Microsoft® Official Course

20483B

Programming in C#

Microsoft

Hello

Instructor: Joshua C. Warner

Saint Martin's University

JWarner@stmartin.edu

Facilities

- **Class hours**: 2:00-4:30 PM on Wednesdays & Fridays
- **Building**: Meeting on Zoom –

  Meeting ID: 823 2470 8920

  Password: 480835

Syllabus details

- Go on moodle … find there a complete syllabus
  - **http://moodle.stmartin.edu**

- Email: JWarner@stmartin.edu

Lab Programming assignments	30% ← open notes
Weekly Homework	30%
Bi-Weekly Quizzes	40% ← closed notes

Quizzes will mostly take place on Wednesdays.
Check moodle calendar!

Grading Standards:

A-: 90-92	A: 93-96	A+:97-100
B-: 80-82	B: 83-86	B+:87-89
C-: 70-72	C: 73-76	C+:77-79
D-: 60-62	D: 63-66	D+:67-69
F : 0–59		

About This Course

- Audience
  - This training course teaches developers the programming skills that are required for developers to create both Windows client and web applications by using the Visual C# language.
    - Read the book for more details …
- Course Prerequisites
    - Read the book for more details …

About This Course

- ## Course Objectives

  After completing this course, students will be able to:

  - Describe the core syntax and features of Visual C#.
  - Create methods, handle exceptions, and describe the monitoring requirements of large-scale applications.
  - Implement the basic structure and essential elements of a typical desktop application.
  - Create classes, define and implement interfaces, and create and use generic collections.
  - Use inheritance to create a class hierarchy and to extend a .NET Framework class.
  - Read & write data by using file input/output and streams, serialize and deserialize data in different formats.
  - Create and use an entity data model for accessing a database and use LINQ to query data.
  - Access and query remote data by using the types in the System.Net namespace and WCF Data Services.
  - Build a graphical user interface by using XAML.
  - Improve the throughput and response time of applications by using tasks and asynchronous operations.
  - Integrate unmanaged libraries and dynamic components into a Visual C# application.
  - Examine the metadata of types by using reflection, create and use custom attributes, generate code at runtime, and manage assembly versions.
  - Encrypt and decrypt data by using symmetric and asymmetric encryption.

  At the end of the course, students should leave the class with a solid knowledge of Visual C# and how to use it to develop .NET Framework applications.

Course Outline

- Module 1: Review of Visual C# Syntax
- Module 2: Creating Methods, Handling Exceptions, and Monitoring Applications
- Module 3: Developing the Code for a Graphical Application
- Module 4: Creating Classes and Implementing Type-Safe Collections
- Module 5: Creating a Class Hierarchy by Using Inheritance
- Module 6: Reading and Writing Local Data
- Module 7: Accessing a Database
- Module 8: Accessing Remote Data
- Module 9: Designing the User Interface for a Graphical Application
- Module 10: Improving Application Performance and Responsiveness
- Module 11: Integrating with Unmanaged Code
- Module 12: Creating Reusable Types and Assemblies
- Module 13: Encrypting and Decrypting Data

Module 1 – Review of Visual C# Syntax

- Using Visual Studio 2017 to create projects
  - Using the Visual Studio 2017 debugger
- Describe data types, operators, expressions and casting
- Using decision statements
- Declaring and using arrays
- Namespaces
- Lab: Developing the Class Enrollment Application
  - Enhance an existing WPF application for student processing
    - Add students to a class, edit student data, delete students
  - Add functionality to assign students to a class
  - Suggested lab time: 105 minutes

Module 2 - Creating Methods, Handling Exceptions, and Monitoring Applications

- Creating and calling methods
  - Method overloading, optional and output parameters
- Using Exceptions
  - Try, Catch, Finally blocks and throwing an exception
- Monitor an application using tracing, application profiling and performance counters
- Lab: Extending the Class Enrollment Application Functionality
  - Avoid code duplication by refactoring existing code
  - Write code to validate user entered student information
  - Use the Visual Studio to debug the application
  - Add exception handling
  - Analyze the solution for duplicate code using application profiler
  - Suggested lab time: 90 minutes

Module 3 – Developing the Code for a Graphical Application

- Create and use enums, structs
  - Include properties, property indexers within a struct
- Describe collections including ArrayList, Hashtable, Queue
  - Using LINQ to query collections
- Create, raise and subscribe to events
- Lab: Writing the Code for the Grades Prototype Application
  - Start developing parts of a prototype application to test proof of concept and obtain client feedback
    - Extend application to enable teachers to record student grades, allow students to view their grades and enable logon
    - Store student, teacher and grade information in basic structs
    - Use an ArrayList of structs to maintain student, teacher, grade data
    - Create and fire event signaling a logon success or failure
    - Suggested lab time: 90 minutes

Module 4 – Creating Classes and Implementing Type-Safe Collections

- Creating and instantiating a class
  - Discuss constructors, methods, properties, static classes
  - Difference between value, reference types (struct v. class)
- Defining and implementing interfaces
  - Implement IComparable, IComparer to sort ArrayList entries
- Using generic collections List, Dictionary for type safety
- Unit testing of class functionality
- Lab: Adding Data Validation and Type-Safety to the Application
  - Replace structs with classes for Student, Teacher and Grade
  - Create a unit test project to validate new user input validations
  - Use IComparable to add display of students alphabetically
  - Suggested lab time: 75 minutes

Module 5 – Creating a Class Hierarchy by Using Inheritance

- Describe inheritance concepts
  - Base and abstract classes, virtual methods, polymorphism, calling base constructors
- Inheriting from and extending .NET classes
  - Creating custom exceptions
    - Throwing a custom extension
  - Creating extension methods
- Lab: Refactoring Common Functionality into the User Class
  - Refactor student, teacher class common functionality into a base class using inheritance
  - Implement user password complexity using an abstract method and polymorphism
  - Throw a custom exception when attempting to enroll a student in a full class
  - Suggested lab time: 60 minutes

Module 6 – Reading and Writing Local Data

- Using file I/O operations to manipulate files and directories, read, write data to the local file system
  - File, FileInfo, Dir, DirectoryInfo and Path in System.IO
- Data serialization and de-serialization
  - Format: JSON, XML, binary, custom using IFormatter
- Perform I/O with streams
  - Describe the purpose of a stream
  - Using different types of steams to read, write data
    - File, memory and network streams
- Lab: Generating the Grades Report
  - Enable users to preview and then save a student's grades as a XML file on local disk
    - Serialize, de-serialize grade data in XML using memory and file streams
  - Suggested lab time: 60 minutes

Module 7 – Accessing a Database

- Use the Entity Framework to model an existing database
  - Create an entity data model (EDM)
  - Customize generated entity classes using partial classes
- Using LINQ to Entities with SQL-based syntax to query and modify data
  - Deferred execution v. forcing query execution using ToList, ToArray
  - Using anonymous types
- Lab: Retrieving and Modifying Grade Data
  - Upgrade prototype application to use an existing SQL database
    - Create a data model and use LINQ to Entities to query, modify data
    - Extend the data model to include input data validation using partial classes
  - Suggested lab time: 75 minutes

Module 8 – Accessing Remote Data

- Access services over the Web using low-level calls
  - Serialize, deserialize types using DataContract, DataMemeber
  - Use HttpWebRequest, HttpWebResponse to consume a service over HTTP
    - Familiarity with authentication techniques
- Using a WCF Data Service to consume data
  - REST services
  - Exposing and consuming (CRUD) data
    - EDM data model
- Lab: Retrieving and Modifying Grade Data in the Cloud
  - Change the application to access data from the cloud using a WCF Data service rather then the local database
    - Expose and consume student data
    - Expose and consume images (if time permits)
  - Suggested lab time: 60 minutes

Module 9 – Designing the User Interface for a Graphical Application

- Use XAML to create a user interface in a WPF application
  - Controls, setting properties, event handling, user controls
    - Basic coverage of XAML, limited in-depth discussion
- Bind data to XAML controls declaratively or using code
  - Creating and using a data template
- Defining a style as a reusable resource
  - Use animations for page transitions
- Lab: Customizing Student Photographs and Styling the Application
  - Improve the UI appearance and have a consistent look and feel
    - Add styles, animations
  - Create and use a WPF user control to display a student photo
  - Animate the user control (if time permits)
  - Suggested lab time: 90 minutes

Module 10 – Improving Application Performance and Responsiveness

- Using multitasking with the Task class in the Task Parallel Library
  - Start, end, cancel return data from a created task
  - Using lambda expressions for defining anonymous delegates
- Performing asynchronous operations using await, async
  - Handling exceptions
- Synchronizing concurrent data access using locks
- Lab: Improving the Responsiveness and Performance of the Application
  - Use asynchronous methods to keep the UI responsive while listing students for a teacher
    - Use the await and async pattern to run a LINQ query
  - Provide the user with visual feedback during a long-running operation
    - Create a user control (progress indicator) that raises events
  - Suggested lab time: 75 minutes

Module 11 – Integrating with Unmanaged Code

- Using the Dynamic Language Runtime (DLR)
  - Interoperate with unmanaged code and dynamic languages
    - Create and invoke methods on a dynamic object
    - Using dynamic objects with Microsoft Word
- Manage object lifetime and unmanaged resources
  - Dispose pattern and IDisposable
- Lab: Upgrading the Grades Report
  - Upgrade the application to generate reports using Word
    - Replace code that generates reports as an XML file
  - Use a wrapper class for Word with dynamic objects
    - Implement the dispose pattern with IDisposable to ensure Word instance is terminated
  - Suggested lab time: 60 minutes

Module 12 – Creating Reusable Types and Assemblies

- Use reflection to inspect and execute an assembly at runtime
- Create and use custom attributes
- Generate code at runtime using CodeDom
- Describe assemblies
  - Name, version and sign using sn.exe
    - Install an assembly into Global assembly cache (GAC) using gacutil.exe
- Lab: Specifying the Data to Include in the Grades Report
  - Define, use a custom attribute to specify fields to include on a student grade report
  - Sign, version and install a utilities assembly into the GAC (if time permits)
  - Suggested lab time: 75 minutes

Module 13 – Encrypting and Decrypting Data

- Describe symmetric encryption
  - Encrypt, decrypt data using Advanced Encryption Standard (AES) algorithm
- Using hashing to ensure message integrity
- Describe asymmetric encryption
  - Encrypt, decrypt data using RSACryptoServiceProvider
  - Using MakeCert to create X509 certificates
- Lab: Encrypting and Decrypting the Grades Report
  - Ensure that student report content is protected
    - Use asymmetric encryption to encrypt report before saving to disk
      - Create an asymmetric certificate with a batch file that uses the MakeCert tool
      - Create a self-signed certificate using the SHA-1 hash algorithm and store it in the LocalMachine certificate store
  - Decrypt saved student report for printing
  - Suggested lab time: 60 minutes

Tentative schedule

	Class- CSC 440 - Summer 2020								
Tue	7-Jul-21	Syllabus + M1-2			Tue	4-Aug-21	M8 - Accessing Remote Data		
Thur	9-Jul-21	M3-4			Thur	6-Aug-21	M8 - Accessing Remote Data		
Weekend					Weekend				
Tue	14-Jul-21	M5 - Inheritance			Tue	11-Aug-21	M9 - User Interface		
Thur	16-Jul-21	M5 - Inheritance			Thur	13-Aug-21	M10 - Performance		
Weekend					Weekend				
Tue	21-Jul-21	M6 - Local Data			Tue	18-Aug-21	M11 - Unmanaged Code		
Thur	23-Jul-21	M6 - Local Data			Thur	20-Aug-21	M12 - Reuasable types		
Weekend					Weekend				
Tue	28-Jul-21	M7 - Accessing Database			Tue	25-Aug-21	M13 Encryption		
Thur	30-Jul-21	M7 - Accessing Database			Thur	27-Aug-21	Graduation		
Weekend									

Course Takeaways

- Students leave the course with
  - Solid knowledge of C# for developing .NET applications
    - Create classes, define, implement interfaces, and create, use generic collections
    - Use inheritance to create a class hierarchy and extend a class
    - Read, write data using file input/output and streams
    - Serialize, de-serialize data in different formats
    - Create, use an entity data model for database access and use LINQ to Entities to query data
    - Improve application response time using tasks and asynchronous operations
    - Integrate unmanaged libraries and dynamic components
    - Examine type metadata using reflection, create and use custom attributes
    - Use CodeDOM to generate managed code at runtime
    - Encrypt and decrypt data by using symmetric and asymmetric encryption