Practice Questions along with Answer Key

Pattern: 3 Questions

2 questions: 15 marks

1 question: 10 marks

15 marks questions will include concepts from First, fourth and fifth modules.

Principles of OOPS. Data Streams. Swing GUI. Exception Handling, etc..

10 Marks questions will include concepts from Modules 2 and 3.

Strings, Arrays, Loops, etc.

Here are some examples of what can be expected:

15-Mark Practice Questions

Q1 (15 Marks – OOP / Inheritance / Method Overriding)

Problem Statement:
A company wants to manage employee salaries with different policies. Create classes **Employee, Designer, and Tester**.

- Each subclass should calculate salary differently:
    - **Designer:** salary = base + 10% of base
    - **Tester:** salary = base + 5% of base + bonus
- Implement **constructor chaining** and **method overriding**.

Input Format:

- Employee details: id, name, base salary, bonus (if any)

Output Format:

- Display all employee salary details

Expected Concepts:

- Inheritance, Method Overriding, Constructor Chaining, super keyword

Answer Key / Guide:

```
class Employee {
    int id;
    String name;
    double baseSalary;

    Employee(int id, String name, double baseSalary) {
        this.id = id;
        this.name = name;
        this.baseSalary = baseSalary;
    }

    double calculateSalary() {
        return baseSalary;
    }

    void display() {
        System.out.println("ID: " + id + ", Name: " + name + ", Salary: " + calculateSalary());
    }
}

class Designer extends Employee {
    Designer(int id, String name, double baseSalary) {
        super(id, name, baseSalary);
    }

    @Override
    double calculateSalary() {
```

```
        return baseSalary + (0.10 * baseSalary);
    }
}

class Tester extends Employee {
    double bonus;

    Tester(int id, String name, double baseSalary, double bonus) {
        super(id, name, baseSalary);
        this.bonus = bonus;
    }

    @Override
    double calculateSalary() {
        return baseSalary + (0.05 * baseSalary) + bonus;
    }
}

public class Main {
    public static void main(String[] args) {
        Employee e1 = new Designer(101, "Alice", 50000);
        Employee e2 = new Tester(102, "Bob", 40000, 2000);

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

Key points to note:

Constructor chaining via super()

Overriding calculateSalary()

Polymorphic behavior (Employee reference, subclass object)

Q2 (15 Marks – Swing GUI / Event Handling)

Problem Statement:
Create a Swing GUI application that accepts a string. Provide buttons to:

1. Count consonants
2. Display string in lowercase
3. Show string length

Input Format:

- Input string via JTextField

Output Format:

- Display result in JLabel

Answer Key / Guide:

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


public class StringGUI {
   public static void main(String[] args) {
      JFrame frame = new JFrame("String Operations");
      JTextField text = new JTextField();
      text.setBounds(50, 50, 200, 30);


      JLabel result = new JLabel();
```

```
        result.setBounds(50, 150, 300, 30);

        JButton consonants = new JButton("Count Consonants");
        consonants.setBounds(50, 100, 150, 30);

        JButton lowercase = new JButton("Lowercase");
        lowercase.setBounds(210, 100, 100, 30);

        JButton length = new JButton("Length");
        length.setBounds(320, 100, 80, 30);

        consonants.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String s = text.getText();
                int count = 0;
                for (char c : s.toLowerCase().toCharArray()) {
                    if (c >= 'a' && c <= 'z' && "aeiou".indexOf(c) == -1) count++;
                }
                result.setText("Consonants: " + count);
            }
        });

        lowercase.addActionListener(e -> result.setText("Lowercase: " +
text.getText().toLowerCase()));
        length.addActionListener(e -> result.setText("Length: " + text.getText().length()));

        frame.add(text); frame.add(consonants); frame.add(lowercase); frame.add(length);
frame.add(result);
        frame.setSize(500, 300);
```

```
        frame.setLayout(null);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

Key Points to note:

Event Handling using ActionListener

Using JTextField and JLabel

String operations inside button actions

10 Mark questions:

(10 Marks – Array / Loops / Conditionals)

Problem Statement:
Input an array of integers. Display:

- Count of odd numbers
- Sum of first and last element
- Numbers divisible by 3

Answer Key / Guide:

```
import java.util.Scanner;

public class ArrayPractice {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter size: ");
```

```
        int n = sc.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter elements:");
        for(int i=0;i<n;i++) arr[i]=sc.nextInt();

        int oddCount = 0, divisibleBy3 = 0;
        for(int num: arr){
            if(num % 2 != 0) oddCount++;
            if(num % 3 == 0) divisibleBy3++;
        }
        int sumFirstLast = arr[0] + arr[n-1];

        System.out.println("Odd numbers count: " + oddCount);
        System.out.println("Sum of first and last: " + sumFirstLast);
        System.out.println("Numbers divisible by 3: " + divisibleBy3);
    }
}
```

Key Points:

- Array traversal
- Conditional checks
- Loop-based computations

Read the question fully → Identify input/output → Decide approach (OOP, array, string, loops) → Implement → Test with sample input.

Prepare well for VIVA.
ALL THE BEST!