A cover of a book

AI-generated content may be incorrect.

**Ejercicio 1**

Motocicletas\_RCM

Programa para mostrar el precio de diferentes marcas de motos, junto con un descuento y el total a pagar.

// Importamos la clase JOptionPane para usar ventanas de entrada/salida gráficas

import javax.swing.JOptionPane;

public class Motocicletas\_RCM { // Clase principal con nombre solicitado por la actividad

    public static void main(String[] args) {

        boolean continuar = true; // Variable para controlar si el menú se sigue mostrando

        // Ciclo que se repite mientras el usuario quiera cotizar otra moto

        while (continuar) {

            // Mostramos el menú de marcas y obtenemos la opción elegida

            String opcionOverlord = JOptionPane.showInputDialog(

                "Seleccione la marca de motocicleta:\n" +

                "1. Honda ($45,000)\n" +

                "2. Yamaha ($50,000)\n" +

                "3. Suzuki ($48,000)\n" +

                "4. Otra marca ($42,000)\n\n" +

                "Presione Cancelar o cierre la ventana para salir."

            );

            // Si el usuario presiona Cancelar o cierra la ventana, se termina el programa

            if (opcionOverlord == null) {

                continuar = false;

                JOptionPane.showMessageDialog(null, "Gracias por usar el sistema de cotizacion. Hasta pronto!");

                break;

            }

            int opcion;

            // Intentamos convertir el texto ingresado en número

            try {

                opcion = Integer.parseInt(opcionOverlord);

            } catch (NumberFormatException e) {

                // Si lo ingresado no es un número válido, mostramos mensaje y reiniciamos el ciclo

                JOptionPane.showMessageDialog(null, "Entrada invalida. Por favor, ingrese un numero del 1 al 4.");

                continue;

            }

            // Declaramos variables con nombres personalizados

            double precioPulgoso = 0;        // Precio base de la moto

            double descuentoSpice = 0;       // Monto del descuento aplicado

            double totalFinalRafa = 0;       // Precio final después del descuento

            String marcaSeleccionada = "";   // Nombre de la marca seleccionada

            switch (opcion) {

                case 1:

                    marcaSeleccionada = "Honda";

                    precioPulgoso = 45000;

                    descuentoSpice = precioPulgoso \* 0.05;

                    break;

                case 2:

                    marcaSeleccionada = "Yamaha";

                    precioPulgoso = 50000;

                    descuentoSpice = precioPulgoso \* 0.08;

                    break;

                case 3:

                    marcaSeleccionada = "Suzuki";

                    precioPulgoso = 48000;

                    descuentoSpice = precioPulgoso \* 0.10;

                    break;

                case 4:

                    marcaSeleccionada = "Otra marca";

                    precioPulgoso = 42000;

                    descuentoSpice = precioPulgoso \* 0.02;

                    break;

                default:

                    // Si el número no está entre 1 y 4, avisamos al usuario

                    JOptionPane.showMessageDialog(null, "Opción no valida. Intente de nuevo.");

                    continue;

            }

            // Calculamos el precio total final después del descuento

            totalFinalRafa = precioPulgoso - descuentoSpice;

            // Mostramos el desglose al usuario: precio, descuento, total a pagar

            JOptionPane.showMessageDialog(null,

                "Marca seleccionada: " + marcaSeleccionada + "\n" +

                "Precio de la moto: $" + String.format("%.2f", precioPulgoso) + "\n" +

                "Descuento aplicado: $" + String.format("%.2f", descuentoSpice) + "\n" +

                "Total a pagar: $" + String.format("%.2f", totalFinalRafa));

        }

    }

}

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Ejercicio 2**

ArregloBidi\_RCM

Usando un loop, se genera un arreglo bidimensional con 2 matrices, una de 3x3 y una que tengo una estructura de 3-2-3 valores por columna.

import javax.swing.JOptionPane;

public class ArregloBidi\_RCM {

    public static void main(String[] args) {

        // Matriz fija 3x2

        int[][] matrizPulgoso = {

            {12, 85},

            {34, 65},

            {99, 21}

        };

        // Matriz irregular 3x3 (con longitudes distintas)

        int[][] matrizSpice = new int[3][];

        matrizSpice[0] = new int[]{15, 37, 88};     // 3 valores

        matrizSpice[1] = new int[]{44, 62};         // 2 valores

        matrizSpice[2] = new int[]{109, 3, 74};     // 3 valores

        // Mostrar ambas matrices usando el método personalizado

        String salidaPulgoso = mostrarMatriz("matrizPulgoso", matrizPulgoso);

        String salidaSpice = mostrarMatriz("matrizSpice", matrizSpice);

        // Mostrar en ventanas emergentes

        JOptionPane.showMessageDialog(null, "Primera matriz (3x2):\n" + salidaPulgoso);

        JOptionPane.showMessageDialog(null, "Segunda matriz irregular:\n" + salidaSpice);

    }

    // Método para recorrer e imprimir cualquier matriz bidimensional como String

    public static String mostrarMatriz(String nombre, int[][] matrizRafa) {

        StringBuilder resultadofinalfinal = new StringBuilder();

        for (int i = 0; i < matrizRafa.length; i++) {

            resultadofinalfinal.append(nombre).append("[").append(i).append("] = ");

            for (int j = 0; j < matrizRafa[i].length; j++) {

                resultadofinalfinal.append(matrizRafa[i][j]).append(" ");

            }

            resultadofinalfinal.append("\n");

        }

        return resultadofinalfinal.toString();

    }

}

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Ejercicio 3**

Orquesta\_RCM

Aplicación de conceptos de herencia, polimorfismo y encapsulamiento

// Importamos la clase JOptionPane de Swing para mostrar ventanas emergentes

import javax.swing.JOptionPane;

// ==============================

// CLASE BASE: Instrumento

// ==============================

// Clase abstracta que representa un instrumento musical genérico

// Aquí aplicamos HERENCIA, ya que otras clases (viento, cuerda, percusión) heredarán de esta

abstract class Instrumento {

    // Atributos privados (ENCAPSULAMIENTO), accesibles solo por métodos de la clase

    private String nombre;

    private String tipo;

    private String material;

    private String barrio;

    // Constructor con parámetros para inicializar todos los atributos del instrumento

    public Instrumento(String nombre, String tipo, String material, String barrio) {

        this.nombre = nombre;

        this.tipo = tipo;

        this.material = material;

        this.barrio = barrio;

    }

    // Métodos públicos para acceder a los atributos privados (getters)

    // Aquí reforzamos el ENCAPSULAMIENTO

    public String getNombre() { return nombre; }

    public String getTipo() { return tipo; }

    public String getMaterial() { return material; }

    public String getbarrio() { return barrio; }

    // Método abstracto que será implementado de forma distinta por cada tipo de instrumento

    // Esto es POLIMORFISMO: cada clase hija lo definirá a su manera

    public abstract String tocar();

}

// ==============================

// SUBCLASE: InstrumentoViento

// ==============================

// Esta clase representa instrumentos de viento (como flauta, clarinete, etc.)

// Hereda de Instrumento

class InstrumentoViento extends Instrumento {

    // Constructor específico que fija el tipo como "Viento"

    public InstrumentoViento(String nombre, String material, String barrio) {

        super(nombre, "Viento", material, barrio);

    }

    // Implementación del método abstracto tocar (POLIMORFISMO)

    @Override

    public String tocar() {

        return "🎵 El instrumento de viento '" + getNombre() + "' sopla con fuerza y elegancia.";

    }

}

// ==============================

// SUBCLASE: InstrumentoCuerda

// ==============================

class InstrumentoCuerda extends Instrumento {

    // Constructor que fija el tipo como "Cuerda"

    public InstrumentoCuerda(String nombre, String material, String barrio) {

        super(nombre, "Cuerda", material, barrio);

    }

    // Cada clase hija implementa su propia versión del método tocar

    @Override

    public String tocar() {

        return "🎻 El instrumento de cuerda '" + getNombre() + "' vibra con notas armoniosas.";

    }

}

// ==============================

// SUBCLASE: InstrumentoPercusion

// ==============================

class InstrumentoPercusion extends Instrumento {

    // Constructor que fija el tipo como "Percusión"

    public InstrumentoPercusion(String nombre, String material, String barrio) {

        super(nombre, "Percusión", material, barrio);

    }

    // Otra versión del método tocar, distinta a viento y cuerda

    @Override

    public String tocar() {

        return "🥁 El instrumento de percusión '" + getNombre() + "' retumba con ritmo poderoso.";

    }

}

// ==============================

// CLASE PRINCIPAL: Orquesta\_RCM

// ==============================

public class Orquesta\_RCM {

    // Método principal que ejecuta el programa

    public static void main(String[] args) {

        // Creamos un arreglo de tipo Instrumento (la clase base)

        // Aquí usamos POLIMORFISMO: podemos guardar cualquier subclase (viento, cuerda o percusión)

        Instrumento[] instrumentosOverlord = new Instrumento[3];

        // Creamos cada instrumento con sus propios datos y lo guardamos en el arreglo

        // Usamos nombres personalizados como Pulgoso, Spice y Octubre(me quede sin inspiracion) como pide el ejercicio

        instrumentosOverlord[0] = new InstrumentoViento("Flauta Pulgoso", "Madera", "Alemania");

        instrumentosOverlord[1] = new InstrumentoCuerda("Violín Spice", "Caoba", "Italia");

        instrumentosOverlord[2] = new InstrumentoPercusion("Timbal Octubre", "Bronce", "África");

        // Usamos un StringBuilder para concatenar el texto que vamos a mostrar

        StringBuilder salida = new StringBuilder();

        // Recorremos el arreglo usando un ciclo for-each (INSTRUCCIÓN DE REPETICIÓN)

        // Aquí también se aplica POLIMORFISMO: el método tocar() se comporta diferente según la clase

        for (Instrumento rafaInstrumento : instrumentosOverlord) {

            salida.append("Instrumento: ").append(rafaInstrumento.getNombre()).append("\n")

                  .append("Tipo: ").append(rafaInstrumento.getTipo()).append("\n")

                  .append("Material: ").append(rafaInstrumento.getMaterial()).append("\n")

                  .append("Origen: ").append(rafaInstrumento.getbarrio()).append("\n")

                  .append(rafaInstrumento.tocar()).append("\n\n");

        }

        // Mostramos toda la información en una ventana emergente con Swing

        // Así no usamos la consola, como pide el ejercicio

        JOptionPane.showMessageDialog(null, salida.toString(), "Orquesta RCM", JOptionPane.INFORMATION\_MESSAGE);

    }

}

A screenshot of a computer

AI-generated content may be incorrect.

**Ejercicio 4**

Excepcion\_RCM

Muestra una ventana para que el usuario ingrese un número entre 1 y 10.

- Si se introduce una letra o algo que no es número, se lanza un `NumberFormatException`.

- Si el número está fuera del rango permitido, se lanza una excepción personalizada con mensaje.

- Si el usuario presiona "Cancelar", se lanza una excepción hecha por nosotros (`CancelarOperacionException`) y se informa que se canceló la operación.

- Si todo está bien, muestra un mensaje de éxito con el número ingresado.

import javax.swing.JOptionPane;

// Clase principal

public class Excepcion\_RCM {

    public static void main(String[] args) {

        try {

            // Llamamos al método que puede lanzar una excepción

            int valorIngresado = pedirNumeroRafa();

            JOptionPane.showMessageDialog(null, "¡Número aceptado! Elegiste: " + valorIngresado);

        } catch (NumberFormatException e) {

            JOptionPane.showMessageDialog(null, "⚠️ Eso no era un número válido. Intenta con un número entre 1 y 10.");

        } catch (CancelarOperacionException e) {

            JOptionPane.showMessageDialog(null, "🚪 Has cancelado la operación. ¡Hasta luego!");

        } catch (Exception e) {

            JOptionPane.showMessageDialog(null, "🚫 Error: " + e.getMessage());

        }

    }

    // Método que pide un número y puede lanzar varias excepciones

    public static int pedirNumeroRafa() throws Exception {

        String entradaSpice = JOptionPane.showInputDialog("Introduce un número entre 1 y 10:");

        // Si el usuario cancela (presiona Cancelar o cierra la ventana)

        if (entradaSpice == null) {

            throw new CancelarOperacionException("Operación cancelada por el usuario.");

        }

        // Posible NumberFormatException si no es número

        int numeroPulgosito = Integer.parseInt(entradaSpice);

        // Validamos si está en el rango correcto

        if (numeroPulgosito < 1 || numeroPulgosito > 10) {

            throw new Exception("El número " + numeroPulgosito + " está fuera del rango permitido.");

        }

        return numeroPulgosito;

    }

}

// Excepción personalizada para manejar cancelaciones del usuario

class CancelarOperacionException extends Exception {

    public CancelarOperacionException(String mensaje) {

        super(mensaje);

    }

}

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer error message

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer error message

AI-generated content may be incorrect.

**Referencias**

Eckel, B. (2006). *Thinking in Java* (4th ed.). Prentice Hall.

Sierra, K., & Bates, B. (2005). *Head First Java* (2nd ed.). O'Reilly Media.

Deitel, P. J., & Deitel, H. M. (2017). *Java: How to Program* (11th ed.). Pearson Education.

Oracle. (n.d.). *The Java™ Tutorials: Creating a GUI With Swing*. <https://docs.oracle.com/javase/tutorial/uiswing/>

Oracle. (n.d.). *The Java™ Tutorials: Object-Oriented Programming Concepts*. <https://docs.oracle.com/javase/tutorial/java/concepts/index.html>