes
- Reading, Writing and Appending to Files

Introduction to WCF Web Services

- Overview of WCF Services
- Understanding SOAP
- Creating a WCF SOAP Web Service
- Creating a Proxy Class
- Calling a WCF SOAP Web Service

Deploying Windows Forms Applications

- Understanding Deployment Options
- Configuring an Application for Deployment
- Using XCOPY Deployment
- Using Installers
- Using Click-Once Deployment

Bright Star Institute is an equal opportunity employer/program.
Auxiliary aids and services are available upon request to individuals with disabilities.