**Lab-4**

Q1. 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 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;

}

// Main method to test the overloaded add methods

public static void main(String[] args) {

Calculator cal = new Calculator();

// two integers

System.out.println("Sum of 5 and 3: " + cal.add(5, 3));

//two double

System.out.println("Sum of 3.3 and 5.5: " + cal.add(3.3, 5.5));

// three integers

System.out.println("Sum of 5, 4, and 3: " + cal.add(5,4,3));

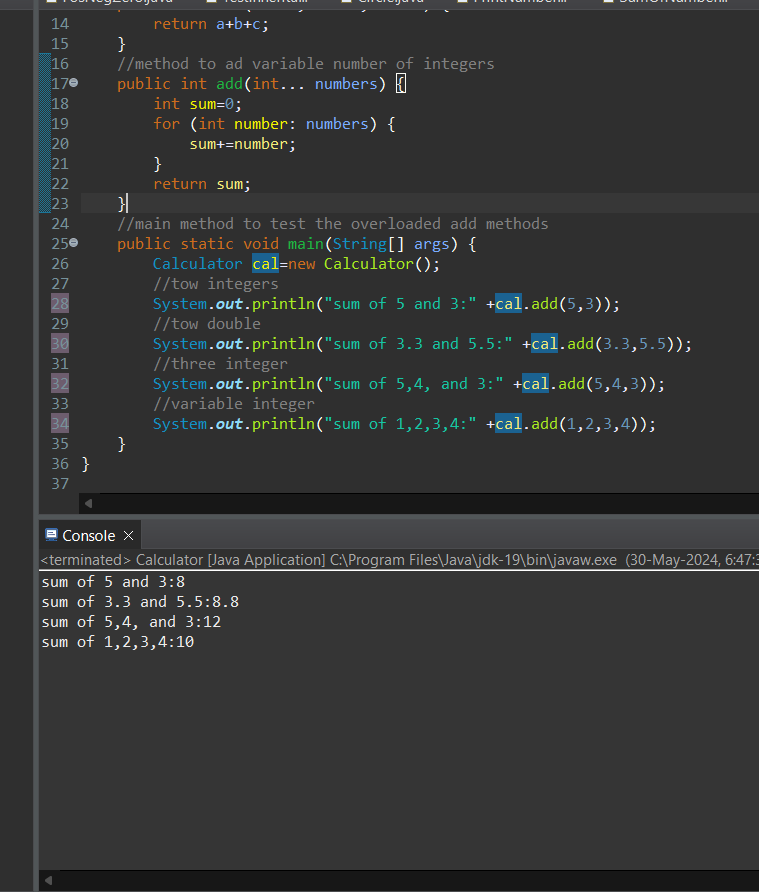
// variable imtegers

System.out.println("Sum of 1, 2, 3, 4: " + cal.add(1, 2, 3, 4));

}

}

**Output:**



Q2. 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:**

**Person class**

public class Person {

protected String name;

protected int age;

// Constructor for Person class

public Person(String name, int age) {

this.name = name;

this.age = age;

}

// Method to display Person details

public void displayPersonInfo() {

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

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

}

}

**Student class**

public class Student extends Person {

private String grade;

// Constructor for Student class

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

super(name, age); // Call to the superclass (Person) constructor

this.grade = grade;

}

// Method to display Student details

public void displayStudentInfo() {

// Display Person details

super.displayPersonInfo();

// Display Student specific details

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

}

// Main method to test the implementation

public static void main(String[] args) {

// Create Student objects

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

Student student2 = new Student("Akanskha", 15, "B");

// Display Student details

System.out.println("Student 1 Details:");

student1.displayStudentInfo();

System.out.println();

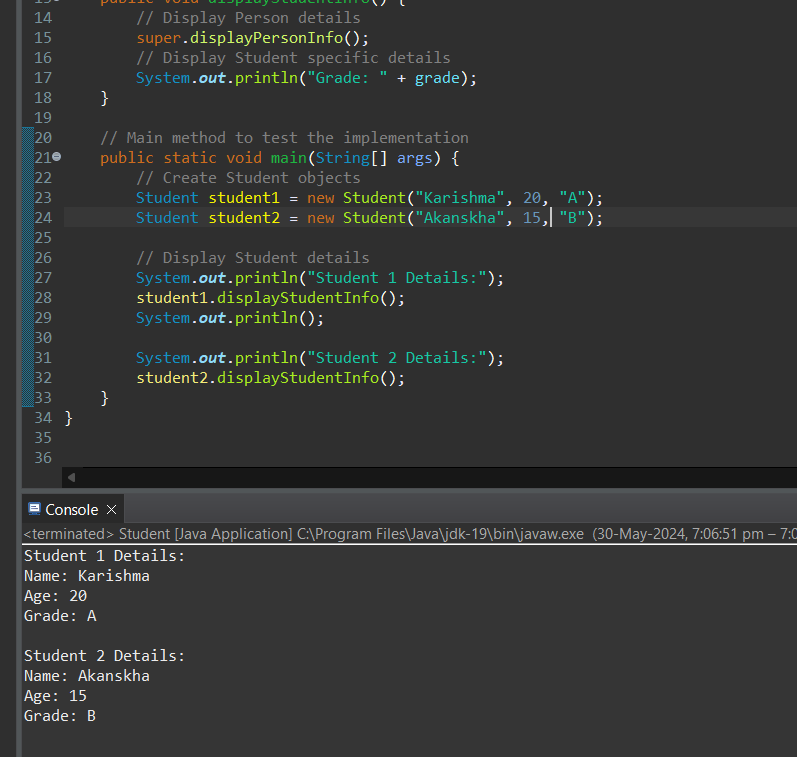
System.out.println("Student 2 Details:");

student2.displayStudentInfo();

}

}

**Output:**

****

Q3.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 {

// Method to draw shape

public void draw() {

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

}

}

// Subclass Circle

class Circle extends Shape {

// Override draw method

public void draw() {

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

// Call draw method of Shape class

super.draw();

}

}

// Main class

public class DrawingShape {

public static void main(String[] args) {

// Create Circle object

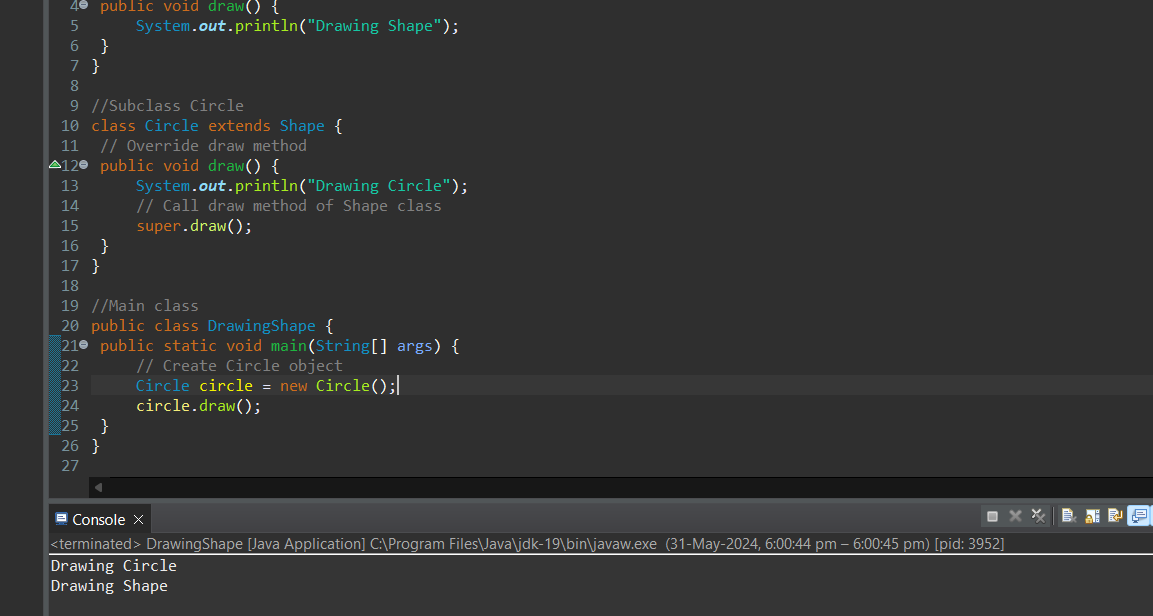
Circle circle = new Circle();

circle.draw();

}

}

**Output**



Q4. 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;

// Base class BankAccount

class BankAccount {

protected double balance;

// Constructor to set the initial balance

public BankAccount(double initialBalance) {

this.balance = initialBalance;

}

// Method to deposit amount

public void deposit(double amount) {

balance += amount;

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

}

// Method to get the current balance

public double getBalance() {

return balance;

}

}

// Subclass SavingsAccount

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

public void deposit(double amount) {

double interest = amount \* interestRate / 100;

double totalAmount = amount + interest;

super.deposit(totalAmount);

System.out.println("Interest: " + interest + ", Deposited with Interest: " + totalAmount);

}

}

public class BankInfo {

public static void main(String[] args) {

// Create SavingsAccount object with initial balance and interest rate

SavingsAccount savingsAccount = new SavingsAccount(1000.0, 5.0);

// Display initial balance

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

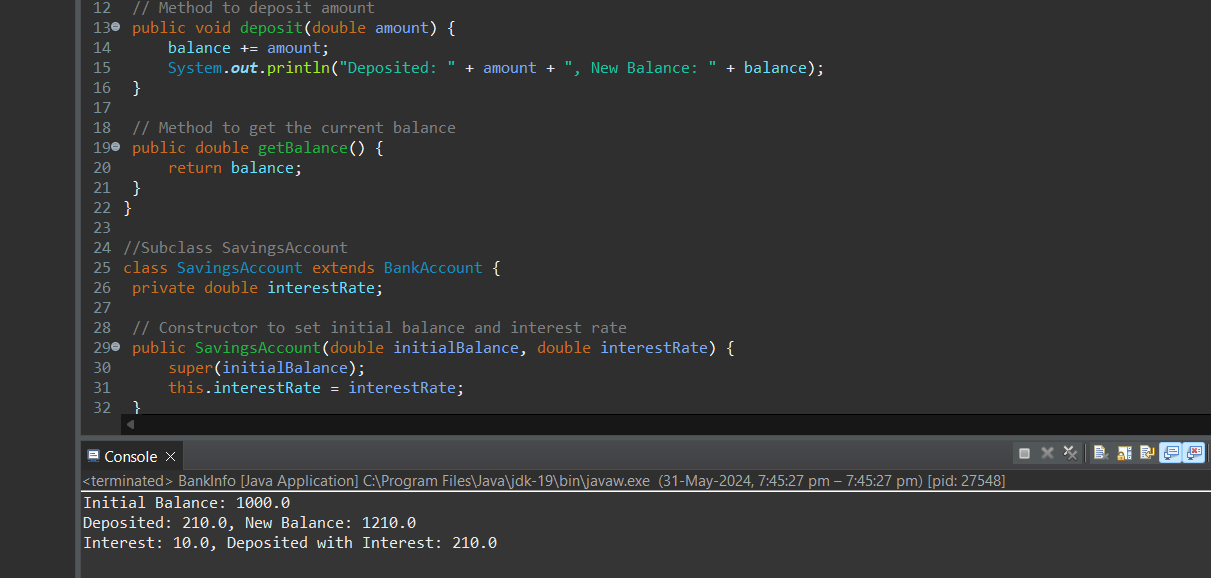
// Deposit an amount to see the effect of interest

savingsAccount.deposit(200.0);

}

}

**output**

****

Q5. 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**

**Employee class**

package demo;

public class Employee {

protected String name;

protected double salary;

// Constructor to initialize 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);

}

}

**Manager class**

package demo;

public class Manager extends Employee {

private String department;

// Constructor to initialize name, salary, and department

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

super(name, salary); // Call the superclass constructor

this.department = department;

}

// Override displayDetails method to include department details

public void displayDetails() {

super.displayDetails(); // Call the superclass displayDetails method

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

}

}

**Properties class(Main)**

package demo;

public class Properties {

public static void main(String[] args) {

// Create Employee object

Employee employee = new Employee("John Doe", 50000);

// Create Manager object

Manager manager = new Manager("Jane Smith", 70000, "IT");

// Display details of Employee

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

employee.displayDetails();

System.out.println();

// Display details of Manager

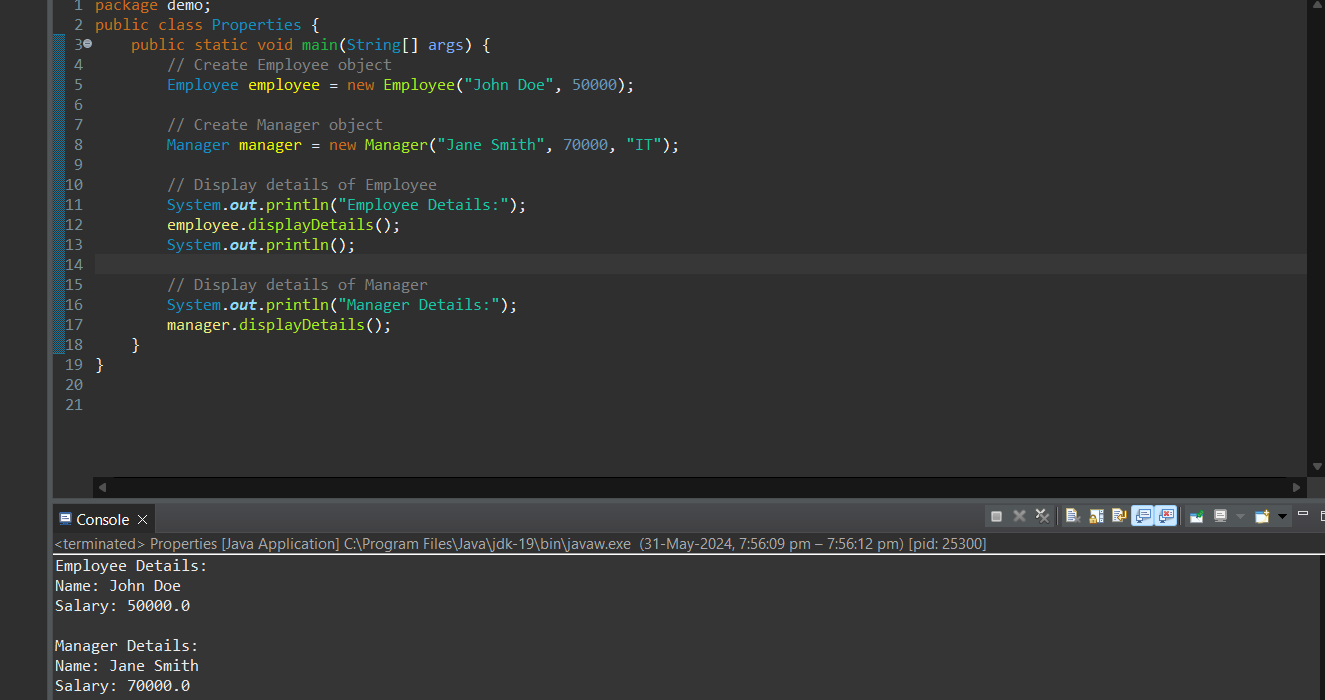
System.out.println("Manager Details:");

manager.displayDetails();

}

}

**Output**

****

Q6. Write the same programme for the class ImmutableExample, to achieve object value ‘Hi’.

**Program**

package demo;

public final class ImmutableExample {

private final String value;

public ImmutableExample(String value) {

this.value = value;

}

// Getter method to retrieve the value

public String getValue() {

return value;

}

// Main method to demonstrate immutability

public static void main(String[] args) {

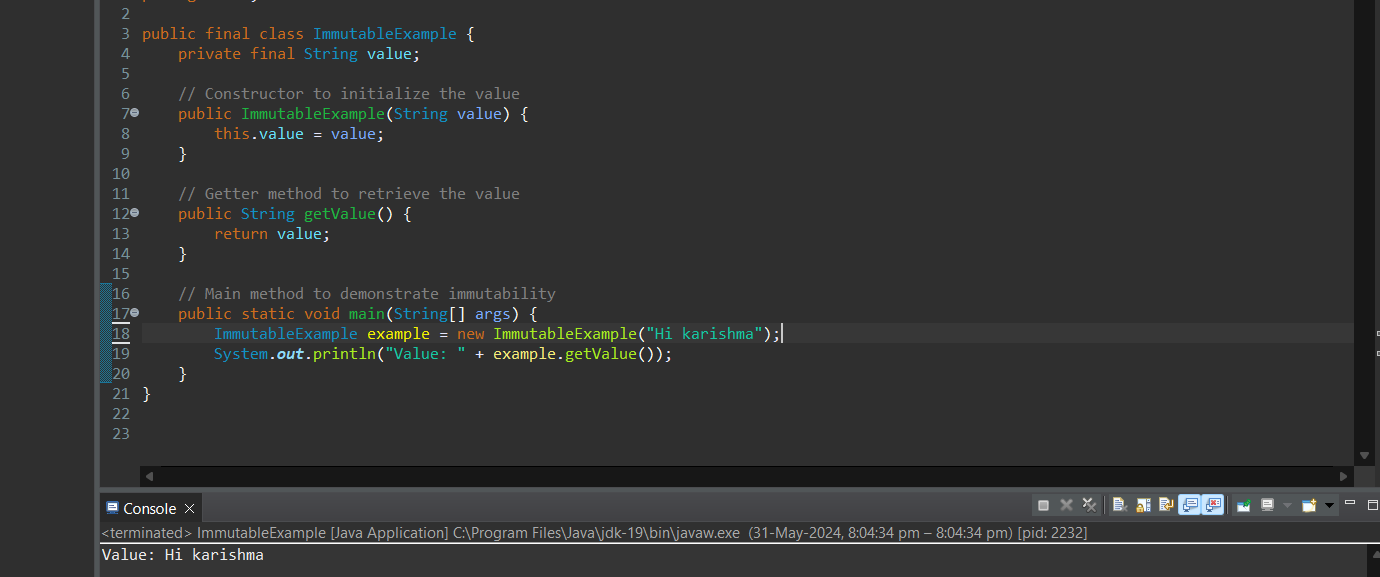
ImmutableExample example = new ImmutableExample("Hi karishma");

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

}

}

**Output**



Q7. Write the same programme for the class MutableExample, to output the object values ‘hello 2’ and ‘hello3’.

**Program**

public class MutableExample {

private String value;

// Constructor to initialize the value

public MutableExample(String value) {

this.value = value;

}

// Getter method to retrieve the value

public String getValue() {

return value;

}

// Setter method to modify the value

public void setValue(String value) {

this.value = value;

}

// Main method to demonstrate mutability

public static void main(String[] args) {

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

MutableExample example = new MutableExample("hello");

example.setValue("hello 2");

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

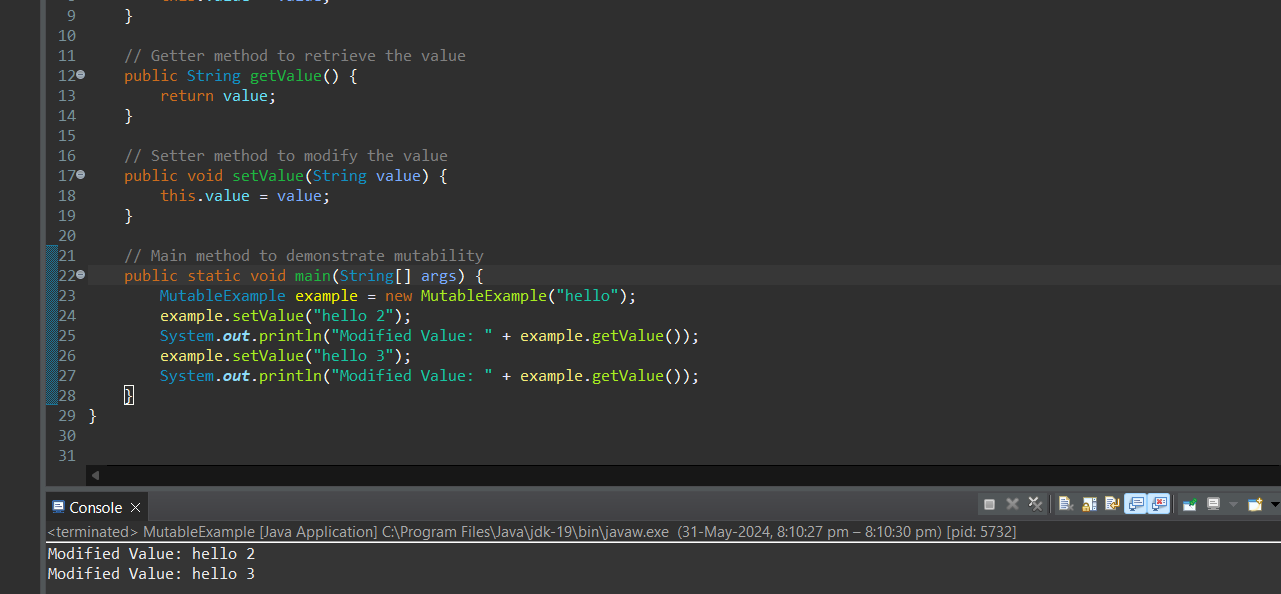
example.setValue("hello 3");

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

}

}

**Output**



Q8.     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**

public class StringMethodExample {

public static void main(String[] args) {

String str = "Hello, World! Welcome to the world of Java programming.";

// replace

String replacedStr = str.replace("World", "Universe");

System.out.println("replace: " + replacedStr);

// contains

boolean containsStr = str.contains("Java");

System.out.println("contains: " + containsStr);

// replaceAll

String replaceAllStr = str.replaceAll("world", "universe");

System.out.println("replaceAll: " + replaceAllStr);

// indexOf

int indexOfStr = str.indexOf("Java");

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

// substring

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

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

// equals

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

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

// lastIndexOf

int lastIndexOfStr = str.lastIndexOf("world");

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

// startsWith

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

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

// endsWith

boolean endsWithStr = str.endsWith("programming.");

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

// equalsIgnoreCase

boolean equalsIgnoreCaseStr = str.equalsIgnoreCase("hello, world! welcome to the world of java programming.");

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

// toLowerCase

String toLowerCaseStr = str.toLowerCase();

System.out.println("toLowerCase: " + toLowerCaseStr);

// toUpperCase

String toUpperCaseStr = str.toUpperCase();

System.out.println("toUpperCase: " + toUpperCaseStr);

// isEmpty

boolean isEmptyStr = str.isEmpty();

System.out.println("isEmpty: " + isEmptyStr);

// length

int lengthStr = str.length();

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

// split

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

System.out.println("split: ");

for (String s : splitStr) {

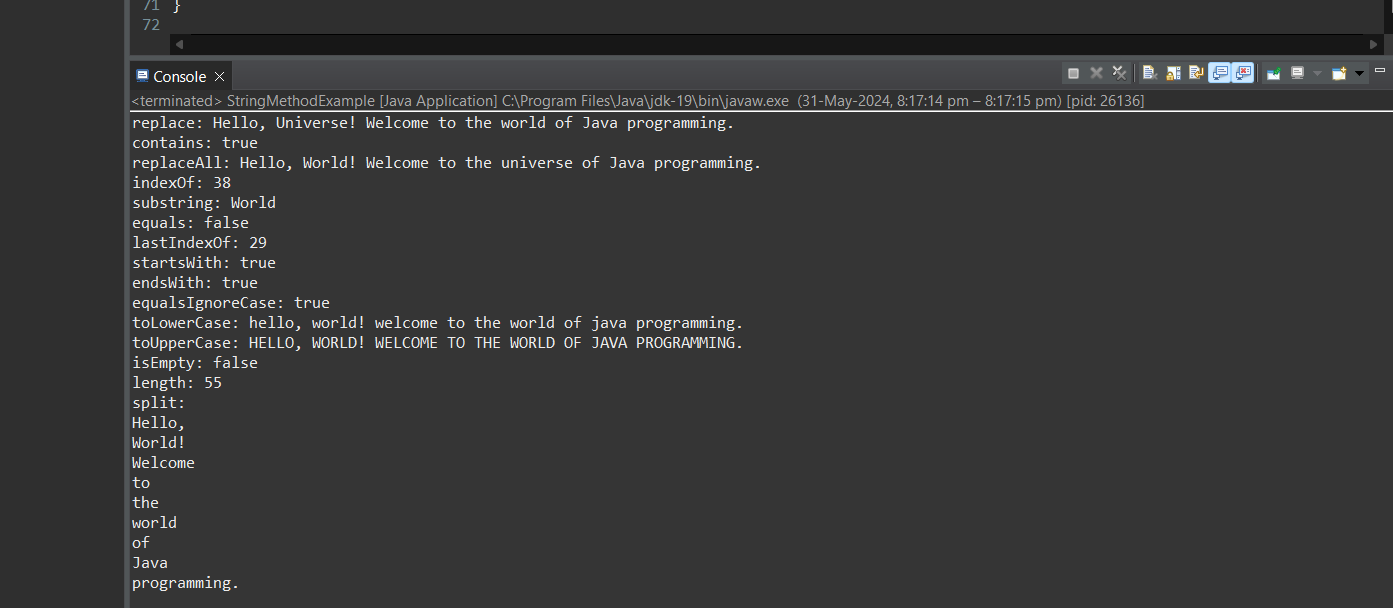
System.out.println(s);

}

}

}

**Output**

****