LAB 4 ANP-C7781

Solve following questions:

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

Program :-

package Day07;

public class MethodOverloadedCalculator {

// 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;

}

public static void main(String[] args) {

// Object Creation

MethodOverloadedCalculator calc = new MethodOverloadedCalculator();

// Testing the add methods

System.***out***.println("Sum of two integers (5 + 6): " + calc.add(5, 6));

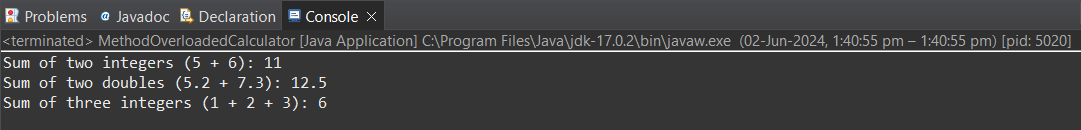
System.***out***.println("Sum of two doubles (5.2 + 7.3): " + calc.add(5.2, 7.3));

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

}

}

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.

Program :-

package Day07;

class Person {

private String name;

private int age;

// Constructor to set name and age

public Person(String name, int age) {

this.name = name;

this.age = age;

}

// Getter for name

public String getName() {

return name;

}

// Getter for age

public int getAge() {

return age;

}

}

class Student extends Person {

private double grade;

// Constructor to set name, age, and grade

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

// Call the constructor of the parent class Person using super

super(name, age);

this.grade = grade;

}

// Getter for grade

public double getGrade() {

return grade;

}

}

public class PersonDetailsProgram {

public static void main(String[] args) {

// Create a Student object

Student student = new Student("Pranay", 21, 10);

// Access the properties of the Student object

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

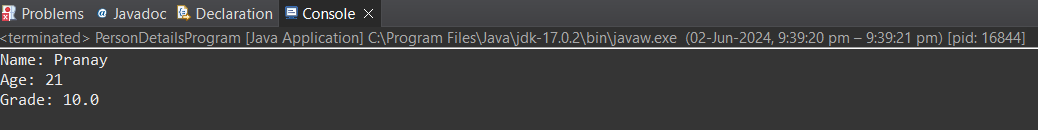
System.***out***.println("Age: " + student.getAge());

System.***out***.println("Grade: " + student.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.

Program :-

package Day07;

// Base class Shape

class Shape {

// Method to draw a shape

void draw() {

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

}

}

// Subclass Circle that extends Shape

class Circle extends Shape {

// Overridden draw() method in Circle

void draw() {

// Call the draw() method of the parent class Shape using super

super.draw();

// Print "Drawing Circle" to demonstrate overriding

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

}

}

public class DrawingProgram {

// Main method to execute the program

public static void main(String[] args) {

// Create an object of the Circle class

Circle circle = new Circle();

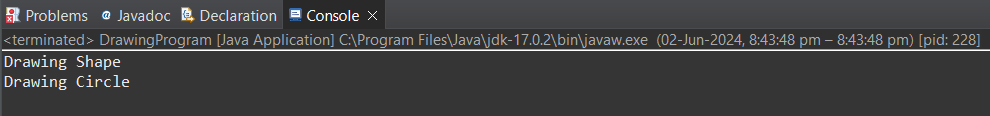
// Call the draw() method on the Circle object

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.

Program :-

package Day07;

// Base class BankAccount

class BankAccount {

// Instance variable to store the account balance

public double balance;

// Constructor to set the initial balance

public BankAccount(double initialBalance) {

balance = initialBalance;

}

// Method to deposit an amount

public void deposit(double amount) {

balance += amount;

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

}

}

// Subclass SavingsAccount that extends BankAccount

class SavingsAccount extends BankAccount {

// Constructor to set the initial balance

public SavingsAccount(double initialBalance) {

// Call the constructor of the parent class BankAccount

super(initialBalance);

}

// Override deposit method to add interest

public void deposit(double amount) {

// Call the deposit method of the parent class BankAccount using super

super.deposit(amount);

// Add interest to the balance and print the new balance

System.***out***.println("Interest added. New balance: " + (balance + (balance \* 0.10)));

}

}

// Main class to execute the program

public class BankingProgram {

// Main method

public static void main(String[] args) {

// Create an object of the SavingsAccount class

SavingsAccount savingsAccount = new SavingsAccount(2000);

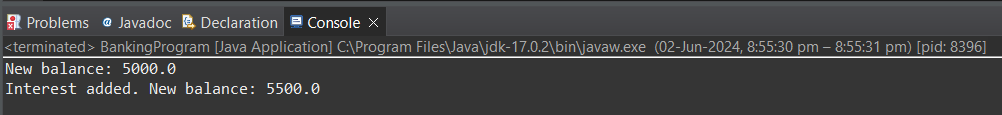
// Call the deposit method on the SavingsAccount object

savingsAccount.deposit(3000);

}

}

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.

Program :-

package Day07;

class Employee {

private String name;

private double salary;

// Constructor to set the name and salary

public Employee(String name, double salary) {

this.name = name;

this.salary = salary;

}

// Method to display employee details

public void displayDetails() {

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

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

}

}

class Manager extends Employee {

private String department;

// Constructor to set the name, salary, and department

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

super(name, salary);

this.department = department;

}

// Override displayDetails method to include department details

public void displayDetails() {

super.displayDetails(); // Call the displayDetails method of the parent class

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

}

}

public class EmployeeProgram {

public static void main(String[] args) {

// Create an Employee object

Employee employee = new Employee("Pranay", 50000);

employee.displayDetails();

// Create a Manager object

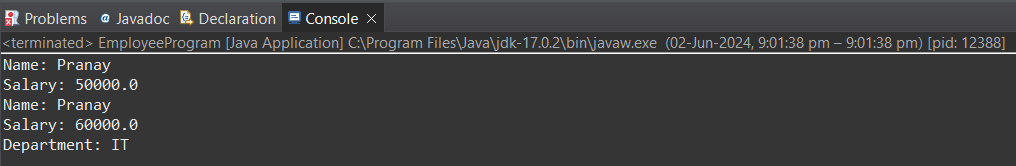
Manager manager = new Manager("Pranay", 60000, "IT");

manager.displayDetails();

}

}

Output :-



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

Program :-

package Day07;

// Class ImmutableExample

public class ImmutableExample {

// Private final field to store the object value

private final String value;

// Constructor to set the value

public ImmutableExample(String value) {

// Set the value in the constructor

this.value = value;

}

// Method to display the value

public void displayValue() {

// Print the value

System.***out***.println("Value: " + value);

}

// Main method to execute the program

public static void main(String[] args) {

// Create an object of ImmutableExample with the value "Hi"

ImmutableExample immutableExample = new ImmutableExample("Hi");

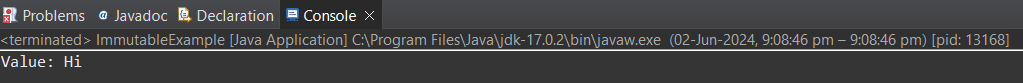
// Call the displayValue method to print the value

immutableExample.displayValue();

}

}

Output :-



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

Program :-

package Day07;

public class MutableExample {

private String value;

// Constructor to set the initial value

public MutableExample(String value) {

this.value = value;

}

// Method to get the value

public String getValue() {

return value;

}

// Method to set the value

public void setValue(String value) {

this.value = value;

}

// Method to display the value

public void displayValue() {

System.***out***.println("Value: " + value);

}

public static void main(String[] args) {

// Create an object of MutableExample with initial value "hello 1"

MutableExample mutableExample = new MutableExample("hello 1");

mutableExample.displayValue();

// Change the value to "hello 2"

mutableExample.setValue("hello 2");

mutableExample.displayValue();

// Change the value to "hello 3"

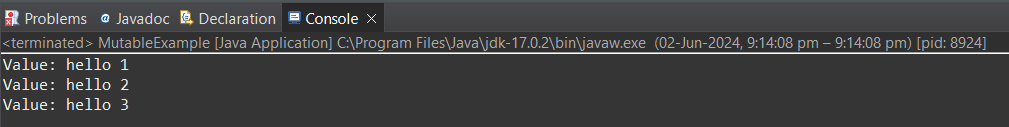
mutableExample.setValue("hello 3");

mutableExample.displayValue();

}

}

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

Program :-

package Day07;

public class StringMethodsExample {

public static void main(String[] args) {

String str1 = "Hello, World!";

String str2 = "hello, world!";

// replace()

System.***out***.println(str1.replace("World", "Java")); // Output: Hello, Java!

// contains()

System.***out***.println(str1.contains("World")); // Output: true

// replaceAll()

System.***out***.println(str1.replaceAll("[a-z]", "\*")); // Output: H\*\*\*\*\*,\*\*\*\*\*!

// indexOf()

System.***out***.println(str1.indexOf("World")); // Output: 7

// substring()

System.***out***.println(str1.substring(7, 12)); // Output: World

// equals()

System.***out***.println(str1.equals(str2)); // Output: false

// lastIndexOf()

System.***out***.println(str1.lastIndexOf("l")); // Output: 12

// startsWith()

System.***out***.println(str1.startsWith("Hello")); // Output: true

// endsWith()

System.***out***.println(str1.endsWith("!")); // Output: true

// toLowerCase()

System.***out***.println(str1.toLowerCase()); // Output: hello, world!

// toUpperCase()

System.***out***.println(str1.toUpperCase()); // Output: HELLO, WORLD!

// isEmpty()

System.***out***.println("".isEmpty()); // Output: true

// length()

System.***out***.println(str1.length()); // Output: 13

// Not Able to Understand these 2

// split()

// equalsIgnoreCase()

}

}

Output :-

