

Tecnicatura Universitaria en Programación a Distancia

PROGRAMACIÓN II

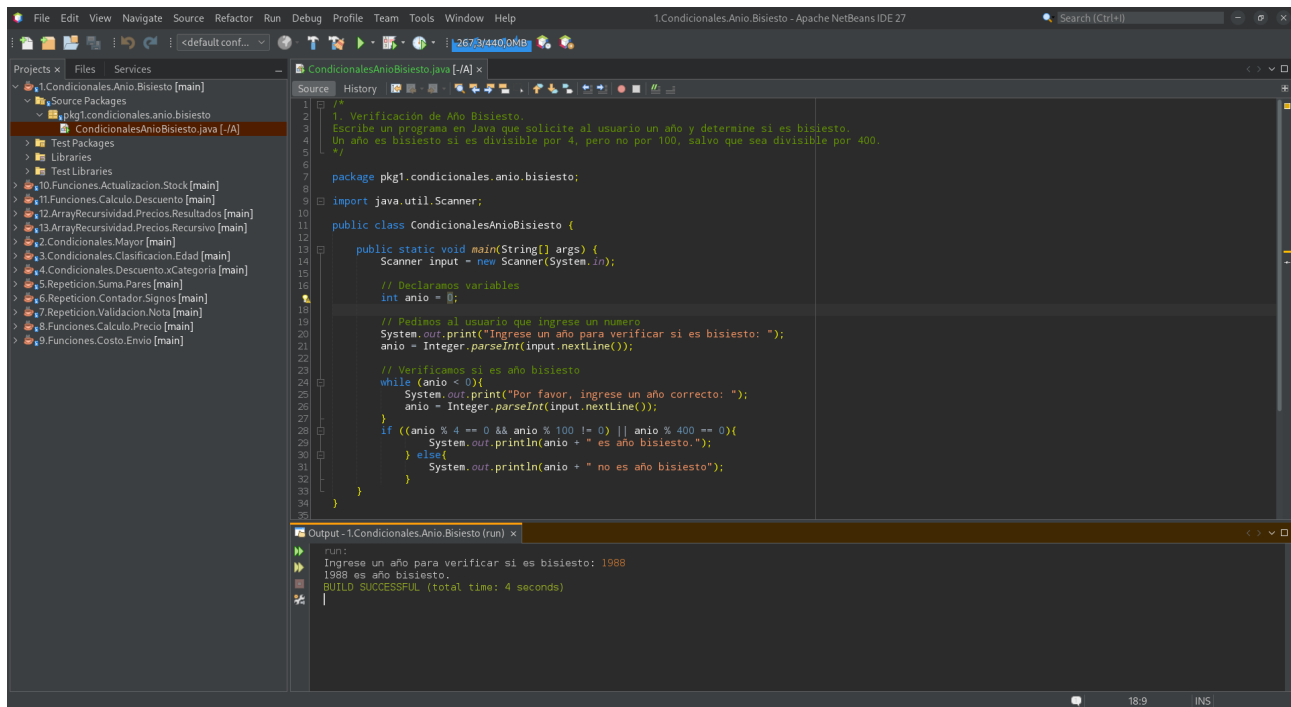
Trabajo Práctico 2: Programación Estructurada

Casos prácticos

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.



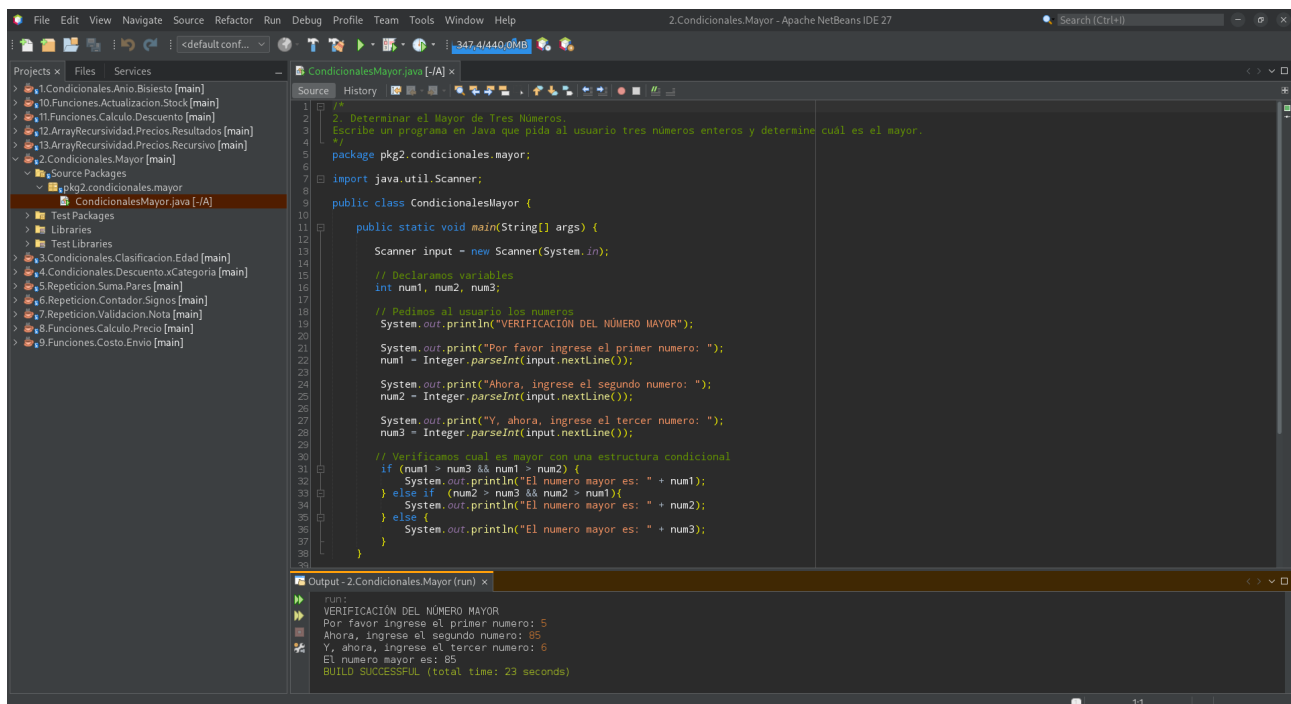
```
1 /*
2  1. Verificación de Año Bisiesto.
3  Escribe un programa en Java que solicite al usuario un año y determine si es bisiesto.
4  Un año es bisiesto si es divisible por 4, pero no por 100, salvo que sea divisible por 400.
5  */
6
7 package pkg1.condicionales.año.bisiesto;
8
9 import java.util.Scanner;
10
11 public class CondicionalesAñoBisiesto {
12
13     public static void main(String[] args) {
14         Scanner input = new Scanner(System.in);
15
16         // Declaramos variables
17         int año = 0;
18
19         // Pedimos al usuario que ingrese un número
20         System.out.print("Ingrese un año para verificar si es bisiesto: ");
21         año = Integer.parseInt(input.nextLine());
22
23         // Verificamos si es año bisiesto
24         while (año < 0) {
25             System.out.print("Por favor, ingrese un año correcto: ");
26             año = Integer.parseInt(input.nextLine());
27         }
28         if ((año % 4 == 0 && año % 100 != 0) || año % 400 == 0) {
29             System.out.println(año + " es año bisiesto.");
30         } else {
31             System.out.println(año + " no es año bisiesto");
32         }
33     }
34 }
```

Output - 1.Condicionales.Año.Bisiesto (run)

```
run:
Ingresar un año para verificar si es bisiesto: 1988
1988 es año bisiesto.
BUILD SUCCESSFUL (total time: 4 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.



```
1 /*
2  2. Determinar el Mayor de Tres Números.
3  Escribe un programa en Java que pida al usuario tres números enteros y determine cuál es el mayor.
4  */
5
6 package pkg2.condicionales.mayor;
7
8 import java.util.Scanner;
9
10 public class CondicionalesMayor {
11
12     public static void main(String[] args) {
13         Scanner input = new Scanner(System.in);
14
15         // Declaramos variables
16         int num1, num2, num3;
17
18         // Pedimos al usuario los números
19         System.out.println("VERIFICACIÓN DEL NÚMERO MAYOR");
20
21         System.out.print("Por favor ingrese el primer número: ");
22         num1 = Integer.parseInt(input.nextLine());
23
24         System.out.print("Ahora, ingrese el segundo número: ");
25         num2 = Integer.parseInt(input.nextLine());
26
27         System.out.print("Y, ahora, ingrese el tercer número: ");
28         num3 = Integer.parseInt(input.nextLine());
29
30         // Verificamos cual es mayor con una estructura condicional
31         if (num1 > num3 && num1 > num2) {
32             System.out.println("El número mayor es: " + num1);
33         } else if (num2 > num3 && num2 > num1) {
34             System.out.println("El número mayor es: " + num2);
35         } else {
36             System.out.println("El número mayor es: " + num3);
37         }
38     }
39 }
```

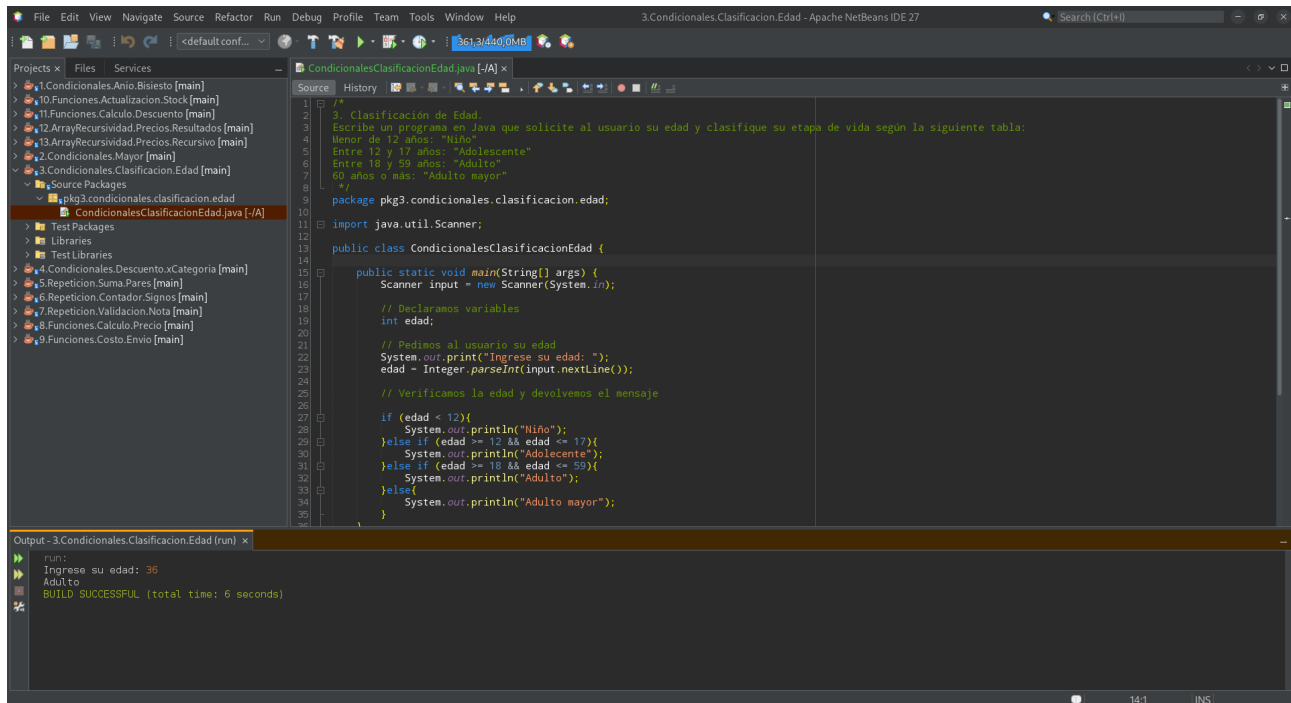
Output - 2.Condicionales.Mayor (run)

```
run:
VERIFICACIÓN DEL NÚMERO MAYOR
Por favor ingrese el primer número: 5
Ahora, ingrese el segundo número: 85
Y, ahora, ingrese el tercer número: 6
El número mayor es: 85
BUILD SUCCESSFUL (total time: 23 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:

1. Menor de 12 años: "Niño"
2. Entre 12 y 17 años: "Adolescente"
3. Entre 18 y 59 años: "Adulto"
4. 60 años o más: "Adulto mayor"



```
1  /**
2  3. Clasificación de Edad.
3  Escribe un programa en Java que solicite al usuario su edad y clasifique su etapa de vida según la siguiente tabla:
4  Menor de 12 años: "Niño"
5  Entre 12 y 17 años: "Adolescente"
6  Entre 18 y 59 años: "Adulto"
7  60 años o más: "Adulto mayor"
8  */
9
10 package pkg3.condicionales.clasificacion.edad;
11
12 import java.util.Scanner;
13
14 public class CondicionalesClasificacionEdad {
15
16     public static void main(String[] args) {
17         Scanner input = new Scanner(System.in);
18
19         // Declaramos variables
20         int edad;
21
22         // Pedimos al usuario su edad
23         System.out.print("Ingrese su edad: ");
24         edad = Integer.parseInt(input.nextLine());
25
26         // Verificamos la edad y devolvemos el mensaje
27
28         if (edad < 12){
29             System.out.println("Niño");
30         }else if (edad >= 12 && edad <= 17){
31             System.out.println("Adolescente");
32         }else if (edad >= 18 && edad <= 59){
33             System.out.println("Adulto");
34         }else{
35             System.out.println("Adulto mayor");
36         }
37     }
38 }
```

Output - 3.Condicionales.Clasificacion.Edad (run) x

```
run:
Ingreso su edad: 36
Adulto
BUILD SUCCESSFUL (total time: 6 seconds)
```

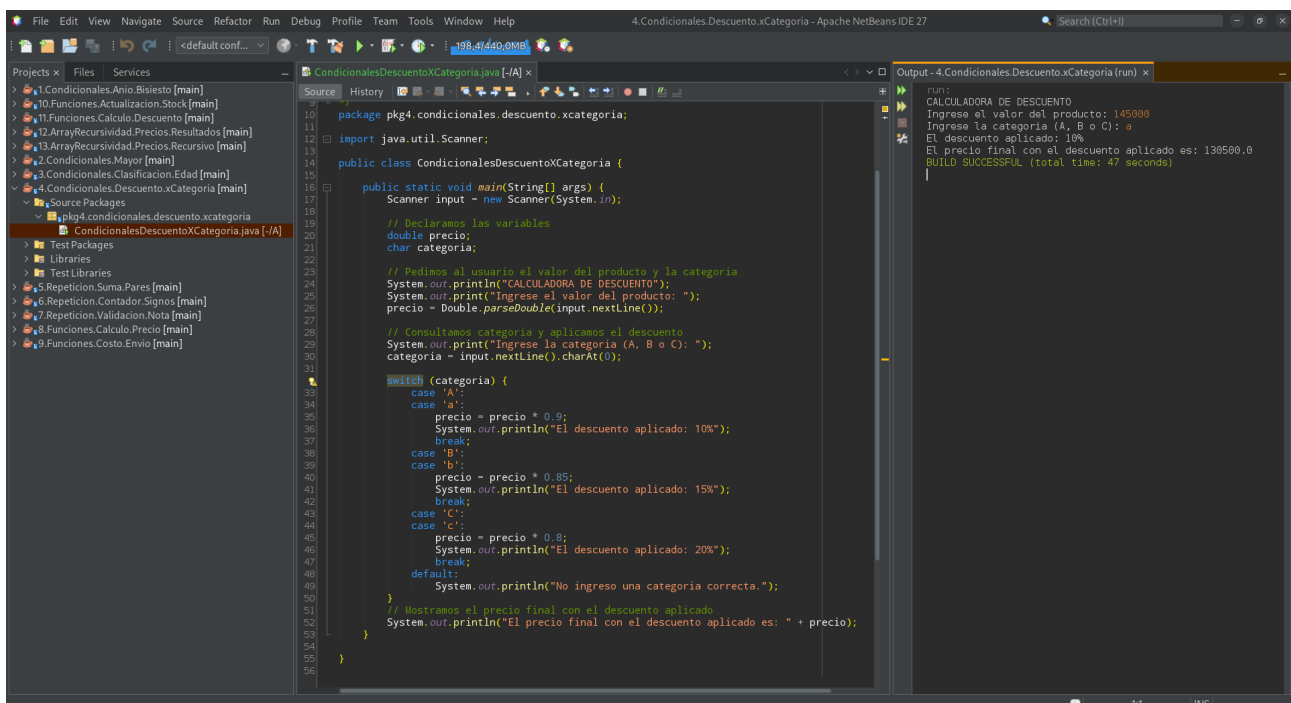
4. Calculadora de Descuento según categoría.

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.



```
1  package pkg4.condicionales.descuento.xcategoria;
2
3  import java.util.Scanner;
4
5  public class CondicionalesDescuentoXCategoria {
6
7     public static void main(String[] args) {
8         Scanner input = new Scanner(System.in);
9
10         // Declaramos las variables
11         double precio;
12         char categoria;
13
14         // Pedimos al usuario el valor del producto y la categoría
15         System.out.println("CALCULADORA DE DESCUENTO");
16         System.out.print("Ingrese el valor del producto: ");
17         precio = Double.parseDouble(input.nextLine());
18
19         // Consultamos categoría y aplicamos el descuento
20         System.out.print("Ingrese la categoría (A, B o C): ");
21         categoria = input.nextLine().charAt(0);
22
23         switch (categoria) {
24             case 'A':
25                 precio = precio * 0.9;
26                 System.out.println("El descuento aplicado: 10%");
27                 break;
28             case 'B':
29                 precio = precio * 0.85;
30                 System.out.println("El descuento aplicado: 15%");
31                 break;
32             case 'C':
33                 precio = precio * 0.8;
34                 System.out.println("El descuento aplicado: 20%");
35                 break;
36             default:
37                 System.out.println("No ingreso una categoría correcta.");
38         }
39
40         // Mostramos el precio final con el descuento aplicado
41         System.out.println("El precio final con el descuento aplicado es: " + precio);
42     }
43 }
```

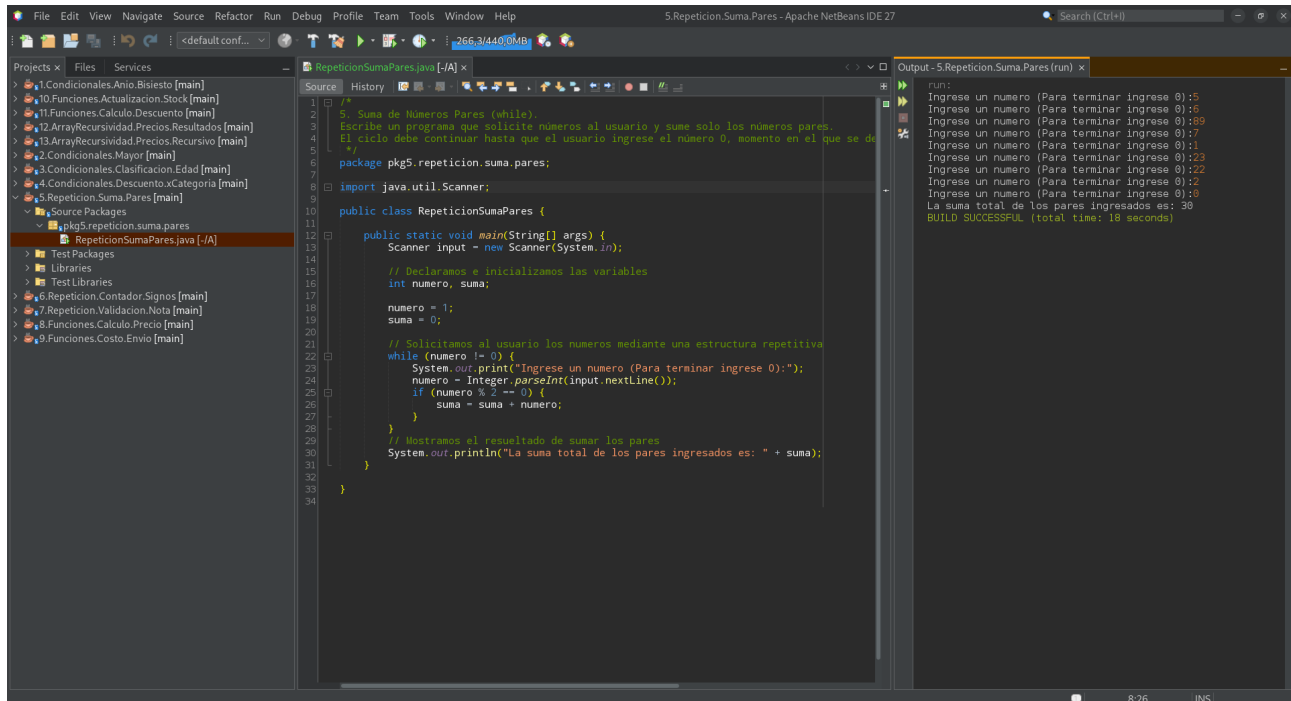
Output - 4.Condicionales.Descuento.xCategoria (run) x

```
run:
CALCULADORA DE DESCUENTO
Ingreso el valor del producto: 145000
Ingreso la categoría (A, B o C): a
El descuento aplicado: 10%
El precio final con el descuento aplicado es: 130500.0
BUILD SUCCESSFUL (total time: 47 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.



The screenshot shows the NetBeans IDE with the file '5.Repetición.Suma.Pares' open. The code is as follows:

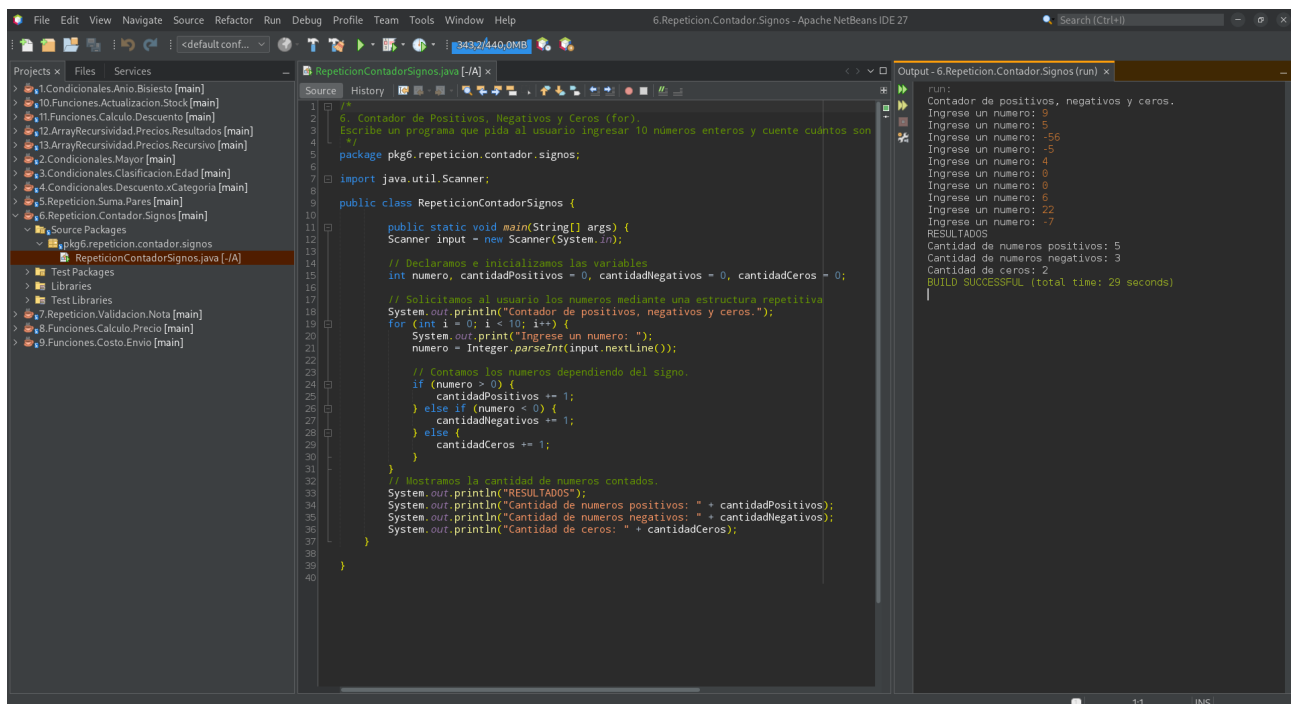
```
1 /*
2  5. Suma de Números Pares (while).
3  Escribe un programa que solicite números al usuario y sume solo los números pares.
4  El ciclo debe continuar hasta que el usuario ingrese el número 0, momento en el que se debe
5  mostrar la suma total de los pares ingresados.
6  */
7 package pkg5.repeticion.suma.pares;
8 import java.util.Scanner;
9
10 public class RepeticionSumaPares {
11
12     public static void main(String[] args) {
13         Scanner input = new Scanner(System.in);
14
15         // Declaramos e inicializamos las variables
16         int numero, suma;
17
18         numero = 1;
19         suma = 0;
20
21         // Solicitamos al usuario los numeros mediante una estructura repetitiva
22         while (numero != 0) {
23             System.out.print("Ingrese un numero (Para terminar ingrese 0):");
24             numero = Integer.parseInt(input.nextLine());
25             if (numero % 2 == 0) {
26                 suma = suma + numero;
27             }
28         }
29         // Mostramos el resultado de sumar los pares
30         System.out.println("La suma total de los pares ingresados es: " + suma);
31     }
32 }
33
34
```

The output window shows the following execution:

```
run:
Ingresar un numero (Para terminar ingrese 0):5
Ingresar un numero (Para terminar ingrese 0):6
Ingresar un numero (Para terminar ingrese 0):89
Ingresar un numero (Para terminar ingrese 0):7
Ingresar un numero (Para terminar ingrese 0):1
Ingresar un numero (Para terminar ingrese 0):23
Ingresar un numero (Para terminar ingrese 0):22
Ingresar un numero (Para terminar ingrese 0):2
Ingresar un numero (Para terminar ingrese 0):8
La suma total de los pares ingresados es: 30
BUILD SUCCESSFUL (total time: 18 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 zeros.



The screenshot shows the NetBeans IDE with the file '6.Repetición.Contador.Signos' open. The code is as follows:

```
1 /*
2  6. Contador de Positivos, Negativos y Ceros (for).
3  Escribe un programa que pida al usuario ingresar 10 numeros enteros y cuente cuantos son
4  positivos, negativos y zeros.
5  */
6 package pkg6.repeticion.contador.signos;
7 import java.util.Scanner;
8
9 public class RepeticionContadorSignos {
10
11     public static void main(String[] args) {
12         Scanner input = new Scanner(System.in);
13
14         // Declaramos e inicializamos las variables
15         int numero, cantidadPositivos = 0, cantidadNegativos = 0, cantidadCeros = 0;
16
17         // Solicitamos al usuario los numeros mediante una estructura repetitiva
18         System.out.println("Contador de positivos, negativos y zeros.");
19         for (int i = 0; i < 10; i++) {
20             System.out.print("Ingrese un numero: ");
21             numero = Integer.parseInt(input.nextLine());
22
23             // Contamos los numeros dependiendo del signo.
24             if (numero > 0) {
25                 cantidadPositivos ++ 1;
26             } else if (numero < 0) {
27                 cantidadNegativos ++ 1;
28             } else {
29                 cantidadCeros ++ 1;
30             }
31         }
32         // Mostramos la cantidad de numeros contados.
33         System.out.println("RESULTADOS");
34         System.out.println("Cantidad de numeros positivos: " + cantidadPositivos);
35         System.out.println("Cantidad de numeros negativos: " + cantidadNegativos);
36         System.out.println("Cantidad de zeros: " + cantidadCeros);
37     }
38 }
39
40
```

The output window shows the following execution:

```
run:
Contador de positivos, negativos y zeros.
Ingresar un numero: 5
Ingresar un numero: -56
Ingresar un numero: -5
Ingresar un numero: 4
Ingresar un numero: 0
Ingresar un numero: 0
Ingresar un numero: 6
Ingresar un numero: 22
Ingresar un numero: -7
RESULTADOS
Cantidad de numeros positivos: 5
Cantidad de numeros negativos: 3
Cantidad de zeros: 2
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.

The screenshot shows the Apache NetBeans IDE with a project named '7.Repeticion.Validacion.Nota'. The source code for 'RepeticionValidacionNota.java' is displayed. The code uses a do-while loop to repeatedly prompt the user for a grade until a valid value (between 0 and 10) is entered. The output window shows the program's execution, including the error message for an invalid input and the success message for a valid input.

```
1  /*
2  7. Validación de Nota entre 0 y 10 (do-while).
3  Escribe un programa que solicite al usuario una nota entre 0 y 10. Si el usuario ingresa u
4  debe seguir pidiéndole la nota hasta que ingrese un valor válido.
5  */
6  package pkg7.repeticion.validacion.nota;
7
8  import java.util.Scanner;
9
10 public class RepeticionValidacionNota {
11
12     public static void main(String[] args) {
13         Scanner input = new Scanner(System.in);
14
15         // Declaramos las variables
16         int nota = 1;
17
18         // Solicitamos al usuario las notas mediante una estructura repetitiva.
19         do {
20             if (nota < 0 || nota > 10) {
21                 System.out.println("Error: Nota inválida. Ingrese una nota entre 0 y 10.");
22             }
23             System.out.print("Ingrese la nota (Debe ser de 0 a 10): ");
24             nota = Integer.parseInt(input.nextLine());
25             while (nota < 0 || nota > 10);
26
27             // Mostramos el mensaje de funcionamiento proceso correcto.
28             System.out.println("La nota se guardo correctamente.");
29         }
30     }
31 }
32
```

Output - 7.Repeticion.Validacion.Nota (run) x

```
run:
Ingrese la nota (Debe ser de 0 a 10): -1
Error: Nota inválida. Ingrese una nota entre 0 y 10.
Ingrese la nota (Debe ser de 0 a 10): 10
La nota se guardo correctamente.
BUILD SUCCESSFUL (total time: 15 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:

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

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.

The screenshot shows the Apache NetBeans IDE with a project named '8.Funciones.Calculo.Precio'. The source code for 'FuncionesCalculoPrecio.java' is displayed. The code defines a static method `calcularPrecioFinal` that calculates the final price based on the base price, tax, and discount. The `main` method prompts the user for these values and calls the `calcularPrecioFinal` method. The output window shows the program's execution, including the prompts for price, tax, and discount, and the final calculated price.

```
1  /*
2  8. Cálculo del Precio Final con impuesto y descuento.
3
4  Crea un método calcularPrecioFinal(double impuesto, double descuento) que calcule el precio final de
5  La fórmula es:
6  PrecioFinal = PrecioBase + (PrecioBase*Impuesto) - (PrecioBase*Descuento)
7  Desde main(), solicita el precio base del producto, el porcentaje de impuesto y el porcentaje de desc
8  final.
9  */
10 package pkg8.funciones.calculo.precio;
11
12 import java.util.Scanner;
13
14 public class FuncionesCalculoPrecio {
15
16     public static void main(String[] args) {
17         Scanner input = new Scanner(System.in);
18         // Declaramos variables
19         double precioFinal, precioProducto, impuesto, descuento;
20
21         // Solicitamos al usuario el precio, porcentaje de impuesto y el porcentaje de descuento.
22         System.out.print("Ingrese el precio del producto: ");
23         precioProducto = Double.parseDouble(input.nextLine());
24
25         System.out.print("Ingrese el porcentaje del impuesto (10 para 10%): ");
26         impuesto = Double.parseDouble(input.nextLine());
27
28         System.out.print("Ingrese el porcentaje del descuento (5 para 5%): ");
29         descuento = Double.parseDouble(input.nextLine());
30
31         // Mostramos el precio final en pantalla
32         precioFinal = calcularPrecioFinal(precioProducto, impuesto, descuento);
33         System.out.println("El precio final es de: " + precioFinal);
34     }
35
36     public static double calcularPrecioFinal(double precioBase, double impuesto, double descuento) {
37         impuesto = impuesto / 100;
38         descuento = descuento / 100;
39         return precioBase + (precioBase * impuesto) - (precioBase * descuento);
40     }
41 }
42
43
44
```

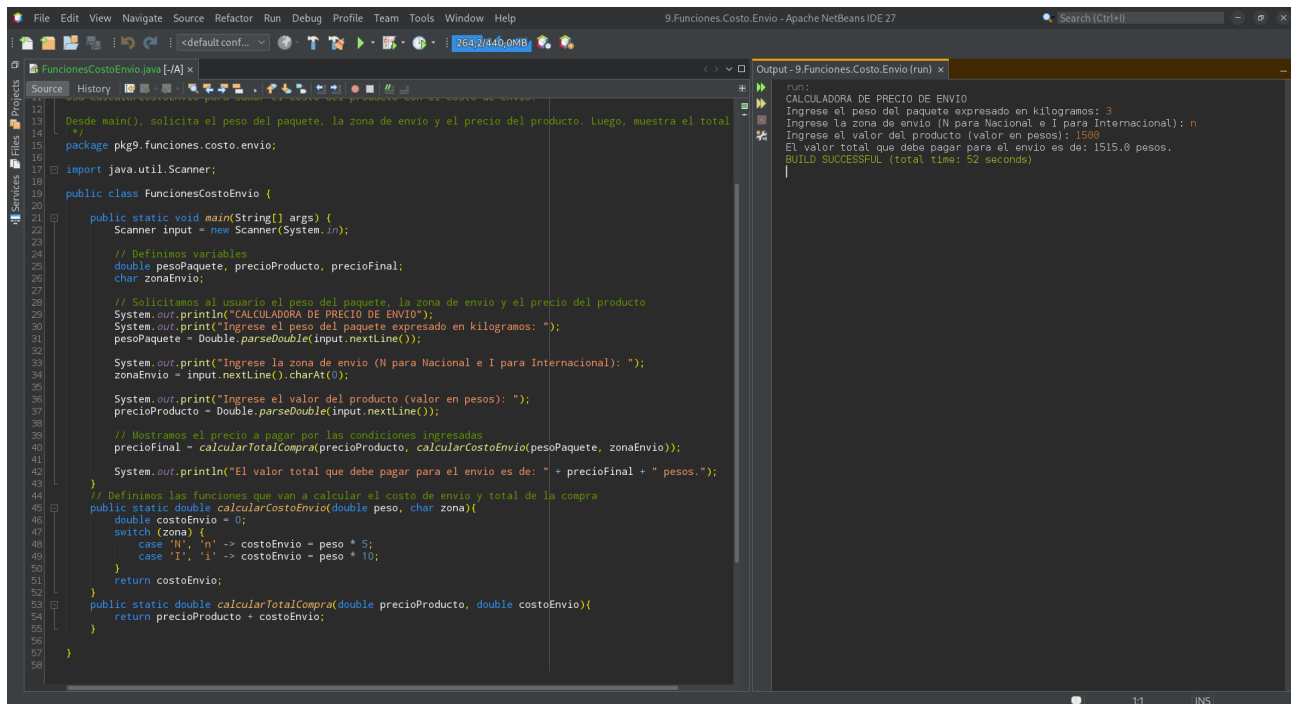
Output - 8.Funciones.Calculo.Precio (run) x

```
run:
Ingrese el precio del producto: 2000
Ingrese el porcentaje del impuesto (10 para 10%): 20
Ingrese el porcentaje del descuento (5 para 5%): 5
El precio final es de: 2300.0
BUILD SUCCESSFUL (total time: 15 seconds)
```

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

- **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 y Internacional: \$10 por kg)
- **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.



```
12 Desde main(), solicita el peso del paquete, la zona de envío y el precio del producto. Luego, muestra el total a pagar.
13
14 package pkg9.funciones.costo.envio;
15
16 import java.util.Scanner;
17
18 public class FuncionesCostoEnvio {
19
20     public static void main(String[] args) {
21         Scanner input = new Scanner(System.in);
22
23         // Definimos variables
24         double pesoPaquete, precioProducto, precioFinal;
25         char zonaEnvio;
26
27         // Solicitamos al usuario el peso del paquete, la zona de envío y el precio del producto
28         System.out.println("CALCULADORA DE PRECIO DE ENVIO");
29         System.out.print("Ingrese el peso del paquete expresado en kilogramos: ");
30         pesoPaquete = Double.parseDouble(input.nextLine());
31
32         System.out.print("Ingrese la zona de envío (N para Nacional e I para Internacional): ");
33         zonaEnvio = input.nextLine().charAt(0);
34
35         System.out.print("Ingrese el valor del producto (valor en pesos): ");
36         precioProducto = Double.parseDouble(input.nextLine());
37
38         // Mostramos el precio a pagar por las condiciones ingresadas
39         precioFinal = calcularTotalCompra(precioProducto, calcularCostoEnvio(pesoPaquete, zonaEnvio));
40
41         System.out.println("El valor total que debe pagar para el envío es de: " + precioFinal + " pesos.");
42
43         // Definimos las funciones que van a calcular el costo de envío y total de la compra
44         public static double calcularCostoEnvio(double peso, char zona){
45             double costoEnvio = 0;
46             switch (zona) {
47                 case 'N', 'n' -> costoEnvio = peso * 5;
48                 case 'I', 'i' -> costoEnvio = peso * 10;
49             }
50             return costoEnvio;
51         }
52
53         public static double calcularTotalCompra(double precioProducto, double costoEnvio){
54             return precioProducto + costoEnvio;
55         }
56     }
57 }
```

Output - 9.Funciones.Costo.Envio (run) x

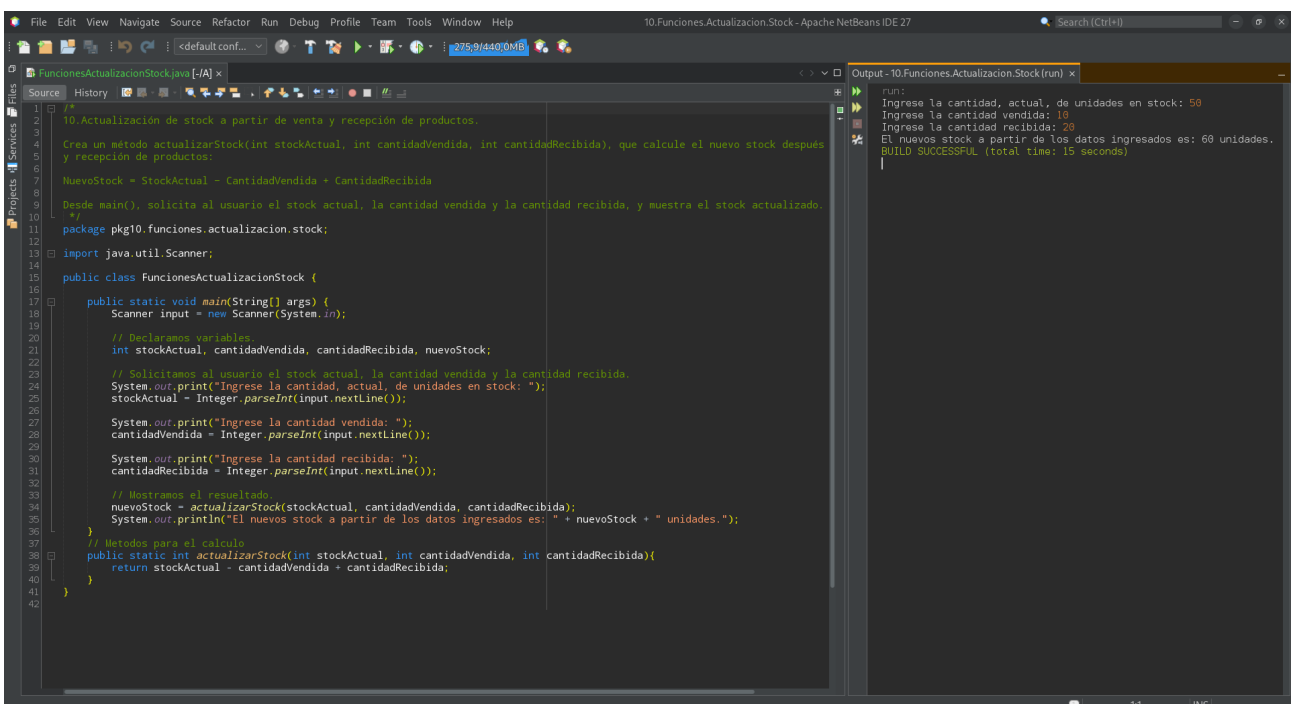
```
run:
CALCULADORA DE PRECIO DE ENVIO
Ingrese el peso del paquete expresado en kilogramos: 3
Ingrese la zona de envío (N para Nacional e I para Internacional): n
Ingrese el valor del producto (valor en pesos): 1500
El valor total que debe pagar para el envío es de: 1515.0 pesos.
BUILD SUCCESSFUL (total time: 52 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:

- $NuevoStock = StockActual - CantidadVendida + CantidadRecibida$

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



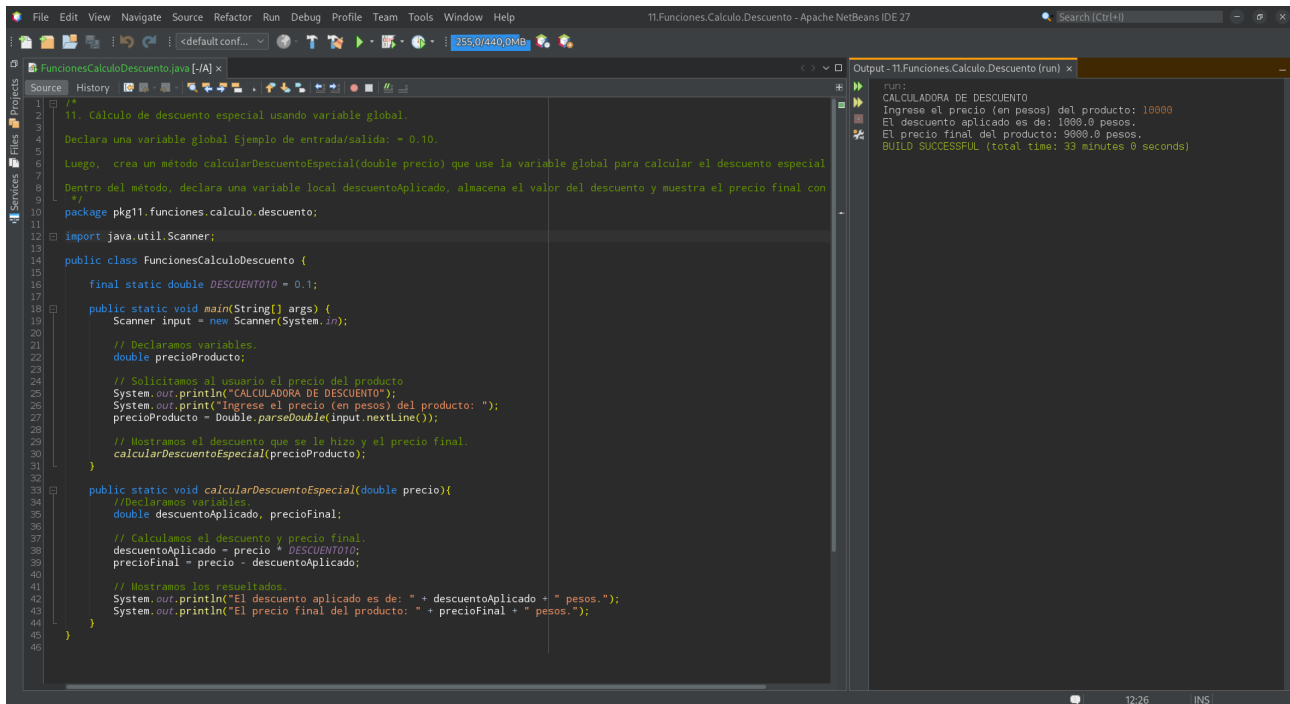
```
1 10.Actualización de stock a partir de venta y recepción de productos.
2 Crea un método actualizarStock(int stockActual, int cantidadVendida, int cantidadRecibida), que calcule el nuevo stock después
3 y recepción de productos.
4
5 NuevoStock = StockActual - CantidadVendida + CantidadRecibida
6
7 Desde main(), solicita al usuario el stock actual, la cantidad vendida y la cantidad recibida, y muestra el stock actualizado.
8
9 package pkg10.funciones.actualizacion.stock;
10
11 import java.util.Scanner;
12
13 public class FuncionesActualizacionStock {
14
15     public static void main(String[] args) {
16         Scanner input = new Scanner(System.in);
17
18         // Declaramos variables
19         int stockActual, cantidadVendida, cantidadRecibida, nuevoStock;
20
21         // Solicitamos al usuario el stock actual, la cantidad vendida y la cantidad recibida.
22         System.out.print("Ingrese la cantidad, actual, de unidades en stock: ");
23         stockActual = Integer.parseInt(input.nextLine());
24
25         System.out.print("Ingrese la cantidad vendida: ");
26         cantidadVendida = Integer.parseInt(input.nextLine());
27
28         System.out.print("Ingrese la cantidad recibida: ");
29         cantidadRecibida = Integer.parseInt(input.nextLine());
30
31         // Mostramos el resultado
32         nuevoStock = actualizarStock(stockActual, cantidadVendida, cantidadRecibida);
33         System.out.println("El nuevo stock a partir de los datos ingresados es: " + nuevoStock + " unidades.");
34
35         // Metodos para el calculo
36         public static int actualizarStock(int stockActual, int cantidadVendida, int cantidadRecibida){
37             return stockActual - cantidadVendida + cantidadRecibida;
38         }
39     }
40 }
41
42 }
```

Output - 10.Funciones.Actualizacion.Stock (run) x

```
run:
Ingrese la cantidad, actual, de unidades en stock: 50
Ingrese la cantidad vendida: 10
Ingrese la cantidad recibida: 20
El nuevo stock a partir de los datos ingresados es: 60 unidades.
BUILD SUCCESSFUL (total time: 15 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.



```
1  /**
2   * 11. Cálculo de descuento especial usando variable global.
3   *
4   * Declara una variable global Ejemplo de entrada/salida: = 0.10.
5   * Luego, crea un método calcularDescuentoEspecial(double precio) que use la variable global para calcular el descuento especial
6   * Dentro del método, declara una variable local descuentoAplicado, almacena el valor del descuento y muestra el precio final con
7   *
8   */
9
10 package pkg11.funciones.calculo.descuento;
11
12 import java.util.Scanner;
13
14 public class FuncionesCalculoDescuento {
15
16     final static double DESCUENTO10 = 0.1;
17
18     public static void main(String[] args) {
19         Scanner input = new Scanner(System.in);
20
21         // Declaramos variables.
22         double precioProducto;
23
24         // Solicitamos al usuario el precio del producto
25         System.out.println("CALCULADORA DE DESCUENTO");
26         System.out.print("Ingrese el precio (en pesos) del producto: ");
27         precioProducto = Double.parseDouble(input.nextLine());
28
29         // Mostramos el descuento que se le hizo y el precio final.
30         calcularDescuentoEspecial(precioProducto);
31     }
32
33     public static void calcularDescuentoEspecial(double precio){
34         //Declaramos variables.
35         double descuentoAplicado, precioFinal;
36
37         // Calculamos el descuento y precio final.
38         descuentoAplicado = precio * DESCUENTO10;
39         precioFinal = precio - descuentoAplicado;
40
41         // Mostramos los resultados.
42         System.out.println("El descuento aplicado es de: " + descuentoAplicado + " pesos.");
43         System.out.println("El precio final del producto: " + precioFinal + " pesos.");
44     }
45 }
46
```

Output - 11.Funciones.Calculo.Descuento (run) x

```
run:
CALCULADORA DE DESCUENTO
Ingrese el precio (en pesos) del producto: 10000
El descuento aplicado es de: 1000.0 pesos.
El precio final del producto: 9000.0 pesos.
BUILD SUCCESSFUL (total time: 33 minutes 0 seconds)
```

(continua en la siguiente página)

Arrays y Recursividad

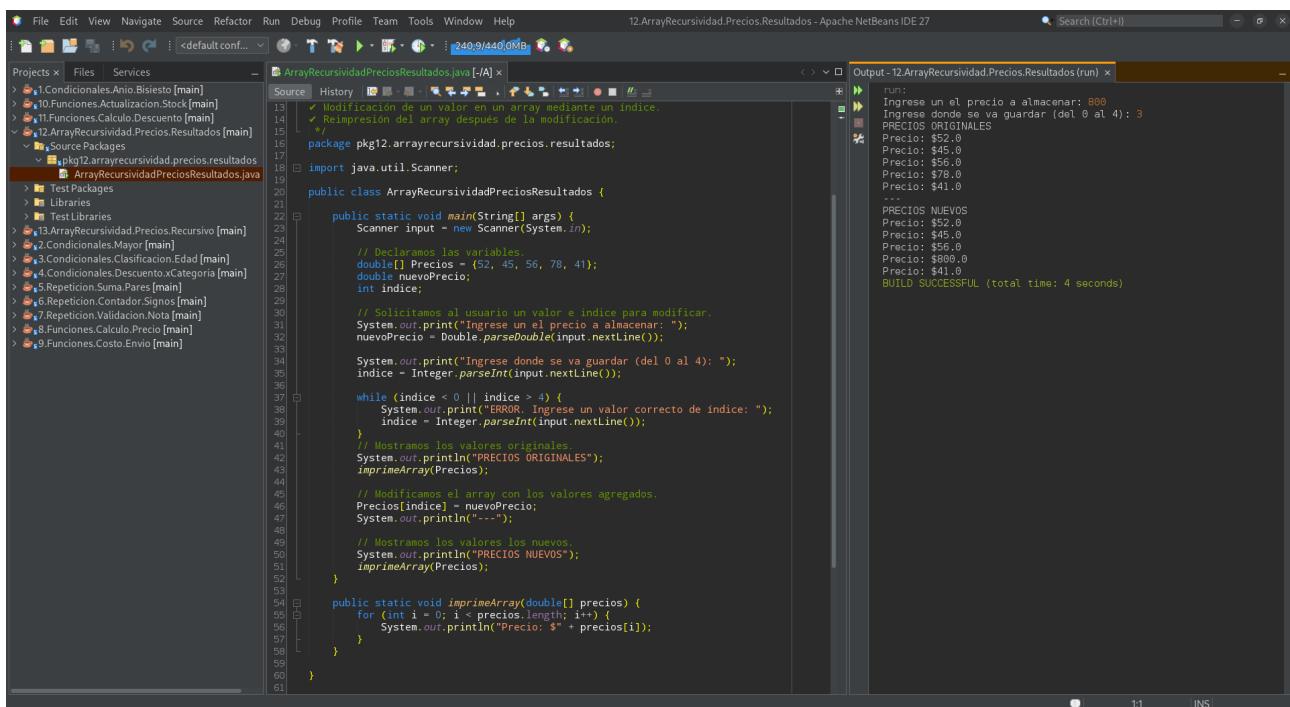
12. Modificación de un array de precios y visualización de resultados.

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.

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 the NetBeans IDE with a project named '12.ArrayRecursividad.Precios.Resultados'. The source file 'ArrayRecursividadPreciosResultados.java' is open, displaying the following code:

```
package pkg12.arrayrecursividad.precios.resultados;

import java.util.Scanner;

public class ArrayRecursividadPreciosResultados {

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

        // Declaramos las variables
        double[] Precios = {52, 45, 56, 78, 41};
        double nuevoPrecio;
        int indice;

        // Solicitamos al usuario un valor e indice para modificar.
        System.out.print("Ingrese un el precio a almacenar: ");
        nuevoPrecio = Double.parseDouble(input.nextLine());

        System.out.print("Ingrese donde se va guardar (del 0 al 4): ");
        indice = Integer.parseInt(input.nextLine());

        while (indice < 0 || indice > 4) {
            System.out.print("ERROR. Ingrese un valor correcto de indice: ");
            indice = Integer.parseInt(input.nextLine());
        }

        // Mostramos los valores originales
        System.out.println("PRECIOS ORIGINALES");
        imprimeArray(Precios);

        // Modificamos el array con los valores agregados.
        Precios[indice] = nuevoPrecio;
        System.out.println("----");

        // Mostramos los valores los nuevos
        System.out.println("PRECIOS NUEVOS");
        imprimeArray(Precios);
    }

    public static void imprimeArray(double[] precios) {
        for (int i = 0; i < precios.length; i++) {
            System.out.println("Precio: $" + precios[i]);
        }
    }
}
```

The output window shows the execution results:

```
run:
Ingrese un el precio a almacenar: 800
Ingrese donde se va guardar (del 0 al 4): 3
PRECIOS ORIGINALES
Precio: $52.0
Precio: $45.0
Precio: $56.0
Precio: $78.0
Precio: $41.0
----
PRECIOS NUEVOS
Precio: $52.0
Precio: $45.0
Precio: $800.0
Precio: $56.0
Precio: $41.0
BUILD SUCCESSFUL (total time: 4 seconds)
```

(Continúa en la siguiente página)

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.

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 screenshot shows the NetBeans IDE with a Java project named '13.ArrayRecursividad.Precios.Recursivo'. The main file is 'ArrayRecursividadPreciosRecursivo.java'. The code is as follows:

```
package pkg13.arrayrecursividad.precios.recursivo;

import java.util.Scanner;

public class ArrayRecursividadPreciosRecursivo {

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

        // Declaramos las variables.
        double[] Precios = {52, 45, 56, 78, 41};
        double nuevoPrecio;
        int indice;

        // Solicitamos al usuario un valor e indice para modificar.
        System.out.print("Ingrese un el precio a almacenar: ");
        nuevoPrecio = Double.parseDouble(input.nextLine());

        System.out.print("Ingrese donde se va guardar (del 0 al 4): ");
        indice = Integer.parseInt(input.nextLine());

        while (indice < 0 || indice > 4) {
            System.out.print("ERROR. Ingrese un valor correcto de indice: ");
            indice = Integer.parseInt(input.nextLine());
        }

        // Mostramos los valores originales.
        System.out.println("PRECIOS ORIGINALES");
        imprimeArray(Precios, Precios.length);

        // Modificamos el array con los valores agregados.
        Precios[indice] = nuevoPrecio;
        System.out.println("----");

        // Mostramos los valores los nuevos.
        System.out.println("PRECIOS NUEVOS");
        imprimeArray(Precios, Precios.length);
    }
}
```

The output window shows the following execution results:

```
run:
Ingrese un el precio a almacenar: 700
ERROR. Ingrese un valor correcto de indice: 2
PRECIOS ORIGINALES
Precio: $41.0
Precio: $78.0
Precio: $56.0
Precio: $45.0
Precio: $52.0
---
PRECIOS NUEVOS
Precio: $41.0
Precio: $78.0
Precio: $700.0
Precio: $45.0
Precio: $52.0
Precio: $52.0
BUILD SUCCESSFUL (total time: 7 seconds)
```