

PROGRAMACIÓN II

Trabajo Práctico 2: Programación Estructurada

Alumno

Fabrizio Nicolás Puccio

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.

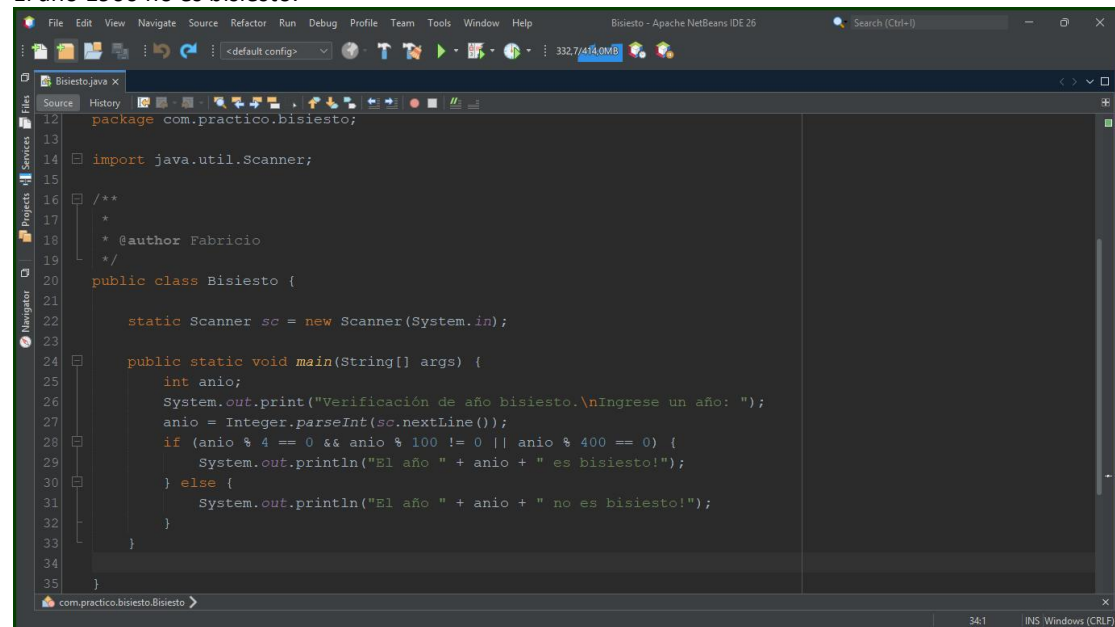
Ejemplo de entrada/salida:

Ingrese un año: 2024

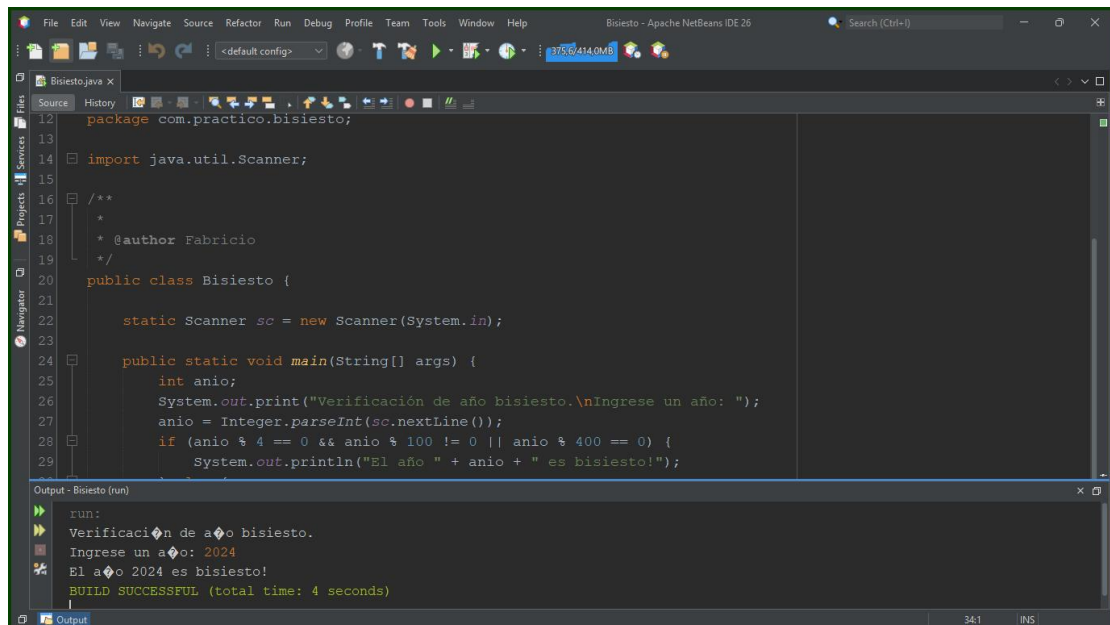
El año 2024 es bisiesto.

Ingrese un año: 1900

El año 1900 no es bisiesto.



```
12 package com.practico.bisiesto;
13
14 import java.util.Scanner;
15
16 /**
17  *
18  * @author Fabrizio
19  */
20 public class Bisiesto {
21
22     static Scanner sc = new Scanner(System.in);
23
24     public static void main(String[] args) {
25         int anio;
26         System.out.print("Verificación de año bisiesto.\nIngrese un año: ");
27         anio = Integer.parseInt(sc.nextLine());
28         if (anio % 4 == 0 && anio % 100 != 0 || anio % 400 == 0) {
29             System.out.println("El año " + anio + " es bisiesto!");
30         } else {
31             System.out.println("El año " + anio + " no es bisiesto!");
32         }
33     }
34 }
35 }
```



The screenshot shows the NetBeans IDE with the file Bisiesto.java open. The code defines a package, imports Scanner, and contains a main method that prompts the user for a year and checks if it is a leap year using a conditional statement. The output window shows the program running successfully with the input 2024, resulting in the output "El año 2024 es bisiesto!".

```
package com.practico.bisiesto;

import java.util.Scanner;

/**
 *
 * @author Fabricio
 */
public class Bisiesto {

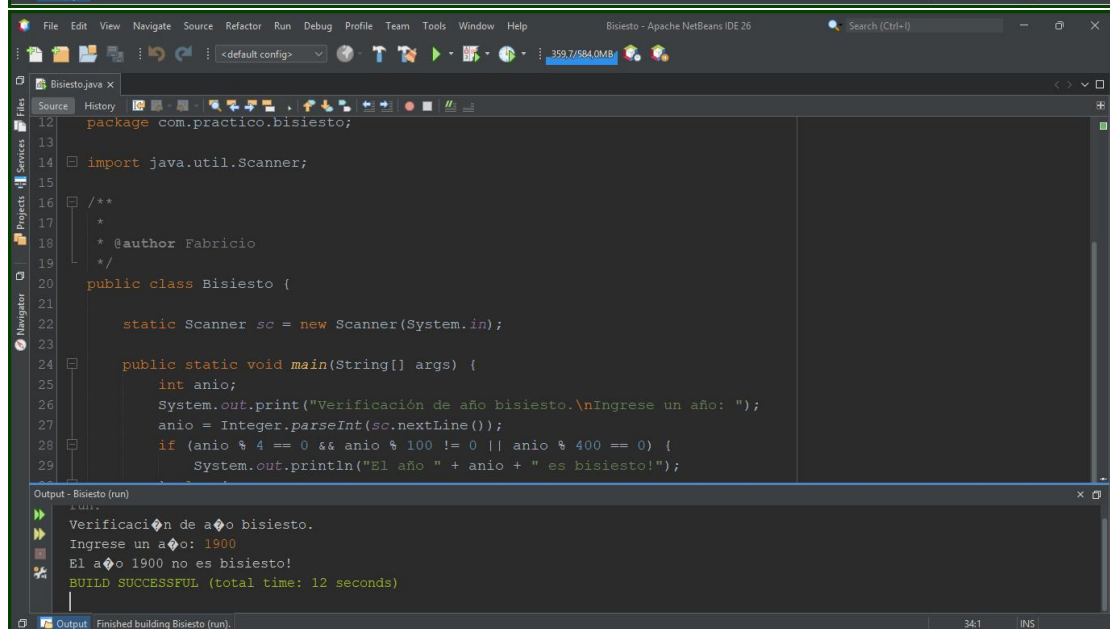
    static Scanner sc = new Scanner(System.in);

    public static void main(String[] args) {

        int anio;
        System.out.print("Verificación de año bisiesto.\nIngrese un año: ");
        anio = Integer.parseInt(sc.nextLine());
        if (anio % 4 == 0 && anio % 100 != 0 || anio % 400 == 0) {
            System.out.println("El año " + anio + " es bisiesto!");
        }
    }
}
```

Output - Bisiesto (run)

```
run:
Verificación de año bisiesto.
Ingrese un año: 2024
El año 2024 es bisiesto!
BUILD SUCCESSFUL (total time: 4 seconds)
```



This screenshot shows the same Bisiesto.java file in NetBeans IDE, but with the output window showing the program running with the input 1900. The output is "El año 1900 no es bisiesto!", indicating that the conditional logic correctly identifies non-leap years.

```
package com.practico.bisiesto;

import java.util.Scanner;

/**
 *
 * @author Fabricio
 */
public class Bisiesto {

    static Scanner sc = new Scanner(System.in);

    public static void main(String[] args) {

        int anio;
        System.out.print("Verificación de año bisiesto.\nIngrese un año: ");
        anio = Integer.parseInt(sc.nextLine());
        if (anio % 4 == 0 && anio % 100 != 0 || anio % 400 == 0) {
            System.out.println("El año " + anio + " es bisiesto!");
        }
    }
}
```

Output - Bisiesto (run)

```
run:
Verificación de año bisiesto.
Ingrese un año: 1900
El año 1900 no es bisiesto!
BUILD SUCCESSFUL (total time: 12 seconds)
```

Link ejercicio resuelto en GitHub:

https://github.com/FabricioPuccio/UTN-TUPaD-P2/tree/main/programacion_estructurada/estructuras_condicionales/Bisiesto/src/com/practico/bisiesto

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

```
public class MayorDeTresNumeros {  
    static Scanner sc = new Scanner(System.in);  
  
    public static void main(String[] args) {  
        int numUno, numDos, numTres;  
        System.out.println("Este programa determina el número mas grande entre tres enteros!\n"  
            + "Ingrese el primer número entero: ");  
        numUno = Integer.parseInt(sc.nextLine());  
        System.out.println("Ingrese el segundo número entero: ");  
        numDos = Integer.parseInt(sc.nextLine());  
        System.out.println("Ingrese el tercer número entero: ");  
        numTres = Integer.parseInt(sc.nextLine());  
  
        if (numUno >= numDos && numUno >= numTres) {  
            System.out.println("El mayor número entero ingresado es: " + numUno);  
        } else if (numDos >= numTres) {  
            System.out.println("El mayor número entero ingresado es: " + numDos);  
        } else {  
            System.out.println("El mayor número entero ingresado es: " + numTres);  
        }  
    }  
}
```

Output - MayorDeTresNumeros (run)

```
>> Este programa determina el número mas grande entre tres enteros!  
>> Ingrese el primer número entero: 8  
>> Ingrese el segundo número entero: 12  
>> Ingrese el tercer número entero: 5  
>> El mayor número entero ingresado es: 12  
BUILD SUCCESSFUL (total time: 13 seconds)
```

Link ejercicio resuelto en GitHub:

https://github.com/FabrizioPuccio/UTN-TUPaD-P2/blob/main/programacion_estructurada/estructuras_condicionales/MayorDeTresNumeros/src/com/practico/mayordetresnumeros/MayorDeTresNumeros.java

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"

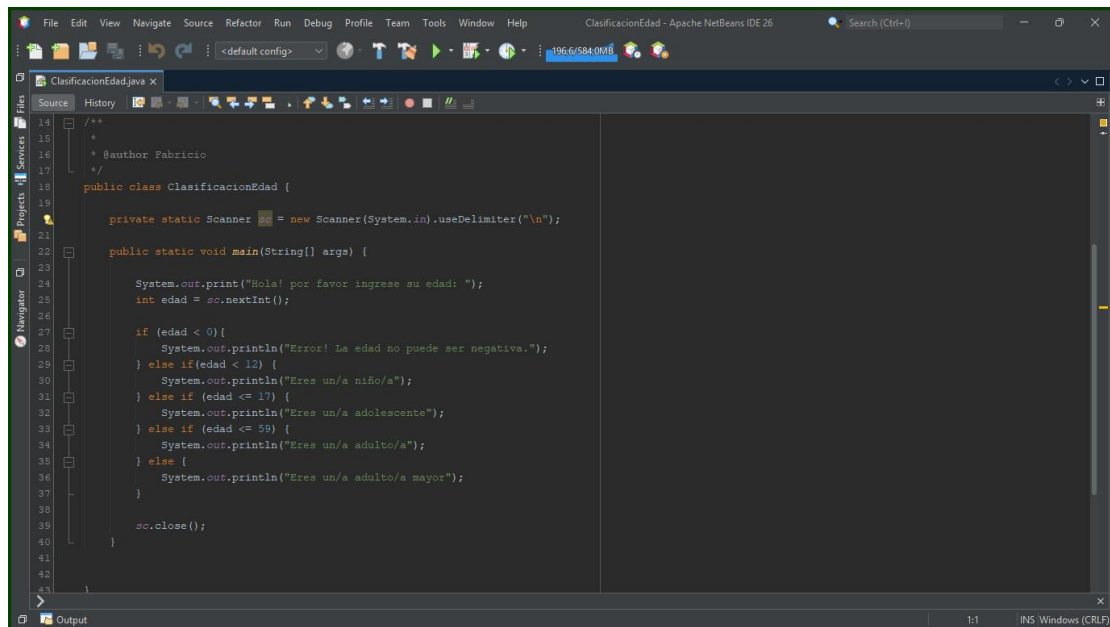
Ejemplo de entrada/salida:

Ingrese su edad: 25

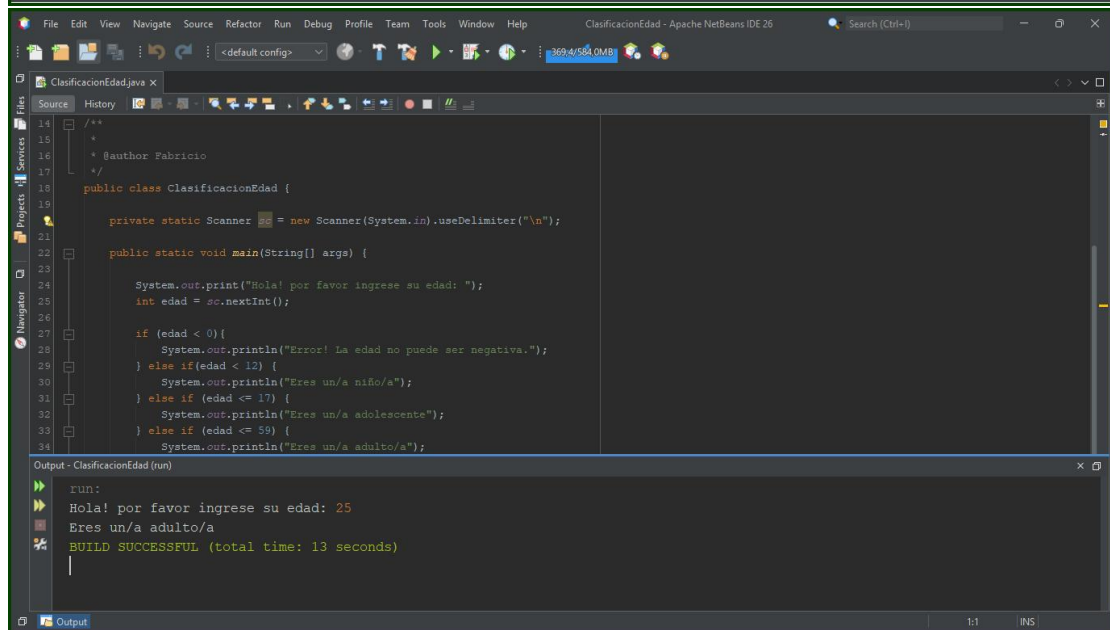
Eres un Adulto.

Ingrese su edad: 10

Eres un Niño.



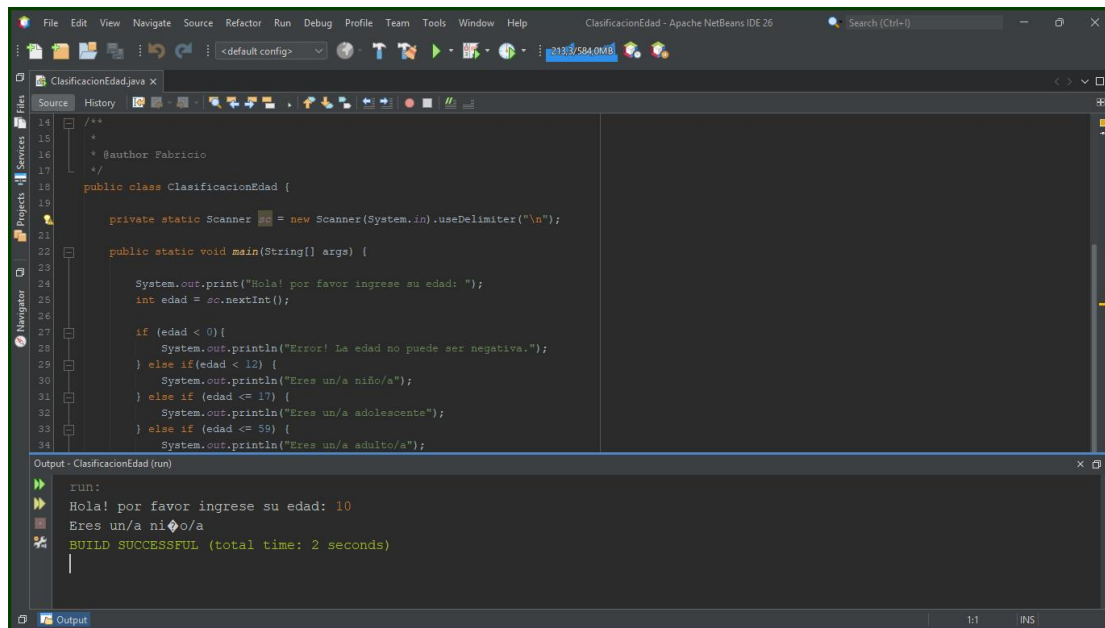
```
14  /**
15   *
16   * @author Fabricio
17   */
18  public class ClasificacionEdad {
19
20      private static Scanner sc = new Scanner(System.in).useDelimiter("\n");
21
22      public static void main(String[] args) {
23
24          System.out.print("Hola! por favor ingrese su edad: ");
25          int edad = sc.nextInt();
26
27          if (edad < 0) {
28              System.out.println("Error! La edad no puede ser negativa.");
29          } else if (edad < 12) {
30              System.out.println("Eres un/a niño/a");
31          } else if (edad <= 17) {
32              System.out.println("Eres un/a adolescente");
33          } else if (edad <= 59) {
34              System.out.println("Eres un/a adulto/a");
35          } else {
36              System.out.println("Eres un/a adulto/a mayor");
37          }
38
39          sc.close();
40      }
41  }
42  }
```



```
14  /**
15   *
16   * @author Fabricio
17   */
18  public class ClasificacionEdad {
19
20      private static Scanner sc = new Scanner(System.in).useDelimiter("\n");
21
22      public static void main(String[] args) {
23
24          System.out.print("Hola! por favor ingrese su edad: ");
25          int edad = sc.nextInt();
26
27          if (edad < 0) {
28              System.out.println("Error! La edad no puede ser negativa.");
29          } else if (edad < 12) {
30              System.out.println("Eres un/a niño/a");
31          } else if (edad <= 17) {
32              System.out.println("Eres un/a adolescente");
33          } else if (edad <= 59) {
34              System.out.println("Eres un/a adulto/a");
35          }
36      }
37  }
```

Output - ClasificacionEdad (run)

```
run:
Hola! por favor ingrese su edad: 25
Eres un/a adulto/a
BUILD SUCCESSFUL (total time: 13 seconds)
```



The screenshot shows the Apache NetBeans IDE interface. The main editor window displays the source code for `ClasificacionEdad.java`. The code includes a package declaration, a comment, and a class definition with a `main` method. The `main` method uses a `Scanner` to read an integer age from the user and prints a message based on the age range. The output window at the bottom shows the execution results, including the prompt, the user input (10), the output message ("Eres un/a niño/a"), and a successful build status.

```
14  /**
15   *
16   * @author Fabricio
17   */
18  public class ClasificacionEdad {
19
20      private static Scanner sc = new Scanner(System.in).useDelimiter("\n");
21
22      public static void main(String[] args) {
23
24          System.out.print("Hola! por favor ingrese su edad: ");
25          int edad = sc.nextInt();
26
27          if (edad < 0) {
28              System.out.println("Error! la edad no puede ser negativa.");
29          } else if (edad < 12) {
30              System.out.println("Eres un/a niño/a");
31          } else if (edad <= 17) {
32              System.out.println("Eres un/a adolescente");
33          } else if (edad <= 59) {
34              System.out.println("Eres un/a adulto/a");
35          }
36      }
37  }
```

Output - ClasificacionEdad (run)

```
run:
Hola! por favor ingrese su edad: 10
Eres un/a niño/a
BUILD SUCCESSFUL (total time: 2 seconds)
```

Link ejercicio resuelto en GitHub:

https://github.com/FabricioPuccio/UTN-TUPaD-P2/tree/main/programacion_estructurada/estructuras_condicionales/ClasificacionEdad/src/com/practico/clasificacionedad

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 com.practico.calculadoradescuento;

import java.util.Scanner;

/**
 *
 * @author Fabricio
 */
public class CalculadoraDescuento {

    static Scanner sc = new Scanner(System.in).useDelimiter("\n");

    public static void main(String[] args) {
        float precio, precioFinal = 0f;
        String categoria;
        System.out.print("Sr. usuario, para calcular el descuento, por favor ingrese el precio del producto: ");
        precio = sc.nextFloat();

        System.out.print("Ingrese la categoria del producto (A, B ó C): ");
        categoria = sc.next().toUpperCase();

        switch (categoria) {
            case "A":
                precioFinal = precio - (precio * 10) / 100;
                break;
            case "B":
                precioFinal = precio - (precio * 15) / 100;
                break;
            case "C":
                precioFinal = precio - (precio * 20) / 100;
                break;
            default:
                System.out.println("Error! La categoria ingresada es invalida.");
        }

        sc.close();
    }
}
```

com.practico.calculadoradescuento.CalculadoraDescuento > main > switch (categoria) > case "C": >

Output

```
        System.out.println("Error! La categoria ingresada es invalida.");
    }

    sc.close();

    switch (categoria) {
        case "A" ->
            System.out.println("Descuento aplicado: 10%\nPrecio final: $" + precioFinal);
        case "B" ->
            System.out.println("Descuento aplicado: 15%\nPrecio final: $" + precioFinal);
        case "C" ->
            System.out.println("Descuento aplicado: 20%\nPrecio final: $" + precioFinal);
    }
}

}
```

com.practico.calculadoradescuento.CalculadoraDescuento > main > switch (categoria) > case "C": >

Output

The screenshot shows the Apache NetBeans IDE with the file 'CalculadoraDescuento.java' open. The code is a Java program that calculates discounts based on categories. The output window shows the program's execution: it prompts the user for a price (1000) and a category (b), applies a 15% discount, and shows the final price as \$850.0. The build was successful.

```
1  /*
2  4. Calculadora de Descuento según categoría.
3  Escribe un programa que solicite al usuario el precio de un producto y
4  su categoría (A, B o C).
5  Luego, aplique los siguientes descuentos:
6  Categoría A: 10% de descuento
7  Categoría B: 15% de descuento
8  Categoría C: 20% de descuento
9  */
10 package com.practico.calculadoradescuento;
11
12 import java.util.Scanner;
13
14 /**
15  *
16  * @author Fabricio
17  */
18 public class CalculadoraDescuento {
19
20     public static void main(String[] args) {
21         Scanner sc = new Scanner(System.in);
22         System.out.print("Sr. usuario, para calcular el descuento, por favor ingrese el precio del producto: ");
23         int precio = sc.nextInt();
24         System.out.print("Ingrese la categoría del producto (A, B o C): ");
25         char categoria = sc.next().charAt(0);
26         double descuento = 0;
27         switch (categoria) {
28             case 'A':
29                 descuento = 0.10;
30             case 'B':
31                 descuento = 0.15;
32             case 'C':
33                 descuento = 0.20;
34         }
35         double precioFinal = precio * (1 - descuento);
36         System.out.println("Descuento aplicado: " + (descuento * 100) + "%");
37         System.out.println("Precio final: $" + precioFinal);
38     }
39 }
```

Output - CalculadoraDescuento (run)

```
run:
Sr. usuario, para calcular el descuento, por favor ingrese el precio del producto: 1000
Ingrese la categoría del producto (A, B o C): b
Descuento aplicado: 15%
Precio final: $850.0
BUILD SUCCESSFUL (total time: 21 seconds)
```

Link ejercicio resuelto en GitHub:

https://github.com/FabricioPuccio/UTN-TUPaD-P2/tree/main/programacion_estructurada/estructuras_condicionales/CalculadoraDescuento/src/com/practico/calculadoradescuento

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

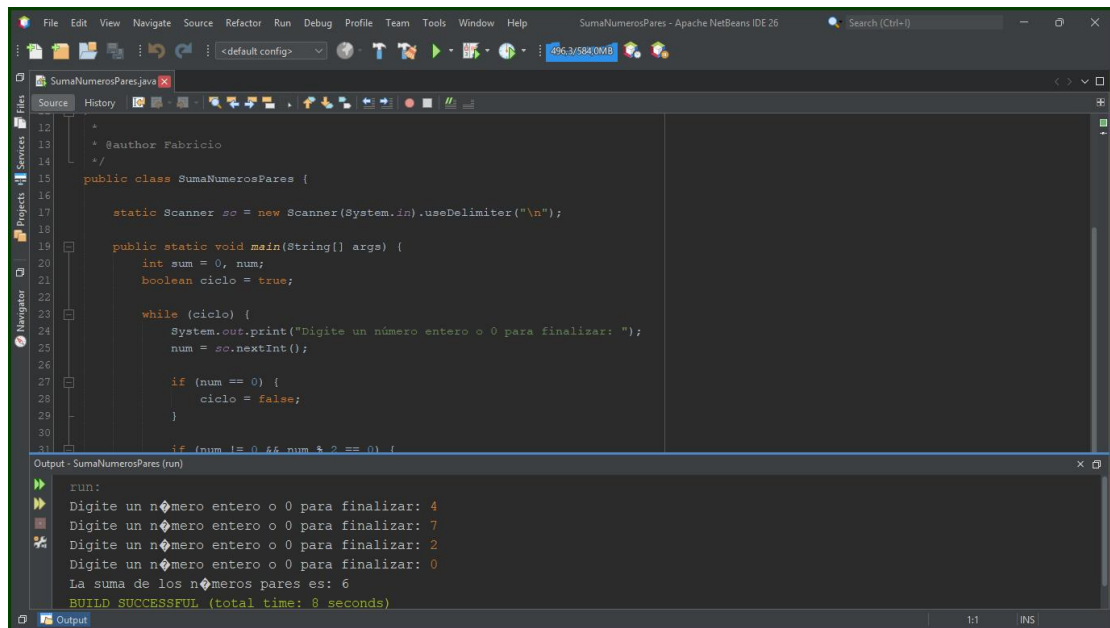
Ingrese un número (0 para terminar): 2

Ingrese un número (0 para terminar): 0

La suma de los números pares es: 6

The screenshot shows the Apache NetBeans IDE with the file 'SumaNumerosPares.java' open. The code is a Java program that uses a while loop to sum even numbers entered by the user until they enter 0.

```
12  *
13  * @author Fabricio
14  */
15  public class SumaNumerosPares {
16
17      static Scanner sc = new Scanner(System.in).useDelimiter("\n");
18
19      public static void main(String[] args) {
20          int sum = 0, num;
21          boolean ciclo = true;
22
23          while (ciclo) {
24              System.out.print("Digite un número entero o 0 para finalizar: ");
25              num = sc.nextInt();
26
27              if (num == 0) {
28                  ciclo = false;
29              }
30
31              if (num != 0 && num % 2 == 0) {
32                  sum += num;
33              }
34          }
35          sc.close();
36          System.out.println("La suma de los números pares es: " + sum);
37      }
38  }
```

The screenshot shows the Apache NetBeans IDE interface. The main editor displays the source code for `SumaNumerosPares.java`. The code includes a package declaration, a class declaration, and a `main` method that uses a `Scanner` to read integers from the user. The `main` method includes a `while` loop that continues until the user enters 0. The output window at the bottom shows the execution results, including the user input and the final sum.

```
12  *
13  * @author Fabricio
14  */
15  public class SumaNumerosPares {
16
17      static Scanner sc = new Scanner(System.in).useDelimiter("\n");
18
19      public static void main(String[] args) {
20          int sum = 0, num;
21          boolean ciclo = true;
22
23          while (ciclo) {
24              System.out.print("Digite un número entero o 0 para finalizar: ");
25              num = sc.nextInt();
26
27              if (num == 0) {
28                  ciclo = false;
29              }
30
31              if (num != 0 && num % 2 == 0) {
```

Output - SumaNumerosPares (run)

```
run:
Digite un número entero o 0 para finalizar: 4
Digite un número entero o 0 para finalizar: 7
Digite un número entero o 0 para finalizar: 2
Digite un número entero o 0 para finalizar: 0
La suma de los números pares es: 6
BUILD SUCCESSFUL (total time: 8 seconds)
```

Link ejercicio resuelto en GitHub:

https://github.com/FabricioPuccio/UTN-TUPaD-P2/tree/main/programacion_estructurada/estructuras_repeticion/SumaNumerosPares/src/com/practica/sumanumerospares

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

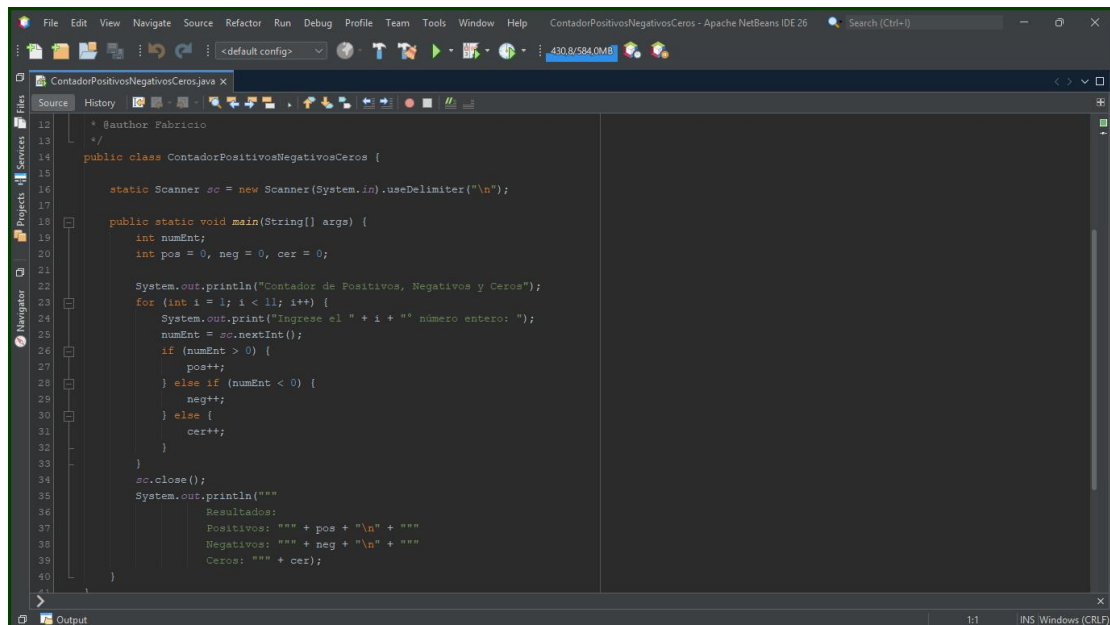
Ingrese el número 10: -8

Resultados:

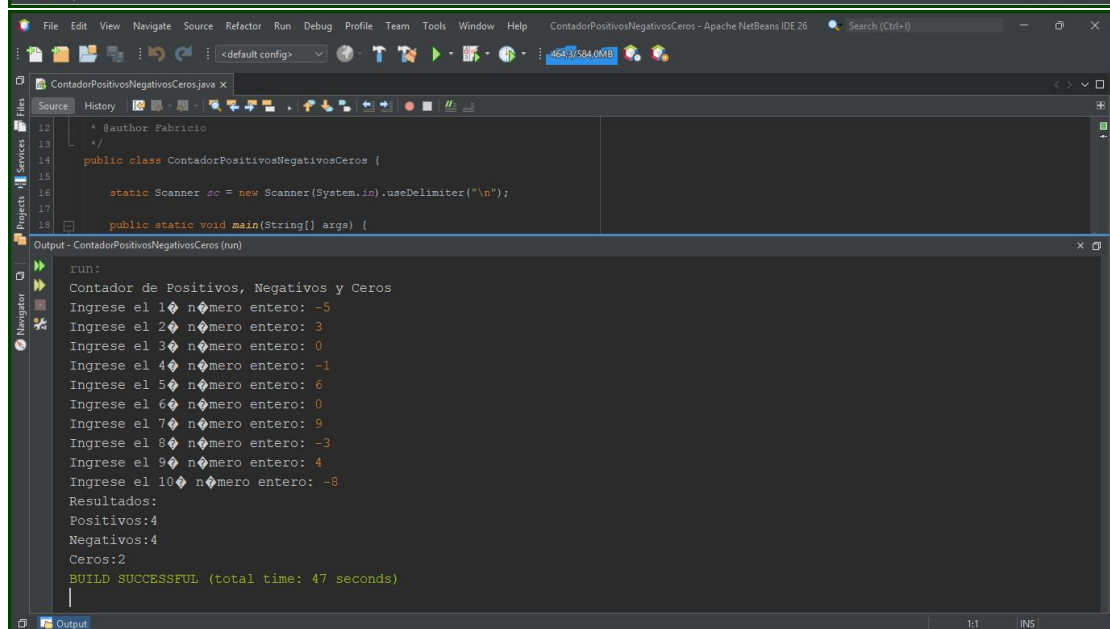
Positivos: 4

Negativos: 4

Ceros: 2



```
12  * @author Fabricio
13  */
14  public class ContadorPositivosNegativosCeros {
15
16      static Scanner sc = new Scanner(System.in).useDelimiter("\n");
17
18      public static void main(String[] args) {
19          int numEnt;
20          int pos = 0, neg = 0, cer = 0;
21
22          System.out.println("Contador de Positivos, Negativos y Ceros");
23          for (int i = 1; i < 11; i++) {
24              System.out.print("Ingrese el " + i + " número entero: ");
25              numEnt = sc.nextInt();
26              if (numEnt > 0) {
27                  pos++;
28              } else if (numEnt < 0) {
29                  neg++;
30              } else {
31                  cer++;
32              }
33          }
34          sc.close();
35          System.out.println("
36              Resultados:
37              Positivos: " + pos + "
38              Negativos: " + neg + "
39              Ceros: " + cer);
40      }
```



```
run:
Contador de Positivos, Negativos y Ceros
Ingrese el 1 número entero: -5
Ingrese el 2 número entero: 3
Ingrese el 3 número entero: 0
Ingrese el 4 número entero: -1
Ingrese el 5 número entero: 6
Ingrese el 6 número entero: 0
Ingrese el 7 número entero: 9
Ingrese el 8 número entero: -3
Ingrese el 9 número entero: 4
Ingrese el 10 número entero: -8
Resultados:
Positivos:4
Negativos:4
Ceros:2
BUILD SUCCESSFUL (total time: 47 seconds)
```

Link ejercicio resuelto en GitHub:

https://github.com/FabricioPuccio/UTN-TUPaD-P2/tree/main/programacion_estructurada/estructuras_repeticion/ContadorPositivosNegativosCeros/src/com/practica/contadorpositivosnegativosceros

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

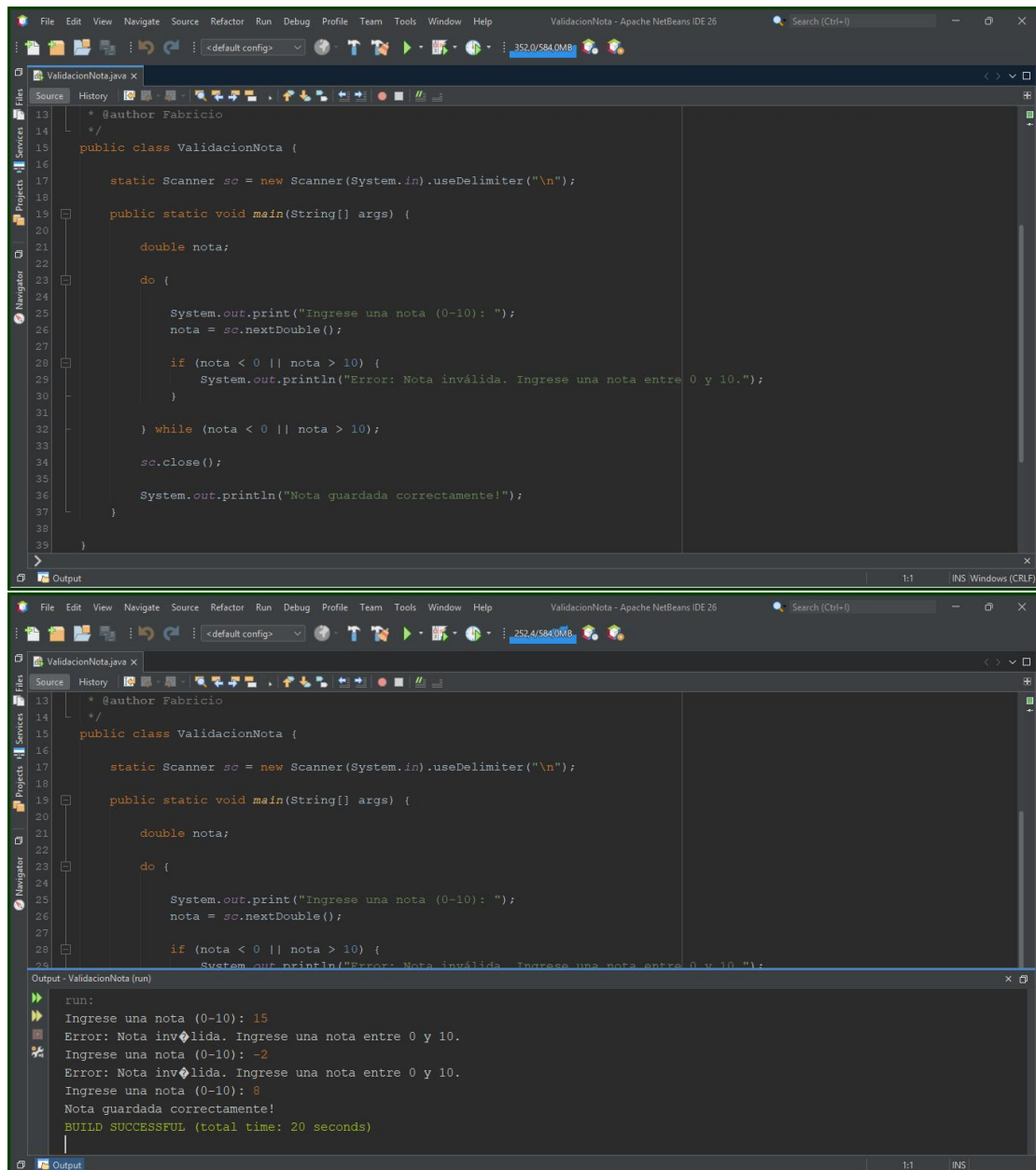
Error: Nota inválida. Ingrese una nota entre 0 y 10.

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.



Link ejercicio resuelto en GitHub:

https://github.com/FabricioPuccio/UTN-TUPaD-P2/tree/main/programacion_estructurada/estructuras_repeticion/ValidacionNota/src/com/practica/validacionnota

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.

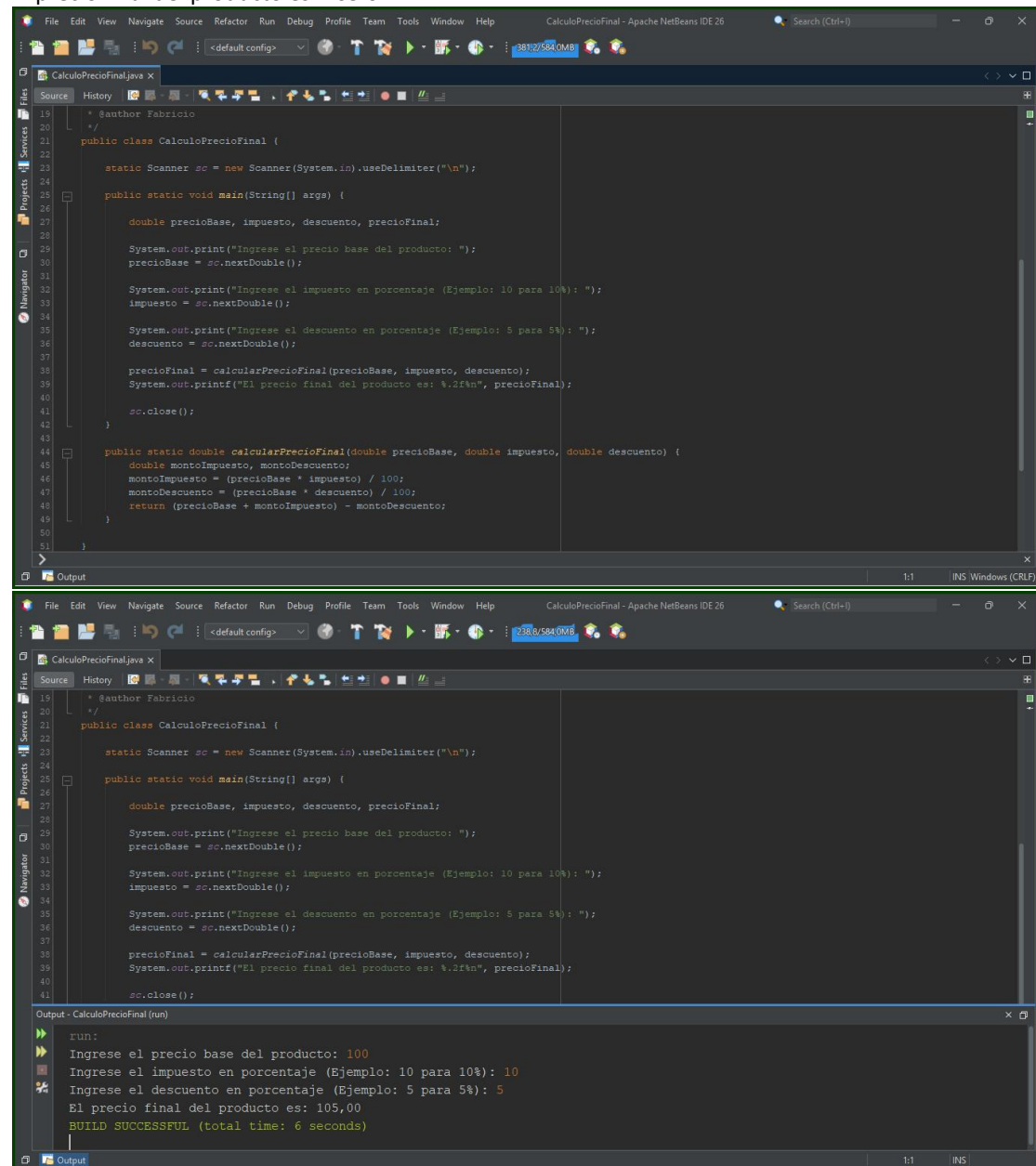
Ejemplo de entrada/salida:

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



```
19  * Santhor Fabricio
20  */
21  public class CalculoPrecioFinal {
22
23      static Scanner sc = new Scanner(System.in).useDelimiter("\n");
24
25      public static void main(String[] args) {
26
27          double precioBase, impuesto, descuento, precioFinal;
28
29          System.out.print("Ingrese el precio base del producto: ");
30          precioBase = sc.nextDouble();
31
32          System.out.print("Ingrese el impuesto en porcentaje (Ejemplo: 10 para 10%): ");
33          impuesto = sc.nextDouble();
34
35          System.out.print("Ingrese el descuento en porcentaje (Ejemplo: 5 para 5%): ");
36          descuento = sc.nextDouble();
37
38          precioFinal = calcularPrecioFinal(precioBase, impuesto, descuento);
39          System.out.printf("El precio final del producto es: %.2f\n", precioFinal);
40
41          sc.close();
42      }
43
44      public static double calcularPrecioFinal(double precioBase, double impuesto, double descuento) {
45          double montoImpuesto, montoDescuento;
46          montoImpuesto = (precioBase * impuesto) / 100;
47          montoDescuento = (precioBase * descuento) / 100;
48          return (precioBase + montoImpuesto) - montoDescuento;
49      }
50
51  }
```

Output - CalculoPrecioFinal (run)

```
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: 105,00
BUILD SUCCESSFUL (total time: 6 seconds)
```

Link ejercicio resuelto en GitHub:

https://github.com/FabricioPuccio/UTN-TUPaD-P2/tree/main/programacion_estructurada/fuciones/CalculoPrecioFinal/src/com/practica/calculopreciofinal

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:

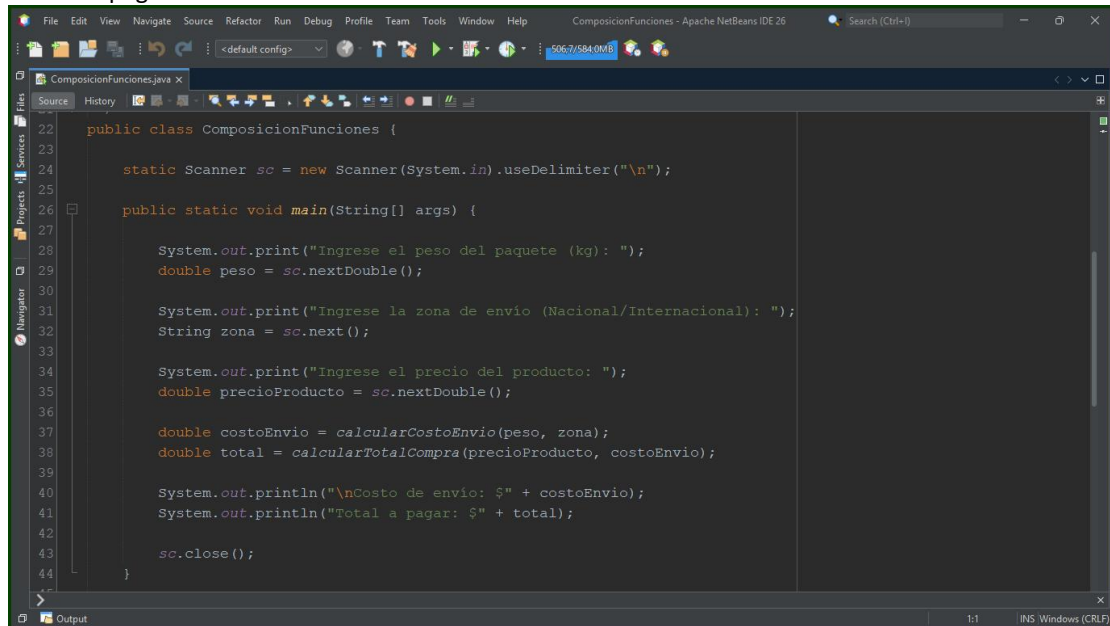
Ingrese el precio del producto: 50

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



```
22 public class ComposicionFunciones {
23
24     static Scanner sc = new Scanner(System.in).useDelimiter("\n");
25
26     public static void main(String[] args) {
27
28         System.out.print("Ingrese el peso del paquete (kg): ");
29         double peso = sc.nextDouble();
30
31         System.out.print("Ingrese la zona de envío (Nacional/Internacional): ");
32         String zona = sc.next();
33
34         System.out.print("Ingrese el precio del producto: ");
35         double precioProducto = sc.nextDouble();
36
37         double costoEnvio = calcularCostoEnvio(peso, zona);
38         double total = calcularTotalCompra(precioProducto, costoEnvio);
39
40         System.out.println("\nCosto de envío: $" + costoEnvio);
41         System.out.println("Total a pagar: $" + total);
42
43         sc.close();
44     }
45 }
```

```
43      sc.close();
44  }
45
46  public static double calcularCostoEnvio(double peso, String zona) {
47
48      if (zona.equalsIgnoreCase("Nacional")) {
49          return peso * 5;
50      } else if (zona.equalsIgnoreCase("Internacional")) {
51          return peso * 10;
52      } else {
53          System.out.println("Zona no válida. Se tomará costo de envío 0.");
54          return 0;
55      }
56  }
57
58
59  public static double calcularTotalCompra(double precioProducto, double costoEnvio) {
60      return precioProducto + costoEnvio;
61  }
62
63  }
64  }
```

Output - ComposicionFunciones (run)

```
run:
Ingrese el peso del paquete (kg): 2
Ingrese la zona de envío (Nacional/Internacional): Nacional
Ingrese el precio del producto: 50

Costo de envío: $10.0
Total a pagar: $60.0
BUILD SUCCESSFUL (total time: 35 seconds)
```

Link ejercicio resuelto en GitHub:

https://github.com/FabricioPuccio/UTN-TUPaD-P2/tree/main/programacion_estructurada/funciones/ComposicionFunciones/src/com/practica/composicionfunciones

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

```
public class ActualizarStock {  
    static Scanner sc = new Scanner(System.in).useDelimiter("\n");  
  
    public static void main(String[] args) {  
        //Declaración de variables  
        int stockActual, cantidadVendida, cantidadRecibida, nuevoStock;  
  
        // Entrada de datos  
        System.out.print("Ingrese el stock actual del producto: ");  
        stockActual = sc.nextInt();  
  
        System.out.print("Ingrese la cantidad vendida: ");  
        cantidadVendida = sc.nextInt();  
  
        System.out.print("Ingrese la cantidad recibida: ");  
        cantidadRecibida = sc.nextInt();  
  
        // Cálculo usando el método  
        nuevoStock = actualizarStock(stockActual, cantidadVendida, cantidadRecibida);  
  
        // Salida de resultados  
        System.out.println("\nEl nuevo stock del producto es: " + nuevoStock);  
  
        sc.close();  
    }  
  
    public static int actualizarStock(int stockActual, int cantidadVendida, int cantidadRecibida) {  
        return stockActual - cantidadVendida + cantidadRecibida;  
    }  
}
```

Output - ActualizarStock (run)

```
run:  
Ingrese el stock actual del producto: 50  
Ingrese la cantidad vendida: 20  
Ingrese la cantidad recibida: 30  
  
El nuevo stock del producto es: 60  
BUILD SUCCESSFUL (total time: 18 seconds)
```

Link ejercicio resuelto en GitHub:

https://github.com/FabrizioPuccio/UTN-TUPaD-P2/tree/main/programacion_estructurada/fuciones/ActualizarStock/src/com/practica/actualizarstock

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%.

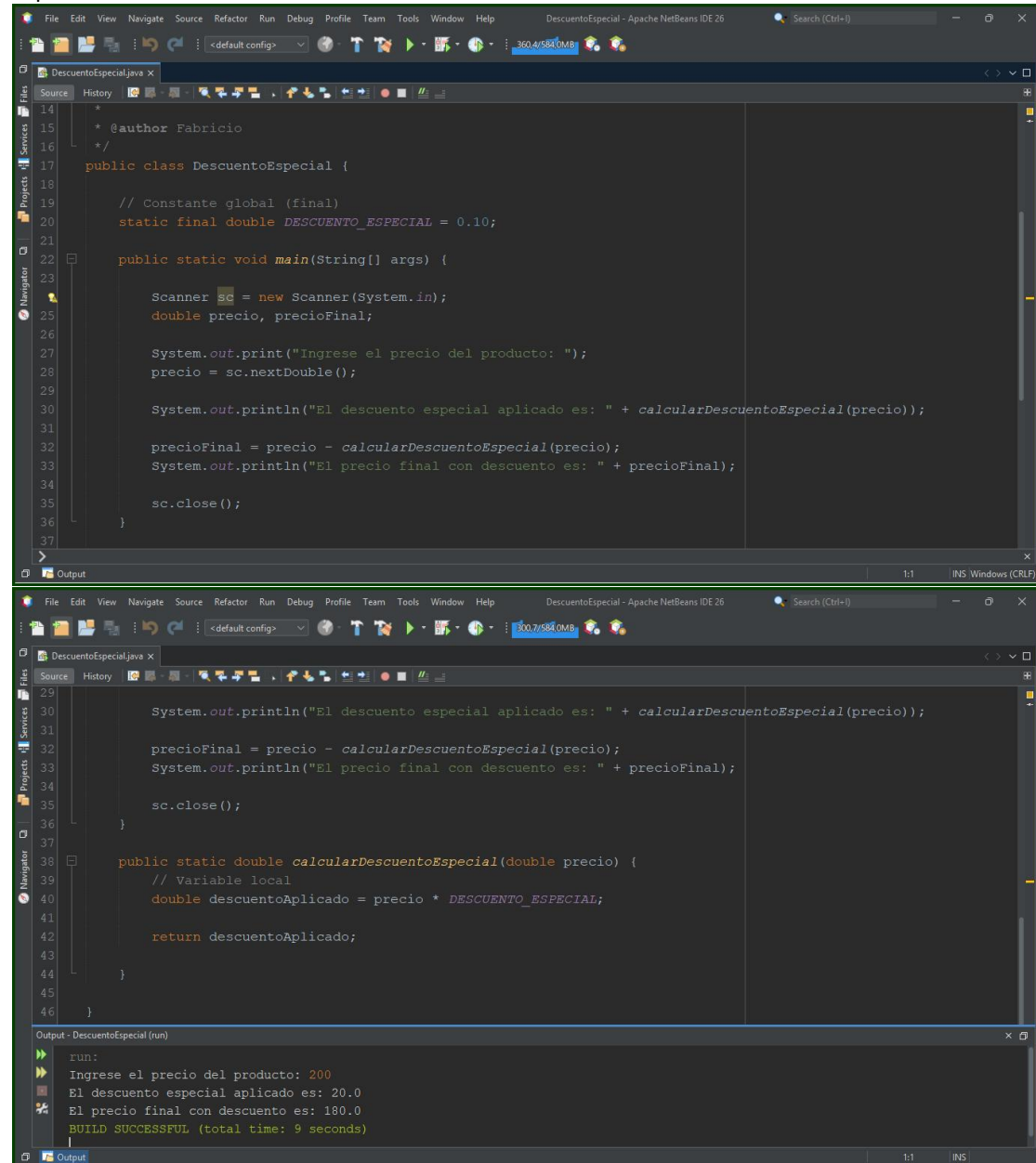
Dentro del método, declara una variable local **descuentoAplicado**, almacena el valor del descuento y muestra el precio final con descuento.

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



The image consists of two screenshots of the Apache NetBeans IDE 26. The top screenshot shows the source code of a Java class named `DescuentoEspecial`. The code includes a package declaration, an author comment, a constant `DESCUENTO_ESPECIAL` set to 0.10, and a `main` method that prompts the user for a price, calculates the discount, and prints the final price. The bottom screenshot shows the same code with an additional `calcularDescuentoEspecial` method. Below the code editor, the 'Output' window shows the program's execution, displaying the input and output values as specified in the example above.

```
14
15  * @author Fabricio
16  */
17  public class DescuentoEspecial {
18
19      // Constante global (final)
20      static final double DESCUENTO_ESPECIAL = 0.10;
21
22      public static void main(String[] args) {
23
24          Scanner sc = new Scanner(System.in);
25          double precio, precioFinal;
26
27          System.out.print("Ingrese el precio del producto: ");
28          precio = sc.nextDouble();
29
30          System.out.println("El descuento especial aplicado es: " + calcularDescuentoEspecial(precio));
31
32          precioFinal = precio - calcularDescuentoEspecial(precio);
33          System.out.println("El precio final con descuento es: " + precioFinal);
34
35          sc.close();
36      }
37
38      public static double calcularDescuentoEspecial(double precio) {
39          // Variable local
40          double descuentoAplicado = precio * DESCUENTO_ESPECIAL;
41
42          return descuentoAplicado;
43      }
44
45  }
46  }
```

Output - DescuentoEspecial (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: 9 seconds)
```

Link ejercicio resuelto en GitHub:

https://github.com/FabricioPuccio/UTN-TUPaD-P2/tree/main/programacion_estructurada/fuciones/DescuentoEspecial/src/com/practica/descuentoespecial

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

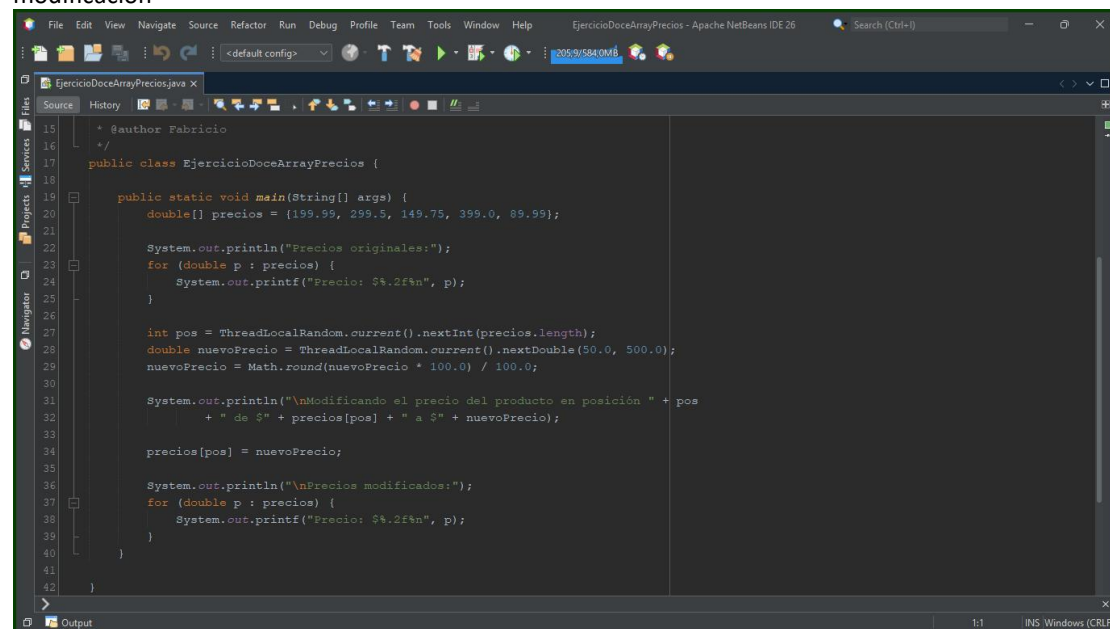
Precio: \$129.99

Precio: \$399.0

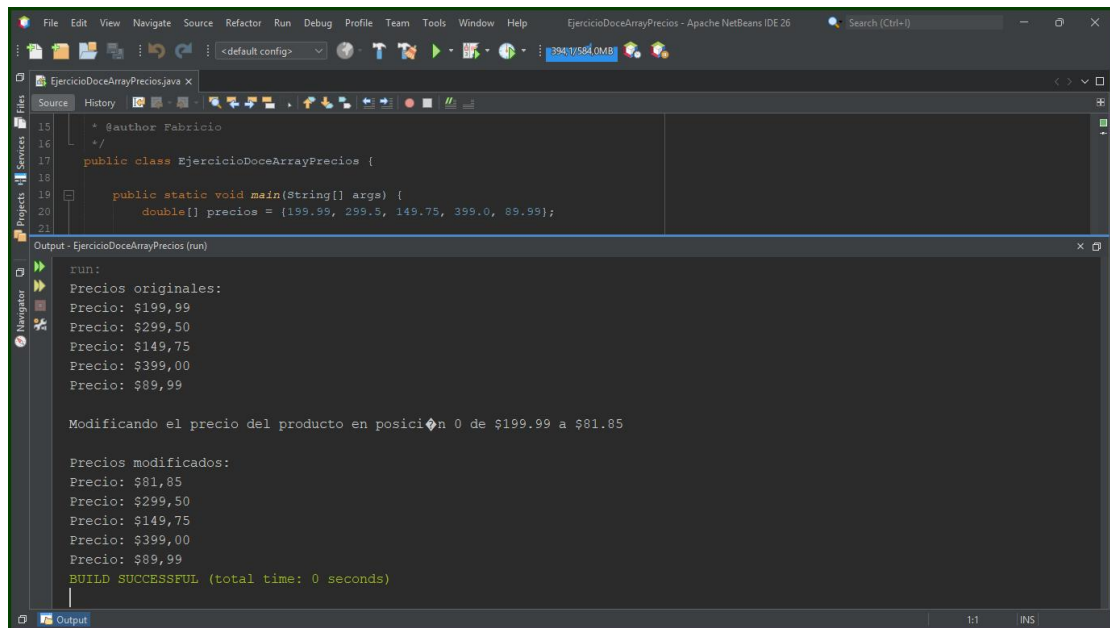
Precio: \$89.99

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



```
15  * @author Fabricio
16  */
17  public class EjercicioDoceArrayPrecios {
18
19      public static void main(String[] args) {
20          double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};
21
22          System.out.println("Precios originales:");
23          for (double p : precios) {
24              System.out.printf("Precio: $%.2f\n", p);
25          }
26
27          int pos = ThreadLocalRandom.current().nextInt(precios.length);
28          double nuevoPrecio = ThreadLocalRandom.current().nextDouble(50.0, 500.0);
29          nuevoPrecio = Math.round(nuevoPrecio * 100.0) / 100.0;
30
31          System.out.println("\nModificando el precio del producto en posición " + pos
32              + " de $" + precios[pos] + " a $" + nuevoPrecio);
33
34          precios[pos] = nuevoPrecio;
35
36          System.out.println("\nPrecios modificados:");
37          for (double p : precios) {
38              System.out.printf("Precio: $%.2f\n", p);
39          }
40      }
41  }
42  }
```



The screenshot shows the Apache NetBeans IDE interface. The main editor window displays a Java file named `EjercicioDoceArrayPrecios.java`. The code includes a package declaration, an author comment, and a `main` method that initializes a `double` array with five values: `{199.99, 299.5, 149.75, 399.0, 89.99}`. Below the editor, the 'Output' window shows the execution results. It first displays the original prices, then a message indicating that the price at index 0 has been modified from `$199.99` to `$81.85`, and finally shows the modified prices. The output concludes with `BUILD SUCCESSFUL (total time: 0 seconds)`.

```
15  * @author Fabricio
16  */
17  public class EjercicioDoceArrayPrecios {
18
19      public static void main(String[] args) {
20          double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};
21      }
```

run:

Precios originales:

Precio: \$199,99

Precio: \$299,50

Precio: \$149,75

Precio: \$399,00

Precio: \$89,99

Modificando el precio del producto en posición 0 de \$199.99 a \$81.85

Precios modificados:

Precio: \$81,85

Precio: \$299,50

Precio: \$149,75

Precio: \$399,00

Precio: \$89,99

BUILD SUCCESSFUL (total time: 0 seconds)

Link ejercicio resuelto en GitHub:

https://github.com/FabricioPuccio/UTN-TUPaD-P2/tree/main/programacion_estructurada/arrays_recurividad/EjercicioDoceArrayPrecios/src/com/practica/ejerciodocearrayprecios

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

The first screenshot shows the source code of a Java class named `EjercicioTreceArrayRecursivoPrecios`. The code includes a `main` method that initializes an array of prices, prints them, modifies the third element, and prints them again. It also includes a recursive method `imprimirPrecios` to print the array elements.

```
32  * @author Fabricio
33  */
34  public class EjercicioTreceArrayRecursivoPrecios {
35
36      public static void main(String[] args) {
37          double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};
38
39          System.out.println("Precios originales:");
40          imprimirPrecios(precios, 0);
41
42          int indiceAModificar = 2; // Por ejemplo, modificamos el tercer producto
43          double nuevoPrecio = 129.99;
44          precios[indiceAModificar] = nuevoPrecio;
45
46          System.out.println("\nPrecios modificados:");
47          imprimirPrecios(precios, 0);
48      }
49
50      public static void imprimirPrecios(double[] precios, int indice) {
51          if (indice < precios.length) {
52              System.out.printf("Precio: %.2f%n", precios[indice]);
53              imprimirPrecios(precios, indice + 1);
54          }
55      }
56  }
```

The second screenshot shows the output of the program. It displays the original prices, followed by the modified prices where the third element has been changed to 129.99. The output ends with a successful build message.

```
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)
```

Link ejercicio resuelto en GitHub:

https://github.com/FabricioPuccio/UTN-TUPaD-P2/tree/main/programacion_estructurada/arrays_recursividad/EjercicioTreceArrayRecursivoPrecios/src/com/practica/ejerciotrecearrayrecursivoprecios

Link repositorio en GitHub:

<https://github.com/FabricioPuccio/UTN-TUPaD-P2>