135

11 page April-24

Scoring Indicators

COURSENAME: 4131- Object Oriented Programming

COURSECODE:4131

QID: 2103230060

QNo	Scoring Indicators	Splits core	Sub Total	Totals core
PART A				9
I.1	class	1	1	9
I.2	constructor	1	1	
I.3	Inheritance refers to an OOP feature where a new class is created from an existing class, inheriting all the properties of the parent class, while also having the ability to add new properties and behaviors.	1	1	
I.4	final	1	1	
I.5	true	1	1	
I.6	javax.swing	1	1	
I.7	False	1	1	
I.8	JDBC stands for Java Database Connectivity, and it is an API that provides a standard way for Java programs to interact with relational databases.	1	1	
I.9	Drop table	1	1	
PART B				24
II.1	• 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	1	1	3
II.2	1. Object 2. Class 3. Encapsulation 4. Abstraction 5. Polymorphism 6. Inheritance 7. Dynamic binding	3	3	3

	8. Message Passing (List Any 6)			
II. 3	• A Java package is a grouping mechanism that allows developers to organize their classes, interfaces, and sub-packages into a hierarchical structure which helps to improve code readability and maintainability. java.lang, java.io, and java.util, java.net, java.awt(any two)	2 1	3	3
II. 4	**Abstract Class** - The superclass that only defines a generalized form ★ Properties of abstract classes must always extended by all of its subclasses ★ To declare a class abstract, simply use the abstract keyword in front of the class keyword at the beginning of the class ★ An Abstract class cannot be instantiated – objects cannot be created	3	3	3
II. 5	• JComboBox is a Swing component that allows users to select an item from a list of predefined choices. • It is a combination of a text field and a drop-down list that displays the available choices. • getSelectedItem() returns the currently selected item	2 1	3	3
II. 6	• Listener is an object that is notified when an event occurs. • They are used to detect user actions, such as mouse clicks or key presses, and to respond to those actions by executing some code or performing some action. • Event listeners in Java are typically implemented as interfaces that define one or more callback methods that are get called when a specific event occurs. Eg. The ActionListener interface defines actionPerformed() method which receives a notification when an action such as clicking a button takes place.	3	3	3

II. 7	• ActionEvent is an event that is generated by a GUI component, such as a button or menu item, when it is activated by the user. • An ActionEvent object is created and sent to all registered event listeners when the component is activated, such as when the user clicks a button or selects a menu item	3	3	3
II. 8	• A relational database is a type of database that organizes data into one or more tables, where each table consists of a set of rows (also called tuples) and columns. The rows in a table represent individual records, and the columns represent specific attributes or fields of those records.	3	3	3
II. 9	CREATE TABLE statement is used to create a new table eg CREATE TABLE students (id INT PRIMARY KEY, name VARCHAR(50), age INT, gender VARCHAR(10), grade FLOAT(2)); • (Any correct example)	1 1 1	3	3
II. 10	• In Java, you can execute SQL statements using the Statement interface or the PreparedStatement interface. • Once a Connection object is created, you can create statements for executing SQL statements as follows: **Statement statement = connection.createStatement();** • An SQL query statement can be executed using executeQuery(String sql). The result of the query is returned in ResultSet • Eg. ResultSet resultSet = statement.executeQuery	2 1	3	3

	("select name from Student where regno=10001")			
	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.	1. **Private:** The access level of a private modifier is only within the class. It cannot be accessed from outside the class. 2. **Default:** The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If we do not specify any access level, it will be the default. 3. **Protected:** The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package. 4. **Public:** The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.	7	7	7

V.	class BankAccount {
	double balance;
	int accountNumber;
	BankAccount(int accountNumber, double balance) {
	this.accountNumber = accountNumber;
	this.balance = balance;
	}
	void deposit(double amount) {
	balance += amount;
	}
	void withdraw(double amount) {
	if (amount <= balance) {
	balance -= amount;
	} else {
	System.out.println("Insufficient funds");
	}
	}
		7	7	7

```
    double getBalance() {
       return balance;
    }
    int getAccountNumber() {
       return accountNumber;
    }
}
class BankBalance {
        public static void main(String[] args)
        {
        BankAccount b=new BankAccount(1000,10000);
        b.deposit(1000);
        b.withdraw(500);
        System.out.println(b.getBalance());
        }
}
```

OR

VI.

- Feature that allows a class to have more than one method having the same name, but their argument lists are different.
- Method overloading is one of the ways that Java supports polymorphism.
- the argument lists of the methods must differ in either of these:

4

Number of parameters.

add(int, int)

add(int, int, int)

Data type of parameters.

add(int, int)

add(int, float)

7

7

Sequence/order of data type of parameters.

add(int, float)

add(float, int)

eg

```
class overload {
// Overloaded sum().
int sum(int x, int y)
{
```

3

	return (x + y);
	}
	int sum(int x, int y, int z)
	{
	return (x + y + z);
	}
	double sum(double x, double y)
	{
	return (x + y);
	}
	}
	(Any example)			

VII.	class Employee {
	String name;
	double salary;
	Employee(String name, double salary) {
	this.name = name;
	this.salary = salary;
	}
	String getName() {
	return name;
	}
	double getSalary() {
	return salary;
	}
	}
	
	class Manager extends Employee {
	String department;
	Manager(String name, double salary, String department) {
	super(name, salary);
	this.department = department;
	}
		7	7	7

	String getDepartment() {
	return department;
	}
	void printDetails() {
	System.out.println("Name: " + getName());
	System.out.println("Salary: " + getSalary());
	System.out.println("Department: " + department);
	}
	}
	class Employeedemo {
	public static void main(String[] args)
	{
	Manager m=new Manager("xxx",50000,"DEP1");
	m.printDetails();
	}
	}			

	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	3	7	7

	{ ------ } • Multilevel inheritance, on the other hand, refers to a situation where a subclass inherits from a superclass, 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.	**//Shape interface**
	public interface Shape {
	public double calculateArea();
	}
	// Rectangle class that implements Shape interface
	public class Rectangle implements Shape {
	private double length;
	private double width;
	Rectangle() {
	length = 10;
	width = 15;
	}
	//Override
	public double calculateArea() {
	return length * width;
	}
	}			

XI.	GUI Event Types ActionEvent: Occurs when a button or menu item is clicked, or when the user presses the Enter key while focused on a text field. ItemEvent: Occurs when the state of a checkbox, radio button, or combo box selection changes. KeyEvent: Occurs when a keyboard key is pressed or released while a component has focus. MouseEvent: Occurs when the user interacts with the mouse, such as clicking, dragging, or hovering over a component. WindowEvent: Occurs when a window is opened, closed, or resized. (Any three)	Listing 1 2 2 2	7 7	7
	OR			
XII.	**JButton**: JButton is a class in the Java Swing library that represents a button component in a graphical user interface (GUI). • It is used to create clickable buttons that can perform various actions or trigger events when clicked. • an ActionListener is added to the button to handle events when it is clicked. **JList**: JList provides a flexible and customizable way to display a list of items. The items in the list can be of any data type, and the list can be set up to allow single or multiple selections. **JTextField**: JTextField is a class in the Java Swing library that represents a GUI component for allowing users to input text. It provides a text box where the user can enter text and allows developers to retrieve the text entered by the user.	3 2 2	7	7

XIII.	1. **Load the JDBC driver:** the **Class.forName()** method, which loads the driver class into memory. 2. **Establish a connection:** Once the driver is loaded, you can establish a connection to the database using the DriverManager.getConnection() method. This method takes a database URL, username, and password as parameters and returns a Connection object 3. **Create a statement:** Once get a connection to the database, you can create a Statement object using the **createStatement()** method of the Connection object. This Statement object can be used to execute SQL statements against the database. 4. **Execute SQL statements**: You can execute DML statements using the **executeUpdate()** method and use the **executeQuery()** method to execute a query that returns a result set. 5. **Process results:** If the SQL statement returns a result set, you can process the results using a ResultSet object. You can use the **next**() method of the ResultSet object to iterate through the rows of the result set 6. **Close the connection**: Once you are done with the database, you should close the connection using the **close()** method of the Connection object.	7	7	
XIV.	**Insert** The SQL INSERT command is used to insert new data into a database table. The basic syntax of the INSERT command is as follows: INSERT INTO table_name (column1, column2, ...) VALUES (value1, value2, ...); **Select** he SQL SELECT command is used to retrieve data from one or more database tables. The basic syntax of the SELECT command is as follows: SELECT column1, column2, ... FROM table_name WHERE condition(s); **Update** The SQL UPDATE command is used to modify existing data in one or more database tables. The basic syntax of the UPDATE command is as follows:	2 2 3	7	7

	UPDATE table_name SET column1 = value1, column2 = value2, ... WHERE condition(s);			