Course Outline

ADVANCEDTRAINING
Inspire. Grow. Achieve.

6367 - Introduction to Object Oriented Programming using Microsoft Visual Studio 2008

General Description

This three-day instructor led course will enable attendees to start designing and developing object-oriented applications using Visual Studio 2008. Attendees will learn object-oriented concepts including classes, methods, properties, inheritance, and interfaces. Also they will learn how to identify opportunities to use these concepts in design, and how to implement these object-oriented concepts using Visual Studio 2008.

Outcomes

At the end of the course, students will be able to:

- Describe the fundamentals of getting started with object-oriented development and review Visual Studio 2008 features.
- Describe classes and their importance in the basic structure of an object-oriented application.
- Add properties and methods to implement the internal functionality of a class.
- Implement inheritance, abstraction and polymorphism to reduce code duplication.
- Create structures that emphasize code reusability.
- Implement interfaces to establish "common" relationships between classes, reduce code dependencies, and facilitate code.
- Create an object-oriented structure design from a business problem.
- Create object-oriented structures based on their knowledge of classes, properties, methods, inheritance, and interfaces.
- Create and use delegates, events and exceptions to establish interclass communications.
- Design class interactions based on a set of business requirements.
- Design class interactions using methods, events, delegates and exceptions.
- Evaluate techniques to improve their own design.
- Evaluate a design pattern and determine its applicability to a business scenario.
- Create and maintain updatable units of software by deploying components and class libraries.
- Maintain an application without deploying the entire application.

Microsoft Visual Studio 2008

Microsoft Partner
Gold Learning
Silver Desktop

Course Details

Course code: 6367
Duration: 3 days
Starting time: 9am
Finishing time: 4.30pm

Booking guidelines

Contact our learning consultants on 1300 86 87246 and we will assist you with your booking.

☎ 1300 86 87246

1300 TO TRAIN

Course Outline

ADVANCEDTRAINING

Module 1: Getting Started with Object-Oriented Programming

This module provides fundamental knowledge required before getting started with object-oriented development. It also reviews Visual Studio 2008 features.

Lessons

- Introduction to Object-Oriented Programming
- Creating Projects in Visual Studio 2008
- Coding in Visual Studio 2008
- Productivity Features in Visual Studio 2008
- Debugging Visual Studio Applications

Lab: Getting Started with Object-Oriented Development in Visual Studio 2008

- Determining the Types of Projects for Solution
- Creating the Solution and Projects
- Adding Code to the Solution
- Adding Comments to the Solution
- Creating the Solution and Projects

After completing this module, students will be able to:

- Describe what makes an object-oriented program different than a procedural based program.
- List influencing factors that affect successful object oriented development.
- Describe the overall design goals of an object-oriented application.
- Explain how Visual Studio and .NET support developing object-oriented applications.
- Describe the various project types supported by Visual Studio 2008 and when to use them.
- Evaluate when to use Web site and Web application projects.
- Create folders and files for the project types included in the 6367A lab application.
- Describe the primary folders/file types included in each project.
- Describe how .NET establishes and references .NET data types.
- Create data types used within the lab application.
- Manage data types used within the lab application.
- Create basic programming structures and control flows within the Visual Studio 2008 IDE.
- Manage basic programming structures and control flows within the Visual Studio 2008 IDE.
- Describe Visual Studio 2008 productivity enhancements.
- Use editing features such as edit marks, code snippets and refactoring in Visual Studio 2008.
- Create code comments.
- Generate documentation from code comments.
- Troubleshoot coding errors using the debugging interface.
- Test for business logic errors during design time.

Module 2: Implementing Classes, Properties and Methods

This module explains classes and their importance in the basic structure of an object-oriented application. It also add properties and methods to implement the internal functionality of a class.

Lessons

- Creating Classes
- Implementing Properties Within a Class
- Implementing Methods Within a Class
- Using Classes, Properties and Methods

Lab: Implementing Classes with Properties and Methods in Visual Studio 2008

- Creating a Class Structure
- Adding Properties to a Class Structure
- Adding Methods to a Class Structure
- Instantiating and Using a Class within an Application
- Implementing a Shared Method

After completing this module, students will be able to:

- Create a class structure within an object-oriented application.
- Describe the relationship between classes, properties and methods.
- Describe the relationship between classes and objects.
- Describe the instantiation process of an object.
- Create properties within class structures to maintain class specific data within Visual Studio projects.
- Control access to properties.
- Simplify property creation syntax by using default properties.
- Create value type and reference type based properties.
- Define the syntax of a basic method.
- Use value and reference types as parameters and as return types of a method.
- Create overloaded methods.
- Control access to methods.
- Control class construction using constructors.
- Create objects from classes.
- Access the properties of a class instance.
- Invoke the methods of a class instance.
- Invoke a shared method of a class instance.

Module 3: Implementing Inheritance, Abstraction, and Polymorphism

This module explains how to implement inheritance, abstraction and polymorphism to reduce code duplication. It also describes how to create structures that emphasize code reusability.

Lessons

- Introduction to Inheritance and Abstraction
- Implementing Inheritance and Abstraction
- Introduction to Polymorphism
- Implementing a Polymorphic Structure

Lab: Implementing Inheritance and Abstraction

Course Outline

ADVANCEDTRAINING

- Implementing Inheritance Within the Class Structures
- Implementing Abstraction Within the Class Structures
- Implementing Polymorphism Within the Lab Application

After completing this module, students will be able to:

- Explain how to reduce code duplication can be reduced by using inheritance and abstraction.
- Explain how to increase code reuse can be increased by using inheritance and abstraction.
- Identify the impact of inheritance on class structures.
- Write code to create an inherited class.
- Write code to create an abstract class.
- Manage the relationship between a base class and a derived class.
- Manage the construction of inherited class structure.
- Explain the importance of implementing dynamic code.
- Explain how polymorphism can create dynamic code at runtime within inherited class structures.
- Explain the importance of decoupling inherited class functionality from runtime control code.
- Explain how polymorphism can simplify the code required by the runtime portion of the application.
- Identify uses of polymorphism within a business scenario.
- Write code to create a polymorphic structure.
- Write code to use a polymorphic structure.
- Create a polymorphic solution that reduces control code logic.
- Create a polymorphic solution that decouples class dependencies.

➢ Module 4: Implementing Interfaces

This module explains how to implement interfaces to establish common relationships between classes, reduce code dependencies, and facilitate code standardization.

Lessons

- Introduction to Interfaces
- Implementing a Custom Interface

Lab: Implementing Interfaces

- Writing Code to Use a System-Defined Interface
- Defining a Custom Interface
- Implementing a Custom Interface
- Implementing Polymorphism with Interfaces

After completing this module, students will be able to:

- Explain the importance of reducing code dependencies.
- Explain how dependencies between classes can be reduced by implementing interfaces.
- Explain the importance of code standardization.
- Explain how code standardization can be increased through implementing interfaces.
- Identify uses of interfaces within a business scenario.
- Create an interface.
- Describe commonly used system-defined interfaces.
- Write code to implement an interface.
- Use an interface to access the functionality within a class.
- Use an interface with a polymorphic class structure.

➢ Module 5: Designing Object-Oriented Structures

This module explains the process of creating an object-oriented structure design from a business problem. It also describes how to create object-oriented structures based on their knowledge of classes, properties, methods, inheritance, and interfaces. And last, the students will review and refine their designs.

Lessons

- Establishing Classes from Business Requirements
- Adding Inheritance to the Design
- Adding Interfaces to the Design
- Reviewing and Refining the Design

Lab: Designing Object-Oriented Structures

- Creating a Draft Class Diagram from the Business Scenario
- Adding Properties and Methods to the Class Diagram
- Adding Inheritance to the Class Diagram
- Adding Interfaces to the Class Diagram
- Refining the Design

After completing this module, students will be able to:

- Review the goals of object-oriented design.
- Describe approaches to create an object-oriented design based on business requirements.
- Create a class diagram to document the design.
- Identify classes based on business requirements.
- Diagram an initial class structure to fulfill business requirements.
- Identify the properties and methods based on business requirements.
- Diagram properties and methods to fulfill business requirements.
- Describe approaches of designing inherited class structures based on business requirements.
- Identify inherited class structures based on business requirements.
- Add inherited structures to the proposed class diagram.
- Critique the methods and property placement within the new class diagram.
- Modify the methods and property placement within the new class diagram.
- Describe approaches of designing interface structures from business requirements.
- Identify interfaces based on business requirements.
- Modify the class diagram to include interfaces.
- Determine if a proposed design meets the business requirements.
- Determine if a proposed design follows the principals of object-oriented application design.

Course Outline

ADVANCEDTRAINING

- Determine the effect of future usage on the completed class diagram.

➢ Module 6: Delegates, Events, and Exceptions

This module explains how to create and use delegates, events and exceptions to establish interclass communications.

Lessons

- Introduction to Delegates
- Implementing Delegates
- Introduction to Events
- Implementing Events
- Introduction to Exceptions
- Implementing Exceptions

Lab: Implementing Delegates and Events

- Implementing a Delegate
- Implementing a Custom Event
- Implementing an Event Handler for System Events

Lab: Implementing Exceptions

- Implementing Custom Exceptions
- Managing System Exceptions

After completing this module, students will be able to:

- Explain the importance of interclass communications.
- Describe the delegate model.
- Explain how the delegate model can decouple class dependencies and increase code reusability.
- Identify uses of delegates within a business scenario.
- Define the syntax required to implement a delegate within a class.
- Define the syntax required to use a delegate from a calling class.
- Use delegates to establish interclass communications.
- Describe the system event model and its importance to object-oriented applications.
- Describe the benefits of creating custom events within an object-oriented application.
- Identify appropriate uses of system and custom events.
- Apply the syntax required to create a system event handler within a class.
- Apply the syntax required to create a custom event.
- Apply the syntax required to create a custom event handler.
- Describe the exception model used within .NET applications.
- Explain how exception management can help mitigate runtime errors within an application.
- Describe differences in object-oriented exception management versus procedural-based exception management.
- Identify uses of exception management given a business scenario.
- Apply the syntax used to handle a system-generated exception within a class.
- Apply the syntax used to create a custom exception.
- Apply the syntax used to throw a custom exception.
- Implement a solution that uses exceptions to mitigate runtime errors.

➢ Module 7: Designing Object Collaboration

This module explains how to design collaborations between classes by using methods, events, exceptions and delegates. It also introduces sequence diagrams as a way of documenting and planning class interactions.

Lessons

- Introduction to Class Interactions
- Adding Interactions to a Design
- Evaluating the Design
- Introduction to Patterns

Lab: Designing Object-Oriented Interactions

- Design Interactions Using Methods
- Design Interactions Using Events, Delegates, and Exceptions
- Evaluating and Refining the Design
- Evaluating a Pattern

After completing this module, students will be able to:

- Define design goals to create class interactions.
- Identify differences between events, exceptions, delegates and direct method invocation to implement an interaction design.
- Describe evaluation approaches and criteria to determine if a design is effective in solving a business problem.
- Determine interaction boundaries within an object-oriented design to decouple class structures.
- Organize an application to decouple dependencies between code.
- Determine the best communication method between classes given a business requirement.
- Diagram class interactions.
- Determine the overall effectiveness of the design in meeting the business requirements.
- Evaluate the overall design against object-oriented design best practices.
- Critique the design using evaluation techniques.
- Describe design patterns and their importance to object-oriented design.
- Determine the applicability of a design pattern to a business scenario.
- Apply a design pattern to an existing solution.

➢ Module 8: Deploying Components and Class Libraries

This module explains how to create and maintain updatable units of software by deploying components and class libraries. It also describes how to maintain an application without redeploying the entire application.

Lessons

- Introduction to Components and Class Libraries
- Deploying a Component/Class Library

- Best Practices for Deploying a Component/Class Library

Lab: Deploying Components/Class Libraries

- Creating a Component/Class Library
- Deploying the Application
- Updating the Component/Class Library
- Deploying an Updated Component/Class Library

After completing this module, students will be able to:

- Explain the importance of creating deployable units of software.
- Describe the software development lifecycle with respect to upgrading and maintenance.
- Determine uses of components and class libraries given a business scenario.
- Prepare an application to use component/class libraries.
- Create a component/class library.
- Deploy a component/class library.
- Manage versioning within an application using a component/class library.
- Describe best practices for deploying component/class libraries.
- Identify code that maximizes component/class library usage.
- Explain how a component/class library meets best practices.