OBJECT ORIENTED PROGRAMMING USING JAVA

Course Code: IS334B **Course Credits: 3:0:0:0**
Prerequisites: NIL **Contact Hours: 42**
Course coordinator(s): Dr.Mydhili K Nair

Course Objectives

The aim of this course is to:

- Familiarize the students with the Java-specific programming constructs and UML Notations for Classes, Objects, Inheritance, Composition, Aggregation, Packages and Interfaces.
- Introduce the students to Object Oriented way of programming in Java through Classes and Objects.
- Acquaint the students with concepts such as command line arguments, static, final, nested classes and inheritance
- Introduce the students to the concepts of Packages, Interfaces and Exception Handling nuances
- Familiarize the students with multi-threaded programming and the Java's Collection framework

Course Contents

UNIT – I

Control Statements: Java's Selection Statements, if, switch, Iteration Statements, while, do-while, for, the For-Each Version of the for Loop, Nested Loops, Jump Statements, Using break, Using continue **Introducing Classes:** Class Fundamentals, Declaring Objects, A Closer Look at new, Assigning Object Reference Variables, Introducing Methods, Constructors, Parameterized Constructors, The this Keyword, Instance Variable Hiding, Garbage Collection, The finalize( ) Method, A Stack Class. **UML Notations** for Classes and Objects

UNIT – II

A Closer Look at Methods and Classes: Overloading Methods, Overloading Constructors, Using Objects as Parameters, A Closer Look at Argument Passing, Returning Objects, Recursion, Introducing Access Control, Understanding static, Introducing final, Arrays Revisited, Introducing Nested and Inner Classes, Exploring the String Class, Using Command-Line Arguments, Varargs: Variable-Length Arguments. **Inheritance:** Inheritance Basics, Using super, Creating a Multilevel Hierarchy, When Constructors Are Executed, Method Overriding. **UML Notations** for Inheritance, Composition and Aggregation

UNIT – III

Inheritance: Dynamic Method Dispatch, Why Overridden Methods? Using Abstract Classes, Using final with Inheritance, The Object Class. **Packages and Interfaces:** Packages, Access Protection, Importing Packages, Interfaces, Defining an Interfaces, Default Interface Methods, Use static Methods in an Interface, Final Thoughts on Packages

and Interfaces. **UML Notations** for Packages and Interfaces

UNIT – IV

Exception Handling: Exception-Handling Fundamentals, Exception Types, Uncaught Exceptions, Using try and catch, multiple catch Clauses, Nested try Statements, throw, throws, finally. **Exception Handling:** Java's Built-in exceptions, Creating Your Own Exception Subclasses, Chained Exceptions, Three Recently Added Exception Features, Using Exceptions.

UNIT – V

java.util Part 1: The Collections Framework: Collections Overview**,** The Collection Interfaces, The Collection Classes, Accessing a Collection via an Iterator -Using an Iterator, The For-Each Alternative to Iterators; Storing User-Defined Classes in Collections; Working with Maps - The Map Interfaces, The Map Classes; The Collection Algorithms; Arrays

Text Books:

1. Herbert Schildt, "Java: The Complete Reference", 9th Edition, McGraw Hill
2. Michael Blaha, James Rambaugh, "Object-Oriented Modeling and Analysis with UML", Pearson, 2nd Edition, 1st Impression

Course Outcomes

Student will be able to:

CO1: Write programs using Java-specific programming constructs and Object Oriented way of programming through Classes and Objects. **(PO1, PO2,PO9) (PSO1)**

CO2: Design solutions to real-world problems using UML Notations for Classes, Objects, Inheritance, Composition, Aggregation, Packages and Interfaces **(PO2, PO3, PO4, PO6, PO9, PO10, PO12) (PSO1, PSO2, PSO3)**

CO3: Write programs using concepts such as command line arguments, static, final, nested classes and inheritance **(PO1, PO2,PO9) (PSO1, PSO3)**

CO4: Apply the concepts of Packages, Interfaces and Exception Handling nuances to solve a given real-world problem. **(PO1, PO2, PO3, PO4, PO6,PO9) (PSO1, PSO2,PSO3)**

CO5: Use the Java's Collection framework to solve computing real-world problems. **(PO1, PO2, PO4, PO6,PO9) (PSO1, PSO2,PSO3)**