**LAB – 4**

1. Method Overloading: Write a class Calculator with overloaded methods add(). Implement add() methods that take:

- Two integers

- Two double values

- Three integers

- A variable number of integers

package anudip;

public class Calculator {

// Method to add two integers

public int add(int a, int b) {

return a + b;

}

// Method to add two double values

public double add(double a, double b) {

return a + b;

}

// Method to add three integers

public int add(int a, int b, int c) {

return a + b + c;

}

// Method to add a variable number of integers

public int add(int... numbers) {

int sum = 0;

for (int number : numbers) {

sum += number;

}

return sum;

}

public static void main(String[] args) {

Calculator calc = new Calculator();

// Testing different add methods

System.out.println(calc.add(1, 2)); // Two integers

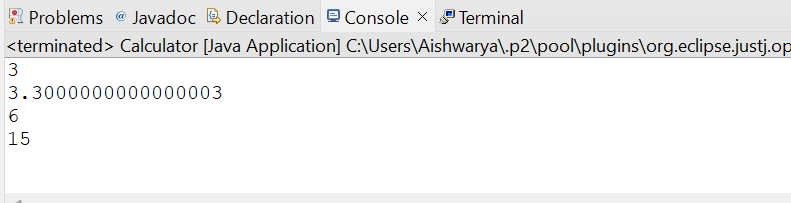
System.out.println(calc.add(1.1, 2.2)); // Two double values

System.out.println(calc.add(1, 2, 3)); // Three integers

System.out.println(calc.add(1, 2, 3, 4, 5)); // Variable number of integers

}

}



2. Super Keyword: Create a class Person with a constructor that accepts and sets name and age.

- Create a subclass Student that adds a grade property and initializes name and age using the super keyword in its constructor.

- Demonstrate the creation of Student objects and the usage of super to call the parent class constructor.

package anudip;

class Person {

private String name;

private int age;

// Constructor

public Person(String name, int age) {

this.name = name;

this.age = age;

}

// Method to display information

public void displayInfo() {

System.out.println("Name: " + name + ", Age: " + age);

}

// Getters

public String getName() {

return name;

}

public int getAge() {

return age;

}

}

class Student extends Person {

private String grade;

// Constructor

public Student(String name, int age, String grade) {

// Call the constructor of the Person class

super(name, age);

this.grade = grade;

}

// Method to display information

@Override

public void displayInfo() {

// Call the displayInfo method of the Person class

super.displayInfo();

System.out.println("Grade: " + grade);

}

// Getter

public String getGrade() {

return grade;

}

}

public class Main {

public static void main(String[] args) {

// Demonstrate the creation of Student objects

Student student1 = new Student("Alice", 20, "A");

Student student2 = new Student("Bob", 22, "B");

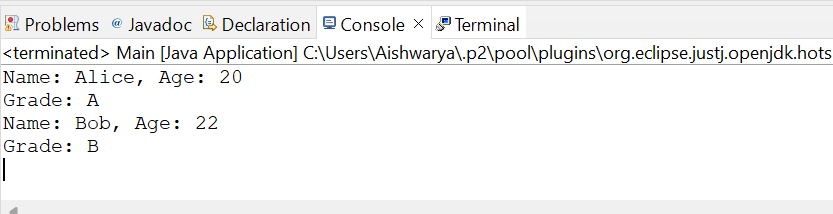
// Display information for the students

student1.displayInfo();

student2.displayInfo();

}

}



3.Super Keyword: Create a base class Shape with a method draw() that prints "Drawing Shape".

- Create a subclass Circle that overrides draw() to print "Drawing Circle".

- Inside the draw() method of Circle, call the draw() method of the Shape class using super.draw().

- Write a main method to demonstrate calling draw() on a Circle object.

package anudip;

class Shape {

// Method to draw a shape

public void draw() {

System.out.println("Drawing Shape");

}

}

class Circle extends Shape {

// Method to draw a circle

@Override

public void draw() {

// Call the draw method of the Shape class

super.draw();

System.out.println("Drawing Circle");

}

}

public class Main {

public static void main(String[] args) {

// Create a Circle object

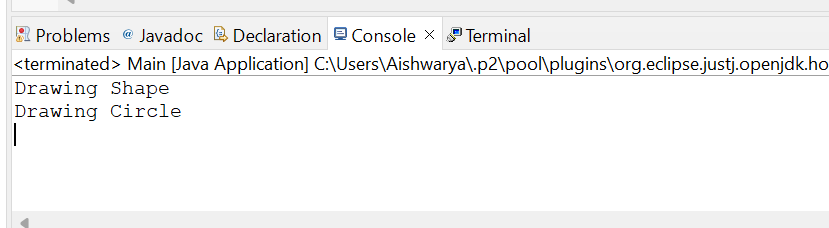
Circle circle = new Circle();

// Call the draw method on the Circle object

circle.draw();

}

}



4. Create a base class BankAccount with a method deposit(amount) and a constructor that sets the initial balance.

- Create a subclass SavingsAccount that overrides deposit(amount) to add interest before depositing. Use the super keyword to call the deposit method of the base class.

- Write a main method to demonstrate creating a SavingsAccount and depositing an amount to see the effect of interest

package anudip;

class BankAccount {

protected double balance;

// Constructor to set the initial balance

public BankAccount(double initialBalance) {

this.balance = initialBalance;

}

// Method to deposit an amount

public void deposit(double amount) {

balance += amount;

System.out.println("Deposited: " + amount);

System.out.println("New Balance: " + balance);

}

// Getter for the balance

public double getBalance() {

return balance;

}

}

class SavingsAccount extends BankAccount {

private double interestRate;

// Constructor to set the initial balance and interest rate

public SavingsAccount(double initialBalance, double interestRate) {

super(initialBalance);

this.interestRate = interestRate;

}

// Overridden method to deposit an amount with interest

@Override

public void deposit(double amount) {

double interest = amount \* interestRate / 100;

double totalAmount = amount + interest;

super.deposit(totalAmount);

System.out.println("Interest added: " + interest);

}

}

public class Main {

public static void main(String[] args) {

// Create a SavingsAccount object

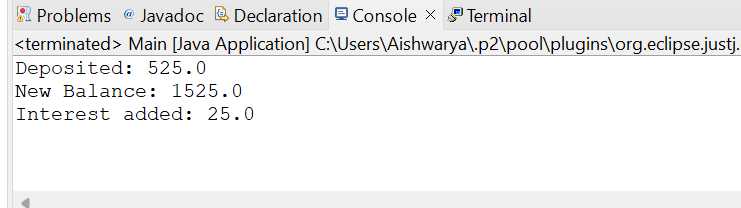
SavingsAccount savingsAccount = new SavingsAccount(1000.0, 5.0);

// Deposit an amount to see the effect of interest

savingsAccount.deposit(500.0);

}

}



5. Define a class Employee with properties name and salary and a method displayDetails().

- Create a subclass Manager that adds a property department and overrides displayDetails() to include department details. Use the super keyword to call the displayDetails() method of Employee within Manager.

- In the main method, create objects of Employee and Manager and call displayDetails() to show the details.

class Employee {

protected String name;

protected double salary;

// Constructor to set name and salary

public Employee(String name, double salary) {

this.name = name;

this.salary = salary;

}

// Method to display details

public void displayDetails() {

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

System.out.println("Salary: " + salary);

}

}

class Manager extends Employee {

private String department;

// Constructor to set name, salary, and department

public Manager(String name, double salary, String department) {

super(name, salary);

this.department = department;

}

// Overridden method to display details including department

@Override

public void displayDetails() {

super.displayDetails(); // Call the displayDetails method of Employee

System.out.println("Department: " + department);

}

}

public class Main {

public static void main(String[] args) {

// Create an Employee object

Employee employee = new Employee("Raj Thakur", 50000);

// Create a Manager object

Manager manager = new Manager("Jiya Sharma", 75000, "IT");

// Display details of Employee

System.out.println("Employee Details:");

employee.displayDetails();

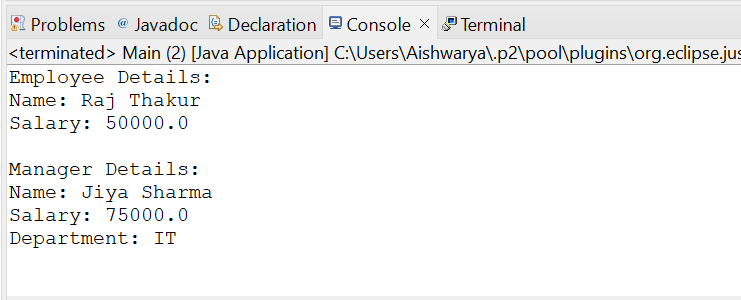
// Display details of Manager

System.out.println("\nManager Details:");

manager.displayDetails();

}

}



.

6.Write the same programme for the class ImmutableExample, to achieve object value ‘Hi

package anudip;

final class ImmutableExample {

private final String value;

// Constructor to set the value

public ImmutableExample() {

this.value = "Hi";

}

// Method to get the value

public String getValue() {

return value;

}

// No setter methods as the class is immutable

}

public class Main {

public static void main(String[] args) {

// Create an instance of ImmutableExample

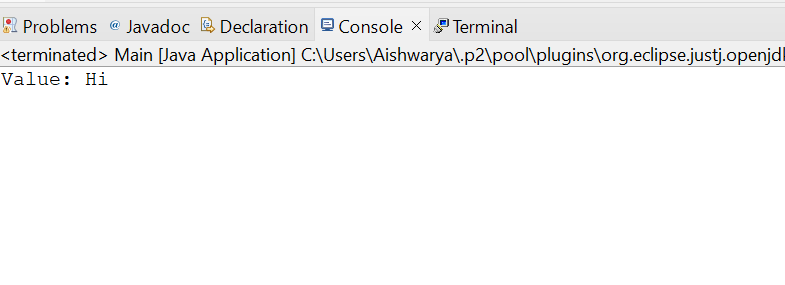
ImmutableExample example = new ImmutableExample();

// Display the value

System.out.println("Value: " + example.getValue());

}

}



.7.. Define a class Employee with properties name and salary and a method displayDetails().

- Create a subclass Manager that adds a property department and overrides displayDetails() to include department details. Use the super keyword to call the displayDetails() method of Employee within Manager.

- In the main method, create objects of Employee and Manager and call displayDetails() to show the details.

package anudip;

class MutableExample {

private String value;

// Constructor to set the initial value

public MutableExample(String value) {

this.value = value;

}

// Getter method to get the value

public String getValue() {

return value;

}

// Setter method to set the value

public void setValue(String value) {

this.value = value;

}

}

public class Main {

public static void main(String[] args) {

// Create an instance of MutableExample with initial value "hello 2"

MutableExample example1 = new MutableExample("hello 2");

// Display the value

System.out.println("Value of example1: " + example1.getValue());

// Modify the value to "hello 3"

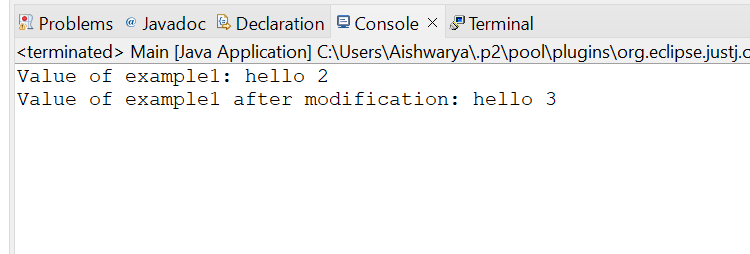
example1.setValue("hello 3");

// Display the modified value

System.out.println("Value of example1 after modification: " + example1.getValue());

}

}



8.Write the same programme for the class ImmutableExample, to achieve object value ‘Hi

package anudip;

public class StringMethodsImplementation {

public static void main(String[] args) {

String str = "Hello World";

// Replace method

System.out.println("Replace method: " + str.replace('o', 'x'));

// Contains method

System.out.println("Contains method: " + str.contains("Hello"));

// ReplaceAll method

System.out.println("ReplaceAll method: " + str.replaceAll("Hello", "Hi"));

// IndexOf method

System.out.println("IndexOf method: " + str.indexOf("o"));

// Substring method

System.out.println("Substring method: " + str.substring(6));

// Equals method

System.out.println("Equals method: " + str.equals("Hello World"));

// LastIndexOf method

System.out.println("LastIndexOf method: " + str.lastIndexOf("o"));

// StartsWith method

System.out.println("StartsWith method: " + str.startsWith("Hello"));

// EndsWith method

System.out.println("EndsWith method: " + str.endsWith("World"));

// EqualsIgnoreCase method

System.out.println("EqualsIgnoreCase method: " + str.equalsIgnoreCase("hello world"));

// ToLowerCase method

System.out.println("ToLowerCase method: " + str.toLowerCase());

// ToUpperCase method

System.out.println("ToUpperCase method: " + str.toUpperCase());

// IsEmpty method

System.out.println("IsEmpty method: " + str.isEmpty());

// Length method

System.out.println("Length method: " + str.length());

// Split method

String[] parts = str.split(" ");

System.out.print("Split method: ");

for (String part : parts) {

System.out.print(part + ", ");

}

}