CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | | |

Internal Assesment Test – II, August 2025

Sub:	Object Oriented Programming using Java							Code:	MMC202
Date:	07/08/2025	Duration:	90 mins	Max Marks:	50	Sem:	II	Branch:	MCA

Answer Any One FULL Question from each part.

		Marks	OBE CO	OBE RBT
	PART I			
1)	What are nested interfaces? Write a program to define and use a nested interface.	10	CO3	L2
	OR			
2)	Explain packages in Java. How are they created and accessed? Give suitable examples.	10	CO3	L2
	PART II			
3)	Demonstrate runtime polymorphism using interfaces. Explain how method resolution occurs at runtime.	10	CO3	L2
	OR			
4)	Explain the working of try, catch, throw, throws, and finally with a complete program.	10	CO3	L3
	PART III			
5)	Write a program that demonstrates multiple catch blocks. Explain how Java decides which catch block to execute	10	CO3	L2
	OR			
6)	Write a Java program to demonstrate how classes in different packages access each other. Explain the output.	10	CO3	L3
	PART IV	10	CO3	L3
7)	Write a Java program (Queue Implementation) to throw a user-defined exception. Explain how to declare, throw, and catch custom exceptions.			
	OR			

8)	Explain the life cycle of a thread in Java with a neat diagram and code examples.	10	CO4	L3
	PART V	10	CO4	L3
9)	Explain thread priorities and thread states in Java. How can thread priority affect program execution?			
	OR			
10)	Discuss the use of MouseListener and MouseMotionListener interfaces. Write a Java program that uses both to capture mouse clicks and movements.	10	CO5	L2

Solution

1) What are nested interfaces? Write a program to define and use a nested interface.

A nested interface is an interface declared inside another class or interface. They are implicitly static and used to logically group interfaces that are only used by the enclosing class.

Example:

```
class Outer {
   interface Nested {
      void show();
   }
}

class Test implements Outer.Nested {
   public void show() {
      System.out.println("Nested Interface Implemented");
   }
   public static void main(String[] args) {
      Outer.Nested obj = new Test();
      obj.show();
   }
}
```

Output:
Nested Interface Implemented

2) Explain packages in Java. How are they created and accessed? Give suitable examples.

A package in Java is a mechanism to group related classes and interfaces. It helps to avoid name conflicts and provides modularization.

Steps:
1. Define the package using the 'package' keyword.
2. Place the class in the corresponding directory.
3. Compile with correct folder structure.
4. Import the package using 'import'.

Example:

```
// File: mypack/Message.java
package mypack;
public class Message {
    public void display() {
        System.out.println("Hello from package");
    }
}

// File: TestPackage.java
import mypack.Message;
public class TestPackage {
    public static void main(String[] args) {
        Message m = new Message();
        m.display();
    }
}
```

Output:
Hello from package

3) Demonstrate runtime polymorphism using interfaces. Explain how method resolution occurs at runtime.

Runtime polymorphism means that a method call is resolved at runtime based on the actual object. Interfaces allow different classes to implement the same method differently.

Example:

```
interface Speaker {
    void speak();
}

class Dog implements Speaker {
    public void speak() { System.out.println("Woof"); }
}

class Cat implements Speaker {
    public void speak() { System.out.println("Meow"); }
}

public class Demo {
```

```
    public static void main(String[] args) {
        Speaker s;
        s = new Dog(); s.speak();
        s = new Cat(); s.speak();
    }
}
```

Output:
Woof
Meow

Explanation: Method resolution occurs at runtime depending on the object type (Dog or Cat).

4) Explain the working of try, catch, throw, throws, and finally with a complete program.

These keywords are used in exception handling.
• try: block of code to test for exceptions
• catch: handles exceptions
• throw: used to explicitly throw an exception
• throws: declares exceptions a method can throw
• finally: executes code regardless of exception

Example:

```
public class ExceptionDemo {
    static int divide(int a, int b) throws ArithmeticException {
        if(b == 0) throw new ArithmeticException("Divide by zero");
        return a / b;
    }
    public static void main(String[] args) {
        try {
            int res = divide(10, 0);
            System.out.println("Result: " + res);
        } catch (ArithmeticException e) {
            System.out.println("Caught: " + e);
        } finally {
            System.out.println("Finally block executes");
        }
    }
}
```

Output:
Caught: java.lang.ArithmeticException: Divide by zero
Finally block executes

5) Write a program that demonstrates multiple catch blocks. Explain how Java decides which catch block to execute.

When multiple catch blocks are present, Java matches the thrown exception with the first compatible catch block.

Example:

```
public class MultiCatchDemo {
   public static void main(String[] args) {
      try {
         int[] arr = {1, 2};
         int x = arr[5];
      } catch (ArrayIndexOutOfBoundsException e) {
         System.out.println("Array error: " + e);
      } catch (Exception e) {
         System.out.println("General error: " + e);
      }
   }
}
```

Output:
Array error: java.lang.ArrayIndexOutOfBoundsException

Explanation: The specific exception block is executed first.

6) Write a Java program to demonstrate how classes in different packages access each other. Explain the output.

Packages allow access to classes across different namespaces using the import statement.

Example:

```
// File: pack1/Message.java
package pack1;
public class Message {
   public String text = "Hello from pack1";
}

// File: pack2/Test.java
package pack2;
import pack1.Message;
public class Test {
   public static void main(String[] args) {
      Message m = new Message();
      System.out.println(m.text);
   }
}
```

Output:
Hello from pack1

7) Write a Java program (Queue Implementation) to throw a user-defined exception. Explain how to declare, throw, and catch custom exceptions.

Custom exceptions extend the Exception class. We can declare them and throw when required.

Example:

```
class QueueEmptyException extends Exception {
   QueueEmptyException(String msg) { super(msg); }
}

class Queue {
   int[] arr = new int[2];
   int size = 0;
   void enqueue(int x) {
      arr[size++] = x;
   }
   int dequeue() throws QueueEmptyException {
      if(size == 0) throw new QueueEmptyException("Queue is empty");
      return arr[--size];
   }
}

public class QueueDemo {
   public static void main(String[] args) {
      Queue q = new Queue();
      try {
         q.dequeue();
      } catch(QueueEmptyException e) {
         System.out.println(e.getMessage());
      }
   }
}
```

Output:

```
Queue is empty
```

8) Explain the life cycle of a thread in Java with a neat diagram and code examples.

Thread states: NEW → RUNNABLE → RUNNING → WAITING/TIMED_WAITING/BLOCKED → TERMINATED.

Example:

```
public class ThreadLifeCycleDemo {
   public static void main(String[] args) throws InterruptedException {
      Thread t = new Thread(() -> {
         System.out.println("Thread is running");
```

```
        });
        System.out.println("State: " + t.getState()); // NEW
        t.start();
        System.out.println("State: " + t.getState()); // RUNNABLE
        t.join();
        System.out.println("State: " + t.getState()); // TERMINATED
    }
}
```

Output:
State: NEW
State: RUNNABLE
Thread is running
State: TERMINATED

9) Explain thread priorities and thread states in Java. How can thread priority affect program execution?

Java threads have priorities from 1 to 10. States include NEW, RUNNABLE, BLOCKED, WAITING, TIMED_WAITING, and TERMINATED.

Example:

```
class Worker extends Thread {
    Worker(String name, int p) {
        super(name);
        setPriority(p);
    }
    public void run() {
        for(int i=0;i<5;i++) {
            System.out.println(getName() + " running");
        }
    }
}

public class PriorityDemo {
    public static void main(String[] args) {
        Worker t1 = new Worker("Low", Thread.MIN_PRIORITY);
        Worker t2 = new Worker("High", Thread.MAX_PRIORITY);
        t1.start(); t2.start();
    }
}
```

Output varies: Higher priority threads may get more CPU time, but scheduling is OS-dependent.

10) Discuss the use of MouseListener and MouseMotionListener interfaces. Write a Java program that uses both to capture mouse clicks and movements.

MouseListener handles mouseClicked, mousePressed, mouseReleased, mouseEntered, and mouseExited.
MouseMotionListener handles mouseDragged and mouseMoved.

Example:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class MouseDemo extends JPanel implements MouseListener, MouseMotionListener {
   String msg = "";
   int x, y;

   public MouseDemo() {
      addMouseListener(this);
      addMouseMotionListener(this);
   }

   public void paintComponent(Graphics g) {
      super.paintComponent(g);
      g.drawString(msg, x, y);
   }

   public void mouseClicked(MouseEvent e) { msg = "Clicked"; x=e.getX(); y=e.getY(); repaint(); }
   public void mousePressed(MouseEvent e) {}
   public void mouseReleased(MouseEvent e) {}
   public void mouseEntered(MouseEvent e) {}
   public void mouseExited(MouseEvent e) {}
   public void mouseDragged(MouseEvent e) { msg = "Dragged"; x=e.getX(); y=e.getY(); repaint(); }
   public void mouseMoved(MouseEvent e) { msg = "Moved"; x=e.getX(); y=e.getY(); repaint(); }

   public static void main(String[] args) {
      JFrame f = new JFrame("Mouse Demo");
      f.add(new MouseDemo());
      f.setSize(400, 300);
      f.setVisible(true);
      f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }
}
```

Output: Window displays events as you click, move, or drag the mouse.