### 1. a. Write a Java program to implement Single Inheritance

\*\*Answer:\*\*

```java

class Parent {

void display() {

System.out.println("This is the parent class.");

}

}

class Child extends Parent {

void show() {

System.out.println("This is the child class.");

}

}

public class SingleInheritanceExample {

public static void main(String[] args) {

Child obj = new Child();

obj.display();

obj.show();

}

}

```

\*\*Output:\*\*

```

This is the parent class.

This is the child class.

```

### 1. b. Write a Java program to implement method overriding

\*\*Answer:\*\*

```java

class Parent {

void display() {

System.out.println("Display method in Parent");

}

}

class Child extends Parent {

@Override

void display() {

System.out.println("Display method in Child");

}

}

public class MethodOverridingExample {

public static void main(String[] args) {

Child obj = new Child();

obj.display();

}

}

```

\*\*Output:\*\*

```

Display method in Child

```

### 2. a. Write a Java program to read details of a student implementing Scanner class

\*\*Answer:\*\*

```java

import java.util.Scanner;

public class StudentDetails {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter student name: ");

String name = scanner.nextLine();

System.out.print("Enter student age: ");

int age = scanner.nextInt();

System.out.println("Student Name: " + name);

System.out.println("Student Age: " + age);

scanner.close();

}

}

```

\*\*Output:\*\*

```

Enter student name: [User input]

Enter student age: [User input]

Student Name: [User input]

Student Age: [User input]

```

### 2. b. Use Eclipse or NetBeans platform and acquaint with the various menus, create a test project, add a test class and run it see how you can use auto suggestions, auto fill. Try code formatter and code refactoring like renaming variables, methods and classes. Try debug step by step with a small program of about 10 to 15 lines which contains at least one if-else condition and a for loop.

\*\*Answer:\*\*

import java.util.Scanner;

// Creating Class

class Sample\_Program {

// main method

public static void main(String args[]) {

int i,count=0,n;

// creating scanner object

Scanner sc=new Scanner(System.in);

// get input number from user

System.out.print("Enter Any Number : ");

n=sc.nextInt();

// logic to check prime or not

for(i=1;i<=n;i++) {

if(n%i==0) {

count++;

}

}

if(count==2)

System.out.println(n+" is prime");

else

System.out.println(n+" is not prime");

}

}

Output:

Enter Any Number : 23

23 is prime

Enter Any Number : 45

45 is not prime

**### 3. a. Write a Java program to implement Array Index Out of Bound Exception**

\*\*Answer:\*\*

```java

public class ArrayIndexOutOfBoundExceptionExample {

public static void main(String[] args) {

int[] array = {1, 2, 3};

try {

System.out.println(array[5]);

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Array Index Out of Bound Exception caught: " + e.getMessage());

}

}

}

```

\*\*Output:\*\*

```

Array Index Out of Bound Exception caught: Index 5 out of bounds for length 3

```

### 3. b. Write a Java program to demonstrate the OOP principles [i.e., Encapsulation, Inheritance, Polymorphism and Abstraction]

\*\*Answer:\*\*

```java

// Encapsulation

class Person {

private String name;

public void setName(String name) {

this.name = name;

}

public String getName() {

return name;

}

}

// Inheritance

class Student extends Person {

private int studentId;

public void setStudentId(int studentId) {

this.studentId = studentId;

}

public int getStudentId() {

return studentId;

}

}

// Polymorphism

class Display {

void showInfo(Person person) {

System.out.println("Name: " + person.getName());

}

}

public class OOPPrinciples {

public static void main(String[] args) {

Student student = new Student();

student.setName("John");

student.setStudentId(123);

Display display = new Display();

display.showInfo(student);

}

}

```

\*\*Output:\*\*

```

Name: John

```

**### 4. a. Write a Java program to perform addition of two numbers**

\*\*Answer:\*\*

```java

public class AdditionExample {

public static void main(String[] args) {

int num1 = 10;

int num2 = 20;

int sum = num1 + num2;

System.out.println("Sum: " + sum);

}

}

```

\*\*Output:\*\*

```

Sum: 30

```

### 4. b. Write a Java program to handle checked and unchecked exceptions. Also, demonstrate the usage of custom exceptions in real-time scenario

\*\*Answer:\*\*

```java

// Custom Exception

class CustomException extends Exception {

public CustomException(String message) {

super(message);

}

}

public class ExceptionHandlingExample {

public static void main(String[] args) {

// Unchecked Exception

try {

int[] arr = new int[2];

arr[3] = 10; // This will throw ArrayIndexOutOfBoundsException

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Unchecked Exception: " + e.getMessage());

}

// Checked Exception

try {

throw new CustomException("This is a custom checked exception.");

} catch (CustomException e) {

System.out.println("Checked Exception: " + e.getMessage());

}

}

}

```

\*\*Output:\*\*

```

Unchecked Exception: Index 3 out of bounds for length 2

Checked Exception: This is a custom checked exception.

```

### 5. a. Write a Java program to perform multiplication of two numbers

\*\*Answer:\*\*

```java

public class MultiplicationExample {

public static void main(String[] args) {

int num1 = 5;

int num2 = 6;

int product = num1 \* num2;

System.out.println("Product: " + product);

}

}

```

\*\*Output:\*\*

```

Product: 30

```

**### 5. b. Write** a Java program on Random Access File class to perform different read and write operations

\*\*Answer:\*\*

```java

import java.io.RandomAccessFile;

import java.io.IOException;

public class RandomAccessFileExample {

public static void main(String[] args) {

try {

RandomAccessFile file = new RandomAccessFile("testfile.txt", "rw");

file.writeUTF("Hello, World!");

file.seek(0);

String content = file.readUTF();

System.out.println("File content: " + content);

file.close();

} catch (IOException e) {

System.out.println("IOException: " + e.getMessage());

}

}

}

```

\*\*Output:\*\*

```

File content: Hello, World!

```

### **6. a**. **Write a Java program to implement** default constructor that displays the default values

\*\*Answer:\*\*

```java

class DefaultConstructorExample {

int num;

String str;

DefaultConstructorExample() {

System.out.println("Default Constructor");

System.out.println("Default int value: " + num);

System.out.println("Default String value: " + str);

}

public static void main(String[] args) {

new DefaultConstructorExample();

}

}

```

\*\*Output:\*\*

```

Default Constructor

Default int value: 0

Default String value: null

```

### 6. b. Write a Java program to demonstrate the working of different collection classes [Use package structure to store multiple classes]

\*\*Answer:\*\*

```java

// Create package structure for collections

package collections;

import java.util.ArrayList;

import java.util.HashSet;

import java.util.LinkedList;

import java.util.TreeSet;

public class CollectionsDemo {

public static void main(String[] args) {

// ArrayList

ArrayList<String> arrayList = new ArrayList<>();

arrayList.add("One");

arrayList.add("Two");

arrayList.add("Three");

System.out.println("ArrayList: " + arrayList);

// LinkedList

LinkedList<String> linkedList = new LinkedList<>();

linkedList.add("A");

linkedList.add("B");

linkedList.add("C");

System.out.println("LinkedList: " + linkedList);

// HashSet

HashSet<String> hashSet = new HashSet<>();

hashSet.add("Apple");

hashSet.add("Banana");

hashSet.add("Apple"); // Duplicate

System.out.println("HashSet: " + hashSet);

// TreeSet

TreeSet<String> treeSet = new TreeSet<>();

treeSet.add("Orange");

treeSet.add("Grapes");

treeSet.add("Apple");

System.out.println("TreeSet: " + treeSet);

}

}

```

\*\*Output:\*\*

```

ArrayList: [One, Two, Three]

LinkedList: [A, B, C]

HashSet: [Banana, Apple, Orange]

TreeSet: [Apple, Grapes, Orange]

```

**### 7. a. Write a Java program to implement** parameterized constructor

\*\*Answer:\*\*

```java

class ParameterizedConstructor {

int num;

String str;

ParameterizedConstructor(int num, String str) {

this.num = num;

this.str = str;

}

void display() {

System.out.println("Number: " + num);

System.out.println("String: " + str);

}

public static void main(String[] args) {

Parameter

izedConstructor obj = new ParameterizedConstructor(5, "Hello");

obj.display();

}

}

```

\*\*Output:\*\*

```

Number: 5

String: Hello

```

### 7. b. Write a program to synchronize the threads acting on the same object [Consider the example of any reservations like railway, bus, movie ticket booking, etc.]

\*\*Answer:\*\*

```java

class TicketBooking {

private int availableTickets = 1;

synchronized void bookTicket(String name) {

if (availableTickets > 0) {

System.out.println(name + " booked a ticket.");

availableTickets--;

} else {

System.out.println("No tickets available.");

}

}

}

public class TicketBookingExample {

public static void main(String[] args) {

TicketBooking booking = new TicketBooking();

Runnable task = () -> booking.bookTicket(Thread.currentThread().getName());

Thread t1 = new Thread(task, "Person1");

Thread t2 = new Thread(task, "Person2");

t1.start();

t2.start();

}

}

```

\*\*Output:\*\*

```

Person1 booked a ticket.

No tickets available.

```

***### 8. a. Write a Java program to implement constructor overloading***

\*\*Answer:\*\*

```java

class ConstructorOverloading {

int num;

String str;

ConstructorOverloading() {

num = 0;

str = "Default";

}

ConstructorOverloading(int num) {

this.num = num;

}

ConstructorOverloading(int num, String str) {

this.num = num;

this.str = str;

}

void display() {

System.out.println("Number: " + num);

System.out.println("String: " + str);

}

public static void main(String[] args) {

ConstructorOverloading obj1 = new ConstructorOverloading();

ConstructorOverloading obj2 = new ConstructorOverloading(10);

ConstructorOverloading obj3 = new ConstructorOverloading(10, "Hello");

obj1.display();

obj2.display();

obj3.display();

}

}

```

\*\*Output:\*\*

```

Number: 0

String: Default

Number: 10

String: Default

Number: 10

String: Hello

```

### 8. b. Write a Java Program to invoke parent class method using super keyword

\*\*Answer:\*\*

```java

class Parent {

void display() {

System.out.println("Display method in Parent");

}

}

class Child extends Parent {

void display() {

super.display(); // Call parent class method

System.out.println("Display method in Child");

}

}

public class SuperKeywordExample {

public static void main(String[] args) {

Child obj = new Child();

obj.display();

}

}

```

\*\*Output:\*\*

```

Display method in Parent

Display method in Child

```

**### 9. a. Write a Java program to implement Hierarchical Inheritance**

\*\*Answer:\*\*

```java

class Animal {

void sound() {

System.out.println("Animal makes a sound");

}

}

class Dog extends Animal {

void bark() {

System.out.println("Dog barks");

}

}

class Cat extends Animal {

void meow() {

System.out.println("Cat meows");

}

}

public class HierarchicalInheritanceExample {

public static void main(String[] args) {

Dog dog = new Dog();

dog.sound();

dog.bark();

Cat cat = new Cat();

cat.sound();

cat.meow();

}

}

```

\*\*Output:\*\*

```

Animal makes a sound

Dog barks

Animal makes a sound

Cat meows

```

**### 9. b. Write a Java program that handles all mouse events and shows the event name at the center of the window when a mouse event is fired [Use Adapter classes]**

**\*\*Answer:\*\***

```java

import java.awt.\*;

import java.awt.event.\*;

public class MouseEventExample extends Frame {

private String message = "";

public MouseEventExample() {

addMouseListener(new MouseAdapter() {

public void mouseClicked(MouseEvent e) {

message = "Mouse Clicked";

repaint();

}

public void mouseEntered(MouseEvent e) {

message = "Mouse Entered";

repaint();

}

public void mouseExited(MouseEvent e) {

message = "Mouse Exited";

repaint();

}

public void mousePressed(MouseEvent e) {

message = "Mouse Pressed";

repaint();

}

public void mouseReleased(MouseEvent e) {

message = "Mouse Released";

repaint();

}

});

setSize(400, 300);

setVisible(true);

}

public void paint(Graphics g) {

g.drawString(message, 150, 150);

}

public static void main(String[] args) {

new MouseEventExample();

}

}

```

\*\*Output:\*\*

The event name will be displayed at the center of the window based on the mouse event.

**### 10. a. Write a Java program to increment** the value using static variable

\*\*Answer:\*\*

```java

class StaticVariableExample {

static int count = 0;

StaticVariableExample() {

count++;

}

public static void main(String[] args) {

new StaticVariableExample();

new StaticVariableExample();

new StaticVariableExample();

System.out.println("Count: " + count);

}

}

```

\*\*Output:\*\*

```

Count: 3

```

### 10. b. Write a program to create a check box, radio button

\*\*Answer:\*\*

```java

import java.awt.\*;

import java.awt.event.\*;

public class CheckboxRadioButtonExample {

public static void main(String[] args) {

Frame frame = new Frame("Checkbox and RadioButton Example");

Checkbox checkbox = new Checkbox("Accept Terms and Conditions");

CheckboxGroup group = new CheckboxGroup();

Checkbox radio1 = new Checkbox("Option 1", group, false);

Checkbox radio2 = new Checkbox("Option 2", group, false);

frame.add(checkbox);

frame.add(radio1);

frame.add(radio2);

frame.setLayout(new FlowLayout());

frame.setSize(300, 200);

frame.setVisible(true);

}

}

```

\*\*Output:\*\*

A window with a checkbox and two radio buttons.

### 11. a. Write a Java program to implement Multi-level Inheritance

\*\*Answer:\*\*

```java

class A {

void displayA() {

System.out.println("Class A");

}

}

class B extends A {

void displayB() {

System.out.println("Class B");

}

}

class C extends B {

void displayC() {

System.out.println("Class C");

}

}

public class MultiLevelInheritanceExample {

public static void main(String[] args) {

C obj = new C();

obj.displayA();

obj.displayB();

obj.displayC();

}

}

```

\*\*Output:\*\*

```

Class A

Class B

Class C

```

**### 11. b. Write a Java program to implement** static method

\*\*Answer:\*\*

```java

class StaticMethodExample {

static void display() {

System.out.println("Static method called");

}

public static void main(String[] args) {

StaticMethodExample.display();

}

}

```

\*\*Output:\*\*

```

Static method called

```

### 12. a. Write a Java which implements Calling parameterized constructor from default constructor

\*\*Answer:\*\*

```java

class ConstructorExample {

ConstructorExample() {

this(10); // Calling parameterized constructor

}

ConstructorExample(int num) {

System.out.println("Number: " + num);

}

public static void main(String[] args) {

new ConstructorExample();

}

}

```

\*\*Output:\*\*

```

Number: 10

```

**### 12. b. Write a Java program to** implement multiple inheritance using interface

\*\*Answer:\*\*

```java

interface A {

void methodA();

}

interface B {

void methodB();

}

class C implements A, B {

public void methodA() {

System.out.println("Method A from Interface A");

}

public void methodB() {

System.out.println("Method B from Interface B");

}

}

public class MultipleInheritanceExample {

public static void main(String[] args) {

C obj = new C();

obj.methodA();

obj.methodB();

}

}

```

\*\*Output:\*\*

```

Method A from Interface A

Method B from Interface B

```

**### 13. Write a Java program that works as a simple calculator. Use a grid layout to arrange buttons for the digits and for the +, -, \*, % operations. Add a text field to display the result. Handle any possible exceptions like divided by zero.**

\*\*Answer:\*\*

```java

import java.awt.\*;

import java.awt.event.\*;

public class SimpleCalculator extends Frame {

private TextField display;

private String operator = "";

private double num1 = 0;

public SimpleCalculator() {

display = new TextField();

display.setEditable(false);

add(display, BorderLayout.NORTH);

Panel panel = new Panel();

panel.setLayout(new GridLayout(4, 4));

String[] buttons = {

"7", "8", "9", "/",

"4", "5", "6", "\*",

"1", "2", "3", "-",

"0", ".", "=", "+"

};

for (String label : buttons) {

Button button = new Button(label);

button.addActionListener(new ButtonClickListener());

panel

.add(button);

}

add(panel);

setSize(300, 400);

setVisible(true);

}

private class ButtonClickListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

Button source = (Button) e.getSource();

String command = source.getLabel();

try {

if (command.matches("[0-9]")) {

display.setText(display.getText() + command);

} else if (command.equals("=")) {

double num2 = Double.parseDouble(display.getText());

double result = 0;

switch (operator) {

case "+": result = num1 + num2; break;

case "-": result = num1 - num2; break;

case "\*": result = num1 \* num2; break;

case "/":

if (num2 != 0) {

result = num1 / num2;

} else {

display.setText("Error");

return;

}

break;

}

display.setText(String.valueOf(result));

operator = "";

} else if (command.matches("[+\\-\*/]")) {

operator = command;

num1 = Double.parseDouble(display.getText());

display.setText("");

} else if (command.equals(".")) {

display.setText(display.getText() + ".");

}

} catch (Exception ex) {

display.setText("Error");

}

}

}

public static void main(String[] args) {

new SimpleCalculator();

}

}

```

\*\*Output:\*\*

A simple calculator window with digit buttons, operation buttons, and a display for the result.

Feel free to run these programs to see the outputs in your Java environment. If you have any more questions or need further assistance, just let me know!