

Internal Assessment Test 1 – November 2025

Sub:	Object Oriented Programming with Java					Sub Code:	BCS306A	Branch:	AI & DS(CSE DS)		
Date:		Duration:	90 minutes	Max Marks:	50	Sem	III A,B,C			OBE	
<u>Answer any FIVE Questions</u>								MARKS	CO	RBT	
1		In what ways does Java’s Object-Oriented approach solve the limitations of procedural programming? Illustrate with an example.						[10]	1	L2	
2		Write a Java program to create a class named Employee with the following: Instance variables:empId, name and salary, A parameterized constructor to initialize the variables, A method display() to print the details Create a main class to create objects for 3 employees and display their information.						[10]	2	L3	
3		Write a Java program to accept marks from the user and display the corresponding grade as per the following: Marks ≥ 90 → Grade A, Marks ≥ 75 → Grade B, Marks ≥ 50 → Grade C, else Fail.						[10]	2	L3	

4	During execution, Java automatically converts smaller data types into larger ones in certain expressions. (a) Explain this concept with an example. (b) Find and correct the error in the following Java program: <pre>int num = 10.5; System.out.println(num);</pre>	[10]	2	L3
---	---	------	---	----

5	(a) Explain how objects are declared and created in Java. (b) Describe the role of the new keyword in object creation. (c) What is a constructor? Write a small example to demonstrate how it is used during object creation.	[10]	1	L2
6	Explain the difference between a for loop and a while loop in Java. Write one simple example for each.	[10]	1	L2

Q1: In what ways does Java's Object-Oriented approach solve the limitations of procedural programming? Illustrate with an example. [10]

Answer:

Procedural programming (like C) organizes code into functions, but as programs grow, it becomes harder to manage data and functions together.

Java's **Object-Oriented Programming (OOP)** solves these problems through concepts like **Encapsulation, Inheritance, Polymorphism, and Abstraction**.

Limitation (Procedural)	OOP Solution in Java
Data and functions are separate	Combines data and behavior into objects
Code reuse is difficult	Inheritance allows reusing and extending existing code
Code maintenance is harder	Encapsulation hides internal details, making code modular
No data security	Access modifiers (private, public, etc.) protect data
Polymorphism absent	Method Overloading & Overriding allow flexible code
Example(Any example that demonstrates the above)	

Q2: Employee Class Program [10]

```

class Employee {
    int empId;
    String name;
    double salary;

    // Parameterized Constructor
    Employee(int empId, String name, double salary) {
        this.empId = empId;
        this.name = name;
        this.salary = salary;
    }

    // Method to display details
    void display() {
        System.out.println("Employee ID: " + empId);
        System.out.println("Name: " + name);
        System.out.println("Salary: " + salary);
        System.out.println("-----");
    }
}

public class EmployeeMain {
    public static void main(String[] args) {
        Employee e1 = new Employee(101, "Ravi", 45000);
        Employee e2 = new Employee(102, "Sneha", 55000);
    }
}

```

```

Employee e3 = new Employee(103, "Arjun", 60000);

e1.display();
e2.display();
e3.display();
}
}

```

Q3: Program to Accept Marks and Display Grade [10]

```

import java.util.Scanner;

public class GradeCalculator {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Marks: ");
        int marks = sc.nextInt();

        if (marks >= 90)
            System.out.println("Grade A");
        else if (marks >= 75)
            System.out.println("Grade B");
        else if (marks >= 50)
            System.out.println("Grade C");
        else
            System.out.println("Fail");

        sc.close();
    }
}

```

Q4. Automatic Type Conversion (Type Promotion) [10 Marks]

(a) Explanation:

In **Java**, when performing operations involving different data types, **smaller data types are automatically promoted to larger data types** to prevent data loss.

This automatic conversion is known as **Type Promotion** or **Implicit Type Conversion**.

During expression evaluation, the conversion happens in the following order:

byte → short → int → long → float → double

Example

```
public class TypePromotionExample {  
    public static void main(String[] args) {  
        int a = 10;  
        double b = 5.5;  
        double result = a + b; // int 'a' is automatically promoted to double  
        System.out.println("Result = " + result);  
    }  
}  
o/p-Result = 15.5
```

b) Find and Correct the Error

Option 1 – Use double type

```
double num = 10.5;  
System.out.println(num);
```

Output:

10.5

Option 2 – Use Explicit Casting

```
int num = (int) 10.5; // fractional part truncated  
System.out.println(num);
```

Output:

10

5)

(a) Explain how objects are declared and created in Java.

In Java, an **object** is an instance of a class.

To use any class, you must first **declare** a reference variable and then **create** the object.

Syntax:

```
ClassName objectName = new ClassName();
```

Explanation:

- `ClassName` → the name of the class.
- `objectName` → the reference variable that holds the object.
- `new` → keyword used to create the object dynamically in memory.
- `ClassName()` → calls the class constructor.

Example:

```
Student s1 = new Student();
```

Here, `s1` is an object (instance) of the `Student` class.

(b) Describe the role of the new keyword in object creation.

The **new** keyword in Java:

- Allocates memory for a new object at runtime.
- Calls the class **constructor** to initialize the object.
- Returns a reference to the created object.

Example:

```
Car c1 = new Car(); // 'new' allocates memory and calls the Car()
constructor
```

(c) What is a constructor? Give an example.

A **constructor** is a special method in Java that:

- Has the **same name as the class**.
- **Initializes** the object when it is created.
- **Does not have a return type**, not even void.

Example:

```
class Student {

    String name;

    int age;


    // Constructor

    Student(String n, int a) {

        name = n;

        age = a;

    }

}
```

```
public class Main {
```

```

public static void main(String[] args) {

    Student s1 = new Student("Ravi", 20); // Constructor called

    System.out.println(s1.name + " " + s1.age);

}
}

```

6) Explain the difference between a for loop and a while loop in Java.

Feature	for loop	while loop
Use	Used when the number of iterations is known	Used when the number of iterations is unknown
Syntax	for(initialization; condition; increment)	while(condition)
Initialization, condition, increment	All are written in a single line	Written separately
Best used for	Counting or fixed repetition	Condition-based repetition

Example:


```
for (int i = 1; i <= 5; i++) {  
    System.out.println(i);  
}
```

Emple 2:

```
int i = 1;  
while (i <= 5) {  
    System.out.println(i);  
    i++;  
}
```

Q1: In what ways does Java's Object-Oriented approach solve the limitations of procedural programming? Illustrate with an example. [10]

Answer:

Procedural programming (like C) organizes code into functions, but as programs grow, it becomes harder to manage data and functions together.

Java's **Object-Oriented Programming (OOP)** solves these problems through concepts like **Encapsulation, Inheritance, Polymorphism, and Abstraction**.