Course Code	**:**	**MCS-206**
Course Title	**:**	**Object Oriented Programming Using Java**
Assignment Number	**:**	**PGDCA_NEW(II)/206/Assignment/2026-27**
Maximum Marks	**:**	**100**
Weightage	**:**	**30%**
Last Date of Submission	**:**	**31st October, 2026 (for July session)** **30th April, 2027 (for January session)**

Note: This assignment has eight questions of 80 Marks. Answer all the questions. The remaining 20 marks are for viva voce. You may use Java code, illustrations and diagrams to enhance the explanations. Please go through the guidelines regarding assignments given in the Programme Guide for the format of presentation.

Question1:

(a) Explain different operators available in the Java programming language, providing a brief explanation for each category. **(5 Marks)**

(b) Define the paradigm of Object-Oriented Programming (OOP) and detail its core architectural features. Additionally, analyze the structural role and significance of the Java Virtual Machine (JVM) in achieving platform independence. **(5 Marks)**

Question2:

(a) Define a class within the context of Java Programming. Explain how a class is declared, and evaluate the scope and visibility of the access specifiers (public, private, protected, and default) using Java code snippets. **(5 Marks)**

(b) Explain the uses of ***this, final*** and ***super*** keywords in Java with the help of example code snippets. **(3 Marks)**

(c) What is the purpose of the Character wrapper class in Java? Explain the operational utility of any four methods of this class. **(2 Marks)**

Question 3:

(a) Write a Java program to find the product of two matrices. Define proper class(es) and methods in your program. **(4 Marks)**

(b) Describe how file handling and I/O operations are managed in Java. Explain the use of the Java File class and its methods using Java code. **(6 Marks)**

Question 4:

(a) Define the concept of inheritance in object-oriented systems and outline its primary advantages. Discuss the specific types of inheritance supported (and not supported) by Java, and demonstrate its implementation using a clear programming example. **(6 Marks)**

(b) What is an abstract class? Explain its advantages. Critically compare it with an interface in Java. **(4 Marks)**

Question 5:

(a) Define polymorphic behavior. Explain the advantages of Polymorphism with example code in Java. **(4 Marks)**

(b) Define an exception and analyse the primary systemic and logical causes behind exception generation. How are exceptions handled in Java? Design a custom, user-defined exception class to validate human age inputs (e.g., throwing an exception if the input is negative or exceeds a realistic biological threshold). **(6 Marks)**

Question 6:

(a) Explain the concept of multithreading and differentiate between the two primary mechanisms used to create threads in Java. Write a Java program that demonstrates the practical implementation of thread synchronization. **(7 Marks)**

(b) Explain the use of the List interface and the Vector class in Java with the help of suitable examples. **(3 Marks)**

Question 7:

(a) Detail the architectural and package hierarchy of the Java Swing component library. Provide a comprehensive technical comparison between the Abstract Window Toolkit (AWT) and the Swing framework. **(6 Marks)**

(b) What is FXML? Explain event handling in JavaFX with the help of a code snippet. **(4 Marks)**

Question 8:

(a) Write a complete Java Database Connectivity (JDBC) program to manage a relational user schema for a library management system. Make necessary design assumptions for the database structure and ensure proper exception handling is implemented. **(8 Marks)**

(b) Explain the architectural role and lifecycle of the ResultSet interface in JDBC programming. Demonstrate how to traverse and retrieve data from a query outcome, using a short and focused code snippet. **(2 Marks)**

Question1:

(a) Explain different operators available in the Java programming language, providing a brief explanation for each category. (5 Marks)

Answer

Java operators are symbols that act on one or more operands and produce a result. The type of operands and the operator determine how an expression is evaluated. Java groups operators by their purpose.

Arithmetic operators: `+`, `-`, `*`, `/` and `%` perform addition, subtraction, multiplication, division and remainder. For integers, `7 / 2` gives `3`, while `7 % 2` gives `1`. The `+` operator also concatenates strings, such as `"Java" + " Code"`.

Unary operators: Unary `+` and `-` indicate sign; `++` and `--` increase or decrease a variable by one; `!` reverses a boolean; and `~` complements the bits of an integral value. Prefix increment changes a variable before its value is used, while postfix increment first uses the previous value.

Relational/equality operators: `<`, `>`, `<=`, `>=`, `==`, and `!=` compare values and return `boolean`. In particular, `==` tests whether two object references refer to the same object; `equals()` is normally used for comparing the meaningful content of strings and many other objects.

Logical operators: `&&`, `||` and `!` combine or negate boolean conditions. `&&` and `||` use short-circuit evaluation: if the first operand decides the answer, Java does not evaluate the second operand. For instance, `x != 0 && 10 / x > 2` guards against division by zero.

Bitwise and shift operators: `&`, `|`, `^` and `~` operate on the individual bits of integer types. `&`, `|` and `^` can also act on booleans without short-circuiting. The shifts `<<`, `>>` and `>>>` move bits left, signed right and unsigned right respectively. They are useful for masks and low-level representation.

Assignment operators: `=` assigns a value, while `+=`, `-=`, `*=`, `/=`, `%=` and bitwise/shift compound assignments update a variable conveniently. For example, `count += 2` increases `count` by two. Compound assignment includes an implicit cast to the left-hand variable's type, which requires care with narrowing conversions.

Conditional operator: `condition ? value1 : value2` selects one of two expressions; for example, `int larger = a > b ? a : b;`.

Type-related operators: `instanceof` checks compatible reference types at runtime. A cast, such as `(double) count`, requests a conversion; casts between