**Lab Sheet 01**

**01.**

public class Item {

protected int location;

protected String description;

public Item(int location, String description) {

this.location = location;

this.description = description;

}

protected int getLocation() {

return location;

}

protected void setLocation(int location) {

this.location = location;

}

protected String getDescription() {

return description;

}

protected void setDescription(String description) {

this.description = description;

}

**02.**

01. super

02. private

03. packages

04. import pkg\*

05. charAt()

06. length()

**03.**

01. attributes and behavior

02. instance variable s

03. methods

04. encapsulation

05. class

06. base class, derived class, extends

07. interface

08. package

09. application programming interface

**Lab sheet 03**

public class Employee {

private String name;

private int age;

private double salary;

public Employee(String name, int age, double salary) {

this.name = name;

this.age = age;

this.salary = salary;

}

public String getName() {

return name;

}

public int getAge() {

return age;

}

public double getSalary() {

return salary;

}

public void setName(String name) {

this.name = name;

}

public void setAge(int age) {

this.age = age;

}

public void setSalary(double salary) {

this.salary = salary;

}

}

**02**

public class Employee {

private String name;

private double basicSalary;

private double bonus;

public Employee(String name, double basicSalary, double bonus) {

this.name = name;

this.basicSalary = basicSalary;

this.bonus = bonus;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getBasicSalary() {

return basicSalary;

}

public void setBasicSalary(double basicSalary) {

this.basicSalary = basicSalary;

}

public double getBonus() {

return bonus;

}

public double calculateBonusAmount() {

return basicSalary + bonus;

}

}

public class TestEmployee {

public static void main(String[] args) {

Employee emp = new Employee("John Doe", 50000.0, 10000.0);

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

System.out.println("Basic Salary: " + emp.getBasicSalary());

System.out.println("Bonus: " + emp.getBonus());

System.out.println("Bonus Amount: " + emp.calculateBonusAmount());

}

}

**Output**

Employee Name: John Doe

Basic Salary: 50000.0

Bonus: 10000.0

Bonus Amount: 60000.0

**Lab sheet 04**

**01**

class Employee {

private int empID;

private String empName;

private String empDesignation;

public int getEmpID() {

return empID;

}

public void setEmpID(int empID) {

this.empID = empID;

}

public String getEmpName() {

return empName;

}

public void setEmpName(String empName) {

this.empName = empName;

}

public String getEmpDesignation() {

return empDesignation;

}

public void setEmpDesignation(String empDesignation) {

this.empDesignation = empDesignation;

}

}

public class TestEmployee {

public static void main(String[] args) {

Employee mrBogdan = new Employee();

Employee msBird = new Employee();

mrBogdan.setEmpID(1);

mrBogdan.setEmpName("Mr. Bogdan");

mrBogdan.setEmpDesignation("Software Engineer");

msBird.setEmpID(2);

msBird.setEmpName("Ms. Bird");

msBird.setEmpDesignation("Product Manager");

System.out.println("Mr. Bogdan Details:");

System.out.println("Employee ID: " + mrBogdan.getEmpID());

System.out.println("Employee Name: " + mrBogdan.getEmpName());

System.out.println("Employee Designation: " + mrBogdan.getEmpDesignation());

System.out.println("\nMs. Bird Details:");

System.out.println("Employee ID: " + msBird.getEmpID());

System.out.println("Employee Name: " + msBird.getEmpName());

System.out.println("Employee Designation: " + msBird.getEmpDesignation());

}

}

**02**

Output

Value of x: 9

**Lab sheet 05**

**01 .**

**02.**

**Output**

false

true

true

**Lab sheet 06**

**01**

public class InterfaceImplemented implements MyFirstInterface {

public static void main(String[] args) {

InterfaceImplemented obj = new InterfaceImplemented();

System.out.println("Value of x before changing: " + obj.x);

obj.display();

System.out.println("Value of x after changing: " + obj.x);

}

public void display() {

x = 20; // Trying to change the value of x

System.out.println("Value of x inside display method: " + x);

}

}

**02**

**Lab sheet 07**

**01**

The outcome would be that the provided code will result in compilation error due to the issues , after correcting the issues the code should compile successfully and you can create objects of the students class access the mark variable and call the display() method. However you wont be able to subclass and override the display () method since it is declared as final.

**02**

**Lab sheet 08**

**01**

**Bank account**

public abstract class BankAccount {

private String accountNumber;

private double balance;

public String getAccountNumber() {

return accountNumber;

}

public void setAccountNumber(String accountNumber) {

this.accountNumber = accountNumber;

}

// Getter for balance

public double getBalance() {

return balance;

}

public void setBalance(double balance) {

this.balance = balance;

}

public abstract double calculateInterest();

}

**Saving account**

public class SavingsAccount extends BankAccount {

private final double interestRate = 0.12; // 12% interest rate for SavingsAccount

public double calculateInterest() {

return getBalance() \* interestRate;

}

}

public class Main {

public static void main(String[] args) {

CheckingAccount checkingAccount = new CheckingAccount();

SavingsAccount savingsAccount = new SavingsAccount();

checkingAccount.setBalance(1000000); // 1 million

savingsAccount.setBalance(20000000); // 20 million

double checkingInterest = checkingAccount.calculateInterest();

double savingsInterest = savingsAccount.calculateInterest();

System.out.println("Interest on Checking Account: " + checkingInterest);

System.out.println("Interest on Savings Account: " + savingsInterest);

}

}

**Output**

Interest on Checking Account: 20000.0

Interest on Savings Account: 2400000.0

**02 .**

**Shape**

public interface Shape {

double calculateArea();

double calculatePerimeter();

}

**Circle**

public class Circle implements Shape {

private double radius;

public Circle(double radius) {

this.radius = radius;

}

public double getRadius() {

return radius;

}

public void setRadius(double radius) {

this.radius = radius;

}

public double calculateArea() {

return Math.PI \* radius \* radius;

}

public double calculatePerimeter() {

return 2 \* Math.PI \* radius;

}

}

**Rectangle**

public class Rectangle implements Shape {

private double length;

private double width;

public Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

public double getLength() {

return length;

}

public double getWidth() {

return width;

}

public void setLength(double length) {

this.length = length;

}

public void setWidth(double width) {

this.width = width;

}

public double calculateArea() {

return length \* width;

}

public double calculatePerimeter() {

return 2 \* (length + width);

}

}

**Triangle**

public class Triangle implements Shape {

private double sideA;

private double sideB;

private double sideC;

public Triangle(double sideA, double sideB, double sideC) {

this.sideA = sideA;

this.sideB = sideB;

this.sideC = sideC;

}

public double getSideA() {

return sideA;

}

public double getSideB() {

return sideB;

}

public double getSideC() {

return sideC;

}

public void setSideA(double sideA) {

this.sideA = sideA;

}

public void setSideB(double sideB) {

this.sideB = sideB;

}

public void setSideC(double sideC) {

this.sideC = sideC;

}

public double calculateArea() {

// Using Heron's formula to calculate the area of a triangle

double s = (sideA + sideB + sideC) / 2.0;

return Math.sqrt(s \* (s - sideA) \* (s - sideB) \* (s - sideC));

}

public double calculatePerimeter() {

return sideA + sideB + sideC;

}

}

**Lab sheet 09**

**01.**

**02.**

public class MoveUpController implements PlayerController {

@Override

public void moveUp() {

System.out.println("Moving Up");

}

public void moveDown() {

// No implementation needed for this direction

}

public void moveLeft() {

// No implementation needed for this direction

}

public void moveRight() {

// No implementation needed for this direction

}

}

public class MoveDownController implements PlayerController {

@Override

public void moveUp() {

// No implementation needed for this direction

}

public void moveDown() {

System.out.println("Moving Down");

}

public void moveLeft() {

}

public void moveRight() {

}

}

public class MoveLeftController implements PlayerController {

@Override

public void moveUp() {

// No implementation needed for this direction

}

public void moveDown() {

// No implementation needed for this direction

}

public void moveLeft() {

System.out.println("Moving Left");

}

public void moveRight() {

// No implementation needed for this direction

}

}

public class MoveRightController implements PlayerController {

@Override

public void moveUp() {

// No implementation needed for this direction

}

public void moveDown() {

// No implementation needed for this direction

}

public void moveLeft() {

// No implementation needed for this direction

}

public void moveRight() {

System.out.println("Moving Right");

**}**

**}**

**Lab sheet 10**

**01**

import java.util.Scanner;

public class DivisionProgram {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the numerator: ");

int numerator = scanner.nextInt();

System.out.print("Enter the denominator: ");

int denominator = scanner.nextInt();

try {

int result = divide(numerator, denominator);

System.out.println("Result of division: " + result);

} catch (ArithmeticException e) {

System.out.println("Error: Division by zero is not allowed.");

}

}

public static int divide(int numerator, int denominator) throws ArithmeticException {

if (denominator == 0) {

throw new ArithmeticException("Division by zero is not allowed.");

}

return numerator / denominator;

}

}

**02**

import java.util.Arrays;

import java.util.Scanner;

public class ArrayAccessProgram {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the size of the array: ");

int size = scanner.nextInt();

int[] array = new int[size];

System.out.println("Enter " + size + " elements for the array:");

for (int i = 0; i < size; i++) {

array[i] = scanner.nextInt();

}

System.out.print("Enter the index to access: ");

int index = scanner.nextInt();

try {

int element = array[index];

System.out.println("Element at index " + index + ": " + element);

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Error: Invalid index. Please enter an index between 0 and " + (size - 1) + ".");

}

}

}

**03**

public class FileReadProgram {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the file path: ");

String filePath = scanner.nextLine();

try {

File file = new File(filePath);

Scanner fileScanner = new Scanner(file);

while (fileScanner.hasNextLine()) {

String line = fileScanner.nextLine();

System.out.println(line);

}

fileScanner.close();

} catch (FileNotFoundException e) {

System.out.println("Error: File not found. Please check the file path and try again.");

}

}

}

**Lab sheet 11**

**Lab sheet 12**

**01**

CREATE TABLE Employee (

ID INT PRIMARY KEY,

Name VARCHAR(50),

Age INT,

Gender VARCHAR(10),

DepartmentNo INT

);

**02**

public class EmployeeDataEntryUI extends JFrame {

private JTextField idField;

private JTextField nameField;

private JTextField ageField;

private JTextField genderField;

private JTextField deptNoField;

public EmployeeDataEntryUI() {

setTitle("Employee Data Entry");

setSize(400, 250);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new GridLayout(6, 2, 10, 10));

JLabel idLabel = new JLabel("ID:");

idField = new JTextField(10);

JLabel nameLabel = new JLabel("Name:");

nameField = new JTextField(50);

JLabel ageLabel = new JLabel("Age:");

ageField = new JTextField(10);

JLabel genderLabel = new JLabel("Gender:");

genderField = new JTextField(10);

JLabel deptNoLabel = new JLabel("Department No.:");

deptNoField = new JTextField(10);

JButton insertButton = new JButton("Insert");

insertButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

insertEmployeeData();

}

});

add(idLabel);

add(idField);

add(nameLabel);

add(nameField);

add(ageLabel);

add(ageField);

add(genderLabel);

add(genderField);

add(deptNoLabel);

add(deptNoField);

add(insertButton);

setVisible(true);

}

private void insertEmployeeData() {

int id = Integer.parseInt(idField.getText());

String name = nameField.getText();

int age = Integer.parseInt(ageField.getText());

String gender = genderField.getText();

int deptNo = Integer.parseInt(deptNoField.getText());

try {

Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/your\_database\_name", "your\_username", "your\_password");

Statement statement = connection.createStatement();

String query = "INSERT INTO Employee (ID, Name, Age, Gender, DepartmentNo) VALUES (" + id + ", '" + name + "', " + age + ", '" + gender + "', " + deptNo + ")";

statement.executeUpdate(query);

statement.close();

connection.close();

JOptionPane.showMessageDialog(this, "Employee data inserted successfully!");

clearFields();

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Error: Failed to insert employee data.");

}

}

private void clearFields() {

idField.setText("");

nameField.setText("");

ageField.setText("");

genderField.setText("");

deptNoField.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

@Override

public void run() {

new EmployeeDataEntryUI();

}

});

}

}

**03**