

PROGRAMACIÓN II

Trabajo Práctico 2: Programación Estructurada

OBJETIVO GENERAL

Desarrollar habilidades en programación estructurada en Java, abordando desde conceptos básicos como operadores y estructuras de control hasta temas avanzados como funciones, recursividad y estructuras de datos. Se busca fortalecer la capacidad de análisis y solución de problemas mediante un enfoque práctico,

MARCO TEÓRICO

Concepto	Aplicación en el proyecto
Estructuras condicionales	Clasificación de edad, verificación de año bisiesto
Ciclos (for, while, do-while)	Repetición de ingreso de datos y cálculos
Funciones	Cálculo modular de descuentos, envíos, stock
Arrays	Gestión de precios de productos
Recursividad	Impresión recursiva de arrays

CONCLUSIONES ESPERADAS

- Aplicar estructuras de control y decisión para resolver problemas.
- Diseñar soluciones usando estructuras iterativas y condicionales.
- Modularizar el código utilizando funciones con y sin retorno.
- Utilizar arrays para almacenamiento y manipulación de datos.
- Comprender y aplicar la recursividad en casos simples.
- Trabajar con variables locales y globales de forma adecuada.
- Fortalecer la capacidad de análisis lógico y la resolución de errores.
- Consolidar el uso del lenguaje Java mediante la práctica estructurada.

Caso Práctico

Desarrollar los siguientes ejercicios en Java utilizando el paradigma de programación estructurada. Agrupados según el tipo de estructuras o conceptos aplicados:

Estructuras Condicionales:

1. Verificación de Año Bisiesto.

Escribe un programa en Java que solicite al usuario un año y determine si es bisiesto. Un año es bisiesto si es divisible por 4, pero no por 100, salvo que sea divisible por 400.

Ejemplo de entrada/salida:

Ingrese un año: 2024

El año 2024 es bisiesto.

Ingrese un año: 1900

El año 1900 no es bisiesto.

```
import java.util.Scanner;

public class VerificacionAnioBisiesto {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in); // Crear un objeto Scanner para leer la entrada del usuario

        // Solicitar al usuario que ingrese un año
        System.out.print("Ingrese un año: ");
        int anio = scanner.nextInt(); // Leer el año ingresado

        // Verificar si el año es bisiesto
        boolean esBisiesto = (anio % 4 == 0 && anio % 100 != 0) || (anio % 400 == 0);

        // Mostrar el resultado
        if (esBisiesto) {
            System.out.println("El año " + anio + " es bisiesto.");
        } else {
            System.out.println("El año " + anio + " no es bisiesto.");
        }

        // Cerrar el scanner
        scanner.close();
    }
}

VerificacionAnioBisiesto > main >
Practico 2 (run) x
run:
Ingrese un año: 2025
El año 2025 no es bisiesto.
BUILD SUCCESSFUL (total time: 30 seconds)
```

2. Determinar el Mayor de Tres Números.

Escribe un programa en Java que pida al usuario tres números enteros y determine cuál es el mayor.

Ejemplo de entrada/salida:

Ingrese el primer número: 8

Ingrese el segundo número: 12

Ingrese el tercer número: 5

El mayor es: 12

```
import java.util.Scanner;

public class DeterminarMayor {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in); // Crear un objeto Scanner para leer la entrada del usuario

        // Solicitar al usuario que ingrese tres números
        System.out.print("Ingrese el primer numero: ");
        int numero1 = scanner.nextInt(); // Leer el primer número

        System.out.print("Ingrese el segundo numero: ");
        int numero2 = scanner.nextInt(); // Leer el segundo número

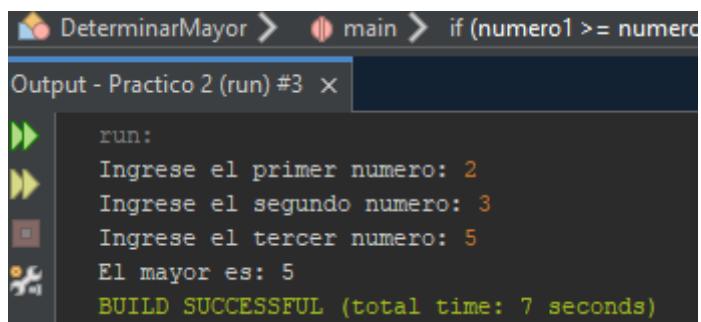
        System.out.print("Ingrese el tercer numero: ");
        int numero3 = scanner.nextInt(); // Leer el tercer número

        // Determinar cuál es el mayor
        int mayor;

        if (numero1 >= numero2 && numero1 >= numero3) {
            mayor = numero1;
        } else if (numero2 >= numero1 && numero2 >= numero3) {
            mayor = numero2;
        } else {
            mayor = numero3;
        }

        // Mostrar el resultado
        System.out.println("El mayor es: " + mayor);

        // Cerrar el scanner
        scanner.close();
    }
}
```



```
DeterminarMayor > main > if (numero1 >= numero3)
Output - Practico 2 (run) #3
run:
Ingrese el primer numero: 2
Ingrese el segundo numero: 3
Ingrese el tercer numero: 5
El mayor es: 5
BUILD SUCCESSFUL (total time: 7 seconds)
```

3. Clasificación de Edad.

Escribe un programa en Java que solicite al usuario su edad y clasifique su etapa de vida según la siguiente tabla:

Menor de 12 años: "Niño"

Entre 12 y 17 años: "Adolescente"

Entre 18 y 59 años: "Adulto"

60 años o más: "Adulto mayor"

Ejemplo de entrada/salida:

Ingrese su edad: 25

Eres un Adulto.

Ingrese su edad: 10

Eres un Niño.

```
import java.util.Scanner;

public class ClasificacionEdad {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in); // Crear un objeto Scanner para leer la entrada del usuario

        // Solicitar al usuario que ingrese su edad
        System.out.print("Ingrese su edad: ");
        int edad = scanner.nextInt(); // Leer la edad ingresada

        // Clasificar la etapa de vida
        if (edad < 12) {
            System.out.println("Eres un Niño.");
        } else if (edad >= 12 && edad <= 17) {
            System.out.println("Eres un Adolescente.");
        } else if (edad >= 18 && edad <= 59) {
            System.out.println("Eres un Adulto.");
        } else {
            System.out.println("Eres un Adulto mayor.");
        }

        // Cerrar el scanner
        scanner.close();
    }
}
```

ClasificacionEdad > main >

put - Practico 2 (run) #3 x

run:
Ingrese su edad: 42
Eres un Adulto.
BUILD SUCCESSFUL (total time: 4 seconds)

4. Calculadora de Descuento según categoría.

Escribe un programa que solicite al usuario el precio de un producto y su categoría (A, B o C).

Luego, aplique los siguientes descuentos:

Categoría A: 10% de descuento

Categoría B: 15% de descuento

Categoría C: 20% de descuento

El programa debe mostrar el precio original, el descuento aplicado y el precio final

Ejemplo de entrada/salida:

Ingrese el precio del producto: 1000

Ingrese la categoría del producto (A, B o C): B

Descuento aplicado: 15%

Precio final: 850.0

```
* @author veron
*/
import java.util.Scanner;

public class CalculadoraDescuento {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in); // Crear un objeto Scanner para leer la entrada del usuario

        // Solicitar el precio del producto
        System.out.print("Ingrese el precio del producto: ");
        double precioOriginal = scanner.nextDouble(); // Leer el precio

        // Solicitar la categoría del producto
        System.out.print("Ingrese la categoría del producto (A, B o C): ");
        char categoria = scanner.next().charAt(0); // Leer la categoría

        double descuento = 0; // Variable para almacenar el valor del descuento

        // Determinar el descuento según la categoría
        switch (Character.toUpperCase(categoria)) {
            case 'A':
                descuento = 0.10; // 10% de descuento
                break;
            case 'B':
                descuento = 0.15; // 15% de descuento
                break;
            case 'C':
                descuento = 0.20; // 20% de descuento
                break;
            default:
                System.out.println("Categoría no válida. No se aplicará descuento.");
                break;
        }

        // Calcular el precio final si la categoría es válida
        double precioFinal = precioOriginal; // Inicializa el precio final con el precio original

        if (descuento > 0) {
            double descuentoAplicado = precioOriginal * descuento; // Calcular el descuento aplicado
            precioFinal = precioOriginal - descuentoAplicado; // Calcular el precio final
            System.out.printf("Descuento aplicado: %.0f%%\n", descuento * 100); // Mostrar el porcentaje de descuento
            System.out.printf("Precio final: %.2f\n", precioFinal); // Mostrar el precio final
        } else {
            // Para categoría inválida, solo muestra el precio original
            System.out.printf("El precio original es: %.2f\n", precioOriginal);
        }

        // Cerrar el scanner
        scanner.close();
    }
}
```



```
CalculadoraDescuento > main > switch (Character.toUpperCas
Output - Practico 2 (run) #6 x
run:
Ingrese el precio del producto: 100
Ingrese la categoria del producto (A, B o C): A
Descuento aplicado: 10%
Precio final: 90,00
BUILD SUCCESSFUL (total time: 8 seconds)
```

```
CalculadoraDescuento > main > switch (Character.toUpperCas
Output - Practico 2 (run) #6 x
run:
Ingrese el precio del producto: 100
Ingrese la categoria del producto (A, B o C): B
Descuento aplicado: 15%
Precio final: 85,00
BUILD SUCCESSFUL (total time: 5 seconds)
```

```
CalculadoraDescuento > main > switch (Character.toUpperCas
Output - Practico 2 (run) #6 x
run:
Ingrese el precio del producto: 100
Ingrese la categoria del producto (A, B o C): c
Descuento aplicado: 20%
Precio final: 80,00
BUILD SUCCESSFUL (total time: 7 seconds)
```

Estructuras de Repetición:

5. Suma de Números Pares (while).

Escribe un programa que solicite números al usuario y sume solo los números pares. El ciclo debe continuar hasta que el usuario ingrese el número 0, momento en el que se debe mostrar la suma total de los pares ingresados.

Ejemplo de entrada/salida:

Ingrese un número (0 para terminar): 4

Ingrese un número (0 para terminar): 7

Ingrese un número (0 para terminar): 2

Ingrese un número (0 para terminar): 0

La suma de los números pares es: 6

```
import java.util.Scanner;

public class SumaNumerosPares {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in); // Crear un objeto Scanner para leer la entrada del usuario
        int sumaPares = 0; // Variable para almacenar la suma de los números pares
        int numero; // Variable para almacenar el número ingresado por el usuario

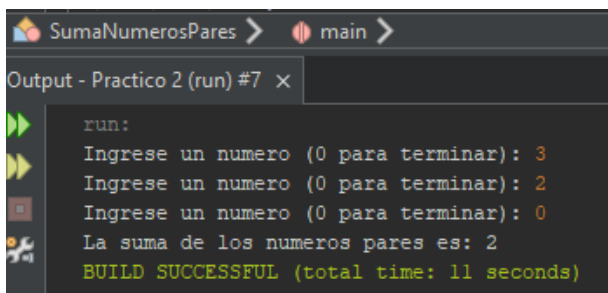
        // Ciclo while que solicita números al usuario
        while (true) {
            System.out.print("Ingrese un numero (0 para terminar): ");
            numero = scanner.nextInt(); // Leer el número ingresado

            if (numero == 0) { // Si el usuario ingresa 0, salir del ciclo
                break;
            }

            // Verificar si el número es par
            if (numero % 2 == 0) {
                sumaPares += numero; // Sumar el número a la suma de los pares
            }
        }

        // Mostrar la suma total de los números pares ingresados
        System.out.println("La suma de los numeros pares es: " + sumaPares);

        // Cerrar el scanner
        scanner.close();
    }
}
```



```
SumaNumerosPares > main >
Output - Practico 2 (run) #7 x
run:
Ingrese un numero (0 para terminar): 3
Ingrese un numero (0 para terminar): 2
Ingrese un numero (0 para terminar): 0
La suma de los numeros pares es: 2
BUILD SUCCESSFUL (total time: 11 seconds)
```

6. Contador de Positivos, Negativos y Ceros (for).

Escribe un programa que pida al usuario ingresar 10 números enteros y cuente cuántos son positivos, negativos y cuántos son ceros.

Ejemplo de entrada/salida:

```
Ingrese el número 1: -5
Ingrese el número 2: 3
Ingrese el número 3: 0
Ingrese el número 4: -1
Ingrese el número 5: 6
Ingrese el número 6: 0
Ingrese el número 7: 9
Ingrese el número 8: -3
Ingrese el número 9: 4
Ingrese el número 10: -8

Resultados:
Positivos: 4
Negativos: 4
Ceros: 2
```

```
import java.util.Scanner;

public class ContadorNumeros {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in); // Crear un objeto Scanner para leer la entrada del usuario

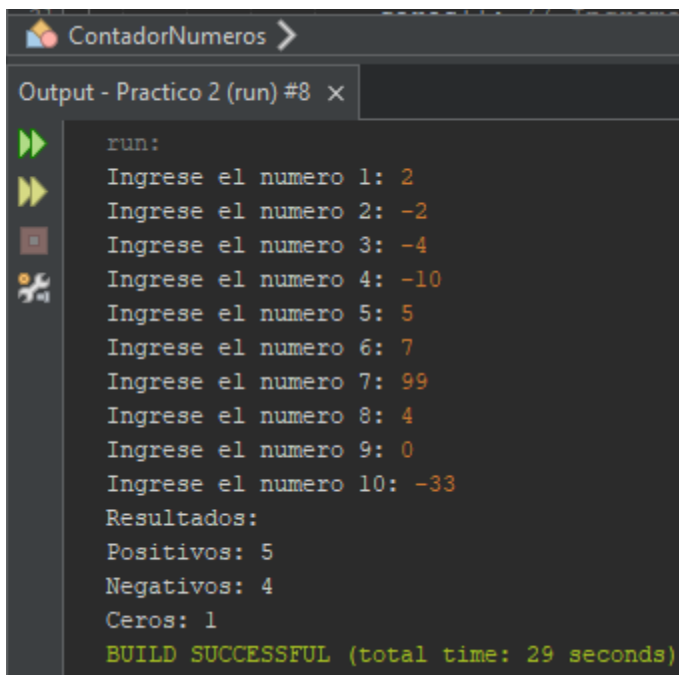
        int positivos = 0; // Contador de números positivos
        int negativos = 0; // Contador de números negativos
        int ceros = 0; // Contador de ceros

        // Ciclo for para solicitar 10 números
        for (int i = 1; i <= 10; i++) {
            System.out.print("Ingrese el numero " + i + ": ");
            int numero = scanner.nextInt(); // Leer el número ingresado

            // Clasificar el número ingresado
            if (numero > 0) {
                positivos++; // Incrementar contador de positivos
            } else if (numero < 0) {
                negativos++; // Incrementar contador de negativos
            } else {
                ceros++; // Incrementar contador de ceros
            }
        }

        // Mostrar los resultados
        System.out.println("Resultados:");
        System.out.println("Positivos: " + positivos);
        System.out.println("Negativos: " + negativos);
        System.out.println("Ceros: " + ceros);

        // Cerrar el scanner
        scanner.close();
    }
}
```



```
run:
Ingrese el numero 1: 2
Ingrese el numero 2: -2
Ingrese el numero 3: -4
Ingrese el numero 4: -10
Ingrese el numero 5: 5
Ingrese el numero 6: 7
Ingrese el numero 7: 99
Ingrese el numero 8: 4
Ingrese el numero 9: 0
Ingrese el numero 10: -33
Resultados:
Positivos: 5
Negativos: 4
Ceros: 1
BUILD SUCCESSFUL (total time: 29 seconds)
```


7. Validación de Nota entre 0 y 10 (do-while).

Escribe un programa que solicite al usuario una nota entre 0 y 10. Si el usuario ingresa un número fuera de este rango, debe seguir pidiéndole la nota hasta que ingrese un valor válido.

Ejemplo de entrada/salida:

Ingrese una nota (0-10): 15

Error: Nota inválida. Ingrese una nota entre 0 y 10.

Ingrese una nota (0-10): -2

Error: Nota inválida. Ingrese una nota entre 0 y 10.

Ingrese una nota (0-10): 8

Nota guardada correctamente.

```
import java.util.Scanner;

public class ValidacionNota {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in); // Crear un objeto Scanner para leer la entrada del usuario
        double nota; // Variable para almacenar la nota ingresada

        do {
            // Solicitar al usuario que ingrese una nota
            System.out.print("Ingrese una nota (0-10): ");
            nota = scanner.nextDouble(); // Leer la nota ingresada

            // Verificar si la nota es válida
            if (nota < 0 || nota > 10) {
                System.out.println("Error: Nota inválida. Ingrese una nota entre 0 y 10.");
            }

        } while (nota < 0 || nota > 10); // Continuar pidiendo la nota hasta que sea válida

        // Nota guardada correctamente
        System.out.println("Nota guardada correctamente.");

        // Cerrar el scanner
        scanner.close();
    }
}
```

ValidationNota > main > do ... while (nota < 0 || nota > 10) > if (

Output - Practico 2 (run) #9 x

```
run:
Ingrese una nota (0-10): 12
Error: Nota invalida. Ingrese una nota entre 0 y 10.
Ingrese una nota (0-10): -2
Error: Nota invalida. Ingrese una nota entre 0 y 10.
Ingrese una nota (0-10): 8
Nota guardada correctamente.
BUILD SUCCESSFUL (total time: 12 seconds)
```

Funciones:

8. Cálculo del Precio Final con impuesto y descuento.

Crea un método `calcularPrecioFinal(double impuesto, double descuento)` que calcule el precio final de un producto en un e-commerce. La fórmula es:

PrecioFinal = PrecioBase + (PrecioBase×Impuesto) – (PrecioBase×Descuento)
PrecioFinal = PrecioBase + (PrecioBase \times Impuesto) - (PrecioBase \times Descuento)

Desde `main()`, solicita el precio base del producto, el porcentaje de impuesto y el porcentaje de descuento, llama al método y muestra el precio final.

Ejemplo de entrada/salida:

Ingrese el precio base del producto: 100

Ingrese el impuesto en porcentaje (Ejemplo: 10 para 10%): 10

Ingrese el descuento en porcentaje (Ejemplo: 5 para 5%): 5

El precio final del producto es: 105.0

```
public class CalculoPrecioFinal {

    // Método para calcular el precio final
    public static double calcularPrecioFinal(double precioBase, double impuesto, double descuento) {
        double precioFinal = precioBase + (precioBase * impuesto / 100) - (precioBase * descuento / 100);
        return precioFinal;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in); // Crear un objeto Scanner para leer la entrada del usuario

        // Solicitar el precio base del producto
        System.out.print("Ingrese el precio base del producto: ");
        double precioBase = scanner.nextDouble(); // Leer el precio base

        // Solicitar el porcentaje de impuesto
        System.out.print("Ingrese el impuesto en porcentaje (Ejemplo: 10 para 10%): ");
        double impuesto = scanner.nextDouble(); // Leer el impuesto

        // Solicitar el porcentaje de descuento
        System.out.print("Ingrese el descuento en porcentaje (Ejemplo: 5 para 5%): ");
        double descuento = scanner.nextDouble(); // Leer el descuento

        // Calcular el precio final llamando al método
        double precioFinal = calcularPrecioFinal(precioBase, impuesto, descuento);

        // Mostrar el precio final
        System.out.println("El precio final del producto es: " + precioFinal);

        // Cerrar el scanner
        scanner.close();
    }
}
```

```

CalculoPrecioFinal >
Output - Practico 2 (run) #9 x
run:
Ingrese el precio base del producto: 100
Ingrese el impuesto en porcentaje (Ejemplo: 10 para 10%): 10
Ingrese el descuento en porcentaje (Ejemplo: 5 para 5%): 5
El precio final del producto es: 105.0
BUILD SUCCESSFUL (total time: 14 seconds)

```

9. Composición de funciones para calcular costo de envío y total de compra.

a. **calcularCostoEnvio(double peso, String zona)**: Calcula el costo de envío basado en la zona de envío (Nacional o Internacional) y el peso del paquete.

Nacional: \$5 por kg

Internacional: \$10 por kg

b. **calcularTotalCompra(double precioProducto, double costoEnvio)**: Usa **calcularCostoEnvio** para sumar el costo del producto con el costo de envío.

Desde **main()**, solicita el peso del paquete, la zona de envío y el precio del producto. Luego, muestra el total a pagar.

Ejemplo de entrada/salida:

Ingrese el precio del producto: 50

Ingrese el peso del paquete en kg: 2

Ingrese la zona de envío (Nacional/Internacional): Nacional

El costo de envío es: 10.0

El total a pagar es: 60.0

```

import java.util.Scanner;

public class CostoEnvioCompra {

    // Método para calcular el costo de envío
    public static double calcularCostoEnvio(double peso, String zona) {
        double costoEnvio = 0;
        // Determinar costo de envío según la zona
        if (zona.equalsIgnoreCase("Nacional")) {
            costoEnvio = 5 * peso; // $5 por kg
        } else if (zona.equalsIgnoreCase("Internacional")) {
            costoEnvio = 10 * peso; // $10 por kg
        } else {
            System.out.println("Zona de envío no valida.");
        }
        return costoEnvio;
    }
}

```

```
// Método para calcular el total de la compra
public static double calcularTotalCompra(double precioProducto, double costoEnvio) {
    return precioProducto + costoEnvio; // Sumar precio del producto y costo de envío
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in); // Crear un objeto Scanner para leer la entrada del usuario

    // Solicitar el precio del producto
    System.out.print("Ingrese el precio del producto: ");
    double precioProducto = scanner.nextDouble(); // Leer el precio del producto

    // Solicitar el peso del paquete
    System.out.print("Ingrese el peso del paquete en kg: ");
    double peso = scanner.nextDouble(); // Leer el peso del paquete

    // Solicitar la zona de envío
    System.out.print("Ingrese la zona de envío (Nacional/Internacional): ");
    String zona = scanner.next(); // Leer la zona de envío

    // Calcular el costo de envío
    double costoEnvio = calcularCostoEnvio(peso, zona);
```

```
// Comprobar si el costo de envío es válido
if (costoEnvio >= 0) {
    // Mostrar el costo de envío
    System.out.printf("El costo de envío es: %.2f\n", costoEnvio);

    // Calcular el total a pagar
    double totalAPagar = calcularTotalCompra(precioProducto, costoEnvio);

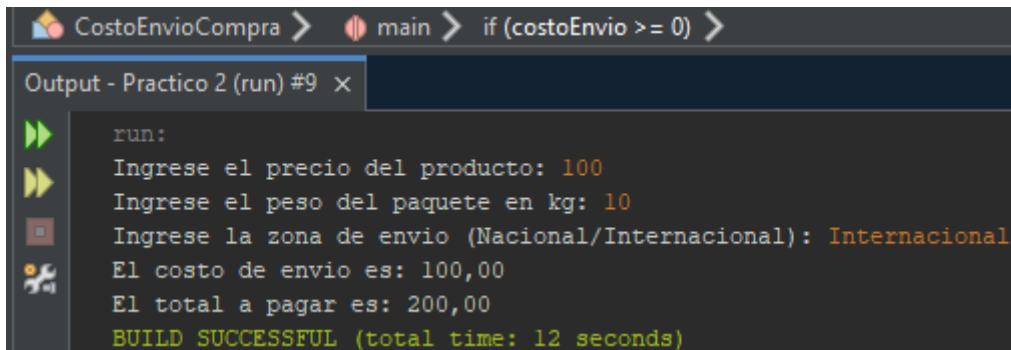
    // Mostrar el total a pagar
    System.out.printf("El total a pagar es: %.2f\n", totalAPagar);
}

// Cerrar el scanner
scanner.close();
```

CostoEnvioCompra > main > if (costoEnvio >= 0) >

Output - Practico 2 (run) #9 x

```
run:
Ingrese el precio del producto: 100
Ingrese el peso del paquete en kg: 10
Ingrese la zona de envío (Nacional/Internacional): Nacional
El costo de envío es: 50,00
El total a pagar es: 150,00
BUILD SUCCESSFUL (total time: 12 seconds)
```



```
CostoEnvioCompra > main > if (costoEnvio >= 0) >
Output - Practico 2 (run) #9 x
run:
Ingrese el precio del producto: 100
Ingrese el peso del paquete en kg: 10
Ingrese la zona de envio (Nacional/Internacional): Internacional
El costo de envio es: 100,00
El total a pagar es: 200,00
BUILD SUCCESSFUL (total time: 12 seconds)
```

10. Actualización de stock a partir de venta y recepción de productos.

Crea un método `actualizarStock(int stockActual, int cantidadVendida, int cantidadRecibida)`, que calcule el nuevo stock después de una venta y recepción

de productos:

NuevoStock = StockActual - CantidadVendida + CantidadRecibida

NuevoStock = CantidadVendida + CantidadRecibida

Desde `main()`, solicita al usuario el stock actual, la cantidad vendida y la cantidad recibida, y muestra el stock actualizado.

Ejemplo de entrada/salida:

Ingrese el stock actual del producto: 50

Ingrese la cantidad vendida: 20

Ingrese la cantidad recibida: 30

El nuevo stock del producto es: 60

```
import java.util.Scanner;

public class ActualizacionStock {

    // Método para actualizar el stock
    public static int actualizarStock(int stockActual, int cantidadVendida, int cantidadRecibida) {
        return stockActual - cantidadVendida + cantidadRecibida; // Calcular nuevo stock
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in); // Crear un objeto Scanner para leer la entrada del usuario

        // Solicitar el stock actual
        System.out.print("Ingrese el stock actual del producto: ");
        int stockActual = scanner.nextInt(); // Leer el stock actual

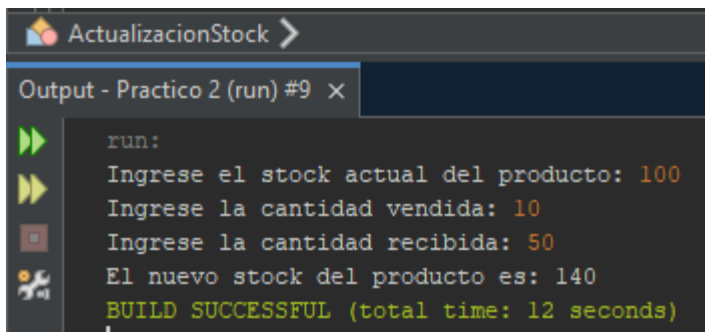
        // Solicitar la cantidad vendida
        System.out.print("Ingrese la cantidad vendida: ");
        int cantidadVendida = scanner.nextInt(); // Leer la cantidad vendida

        // Solicitar la cantidad recibida
        System.out.print("Ingrese la cantidad recibida: ");
        int cantidadRecibida = scanner.nextInt(); // Leer la cantidad recibida

        // Calcular el nuevo stock llamando al método
        int nuevoStock = actualizarStock(stockActual, cantidadVendida, cantidadRecibida);

        // Mostrar el nuevo stock
        System.out.println("El nuevo stock del producto es: " + nuevoStock);

        // Cerrar el scanner
    }
}
```



```
ActualizacionStock >
Output - Practico 2 (run) #9 x
run:
Ingrese el stock actual del producto: 100
Ingrese la cantidad vendida: 10
Ingrese la cantidad recibida: 50
El nuevo stock del producto es: 140
BUILD SUCCESSFUL (total time: 12 seconds)
```

11. Cálculo de descuento especial usando variable global.

Declara una variable global **Ejemplo de entrada/salida:** = 0.10. Luego, crea un método **calcularDescuentoEspecial(double precio)** que use la variable global para calcular el descuento especial del 10%.

Dentro del método, declara una variable local **descuentoAplicado**, almacena el valor del descuento y muestra el precio final con descuento.

Ejemplo de entrada/salida:

Ingrese el precio del producto: 200

El descuento especial aplicado es: 20.0

El precio final con descuento es: 180.0


```
import java.util.Scanner;

public class DescuentoEspecial {

    // Variable global para el descuento
    static final double DESCUENTO_ESPECIAL = 0.10;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

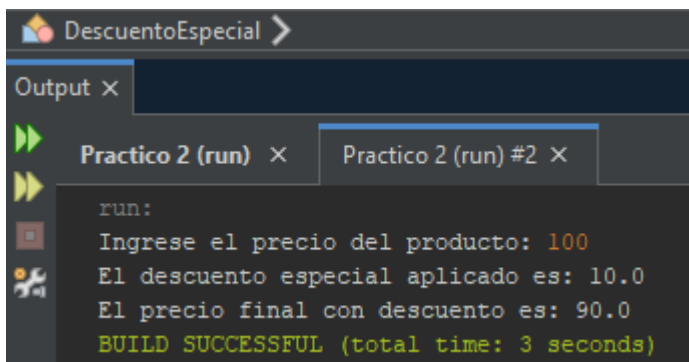
        // Solicitar al usuario que ingrese el precio del producto
        System.out.print("Ingrese el precio del producto: ");
        double precio = scanner.nextDouble();

        // Calcular y mostrar el descuento especial y el precio final
        calcularDescuentoEspecial(precio);
    }

    // Método que calcula el descuento especial
    public static void calcularDescuentoEspecial(double precio) {
        // Variable local para almacenar el descuento aplicado
        double descuentoAplicado = precio * DESCUENTO_ESPECIAL;

        // Precio final con descuento
        double precioFinal = precio - descuentoAplicado;

        // Mostrar resultados
        System.out.println("El descuento especial aplicado es: " + descuentoAplicado);
        System.out.println("El precio final con descuento es: " + precioFinal);
    }
}
```



DescuentoEspecial >

Output X

Practico 2 (run) X Practico 2 (run) #2 X

run:
Ingrese el precio del producto: 100
El descuento especial aplicado es: 10.0
El precio final con descuento es: 90.0
BUILD SUCCESSFUL (total time: 3 seconds)

Arrays y Recursividad:

12. Modificación de un array de precios y visualización de resultados.

Crea un programa que:

- a. Declare e inicialice un array con los precios de algunos productos.
- b. Muestre los valores originales de los precios.
- c. Modifique el precio de un producto específico.
- d. Muestre los valores modificados.

Salida esperada:

Precios originales:

Precio: \$199.99

Precio: \$299.5

Precio: \$149.75

Precio: \$399.0

Precio: \$89.99

Precios modificados:

Precio: \$199.99

Precio: \$299.5

Precio: \$129.99

Precio: \$399.0

Precio: \$89.99

Conceptos Clave Aplicados:

- ✓ Uso de arrays (double[]) para almacenar valores.
- ✓ Recorrido del array con for-each para mostrar valores.
- ✓ Modificación de un valor en un array mediante un índice.
- ✓ Reimpresión del array después de la modificación.

```
* @author veron
*/
public class ModificacionArrayPrecios {

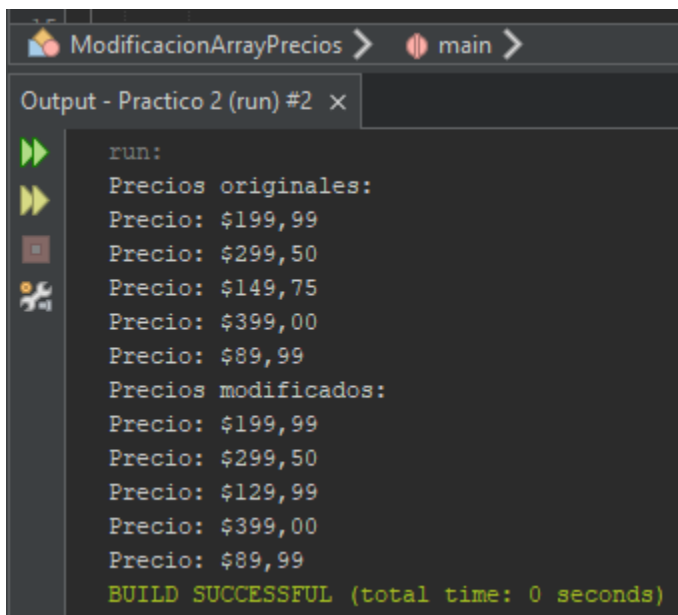
    public static void main(String[] args) {
        // a. Declarar e inicializar un array con los precios de algunos productos
        double[] precios = {199.99, 299.50, 149.75, 399.00, 89.99};

        // b. Mostrar los valores originales de los precios
        System.out.println("Precios originales:");
        mostrarPrecios(precios);

        // c. Modificar el precio de un producto específico
        // Por ejemplo, modificar el tercer producto (índice 2)
        precios[2] = 129.99;

        // d. Mostrar los valores modificados
        System.out.println("Precios modificados:");
        mostrarPrecios(precios);
    }

    // Método para mostrar los precios
    public static void mostrarPrecios(double[] precios) {
        for (double precio : precios) {
            System.out.printf("Precio: $%.2f%n", precio);
        }
    }
}
```



The screenshot shows an IDE window with the file `ModificacionArrayPrecios` and the `main` method selected. The output window, titled "Output - Practico 2 (run) #2", displays the following text:

```
run:
Precios originales:
Precio: $199,99
Precio: $299,50
Precio: $149,75
Precio: $399,00
Precio: $89,99
Precios modificados:
Precio: $199,99
Precio: $299,50
Precio: $129,99
Precio: $399,00
Precio: $89,99
BUILD SUCCESSFUL (total time: 0 seconds)
```

13. Impresión recursiva de arrays antes y después de modificar un elemento.

Crea un programa que:

- Declare e inicialice un array con los precios de algunos productos.
- Use una función recursiva para mostrar los precios originales.
- Modifique el precio de un producto específico.
- Use otra función recursiva para mostrar los valores modificados.

Salida esperada:

Precios originales:

Precio: \$199.99

Precio: \$299.5

Precio: \$149.75

Precio: \$399.0

Precio: \$89.99

Precios modificados:

Precio: \$199.99

Precio: \$299.5

Precio: \$129.99

Precio: \$399.0

Precio: \$89.99

Conceptos Clave Aplicados:

- ✓ Uso de arrays (double[]) para almacenar valores.
- ✓ Recorrido del array con una función recursiva en lugar de un bucle.
- ✓ Modificación de un valor en un array mediante un índice.
- ✓ Uso de un índice como parámetro en la recursión para recorrer el array.

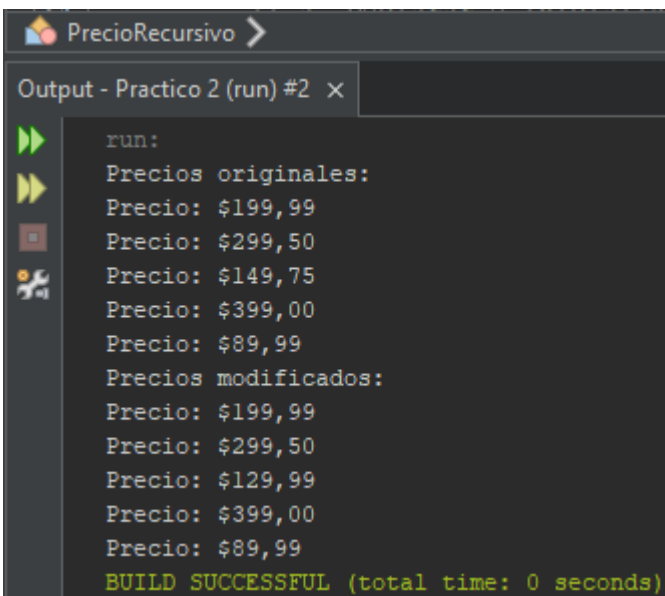
```

public class PrecioRekursivo {

    public static void main(String[] args) {
        // a. Declarar e inicializar un array con algunos precios
        double[] precios = {199.99, 299.50, 149.75, 399.00, 89.99};
        // b. Usar una función recursiva para mostrar los precios originales
        System.out.println("Precios originales:");
        mostrarPreciosOriginales(precios, 0);
        // c. Modificar el precio de un producto específico
        // Por ejemplo, modificar el tercer producto (índice 2)
        precios[2] = 129.99;

        // d. Usar otra función recursiva para mostrar los valores modificados
        System.out.println("Precios modificados:");
        mostrarPreciosModificados(precios, 0);
    }
    // Función recursiva para mostrar precios originales
    public static void mostrarPreciosOriginales(double[] precios, int index) {
        if (index >= precios.length) {
            return; // Fin de la recursión
        }
        System.out.printf("Precio: $%.2f%n", precios[index]); // Mostrar precio
        mostrarPreciosOriginales(precios, index + 1); // Llamada recursiva
    }
    // Función recursiva para mostrar precios modificados
    public static void mostrarPreciosModificados(double[] precios, int index) {
        if (index >= precios.length) {
            return; // Fin de la recursión
        }
        System.out.printf("Precio: $%.2f%n", precios[index]); // Mostrar precio
        mostrarPreciosModificados(precios, index + 1); // Llamada recursiva
    }
}

```



```

PrecioRekursivo >
Output - Practico 2 (run) #2 x
run:
Precios originales:
Precio: $199,99
Precio: $299,50
Precio: $149,75
Precio: $399,00
Precio: $89,99
Precios modificados:
Precio: $199,99
Precio: $299,50
Precio: $129,99
Precio: $399,00
Precio: $89,99
BUILD SUCCESSFUL (total time: 0 seconds)

```