LAB 4

Poorvaja Bhorunde

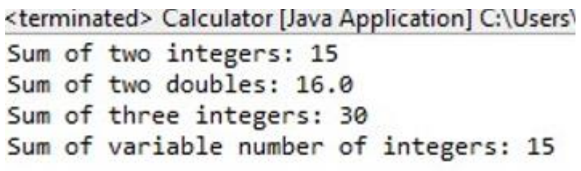
Q.1. Method Overloading: Write a class Calculator with overloaded methods add(). Implement add() methods that take:

- Two integers
- Two double values
- Three integers
- A variable number of integers

Program:

```
package demo;
public class Calculator {
    // Method to add two integers
    public int add(int a, int b) {
        return a + b;
    }
    // Method to add two doubles with a descriptive name
    public double addDoubles(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 num : numbers) {
            sum += num;
        }
        return sum;
    }
    // Example usage in a separate method for better organization
    public static void main(String[] args) {
        Calculator calc = new Calculator();
        System.out.println("Sum of two integers: " + calc.add(5, 10));
        System.out.println("Sum of two doubles: " + calc.addDoubles(5.5,
10.5));
        System.out.println("Sum of three integers: " + calc.add(5, 10, 15));
        System.out.println("Sum of variable number of integers: " +
calc.add(1, 2, 3, 4, 5));
    }
}
```

Output:



```
<terminated> Calculator [Java Application] C:\Users\
Sum of two integers: 15
Sum of two doubles: 16.0
Sum of three integers: 30
Sum of variable number of integers: 15
```

LAB 4

Poorvaja Bhorunde

Q.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:

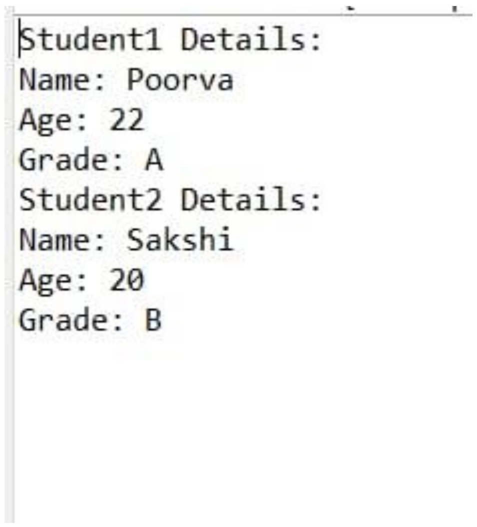
```
package demo;
 public class Person {
    private String name;
    private int age;
    public Person(String name, int age) { // Constructor to initialize name
and age
        this.name = name;
        this.age = age;
    }
    public String getName() {     // Getter methods for name and age
        return name;
    }
    public int getAge() {
        return age;
    }
      public static void main(String[] args) {
      }

}

Main2.java
package demo;
 class stud extends Person {
    private String grade;
    // Constructor to initialize name, age, and grade
    public stud(String name, int age, String grade) {
        super(name, age); // Calling parent class constructor
        this.grade = grade;
    }
    public String getGrade() {     // Getter method for grade
        return grade;
    }
    public void displayInfo() {     // Method to display student details
        System.out.println("Name: " + getName());
        System.out.println("Age: " + getAge());
        System.out.println("Grade: " + getGrade());
    }
}
```

LAB 4

Poorvaja Bhorunde

```java
public class Main2 {
	public static void main(String[] args) {
	  stud student1 = new stud("Poorva", 22, "A");
	  stud student2 = new stud("Sakshi", 20, "B");
	  //displaying student details
	  System.out.println("Student1 Details:");
	  student1.displayInfo();
	  System.out.println("Student2 Details:");
	  student2.displayInfo();
	}
}
```

Output:



```
Student1 Details:
Name: Poorva
Age: 22
Grade: A
Student2 Details:
Name: Sakshi
Age: 20
Grade: B
```

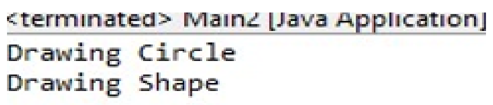
Q.3. Super Keyword: Create a base class Shape with a method draw() that prints "Drawing Shape".

- 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 demo;
class Shape {
    public void draw() {
        System.out.println("Drawing Shape");
    }
}
class Circle extends Shape {
    @Override
    public void draw() {
        System.out.println("Drawing Circle");
        super.draw(); // Calling draw() method of the Shape class
    }
}
public class Main2 {
    public static void void main(String[] args) {
        Circle circle = new Circle();
        circle.draw();
    }
}
```

Output:



```
<terminated> Main2 [Java Application]
Drawing Circle
Drawing Shape
```

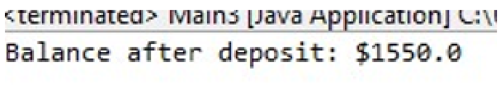
Q.4. Create a base class BankAccount with a method deposit(amount) and a constructor that sets the initial balance.

- 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 demo;
class BankAccount {
    protected double balance;
    // Constructor to set initial balance
    public BankAccount(double initialBalance) {
        this.balance = initialBalance;
    }
    // Method to deposit amount
    public void deposit(double amount) {
        balance += amount;
    }
}
class SavingsAccount extends BankAccount {
    private double interestRate;
    // Constructor to set initial balance and interest rate
    public SavingsAccount(double initialBalance, double interestRate) {
        super(initialBalance);
        this.interestRate = interestRate;
    }
    // Override deposit method to add interest before depositing
    @Override
    public void deposit(double amount) {
        double interest = balance * (interestRate / 100); // Calculate
interest
        balance += interest; // Add interest to balance
        super.deposit(amount); // Call deposit method of the base class
    }
}
public class Main3 {
    public static void main(String[] args) {
        // Create a SavingsAccount with initial balance and interest rate
        SavingsAccount savingsAccount = new SavingsAccount(1000, 5);
        // Initial balance: $1000, Interest rate: 5%
        // Deposit an amount and observe the effect of interest
        savingsAccount.deposit(500); // Deposit $500
        System.out.println("Balance after deposit: $" +
savingsAccount.balance); // Print updated balance
    }
}
```

<u>Output:</u>



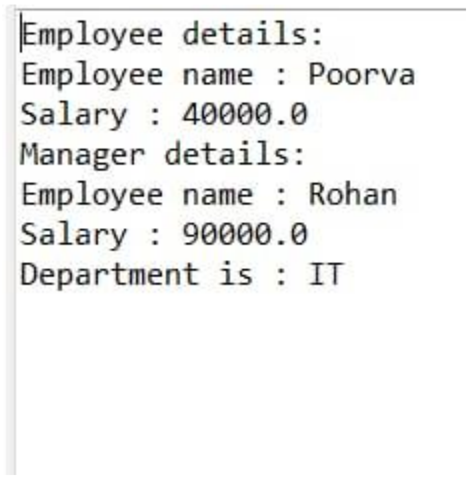
```
<terminated> Main3 [Java Application] C:\
Balance after deposit: $1550.0
```

Q.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 demo;
class Employee {
    private String name;
    private double salary;
    public Employee(String name, double salary) {
        this.name = name;
        this.salary = salary;
    }
    public String getName() {//get() method
        return name;
    }
    public double getSalary() {
        return salary;
    }
    public void displayDetails() {
        System.out.println("Name of employee is : " + name);
        System.out.println("Salary is : " + salary);
    }
}


class Manager extends Employee {
    private String department;

    public Manager(String name, double salary, String department) {
        super(name, salary);
        this.department = department;
    }
    public void displayDetails() {//Override
        super.displayDetails();
        System.out.println("Department is : " + department);
    }
}

public class Main5 {
    public static void main(String[] args) {
        Employee emp = new Employee("Poorva", 40000.0);
        Manager manger = new Manager("Rohan", 90000.0, "IT");
        System.out.println("Employee details:");
        emp.displayDetails();
        System.out.println("Manager details:");
        manger.displayDetails();
    }
}
```

Output:



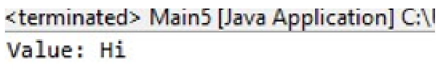
```
Employee details:
Employee name : Poorva
Salary : 40000.0
Manager details:
Employee name : Rohan
Salary : 90000.0
Department is : IT
```

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

Program:

```
package demo;
final class ImmutableExample {
    private final String value;
    // Private constructor to prevent external instantiation
    private ImmutableExample(String value) {
        this.value = value;
    }
    // Static factory method to create instances of ImmutableExample
    public static ImmutableExample createInstance() {
        // Return a new instance with the value "Hi"
        return new ImmutableExample("Hello");
    }
    // Getter method to access the value
    public String getValue() {
        return value;
    }
}
public class Main {
    public static void main(String[] args) {
        // Create an instance of ImmutableExample
        ImmutableExample immutableObj = ImmutableExample.createInstance();
        // Access and print the value
        System.out.println("Value: " + immutableObj.getValue());
    }
}
```

Output:



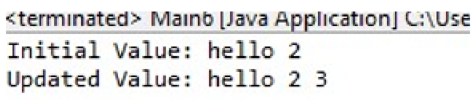
```
<terminated> Main5 [Java Application] C:\
Value: Hi
```

Q.7. Write the same programme for the class MutableExample, to output the object values 'hello 2' and 'hello3'.z

Program:

```
package demo;
class MutableExample {
    private String value; // Creating the string
    // Constructor
    public MutableExample(String value) { // Constructor
        this.value = value;
    }
    // Method to set the value
    public void setValue(String value) {
        this.value = value;
    }
    // Method to append a number to the value
    public void appendNumber(int number) {
        this.value += " " + number;
    }
    // Getter method to access the value
    public String getValue() {
        return value;
    }
}
public class Main {
    public static void main(String[] args) {
        // Create an instance of MutableExample with initial value "hello 2"
        MutableExample mutableObj = new MutableExample("hello 2");
        // Print the initial value
        System.out.println("Initial Value: " + mutableObj.getValue());
        // Append number 3 to the value
        mutableObj.appendNumber(3);
        // Print the updated value
        System.out.println("Updated Value: " + mutableObj.getValue());
    }
}
```

Output:



```
<terminated> Mainb [Java Application] C:\Use
Initial Value: hello 2
Updated Value: hello 2 3
```

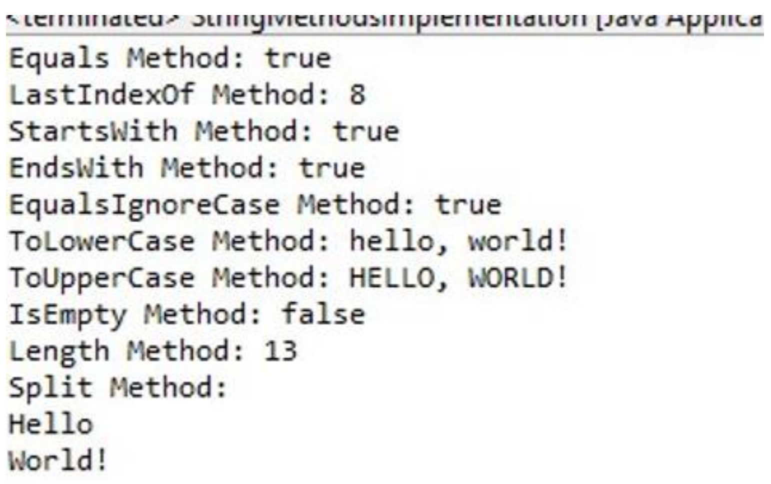
LAB 4
Poorvaja Bhorunde

Q.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 demo;
public class StringMethodsImplementation {//creating the class
    public static void main(String[] args) {
        // Example string
        String str = "Hello, World!";
        // Replace method
        System.out.println("Replace Method: " + str.replace('l',
'x'));//printing the statements
        // Contains method
        System.out.println("Contains Method: " +
str.contains("World"));
        // ReplaceAll method
        System.out.println("ReplaceAll Method: " +
str.replaceAll("[aeiou]", "*"));
        // IndexOf method
        System.out.println("IndexOf Method: " + str.indexOf('o'));
        // Substring method
        System.out.println("Substring Method: " + str.substring(7));
        // 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[] splitArray = str.split(",");
System.out.println("Split Method: ");         for (String s : splitArray) {
System.out.println(s.trim());
        }
    }
}
```

Output:



```
<terminated> StringMethodsImplementation [Java Applica
Equals Method: true
LastIndexOf Method: 8
StartsWith Method: true
EndsWith Method: true
EqualsIgnoreCase Method: true
ToLowerCase Method: hello, world!
ToUpperCase Method: HELLO, WORLD!
IsEmpty Method: false
Length Method: 13
Split Method:
Hello
World!
```