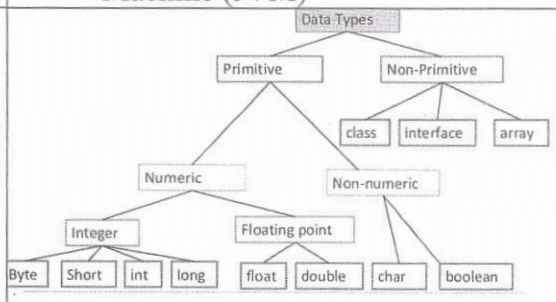
Scoring Indicators

COURSENAME: 4343- Object Oriented Programming Concepts Using Java

COURSECODE: 4343

QID: 2102250073

QNo	Scoring Indicators	Split score	Sub Total	Totals core
	PART A			**9**
I.1	True	1	1	9
I.2	Java virtual machine	1	1	
I.3	0	1	1	
I.4	10	1	1	
I.5	class	1	1	
I.6	break	1	1	
I.7	True	1	1	
I.8	import	1	1	
I.9	Init()	1	1	
	PART B			**24**
II.1	Dynamic binding is associated with polymorphism and inheritance. It means deciding which method to call at runtime, not at compile time. This happens when a method is overridden in a subclass.	3	3	3
II.2	When a Java program is compiled, the source code is transformed into a binary representation of the program called bytecode, which can be executed by the Java Virtual Machine (JVM). Bytecode is a low-level, platform-independent code that is designed to be easily interpreted and executed by the JVM	3	3	3
II. 3	- Set Up the Development Environment-Install JDK (Java Development Kit). - Write the Java Program :Create a .java file with a class definition. - Implement the main method - Compile the Program	3 1	3	3

	- Use the javac compiler to convert the .java file into bytecode - Run the Java Program - Execute the compiled .class file using the Java Virtual Machine (JVM)			
II. 4	 Primitive data types- integer & floating point Interger types Floating point types	1 1 1	3	3
II. 5	< Is less than <= Is less tan or equal to > Is greater than >= Is greater than or equal to == Is equal to != Is not equal to Used to check the relationship between two operands Binary operator Returns either true or false	Listing 2 Expln 1	3	3

II. 6	import java.util.Scanner;
	public class PersonDetails {
	public static void main(String[] args) {
	Scanner scanner = new Scanner(System.in);
	System.out.print("Enter your full name: ");
	String fullName = scanner.nextLine();
	System.out.print("Enter your age: ");
	int age = scanner.nextInt();
	System.out.println("Full Name: " + fullName);
	System.out.println("Age: " + age);
	}
	}
	(Or program may be written using BufferedReader or DataInputStream)
		Import - 1
		Input-1
		Output-1
			3	3

II. 7	if statement if-else statement if-else-if ladder switch statement (any three)	1 1 1	3	3
II. 8	A superclass (also known as a parent class or base class) is a class from which other classes inherit properties and behaviors. The subclass (child class) derives attributes and methods from the superclass using **extends** keyword. Any correct example	2 1	3	3
II. 9	- An interface in Java is a kind of a class that contains only abstract methods and final variables. It is used to achieve multiple inheritance and abstraction. - Defined using the **interface** keyword. - A class implements an interface using the **implements** keyword.	3 points	3	3
II. 10	- Named using the standard Java naming rules - Packages begins with lowercase letters - Class names begins with uppercase letters - Method names begins with lowercase letters Eg `double y = java.lang.Math.sqrt(x);` Package Name / Class Name / Method Name	2 1	3	3
	PART C			**42**
III.	• Object is the basic entity in object-oriented programming having its own set of properties and behavior. • a class is a blueprint for creating objects. It defines the properties and behavior of the objects of that class. An object is an instance of a class **any example**	2 2 3	7	7
	OR			
IV.	Simple, Object Oriented, Distributed, Compiled and Interpreted, Robust, Secure, Architecture Neutral, Portable, High Performance Multithreaded, Dynamic, Extensible	Expln Any seven points	7	7

V.	Constructor overloading : Feature that allows a class to have multiple constructors with different input arguments. Java differentiates constructors based on their parameter lists.(number, type, or order) Any correct example	Expln 4 Eg 3	7	7
OR				

VI.	The looping statements are
	While()
	Do while()
	For()
		Listing - 1
		Expln with eg- 2 each
			7	7

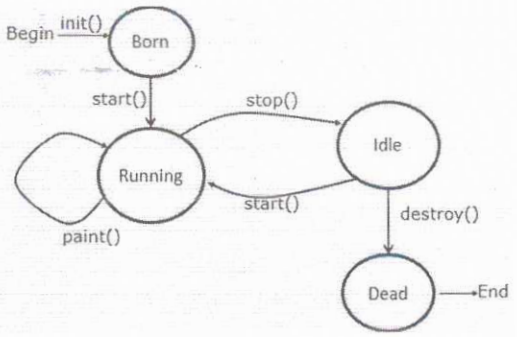
VII.	class Programmer {
	String name;
	int experienceYears;
	// Constructor
	public Programmer(String name, int experienceYears) {
	this.name = name;
	this.experienceYears = experienceYears;
	}
	}
	// Subclass: Project
	class Project extends Programmer {
	String technologyUsed;
	// Constructor
	public Project(String name, int experienceYears, String technologyUsed) {
	super(name, experienceYears); // Calling superclass constructor
	this.technologyUsed = technologyUsed;
	}
	// Method to display details
	public void displayDetails() {
		7	7	7

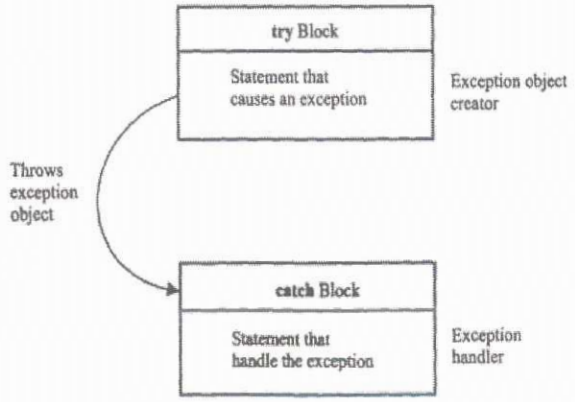
	System.out.println("Programmer Name: " + name);
	System.out.println("Experience Years: " + experienceYears);
	System.out.println("Technology Used: " + technologyUsed);
	}
	}
	// Main Class
	public class ProgrammerDemo {
	public static void main(String[] args) {
	// Creating an object of Project class
	Project project1 = new Project("Alice", 5, "Java");
	project1.displayDetails();
	}
	}			

	OR			
VIII.	Final variables, methods, and classes are all features in object-oriented programming languages that restrict the modification of their definitions once they are declared. **Final variables** are variables that cannot be reassigned after their initial value is assigned. **Final methods** are methods that cannot be overridden in a subclass. **Final classes** are classes that cannot be subclassed, meaning that they cannot be extended by another class.	7	7	7
IX.	• Single inheritance refers to a situation where a subclass inherits from only one superclass. • In this case, the subclass can access all the public and protected members of the superclass. Here is an example of single inheritance in Java: • eg Class person { -------- } Class student extends person { ------ } • Multilevel inheritance, on the other hand, refers to a situation where a subclass inherits from a superclass,	3	7	7

	which in turn inherits from another superclass. In this case, the subclass can access the public and protected members of both superclasses. Class person { --------} Class student extends person { ----- } Class result extends student {-----}	4		
	OR			

X.	interface Item {
	double getAmount(); // Method to calculate the total
	}
	// Class: Purchase implementing Item interface
	class Purchase implements Item {
	int quantity;
	double unitPrice;
	// Constructor
	public Purchase(int quantity, double unitPrice) {
	this.quantity = quantity;
	this.unitPrice = unitPrice;
	}
	// Overriding getAmount() method
	public double getAmount() {
	return quantity * unitPrice; // Calculate total amount
	}
	// Method to display purchase details
	public void displayDetails() {
	System.out.println("Quantity Purchased: " + quantity);
	System.out.println("Unit Price: " + unitPrice);
	System.out.println("Total Amount: " + getAmount());
	}
	}
	// Main Class
	public class ItemPurchaseDemo {
	public static void main(String[] args) {
	// Creating an object of Purchase class
	Purchase purchase1 = new Purchase(5, 20.5);
	purchase1.displayDetails();
	}
	}
		Interface 1
		
		Abstract method 1
		
		Implementation class &overriding 4 marks
		
		Main class 1
			7	7

XI.	Applet life cycle 1. Initialisation – invokes init() – only once - Invoked when applet is first loaded. 2. Running – invokes start() – more than once - For the first time, it is called automatically by the system after init() method execution. - It is also invoked when applet moves from idle/stop() state to active state. 3. Display – invokes paint() - more than once - It happens immediately after the applet enters into the running state. It is responsible for displaying output. 4. Idle – invokes stop() - more than once - It is invoked when the applet is stopped from running. For example, it occurs when we leave a web page. 5. Dead/Destroyed State – invokes destroy() - only once - This occurs automatically by invoking destroy() method when we quite the 	Listing 1 2 2 2	7	7
OR				
XII.	*Tasks of Exception Handling* 1) Find the problem (Hit the exception) 2) Inform that an error has occurred (Throw the exception) 3) Receive the error information (catch the exception) 4) Take corrective actions (Handle the exception)	3 2	7	7

<table>
<tr><td></td><td>


Exception handling mechanism

Syntax of exception handling code
Syntax
<pre>try {
  // Block of code to try
}
catch(Exception e) {
  // Block of code to handle errors
}
</td><td>2</td><td></td><td></td></tr>
<tr><td colspan="5"></td></tr>
<tr><td>XIII.</td><td>1. Declare the package at the beginning of a file using the form package packagename;
2. Define the class that is to be put in the package and declare it public
3. Create a subdirectory with the same name of package under the directory where the main source file is stored. The subdirectory name must match the package name exactly in cases.
4. The code is saved with the class name in the subdirectory created.ie. classname.java
5. Compile the file. This will create a class file in the directory.</td><td>7</td><td>7</td><td></td></tr>
<tr><td>XIV.</td><td>import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;

public class ButtonApplet extends Applet implements ActionListener {
Button clickButton;
String message = "";

public void init() {</td><td>Import 1

Extends Applet class and button cration-3

Event</td><td>7</td><td>7</td></tr>
</table>

	// Create a button
	clickButton = new Button("Click Me");
	
	// Add action listener to the button
	clickButton.addActionListener(this);
	
	// Add button to the applet
	add(clickButton);
	}
	
	// Event handling method
	public void actionPerformed(ActionEvent e) {
	message = "Button Clicked!";
	repaint(); // Refresh applet to show the updated message
	}
	
	// Paint method to display the message
	public void paint(Graphics g) {
	g.drawString(message, 50, 100);
	}
	}
		handling
		3
		
		3		