

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

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.

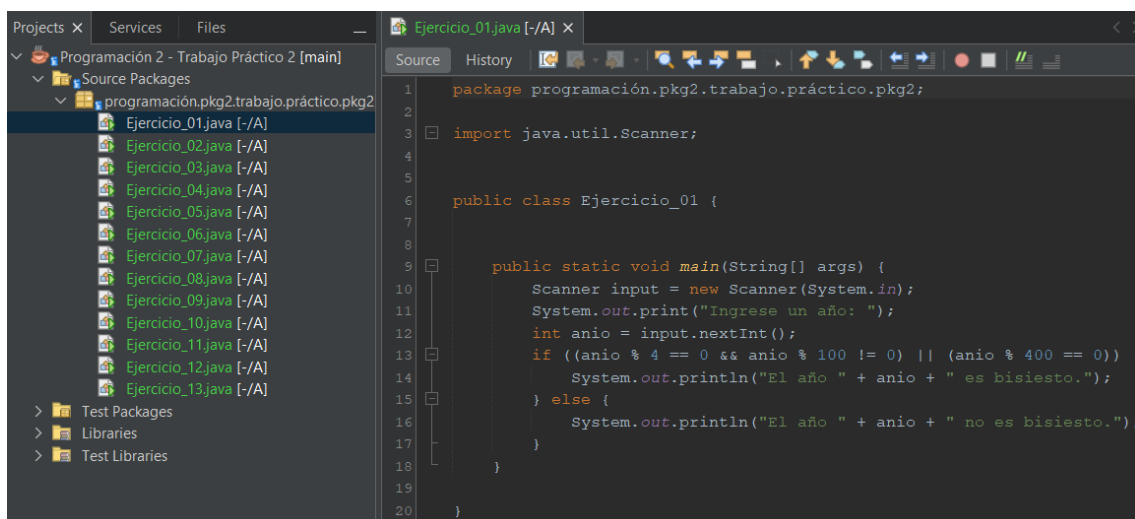
Ejemplo de entrada/salida:

Ingresa un año: 2024

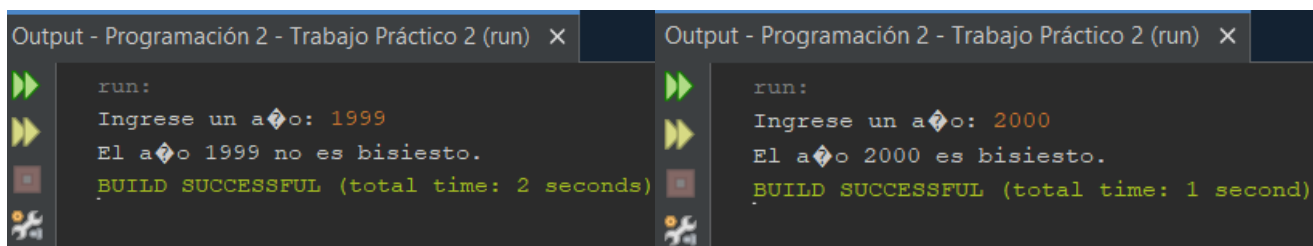
El año 2024 es bisiesto.

Ingresa un año: 1900

El año 1900 no es bisiesto.



```
1 package programación.pkg2.trabajo.práctico.pkg2;
2
3 import java.util.Scanner;
4
5
6 public class Ejercicio_01 {
7
8
9     public static void main(String[] args) {
10         Scanner input = new Scanner(System.in);
11         System.out.print("Ingresa un año: ");
12         int anio = input.nextInt();
13         if ((anio % 4 == 0 && anio % 100 != 0) || (anio % 400 == 0)) {
14             System.out.println("El año " + anio + " es bisiesto.");
15         } else {
16             System.out.println("El año " + anio + " no es bisiesto.");
17         }
18     }
19
20 }
```



```
run:
Ingresa un año: 1999
El año 1999 no es bisiesto.
BUILD SUCCESSFUL (total time: 2 seconds)

run:
Ingresa un año: 2000
El año 2000 es bisiesto.
BUILD SUCCESSFUL (total time: 1 second)
```

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.

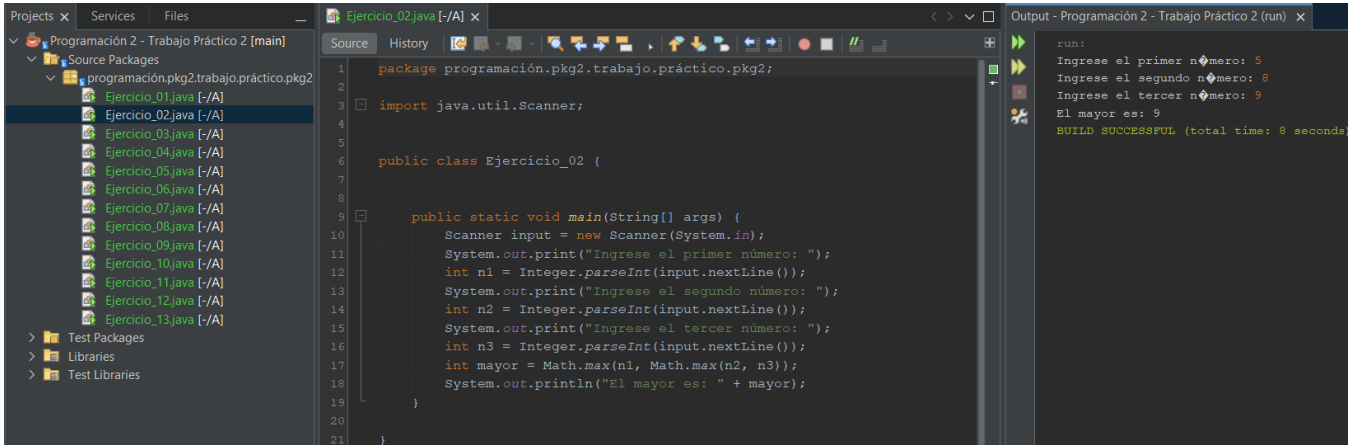
Ejemplo de entrada/salida:

Ingresa el primer número: 8

Ingrese el segundo número: 12

Ingrese el tercer número: 5

El mayor es: 12



```
1 package programación.pkg2.trabajo.práctico.pkg2;
2
3 import java.util.Scanner;
4
5 public class Ejercicio_02 {
6
7
8
9
10     public static void main(String[] args) {
11         Scanner input = new Scanner(System.in);
12         System.out.print("Ingrese el primer número: ");
13         int n1 = Integer.parseInt(input.nextLine());
14         System.out.print("Ingrese el segundo número: ");
15         int n2 = Integer.parseInt(input.nextLine());
16         System.out.print("Ingrese el tercer número: ");
17         int n3 = Integer.parseInt(input.nextLine());
18         int mayor = Math.max(n1, Math.max(n2, n3));
19         System.out.println("El mayor es: " + mayor);
20     }
21 }
```

run:
Ingrese el primer número: 5
Ingrese el segundo número: 12
Ingrese el tercer número: 5
El mayor es: 12
BUILD SUCCESSFUL (total time: 8 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"

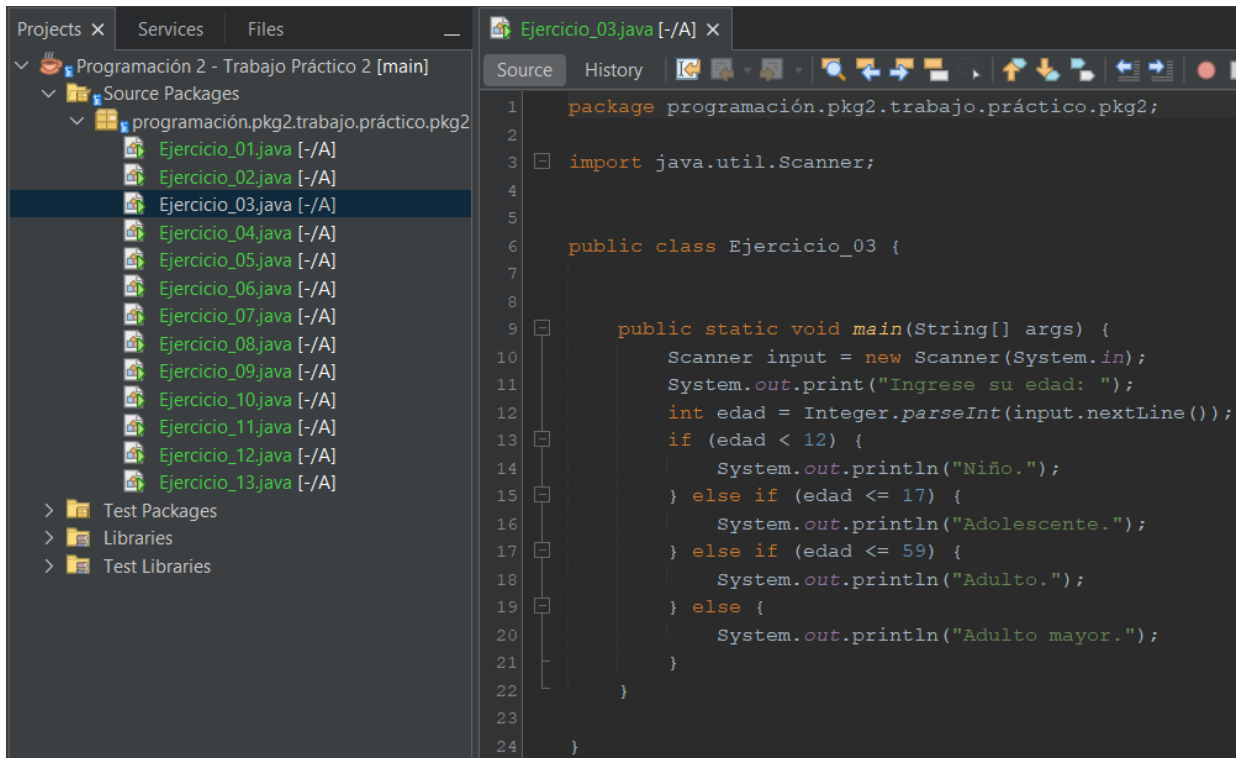
60 años o más: "Adulto mayor"

Ejemplo de entrada/salida:

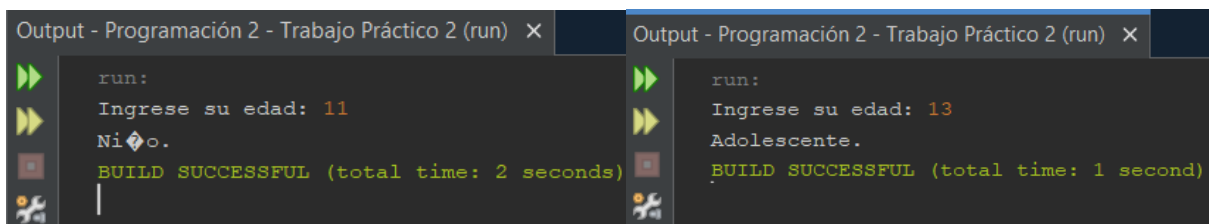
Ingrese su edad: 25 Eres un

Adulto. Ingrese su edad: 10

Eres un Niño.

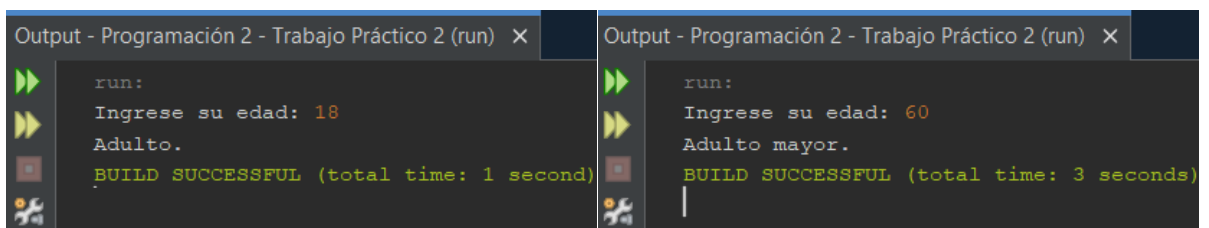


```
1 package programación.pkg2.trabajo.práctico.pkg2;
2
3 import java.util.Scanner;
4
5
6 public class Ejercicio_03 {
7
8
9     public static void main(String[] args) {
10         Scanner input = new Scanner(System.in);
11         System.out.print("Ingrese su edad: ");
12         int edad = Integer.parseInt(input.nextLine());
13         if (edad < 12) {
14             System.out.println("Niño.");
15         } else if (edad <= 17) {
16             System.out.println("Adolescente.");
17         } else if (edad <= 59) {
18             System.out.println("Adulto.");
19         } else {
20             System.out.println("Adulto mayor.");
21         }
22     }
23
24 }
```



```
run:
Ingrese su edad: 11
Niño.
BUILD SUCCESSFUL (total time: 2 seconds)

run:
Ingrese su edad: 13
Adolescente.
BUILD SUCCESSFUL (total time: 1 second)
```



```
run:
Ingrese su edad: 18
Adulto.
BUILD SUCCESSFUL (total time: 1 second)

run:
Ingrese su edad: 60
Adulto mayor.
BUILD SUCCESSFUL (total time: 3 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

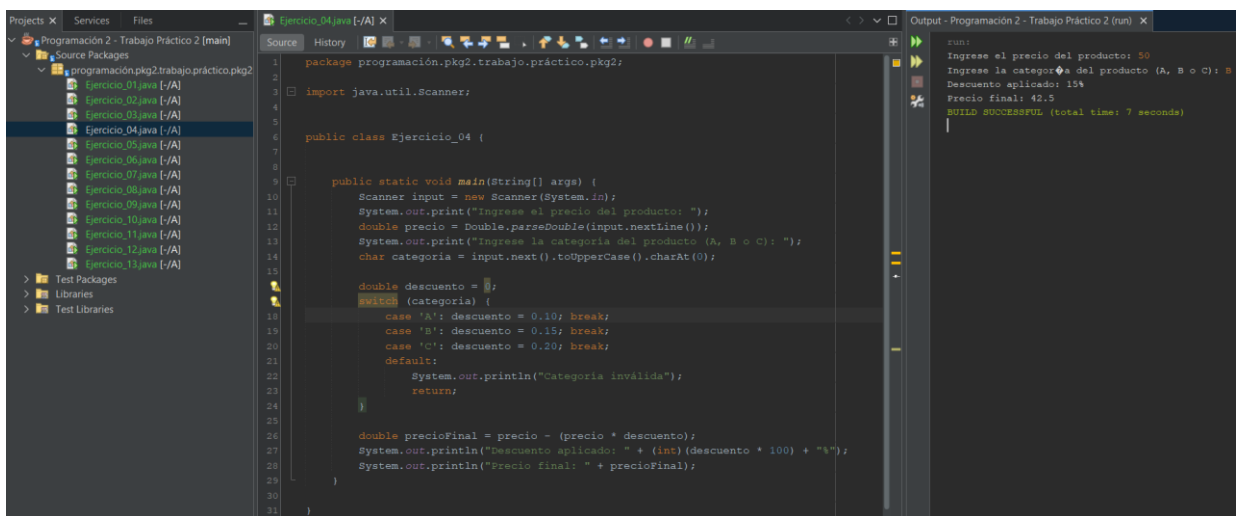
Ejemplo de entrada/salida:

Ingrese el precio del producto: 1000

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

Descuento aplicado: 15%

Precio final: 850.0



```
package programación.pkg2.trabajo.práctico.pkg2;

import java.util.Scanner;

public class Ejercicio_04 {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Ingrese el precio del producto: ");
        double precio = Double.parseDouble(input.nextLine());
        System.out.print("Ingrese la categoría del producto (A, B o C): ");
        char categoria = input.next().toUpperCase().charAt(0);

        double descuento = 0;
        switch (categoria) {
            case 'A': descuento = 0.10; break;
            case 'B': descuento = 0.15; break;
            case 'C': descuento = 0.20; break;
            default:
                System.out.println("Categoría inválida");
                return;
        }

        double precioFinal = precio - (precio * descuento);
        System.out.println("Descuento aplicado: " + (int)(descuento * 100) + "%");
        System.out.println("Precio final: " + precioFinal);
    }
}
```

run:
Ingrese el precio del producto: 50
Ingrese la categoría del producto (A, B o C): B
Descuento aplicado: 15%
Precio final: 42.5
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:

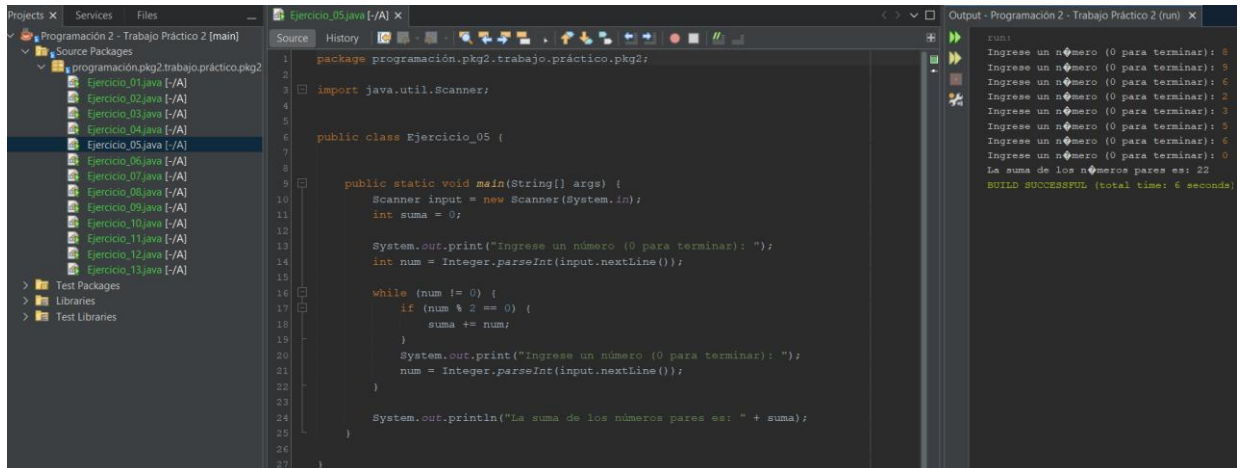
Ingrese un número (0 para terminar): 4

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



```
1 package programación.pkg2.trabajo.práctico.pkg2;
2
3 import java.util.Scanner;
4
5
6 public class Ejercicio_05 {
7
8
9     public static void main(String[] args) {
10         Scanner input = new Scanner(System.in);
11         int suma = 0;
12
13         System.out.print("Ingrese un número (0 para terminar): ");
14         int num = Integer.parseInt(input.nextLine());
15
16         while (num != 0) {
17             if (num % 2 == 0) {
18                 suma += num;
19             }
20             System.out.print("Ingrese un número (0 para terminar): ");
21             num = Integer.parseInt(input.nextLine());
22         }
23
24         System.out.println("La suma de los números pares es: " + suma);
25     }
26 }
27 }
```

run:
Ingrese un número (0 para terminar): 8
Ingrese un número (0 para terminar): 5
Ingrese un número (0 para terminar): 6
Ingrese un número (0 para terminar): 2
Ingrese un número (0 para terminar): 3
Ingrese un número (0 para terminar): 5
Ingrese un número (0 para terminar): 6
Ingrese un número (0 para terminar): 0
La suma de los números pares es: 22
BUILD SUCCESSFUL (total time: 6 seconds)

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

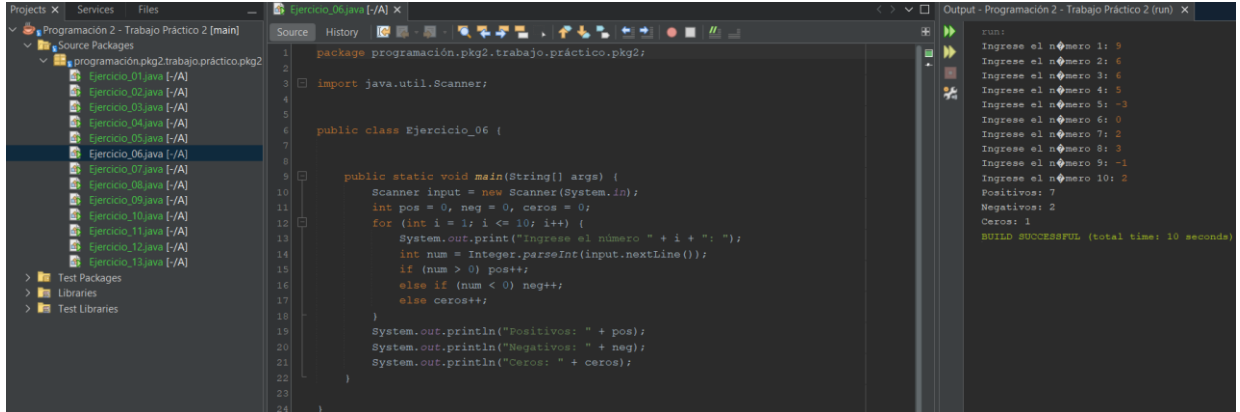
Escribe un programa que pida al usuario ingresar 10 números enteros y cuente cuántos son positivos, negativos y cuántos son ceros.

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



```
package programación.pkg2.trabajo.práctico.pkg2;

import java.util.Scanner;

public class Ejercicio_06 {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int pos = 0, neg = 0, ceros = 0;
        for (int i = 1; i <= 10; i++) {
            System.out.print("Ingrese el número " + i + ": ");
            int num = Integer.parseInt(input.nextLine());
            if (num > 0) pos++;
            else if (num < 0) neg++;
            else ceros++;
        }
        System.out.println("Positivos: " + pos);
        System.out.println("Negativos: " + neg);
        System.out.println("Ceros: " + ceros);
    }
}
```

run:
Ingrese el número 1: 9
Ingrese el número 2: 6
Ingrese el número 3: 6
Ingrese el número 4: 5
Ingrese el número 5: -3
Ingrese el número 6: 0
Ingrese el número 7: 2
Ingrese el número 8: 3
Ingrese el número 9: -1
Ingrese el número 10: 2
Positivos: 7
Negativos: 2
Ceros: 1
BUILD SUCCESSFUL (total time: 10 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

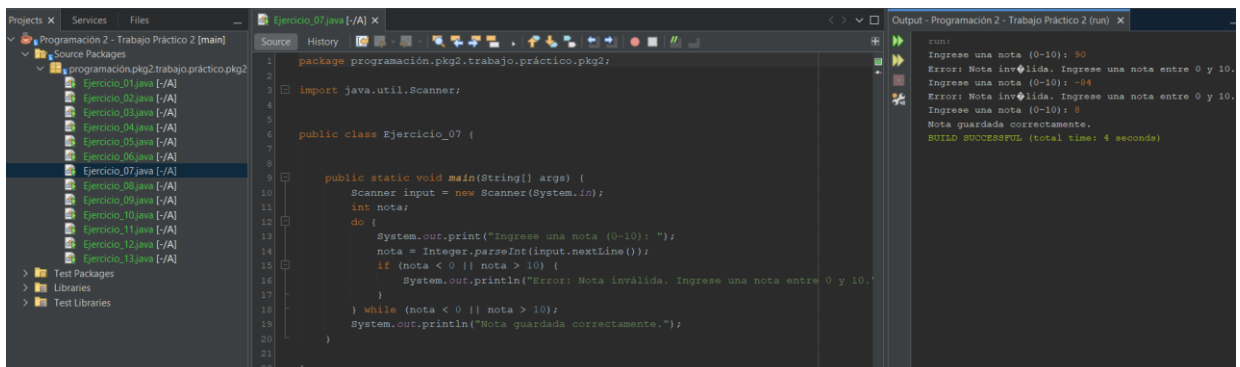
Error: Nota inválida. Ingrese una nota entre 0 y 10.

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.



```
package programación.pkg2.trabajo.práctico.pkg2;

import java.util.Scanner;

public class Ejercicio_07 {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int nota;
        do {
            System.out.print("Ingrese una nota (0-10): ");
            nota = Integer.parseInt(input.nextLine());
            if (nota < 0 || nota > 10) {
                System.out.println("Error: Nota inválida. Ingrese una nota entre 0 y 10.");
            }
        } while (nota < 0 || nota > 10);
        System.out.println("Nota guardada correctamente.");
    }
}
```

run:
Ingrese una nota (0-10): 90
Error: Nota inválida. Ingrese una nota entre 0 y 10.
Ingrese una nota (0-10): -84
Error: Nota inválida. Ingrese una nota entre 0 y 10.
Ingrese una nota (0-10): 8
Nota guardada correctamente.
BUILD SUCCESSFUL (total time: 4 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:

$$\text{PrecioFinal} = \text{PrecioBase} + (\text{PrecioBase} \times \text{Impuesto}) - (\text{PrecioBase} \times \text{Descuento})$$
$$\text{PrecioFinal} = \text{PrecioBase} + (\text{PrecioBase} \times \text{Impuesto}) - (\text{PrecioBase} \times \text{Descuento})$$

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.

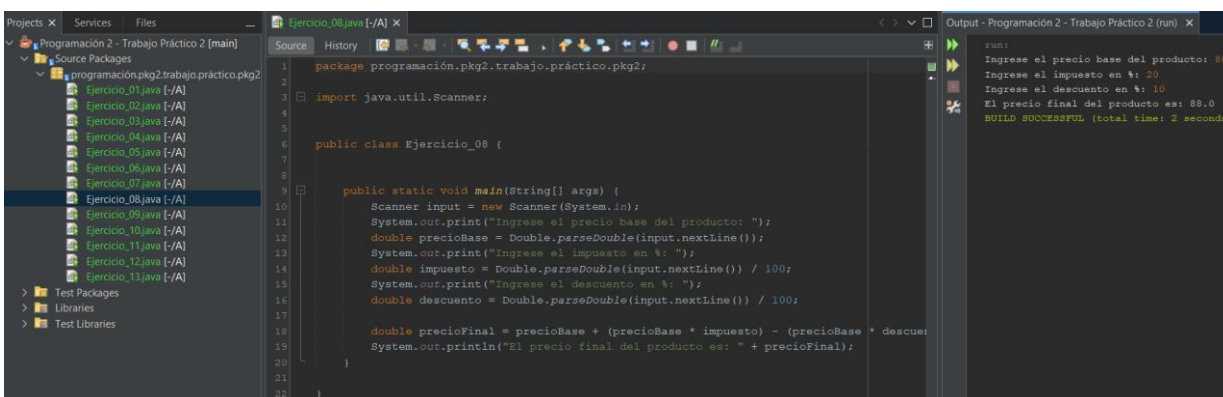
Ejemplo de entrada/salida:

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



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:

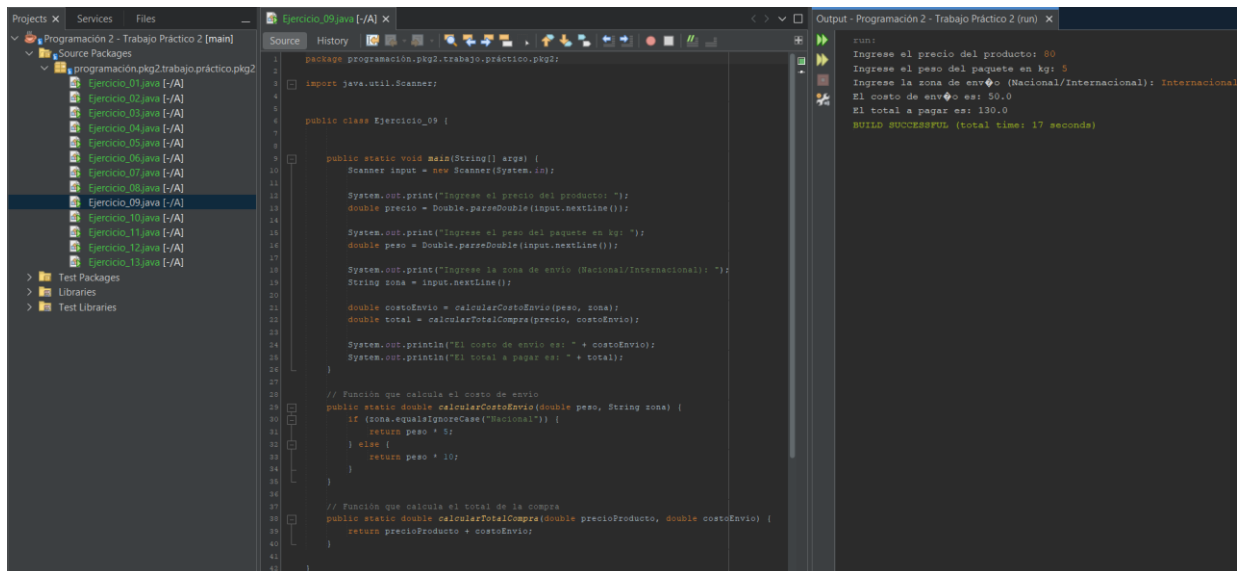
Ingrese el precio del producto: 50

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



```
package programación.pkg2.trabajo.práctico.pkg2;

import java.util.Scanner;

public class Ejercicio_09 {

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

        System.out.print("Ingrese el precio del producto: ");
        double precio = Double.parseDouble(input.nextLine());

        System.out.print("Ingrese el peso del paquete en kg: ");
        double peso = Double.parseDouble(input.nextLine());

        System.out.print("Ingrese la zona de envío (Nacional/Internacional): ");
        String zona = input.nextLine();

        double costoEnvio = calcularCostoEnvio(peso, zona);
        double total = calcularTotalCompra(precio, costoEnvio);

        System.out.println("El costo de envío es: " + costoEnvio);
        System.out.println("El total a pagar es: " + total);
    }

    // Función que calcula el costo de envío
    public static double calcularCostoEnvio(double peso, String zona) {
        if (zona.equalsIgnoreCase("Nacional")) {
            return peso * 5;
        } else {
            return peso * 10;
        }
    }

    // Función que calcula el total de la compra
    public static double calcularTotalCompra(double precioProducto, double costoEnvio) {
        return precioProducto + costoEnvio;
    }
}
```

run:
Ingrese el precio del producto: 60
Ingrese el peso del paquete en kg: 5
Ingrese la zona de envío (Nacional/Internacional): Internacional
El costo de envío es: 50.0
El total a pagar es: 110.0
BUILD SUCCESSFUL (total time: 17 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.

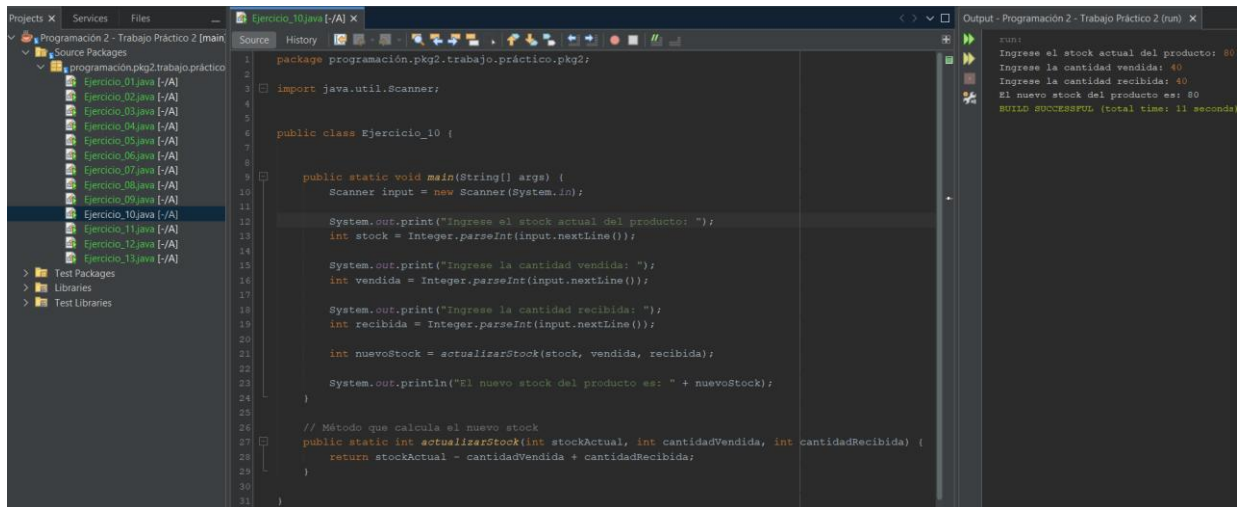
Ejemplo de entrada/salida:

Ingrese el stock actual del producto: 50

Ingrese la cantidad vendida: 20

Ingrese la cantidad recibida: 30

El nuevo stock del producto es: 60



```
1 package programación.pkg2.trabajo.práctico.pkg2;
2
3 import java.util.Scanner;
4
5
6 public class Ejercicio_10 {
7
8
9
10     public static void main(String[] args) {
11         Scanner input = new Scanner(System.in);
12
13         System.out.print("Ingrese el stock actual del producto: ");
14         int stock = Integer.parseInt(input.nextLine());
15
16         System.out.print("Ingrese la cantidad vendida: ");
17         int vendida = Integer.parseInt(input.nextLine());
18
19         System.out.print("Ingrese la cantidad recibida: ");
20         int recibida = Integer.parseInt(input.nextLine());
21
22         int nuevoStock = actualizarStock(stock, vendida, recibida);
23
24         System.out.println("El nuevo stock del producto es: " + nuevoStock);
25
26         // Método que calcula el nuevo stock
27         public static int actualizarStock(int stockActual, int cantidadVendida, int cantidadRecibida) {
28             return stockActual - cantidadVendida + cantidadRecibida;
29         }
30
31     }
```

Output - Programación 2 - Trabajo Práctico 2 (run)

```
run:
Ingrese el stock actual del producto: 80
Ingrese la cantidad vendida: 40
Ingrese la cantidad recibida: 40
El nuevo stock del producto es: 80
BUILD SUCCESSFUL (total time: 11 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%.

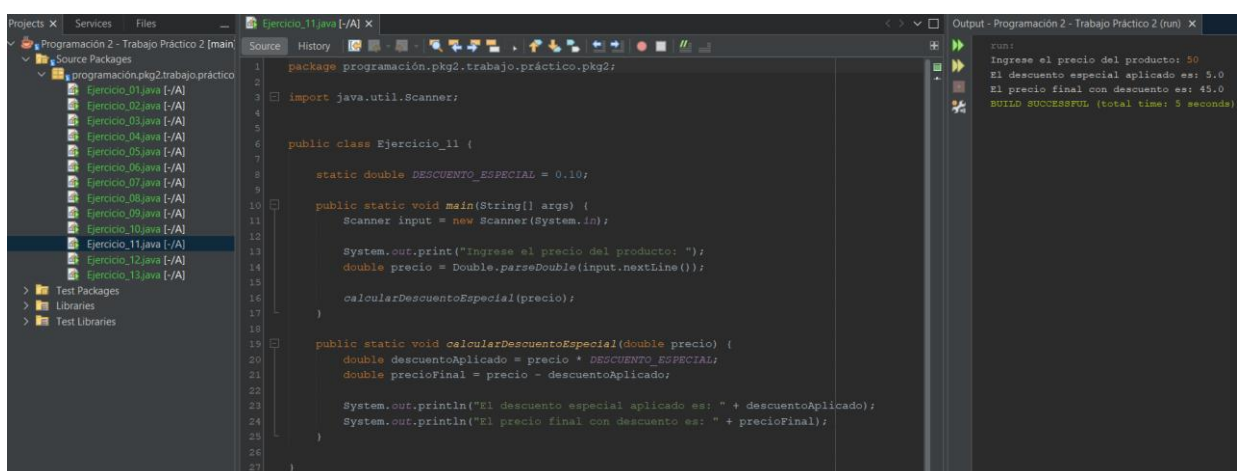
Dentro del método, declara una variable local **descuentoAplicado**, almacena el valor del descuento y muestra el precio final con descuento.

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



```
1 package programación.pkg2.trabajo.práctico.pkg2;
2
3 import java.util.Scanner;
4
5
6 public class Ejercicio_11 {
7
8     static double DESCUENTO_ESPECIAL = 0.10;
9
10     public static void main(String[] args) {
11         Scanner input = new Scanner(System.in);
12
13         System.out.print("Ingrese el precio del producto: ");
14         double precio = Double.parseDouble(input.nextLine());
15
16         calcularDescuentoEspecial(precio);
17     }
18
19     public static void calcularDescuentoEspecial(double precio) {
20         double descuentoAplicado = precio * DESCUENTO_ESPECIAL;
21         double precioFinal = precio - descuentoAplicado;
22
23         System.out.println("El descuento especial aplicado es: " + descuentoAplicado);
24         System.out.println("El precio final con descuento es: " + precioFinal);
25     }
26
27 }
```

Output - Programación 2 - Trabajo Práctico 2 (run)

```
run:
Ingrese el precio del producto: 200
El descuento especial aplicado es: 20.0
El precio final con descuento es: 180.0
BUILD SUCCESSFUL (total time: 5 seconds)
```

Arrays y Recursividad:

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

Crea un programa que:

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

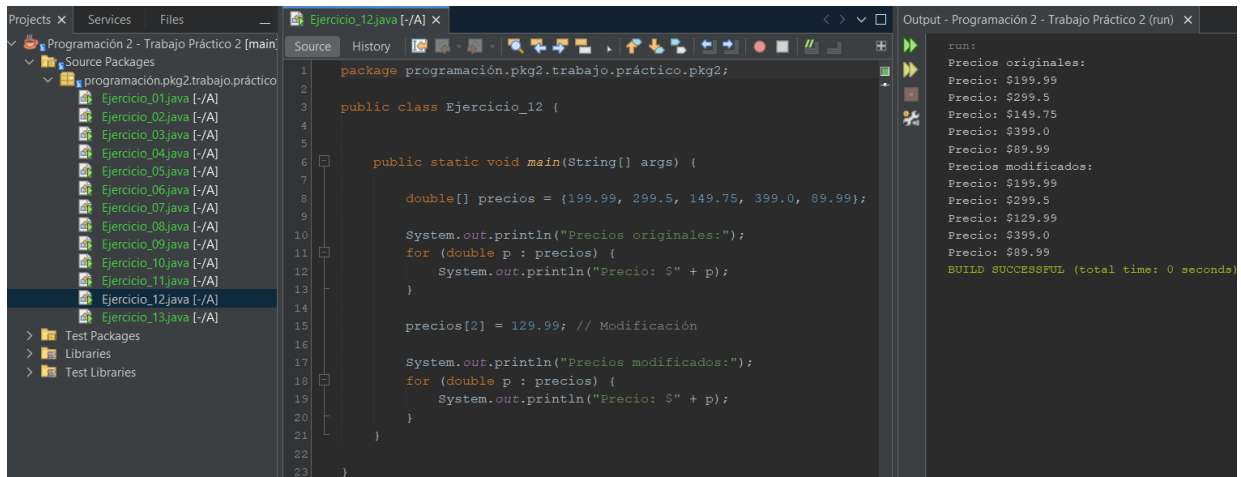
Precio: \$129.99

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.



The screenshot shows an IDE with a project named 'Programación 2 - Trabajo Práctico 2'. The source code for 'Ejercicio_12.java' is displayed. It defines a package 'programación.pkg2.trabajo.práctico.pkg2' and a class 'Ejercicio_12'. The 'main' method initializes an array 'precios' with values {199.99, 299.5, 149.75, 399.0, 89.99}. It prints the original prices, then modifies 'precios[2]' to 129.99, and prints the modified prices. The output window shows the execution results, confirming the successful build and the printed values.

```
package programación.pkg2.trabajo.práctico.pkg2;

public class Ejercicio_12 {

    public static void main(String[] args) {

        double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};

        System.out.println("Precios originales:");
        for (double p : precios) {
            System.out.println("Precio: $" + p);
        }

        precios[2] = 129.99; // Modificación

        System.out.println("Precios modificados:");
        for (double p : precios) {
            System.out.println("Precio: $" + p);
        }
    }
}
```

run:
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
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

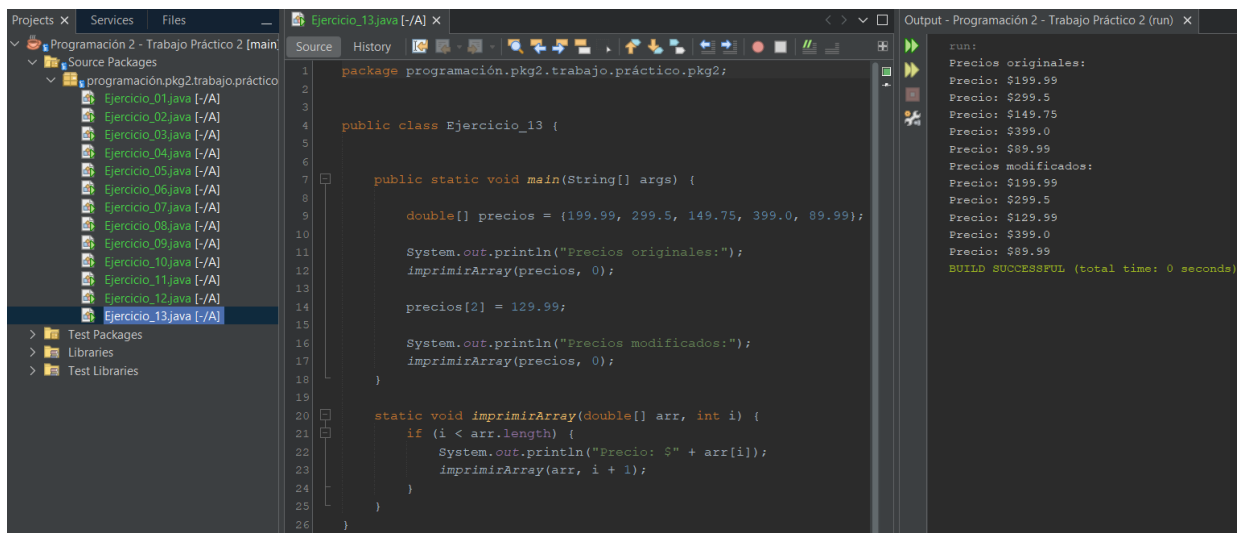
Precio: \$129.99

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.



```
1 package programación.pkg2.trabajo.práctico.pkg2;
2
3
4 public class Ejercicio_13 {
5
6
7     public static void main(String[] args) {
8
9         double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};
10
11         System.out.println("Precios originales:");
12         imprimirArray(precios, 0);
13
14         precios[2] = 129.99;
15
16         System.out.println("Precios modificados:");
17         imprimirArray(precios, 0);
18     }
19
20     static void imprimirArray(double[] arr, int i) {
21         if (i < arr.length) {
22             System.out.println("Precio: $" + arr[i]);
23             imprimirArray(arr, i + 1);
24         }
25     }
26 }
```

run:
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
BUILD SUCCESSFUL (total time: 0 seconds)

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.