Software Development II Course Information

Software Development Fundamentals I & II introduces learners to the core components of writing programs. From a basic introduction to using the software development interface to the fundamentals of data structures, data types and variables. The course introduces the core concepts of object-oriented programming and provides an important primer for any future career in the field.

Learning Outcomes for Software Development Fundamentals II

1. Understanding advanced object-oriented programming techniques.
2. Understanding generic types and methods and being able to use them appropriately.
3. Be able to query, update and delete data objects and data structures from a data source.
4. Create and implement delegates in an application.
5. Basic understanding and application of exception handling.
6. Incorporate input, output and file handling techniques while developing applications.
7. Know how to tier applications for code reusability.

Admission Requirements for Software Development II

Minimum admission requirement is a National Senior Certificate (NSC) or Senior Certificate (SC) or a National Certificate Vocational (NCV).

Applicants must also have completed Programming Fundamentals I or successfully passed an admission assessment. A portfolio of an applicant's software development experience could also be used to waiver the admission requirements.

Course Content for Software Development II

1. Object Oriented Programming
    - Constructors and Finalizes
        - Constructors
        - Static Constructors
        - Destructors
    - Operators, Overloading and Conversions
        - Operators
        - Conversions
    - Inheritance
    - Polymorphism and Virtual Members
    - Encapsulation
    - Abstract Classes
    - Interfaces
    - Delegates
        - Functional Programming
        - What are delegates?
        - Delegate Types and Delegate Instances
        - Invoking Delegates
    - Events
        - How events work?
        - Raising Events
        - Add and Remove Accessors
    - Detach you event handlers
2. Generic Types and Methods
    - Generics

- Making the Case for Generics
- Building a Generic Class
- Using a Generic Class
- Defining Generic Methods
- Leveraging Generic Constraints

- Lists
  - Declaring and Populating a Generic List
  - Using Collection Initializers
  - Initializing a List of Objects
  - Retrieving an Element from a Generic List
  - Iterating Through a Generic List
  - Types of C# Lists
- Dictionaries
  - Declaring and Populating a Generic Dictionary
  - Using Collection Initializers
  - Initializing a Dictionary of Objects
  - Retrieving an Element from a Generic Dictionary
  - Iterating Through a Generic Dictionary
  - Types of C# Dictionaries
- Interfaces
  - Making the Case for Using Interfaces
  - Built-in Generic Collection Interfaces
  - Using an Interface as a Parameter
  - Using an Interface as a Return Type
  - Returning IEnumerable T
  - Defining an Iterator with Yield
- Hash Set
  - Introducing HashSet<T>
  - HashSet<T> and Uniqueness
  - HashSet<T> and Comparers
  - Comparing Elements and SetEquals()
  - Set Comparisons and Subsets
  - SortedSet<T>
- Link Lists
  - Understanding Linked Lists
  - LinkedList<T> and LinkedListNode<T>
  - Stack<T>
  - Queue<T>
- Enumerators
  - Enumerators and IEnumerator<T>
  - The foreach Loop
  - Why Don't Collections Enumerate Themselves?
  - Modifying While Enumerating
  - Writing Your Own Enumerator
  - Enumerable Covariance

3. Language Integrated Query Essentials
   - LINQ by example
   - Query Expressions
   - Building a LINQ Query: Query Syntax
   - Building a LINQ Query: Method Syntax
   - Using Lambda Expressions
   - LINQ and Collections

- **Stand Query Expressions**

4. **Exceptions**
   - **Introduction to Exceptions**
   - **Exception Handling and Throwing Exception**
   - **Working With IO**
5. **Object Relation Mapping**
   - **Connecting to a data source**
   - **Retrieving data from a data source**
   - **Writing data to a data source**
   - **Deleting data from a data source**
   - **Querying Data from a data source**
6. **Building Tiered Applications**
   - **Why is application design important?**
   - **Separating logic into a different layer**
7. **File Handling**
   - **Open Files**
   - **Read, Write data**
     - **Streamreader, Streamwrite, Binaryreader, Binarywriter**
   - **Use Memory Streams**
     - **File, Memory, Buffered, Network, Pipe, Crypto**
   - **Close Files**
8. **Building an application using technologies covered in this module**