

Matrix Multiplication:

Write a Java program to perform matrix multiplication.

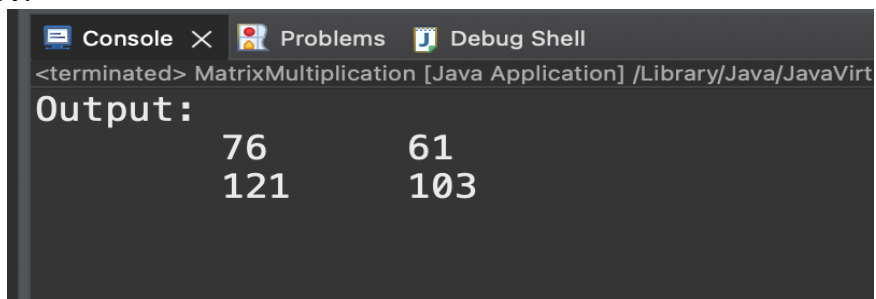
```
package matrix;

public class MatrixMultiplication {
    public static void main(String[] args) {
        int[][] num1 = { { 3, 4, 5 }, { 8, 6, 7 } };
        int[][] num2 = { { 3, 4 }, { 8, 6 }, { 7, 5 } };

        int result[][] = new int[num1.length][num2[0].length];

        for (int i = 0; i < num1.length; i++) {
            for (int j = 0; j < num2[0].length; j++) {
                for (int k = 0; k < num1[0].length; k++) {
                    result[i][j] += num1[i][k] * num2[k][j];
                }
            }
        }
        // Print the result
        for (int i = 0; i < result.length; i++) {
            for (int j = 0; j < result[0].length; j++) {
                System.out.print(result[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

OUTPUT:



The screenshot shows an IDE window with three tabs: 'Console', 'Problems', and 'Debug Shell'. The 'Console' tab is active, displaying the output of the Java application. The output is a 2x2 grid of numbers: 76, 61, 121, and 103. The text '<terminated> MatrixMultiplication [Java Application] /Library/Java/JavaVirt' is visible at the top of the console area.

```
<terminated> MatrixMultiplication [Java Application] /Library/Java/JavaVirt
Output:
      76      61
     121     103
```

Swing GUI:

Create a Java Swing GUI application that allows users to input their name and displays a personalized greeting message using JLabel and JTextField.

```
package Swing;
```

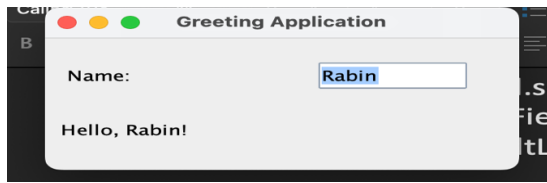
```
// imports
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class JavaGui {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Greeting Application");
        JLabel label = new JLabel(" Name:");
//    create objects
        final JTextField textField = new JTextField();
        final JLabel resultLabel = new JLabel();

        // position set Display
        label.setBounds(10, 20, 150, 30);
        textField.setBounds(170, 20, 100, 30);
        resultLabel.setBounds(10, 70, 300, 30);

        textField.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String name = textField.getText();
                resultLabel.setText("Hello, " + name + "!");
            }
        });
        // add to frames
        frame.add(label);
        frame.add(textField);
        frame.add(resultLabel);
        frame.setLayout(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 150);
        frame.setVisible(true);
    }
}
```

Output:



Java Applet:

Develop a simple Java applet program

Interface:

Define an interface named Shape with methods calculateArea() and calculatePerimeter(). Implement this interface in two classes: Circle and Rectangle. Write a program to demonstrate the use of the interface.

```
// interface class
package interfaces;
public interface Shape {
    double calculateArea();
    double calculatePerimeter();
}

package interfaces;
public class Circle implements Shape{
    private double radius;
    public Circle(double radi) {
        this.radius= radi;
    }
    public double calculateArea() {
        return Math.PI * radius * radius;
    }
    public double calculatePerimeter() {
        return 2 * Math.PI * radius;
    }
}

package interfaces;
class Rectangle implements Shape {
    private double length;
    private double width;

    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
    public double calculateArea() {
        return length * width;
    }
}
```

```

    }
    public double calculatePerimeter() {
        return 2 * (length + width);
    }
}

package interfaces;
public class Main {
    public static void main(String[] args) {
        Circle circle = new Circle(5.0);
        Rectangle rectangle = new Rectangle(7.0, 6.0);

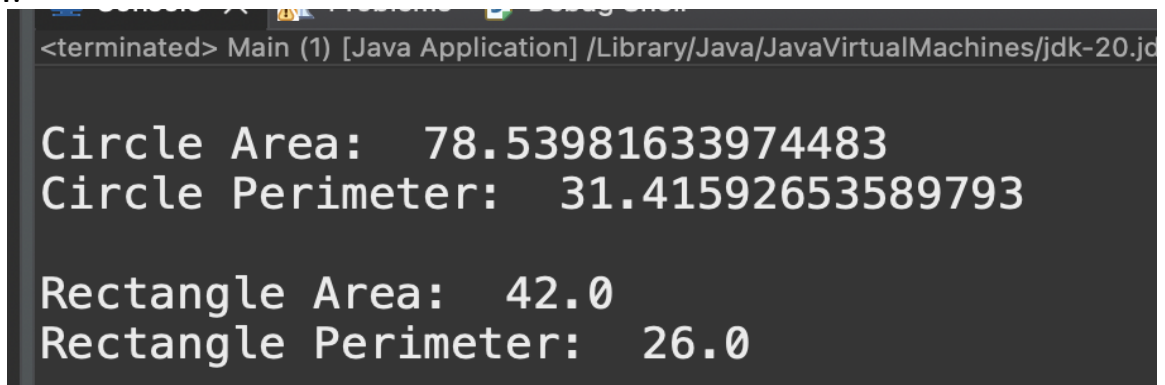
        System.out.println("\nCircle Area: " + circle.calculateArea());
        System.out.println("Circle Perimeter: " + circle.calculatePerimeter());

        System.out.println("\nRectangle Area: " + rectangle.calculateArea());
        System.out.println("Rectangle Perimeter: " + rectangle.calculatePerimeter());

    }
}

```

OUTPUT:



```

<terminated> Main (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk-20.jc

Circle Area:  78.53981633974483
Circle Perimeter:  31.41592653589793

Rectangle Area:  42.0
Rectangle Perimeter:  26.0

```

Abstract Class:

Create an abstract class called Vehicle with abstract methods start() and stop(). Implement this abstract class in two concrete classes: Car and Motorcycle. Demonstrate the use of abstract classes and their methods.

```
package abstracts;
abstract class Vehicle {
    public abstract void start();
    public abstract void stop();
}

package abstracts;
public class Car extends Vehicle {
    public void start() {
        System.out.println("\nRabin Car Enginee start");
    }
    public void stop() {
        System.out.println("Rabin Car Enginee stop");
    }
}

package abstracts;

public class Motorcycle extends Vehicle {
    public void start() {
        System.out.println("\nDuke 200 Enginee start");
    }
    public void stop() {
        System.out.println("Duke 200 Enginee stop");
    }
}

package abstracts;
public class Vehicle_Main {

    public static void main(String[] args) {
        // create an objects
        Motorcycle bike = new Motorcycle();
        Car car = new Car();

        // call methods
        bike.start();
        bike.stop();

        car.start();
        car.stop();
    }
}
```

OUTPUT:

```
<terminated> Vehicle_Main [Java Application] /Library/Java/JavaVirtualMachines/jdk-
|
Duke 200 Enginee start
Duke 200 Enginee stop

Rabin Car Enginee start
Rabin Car Enginee stop
```

Super Class / Inheritance:

Design a class hierarchy representing different types of animals. Include a superclass Animal with subclasses such as Mammal, Bird, and Fish. Implement the concept of inheritance and demonstrate it through a program.

```
package inheritance;

//Animal.java
public class Animal {
    //common attributes of all animals
    protected String name;

    //constructor
    public Animal(String name) {
        this.name = name;
    }
    //common behaviors of all animals
    public void eat() {
        System.out.println(name + " is eating.");
    }

    public void sleep() {
        System.out.println(name + " is sleeping.");
    }
}

public class Birds extends Animal {
    public Birds(String name) {
        super(name); // call the constructor of the superclass
    }
    public void fly() {
        System.out.println(name + " is flying. \n");
    }
}

public class Fish extends Animal {
    // constructor
    public Fish(String name) {
        super(name);
    }
    // specific behavior of fish
    public void breathe() {
        System.out.println(name + " is breathing with gills.\n");
    }
}

public class Mammal extends Animal {
    //constructor
```

```

        public Mammal(String name) {
            super(name);
        }

//specific behavior of mammals
        public void breathe() {
            System.out.println(name + " is breathing with lungs.\n");
        }
    }

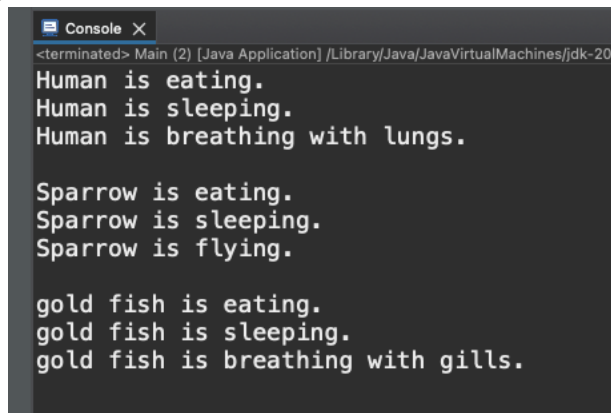
    public class Main {
        public static void main(String[] args) {
            // Creating objects
            Mammal mam = new Mammal("Human");
            Birds sparrow = new Birds("Sparrow");
            Fish goldfish = new Fish("gold fish");
            mam.eat();
            mam.sleep();
            mam.breathe();

            sparrow.eat();
            sparrow.sleep();
            sparrow.fly();

            goldfish.eat();
            goldfish.sleep();
            goldfish.breathe();
        }
    }
}

```

OUTPUT:



```

<terminated> Main (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk-20
Human is eating.
Human is sleeping.
Human is breathing with lungs.

Sparrow is eating.
Sparrow is sleeping.
Sparrow is flying.

gold fish is eating.
gold fish is sleeping.
gold fish is breathing with gills.

```


Constructors:

Create a class called Book with attributes title, author, and price. Include a default constructor and a parameterized constructor. Write a program that creates instances of the Book class using both constructors.

```
package constructor;

public class Book {

    String name;
    String author;
    double price;

    // Default constructor
    Book(){
        name= "Seto Dharti";
        author = "Baburam Neupane";
        price = 345.00;
        System.out.println("Default Constructor: ");
    }

    // Parameterized Constructor
    public Book(String nam, String auth, double paisa) {
        this.name = nam;
        this.author = auth;
        this.price = paisa;
        System.out.println("\nParameterized Construcor: ");
    }

    // display
    void show() {
        System.out.println(" Name: " + name + "\n Author: " + author + "\n Price: " + price);
    }

}

public class Book_Main {
    public static void main(String[] args) {

        Book bok = new Book();
        bok.show();

        Book book = new Book("11 Minutes", "Paulo ", 800.00);
        book.show();

    }
}
```

OUTPUT:

```
Console X
<terminated> Book_Main [Java Application] /Library/Java/JavaVirtualMachines/jdk-2
Default Constructor:
  Name: Seto Dharti
  Author: Baburam Neupane
  Price: 345.0

Parameterized Construcor:
  Name: 11 Minutes
  Author: Paulo
  Price: 800.0
```

Error Handling:

Write a Java program that reads an integer from the user. Handle the possibility of the user entering a non-integer value and provide appropriate error messages. Ensure the program does not crash due to invalid input.

```
package error_Handling;
import java.util.Scanner;

public class IntegerErrors {

    public static void main(String[] args) {

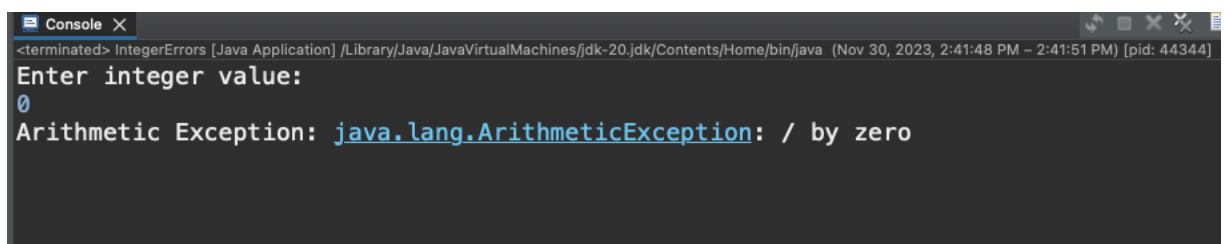
        Scanner sc = new Scanner(System.in); // Scanner input
        int num2 = 12;
        System.out.println("Enter integer value: ");
        try {
            int num = sc.nextInt();
            int result = num2 / num;
            System.out.println(result);

        } catch (ArithmeticException ex) {
            System.out.println("Arithmetic Exception: " + ex);

        } catch (Exception ex) {
            System.out.println("Error: ");
        }

    }
}
```

OUTPUT:



```
Console X
<terminated> IntegerErrors [Java Application] /Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java (Nov 30, 2023, 2:41:48 PM - 2:41:51 PM) [pid: 44344]
Enter integer value:
0
Arithmetic Exception: java.lang.ArithmeticException: / by zero
```

GUI-Based Functional Calculator:

Design a GUI-based functional calculator using Java Swing. Include basic arithmetic operations (addition, subtraction, multiplication, division). Implement error handling for division by zero and invalid input.

```
package calculator_app;
public class Calculator {
    public static void main(String[] args) {
        new Calculation();
    }
}

package calculator_app;
import java.awt.Color;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.JButton;

// create class and implements the actionListener
public class Calculation implements ActionListener {
    // create objects of the different frame
    JFrame frame;
    JTextField textfield;
    JButton[] numberButtons = new JButton[10];
    JButton[] functionButtons = new JButton[8];
    JButton addButton, subButton, mulButton, divButton;
    JButton decButton, equButton, delButton, clrButton;
    JPanel panel;
    Font myFont = new Font("Ink Free", Font.BOLD, 30);
    double num1 = 0, num2 = 0, result = 0;
    char operator;
    Calculation() {
        frame = new JFrame("Calculator");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(420, 550);
        frame.setLayout(null);
        textfield = new JTextField();
        textfield.setBounds(50, 25, 300, 50);
        textfield.setFont(myFont);
        textfield.setEditable(false);

        addButton = new JButton("+");
```

```

subButton = new JButton("-");
mulButton = new JButton("*");
divButton = new JButton("/");
decButton = new JButton(".");
equButton = new JButton("=");
delButton = new JButton("Delete");
clrButton = new JButton("clear");
functionButtons[0] = addButton;
functionButtons[1] = subButton;
functionButtons[2] = mulButton;
functionButtons[3] = divButton;
functionButtons[4] = decButton;
functionButtons[5] = equButton;
functionButtons[6] = delButton;
functionButtons[7] = clrButton;
for (int i = 0; i < 8; i++) {
    functionButtons[i].addActionListener(this);
    functionButtons[i].setFont(myFont);
    functionButtons[i].setFocusable(false);
}
for (int i = 0; i < 10; i++) {
    numberButtons[i] = new JButton(String.valueOf(i));
    numberButtons[i].addActionListener(this);
    numberButtons[i].setFont(myFont);
    numberButtons[i].setFocusable(false);
}
// Buttons positioned
delButton.setBounds(50, 430, 145, 50); // delete button position
clrButton.setBounds(205, 430, 145, 50); // clear button position
panel = new JPanel();
panel.setBounds(50, 100, 300, 300);
panel.setLayout(new GridLayout(4, 4, 10, 10));
panel.setBackground(Color.gray);
panel.add(numberButtons[1]);
panel.add(numberButtons[2]);
panel.add(numberButtons[3]);
panel.add(addButton);
panel.add(numberButtons[4]);
panel.add(numberButtons[5]);
panel.add(numberButtons[6]);
panel.add(subButton);
panel.add(numberButtons[7]);
panel.add(numberButtons[8]);
panel.add(numberButtons[9]);
panel.add(mulButton);
panel.add(decButton);
panel.add(numberButtons[0]);
panel.add(equButton);

```

```

        panel.add(divButton);
        // Buttons add to the frame
        frame.add(textfield);
        frame.add(delButton);
        frame.add(clrButton);
        frame.add(panel);
        frame.setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        for(int i=0; i<10; i++) {
            if(e.getSource() == numberButtons[i]) {
                textfield.setText(textfield.getText().concat(String.valueOf(i)));
            }
        }
        if(e.getSource() == decButton) {
            textfield.setText(textfield.getText().concat("."));
        }
        if(e.getSource() == addButton) {
            num1 = Double.parseDouble(textfield.getText());
            operator = '+';
            textfield.setText("");
        }
        if(e.getSource() == subButton) {
            num1 = Double.parseDouble(textfield.getText());
            operator = '-';
            textfield.setText("");
        }
        if(e.getSource() == mulButton) {
            num1 = Double.parseDouble(textfield.getText());
            operator = '*';
            textfield.setText("");
        }
        if(e.getSource() == divButton) {
            num1 = Double.parseDouble(textfield.getText());
            operator = '/';
            textfield.setText("");
        }

        if(e.getSource() == equButton) {
            num2 = Double.parseDouble(textfield.getText());
            switch(operator) {
                case '+':
                    result = num1 + num2;
                    break;
                case '-':

```

```

        result = num1 - num2;
        break;
    case '*':
        result = num1 * num2;
        break;
    case '/':
        result = num1 / num2;
        break;
    }
    textfield.setText(String.valueOf(result));
    num1 =result;
}
if(e.getSource()==clrButton) {
    textfield.setText("");
}
if(e.getSource()==delButton) {
    String string = textfield.getText();
    textfield.setText("");
    for(int i=0; i<string.length()-1;i++) {
        textfield.setText(textfield.getText()+string.charAt(i));
    }
}
}
}
}

```

OUTPUT:

