Teaching Guidelines for

**Object Oriented Programming with Java**

PG-DAC August 2024

**Duration: 116 hours** (60 theory hours + 56 lab hours)

**Objective:** To reinforce knowledge of Object Oriented Programming concepts using Core Java.

**Prerequisites:** Basic knowledge of computer programming

**Evaluation:** Total 100 marks

**Weightage:** CCEE – 40%, Lab exam – 40%, Internals – 20%

# Text Book:

* Core and Advanced Java Black Book / Dreamtech Press

# References:

* Java 8 Programming Black Book / Dreamtech Press
* Core Java : Volume 1 - Fundamentals by Cay S. Horstmann / Prentice Hall
* Core Java : Volume 2 - Advanced Features by Cay S. Horstmann / Prentice Hall
* Programming in Java by Sachin Malhotra, Saurabh Choudhary / Oxford University Press
* Java The Complete Reference by Herbert Schildt / McGraw Hill
* Core Java 8 for Beginners by Sharanam Shah, Vaishali Shah / Shroff Publishers
* Murach’s Java Programming by Joel Murach / Mike Murach
* Object-Oriented Analysis and Design with applications by Grady Booch / Pearson

# (Note: Each Session is of 2 hours)

**Session 1: Introduction to Java**

**Lecture:**

* Introduction to java
* Features of java
* JVM Architecture
* JDK and its usage
* Structure of java class
* Working with data types: Primitive data types

**Sessions 2 & 3: Basic programming concepts**

**Lecture:**

* Java Tokens
* Declaring variables and methods
* Data type compatibility
* Operators
* Control statements
* Arrays 1-D and multidimensional array

**Lab 1 & 2:**

* Get yourself acquainted with java environment.
* Print different patterns of asterisk (\*) using loops (e.g. triangle of \*).

**Tutorial:**

* Compare syntactical similarities and dissimilarities between Java and C++.

**Object Oriented Programming Concepts**

**Session 4: Object Oriented Programming Concepts**

**Lecture:**

* Introduction to OOP
* Classes and Objects
* OOP principles
* Encapsulation, Abstraction, Inheritance and Polymorphism

**Session 5:**

**Lecture:**

* Static variables and methods
* Accessing static variables and methods of different class
* Introduction to reference data types
* Reference variables and methods
* Difference between reference data types and primitive data types
* Difference between reference variable and static variable

**Session 6:**

**Lecture:**

* Constructors, initializing reference variables using constructors.
* Pass by value v/s pass by reference.
* Re-assigning a reference variable.
* Passing reference variable to method
* Initializing reference variable of different class
* Heap memory and stack memory

**Lab 3 & 4:**

* Print default values of static & instance variables for different data types.
* Build a class Employee which contains details about the employee and compile and run its instance.
* Build a class which has references to other classes. Instantiate these reference variables and invoke instance methods.

**Tutorial:**

* Understand role of stack and heap memory in method invocation and object creation.

**Session 7:**

**Lecture:**

* Inheritance: single & multilevel
* Inheritance: Hierarchical
* Association, Aggregation and Composition
* Polymorphism: Compile time and runtime polymorphism
* Rules of overriding and overloading of methods
* super and this keyword

**Lab 5 & 6:**

* Create a class Employee and encapsulate the data members.
* Create demo applications to illustrate different types of inheritance.

**Session 8:**

**Lecture:**

* Upcastingand downcasting of a reference variable
* Abstract class and abstract methods
* Interface (implementing multiple interfaces)

**Sessions 9 & 10:**

**Lecture:**

* Final variables, final methods and final class
* Functional interface
* New interface features (Java 8 &11)
* Lambda Expression
* Inner Class (Regular, Method local, Anonymous & static inner class)
* Enum

**Lab 7, 8 & 9:**

* Create an Array of Employee class and initialize array elements with different employee objects.
* Try to understand the no. of objects on heap memory when any array is created.
* Implementaion of functional interface with anonymous class and lambda expression.

**Session 11:**

**Lecture:**

* Access modifiers (public, private, protected and default)
* Packages and import statements
* Static imports
* Constructor chaining (with and without packages)
* Accessing protected variables and methods outside the package

**Session 12:**

**Lecture:**

* Garbage collection in java
* Requesting JVM to run garbage collection.
* Different ways to make object eligible for garbage collection: (Nulling a reference variable, Re-assigning a reference variable & island of isolation)
* Finalize method.

**Lab 10 & 11:**

* Create a demo application to understand the role of access modifiers.
* Implement multilevel inheritance using different packages.
* Access/invoke protected members/methods of a class outside the package.
* Override finalize method to understand the behavior of JVM garbage collector.

**Sessions 13 & 14:**

**Wrapper Classes and String Class**

**Lecture:**

* Wrapper classes and constant pools
* String class, StringBuffer&StringBuilder class
* String pool

**Lab 12 & 13:**

* Create sample classes to understand boxing & unboxing.
* Use different methods of java defined wrapper classes.
* Create StringDemo class and perform different string manipulation methods.

**Tutorial:**

* Understand the difference between String / StringBuffer / StringBuilder.

**Sessions 15 & 16:**

**Exception Handling**

**Lecture:**

* Exception hierarchy, Errors, Checked and un-checked exceptions.
* Exception propagation
* try-catch-finallyblock, throws clause and throw keyword.
* Multi catch block.
* Creating user defined checked and unchecked exceptions.

**Lab 14 & 15:**

* Create user defined checked and unchecked exceptions.

**Sessions 17:**

**java.io & java.nio Package**

**Lecture:**

* Brief introduction to InputStream, OutputStream, Reader and Writer interfaces
* NIO package
* Serialization and de-serialization
* Shallow copy and deep copy

**Session 18:**

**Lecture:**

**Object Class & java.util Package**

* Date, DateTime, Calendar class
* Converting Date to String and String to Date using SimpleDateFormat class
* Object Class: Overriding to String, equals &hashcodemethod

**Lab 16 & 17:**

* Create a Demo class to read & write image/text files.
* Create Serialization Demo class to illustrate serialization and de-serialization process.
* Create a demo class for Date, Time and Calendar

**Collections**

**Sessions 19, 20, 21 & 22:**

**Lecture:**

* Introduction to collections: Collection hierarchy
* List, Queue, Set and Map Collections
* List Collection:
  + ArrayList, LinkedList
  + Vector (insert, delete, search, sort, iterate, replace operations)
* Collections class
* Comparable and Comparator interfaces
* Queue collection

**Lab 18, 19, 20 & 21:**

* Create DateManipulator class to convert String to date, date to String and to find out number of days between two dates.
* Create a list of java defined wrapper classes and perform insert/delete/search/iterate/sort operations.
* Create a collection of Employee class and sort objects using comparable and comparator interfaces.
* Implement Queue data structure using LinkedList and Queue collection.

**Sessions 23, 24, 25 & 26:**

**Lecture:**

* Set Collection:
  + HashSet, LinkedHashSet&TreeSet collection
  + Backed set collections.
* Map Collection:
  + HashTable, HashMap, LinkedHashMap&TreeMap classes
  + Backed Map collections.
* Concurrent collections
* Implentation of Java 8 stream API

**Lab 22, 23 & 24:**

* Create an Employee HashSet collection and override equals &hashCode methods to understand how the set maintains uniqueness using these methods.
* Create a Sample class to understand generic assignments using “? extendsSomeClass” , “? supersomeclass ” and “?”
* Implentation of streams methods: Map, Reduce, Count, Sort, etc.

**Session 27:**

**Lecture:**

* MultiThreading : Thread class and Runnable Interface
* sleep, join, yield, setPriority, getPrioritymethods.
* ThreadGroup class

**Lab 25:**

* Create multiple threads using Thread class and Runnable interfaces.
* Assign same task and different task to multiple threads.
* Understand sleep, join, yield methods.

**Sessions 28 & 29:**

**Lecture:**

* Synchronization
* Deadlock
* Wait, notify and notifyAllmethods.
* Producer & Consumer problem

**Lab 26 & 27:**

* Create a Deadlock class to demonstrate deadlock in multithreading environment.
* Implement wait, notify and notifyAll methods.
* Demonstrate how to share threadlocal data between multiple threads.

**Session 30: Generics and Reflection API**

**Lecture:**

* Introduction to generics
* Generic classes
* Generic methods
* Wild cards (upper and lower)
* Metadata & Reflection

**Lab 28:**

* Invoke private methods of some other class using reflection.
* Create multiple threads using anonymous inner classes.
* Create multiple threads using lambda expressions.