**Lab4**

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

     - Two integers

     - Two double values

     - Three integer

- A variable number of integers

**Code-**

package demo1;

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 (using varargs)

public int add(int... numbers) {

int sum = 0;

for (int number : numbers) {

sum += number;

}

return sum;

}

public static void main(String[] args) {

Calculator calculator = new Calculator();

// Testing the overloaded add methods

System.***out***.println("Sum of 2 integers (3, 4): " + calculator.add(3, 4));

System.***out***.println("Sum of 2 double values (3.5, 4.5): " + calculator.add(3.5, 4.5));

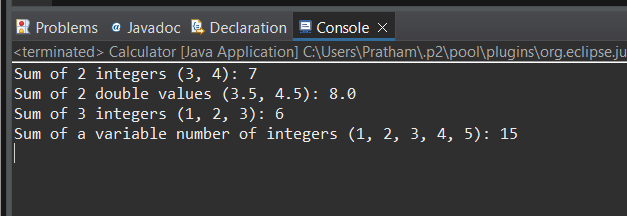
System.***out***.println("Sum of 3 integers (1, 2, 3): " + calculator.add(1, 2, 3));

System.***out***.println("Sum of a variable number of integers (1, 2, 3, 4, 5): " + calculator.add(1, 2, 3, 4, 5));

}

}

**Output-**



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.

**Code**-

package demo1;

class Person2 {

private String name;

private int age;

// Constructor to initialize name and age

public Person2(String name, int age) {

this.name = name;

this.age = age;

}

// Public getter methods

public String getName() {

return name;

}

public int getAge() {

return age;

}

}

class Students extends Person2 {

private String grade;

// Constructor to initialize name, age, and grade

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

// Call the parent class constructor to initialize name and age

super(name, age);

this.grade = grade;

}

// Public getter method for grade

public String getGrade() {

return grade;

}

}

public class Main21 {

public static void main(String[] args) {

// Create Student objects

Students student1 = new Students("Pratham", 21, "A");

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

// Display the details of the students

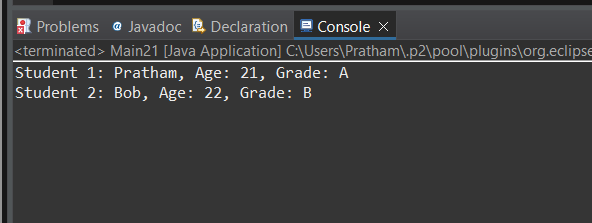
System.***out***.println("Student 1: " + student1.getName() + ", Age: " + student1.getAge() + ", Grade: " + student1.getGrade());

System.***out***.println("Student 2: " + student2.getName() + ", Age: " + student2.getAge() + ", Grade: " + student2.getGrade());

}

}

Output-



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.

**Code-**

package demo1;

// Shape.java

class Shape {

public void draw() {

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

}

}

// Circle.java

class Circle extends Shape {

*@Override*

public void draw() {

super.draw(); // Call the draw method of Shape

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

}

}

// Main.java

public class Main2 {

public static void main(String[] args) {

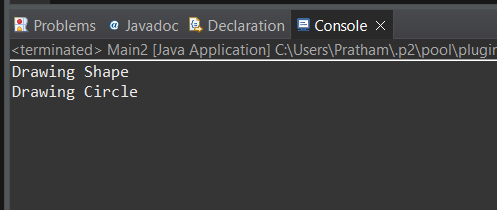
Circle circle = new Circle();

circle.draw();

}

}

**Output-**

****

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.

**Code-**

package demo1;

// BankAccount.java

class BankAccount {

protected double balance;

public BankAccount(double initialBalance) {

this.balance = initialBalance;

}

public void deposit(double amount) {

balance += amount;

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

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

}

public double getBalance() {

return balance;

}

}

// SavingsAccount.java

class SavingsAccount extends BankAccount {

private double interestRate;

public SavingsAccount(double initialBalance, double interestRate) {

super(initialBalance);

this.interestRate = interestRate;

}

*@Override*

public void deposit(double amount) {

double interest = amount \* interestRate / 100;

double totalAmount = amount + interest;

super.deposit(totalAmount);

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

}

}

// Main.java

public class Main2 {

public static void main(String[] args) {

SavingsAccount savingsAccount = new SavingsAccount(1000.0, 5.0); // Initial balance of 1000 and interest rate of 5%

System.***out***.println("Initial balance: " + savingsAccount.getBalance());

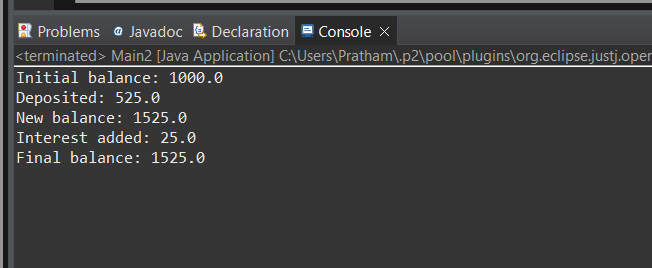
savingsAccount.deposit(500.0); // Deposit 500

System.***out***.println("Final balance: " + savingsAccount.getBalance());

}

}

**Output-**

****

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.

**Code-**

package demo1;

class Employee1 {

private String name;

private double salary;

public Employee1(String name, double salary) {

this.name = name;

this.salary = salary;

}

// Getter for name

public String getName() {

return name;

}

// Getter for salary

public double getSalary() {

return salary;

}

// Method to display employee details

public void displayDetails() {

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

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

}

}

class Manager extends Employee1 {

private String department;

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

super(name, salary);

this.department = department;

}

// Getter for department

public String getDepartment() {

return department;

}

*@Override*

public void displayDetails() {

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

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

}

}

public class Main2 {

public static void main(String[] args) {

// Create an Employee object

Employee1 emp = new Employee1("Prachit", 50000);

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

emp.displayDetails();

// Create a Manager object

Manager mgr = new Manager("Pratham", 75000, "HR");

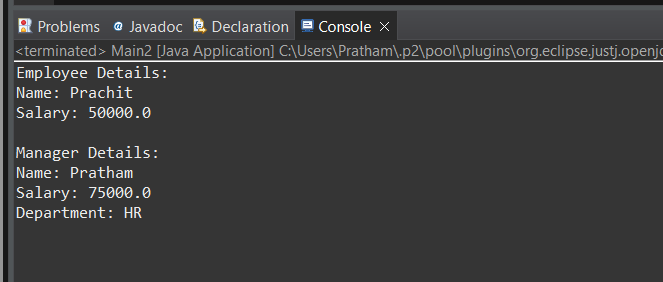
System.***out***.println("\nManager Details:");

mgr.displayDetails();

}

}

**Output-**

****

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

**Code-**

package demo1;

public final class Immutabledemo {

private String name;

Immutabledemo (String name) {

this.name = name;

}

public String getName() {

return name;

}

public static void main(String[] args) {

Immutabledemo obj = new Immutabledemo("hi");

System.***out***.println(obj.getName());

obj.setName("new hi");

System.***out***.println(obj.getName());

}

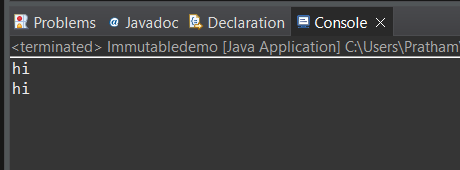
private void setName(String string) {

// **TODO** Auto-generated method stub

}

}

Output-



7. Write the same programme for the class MutableExample, to output the object values ‘hello 2’ and ‘hello3’.

Code-

package demo1;

public class Mutabledemo {

private String name;

Mutabledemo(String name) {

this.name = name;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public static void main(String[] args) {

Mutabledemo obj = new Mutabledemo("hello2");

System.***out***.println(obj.getName());

obj.setName("new hello3");

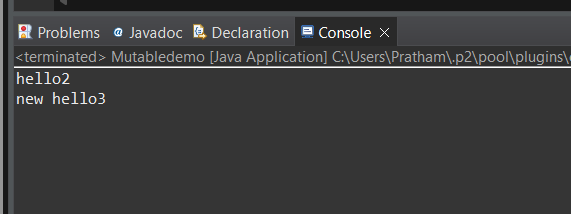
System.***out***.println(obj.getName());

}

}

}

Output-



8.  Write a java class to implement any 10 string methods:

● replace ● contains ● replaceAll ● indexOf ● substring ● Equals ● lastIndexOf ● startsWith

● endsWith ● EqualsIgnoreCase ● toLowerCase ● toUpperCase ● isEmpty ● Length ● split

**Code-**

package demo1;

public class StringMethodsDemo {

public static void main(String[] args) {

String str = "Hello, World!";

// replace

String replacedStr = str.replace('o', 'a');

System.***out***.println("replace: " + replacedStr); // Hella, Warld!

// replaceAll

String replacedAllStr = str.replaceAll("o", "a");

System.***out***.println("replaceAll: " + replacedAllStr); // Hella, Warld!

// indexOf

int indexOfStr = str.indexOf('o');

System.***out***.println("indexOf: " + indexOfStr); //4

// substring

String substringStr = str.substring(7, 12);

System.***out***.println("substring: " + substringStr); // World

// equals

boolean equalsStr = str.equals("Hello, World!");

System.***out***.println("equals: " + equalsStr); // true

// lastIndexOf

int lastIndexOfStr = str.lastIndexOf('o');

System.***out***.println("lastIndexOf: " + lastIndexOfStr); // 8

// startsWith

boolean startsWithStr = str.startsWith("Hello");

System.***out***.println("startsWith: " + startsWithStr); // true

// endsWith

boolean endsWithStr = str.endsWith("World!");

System.***out***.println("endsWith: " + endsWithStr); // true

// equalsIgnoreCase

boolean equalsIgnoreCaseStr = str.equalsIgnoreCase("hello, world!");

System.***out***.println("equalsIgnoreCase: " + equalsIgnoreCaseStr); // true

// toLowerCase

String lowerCaseStr = str.toLowerCase();

System.***out***.println("toLowerCase: " + lowerCaseStr); // hello, world!

// toUpperCase

String upperCaseStr = str.toUpperCase();

System.***out***.println("toUpperCase: " + upperCaseStr); // HELLO, WORLD!

// length

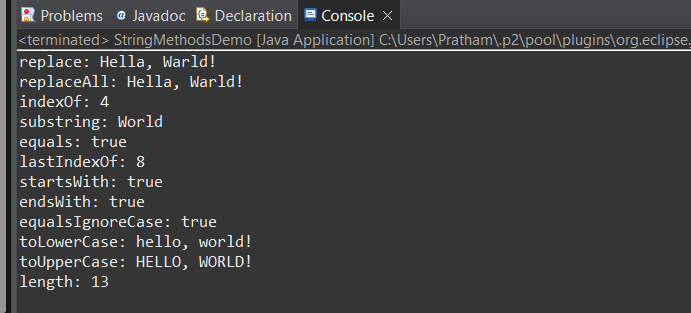
int lengthStr = str.length();

System.***out***.println("length: " + lengthStr); // 13

}

}

**Output-**

****