



Universidad Tecnológica Nacional

## TP 2 – Programación Estructurada

🔗 [Enlace](#) al repositorio de GitHub

Programación II  
Belén Yarde Buller  
Comisión 3

## **Aclaraciones preliminares**

1. Con el fin de organizar el código de cada ejercicio de manera separada, y aplicando los conocimientos adquiridos respecto a funciones, cada ejercicio se ha resuelto con un método de la clase Funciones (presente en Funciones.java), que puede llamarse desde la función principal del programa, main(), hallada en el archivo Ejercicios.java.

2. Por razones de practicidad, teniendo en cuenta que en más de un ejercicio se le pide al usuario que ingrese un valor (int o double) por consola, se han creado dos funciones que permiten validar si se ha recibido un Integer o Double válido. De lo contrario, la función ejecuta un bucle hasta dar con el valor esperado. El código se detalla a continuación:

```
public static double returnValidDouble(String mensaje) {  
    double valor = 0;  
    boolean isValidDouble = false;  
  
    while (!isValidDouble) {  
        System.out.print(mensaje);  
        try {  
            valor = Double.parseDouble(scanner.nextLine());  
            isValidDouble = true;  
        } catch (NumberFormatException e) {  
            System.out.println("Error. Por favor ingrese un número válido.");  
        }  
    }  
  
    return valor;  
}  
  
static int returnValidInteger(String mensaje) {  
  
    int valor = 0;  
    boolean isValidInteger = false;  
  
    while (!isValidInteger) {  
        System.out.print(mensaje);  
        try {  
            valor = Integer.parseInt(scanner.nextLine());  
            isValidInteger = true;  
        } catch (NumberFormatException e) {  
            System.out.println("Error. Por favor ingrese un número entero.");  
        }  
    }  
    return valor;  
}
```

### Estructuras Condicionales:

#### **1. Verificación de Año Bisiesto.**

The screenshot shows the NetBeans IDE interface. The code editor displays a Java program named 'ejercicio1' which checks if a given year is a leap year. The output window shows the program's execution results.

```
51  /**
52   * Escribe un programa en Java que solicite al usuario un año y determine si
53   * es bisiesto. Un año es bisiesto si es divisible por 4, pero no por 100,
54   * salvo que sea divisible por 400.
55   *
56   */
57  public static void ejercicio1() {
58      int year = returnValidInteger("Ingrese un año: ");
59
60      if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0) {
61          System.out.println("El año " + year + " es bisiesto.");
62      } else {
63          System.out.println("El año " + year + " no es bisiesto.");
64      }
65  }
66
67  /-->
```

Ejecicios.Funciones >

Output x

netbeans - /Users/belenyardebuller/code/netbeans x TP2 (run) #2 x

run:  
Ingrese un año: 2024  
El año 2024 es bisiesto.  
BUILD SUCCESSFUL (total time: 7 seconds)

## 2. Determinar el Mayor de Tres Números.

```
67  /**
68   * Escribe un programa en Java que pida al usuario tres números enteros y
69   * determine cuál es el mayor.
70   *
71   */
72 public static void ejercicio2() {
73     int mayor = 0;
74
75     int a = returnValidInteger("Ingrese el primer número: ");
76     int b = returnValidInteger("Ingrese el segundo número: ");
77     int c = returnValidInteger("Ingrese el tercer número: ");
78
79     if (a > b && a > c) {
80         mayor = a;
81     } else if (b > a && b > c) {
82         mayor = b;
83     } else {
84         mayor = c;
85     }
86
87     System.out.println("El mayor es: " + mayor);
88
89 }
```

Ejercicios.Funciones > ejercicio2 > a >

Output x

```
netbeans - /Users/belenyardebuller/code/netbeans x TP2 (run) #2 x
run:
Ingrese el primer número: 8
Ingrese el segundo número: 12
Ingrese el tercer número: 5
El mayor es: 12
BUILD SUCCESSFUL (total time: 10 seconds)
```

### 3. Clasificación de Edad.

The screenshot shows the NetBeans IDE interface. The code editor displays a Java program named `ejercicio3`. The code prompts the user for their age and prints a classification based on their input. The output window shows the program's execution, including the input age and the resulting classification message.

```
91  */
92  * Escribe un programa en Java que solicite al usuario su edad y clasifique
93  * su etapa de vida según la siguiente tabla: Menor de 12 años: "Niño" Entre
94  * 12 y 17 años: "Adolescente" Entre 18 y 59 años: "Adulto" 60 años o más:
95  * "Adulto mayor"
96  */
97  public static void ejercicio3() {
98      int age = returnValidInteger("Ingrese su edad: ");
99
100     if (age < 12) {
101         System.out.println("Eres un Niño");
102     } else if (age <= 17) {
103         System.out.println("Eres un Adolescente");
104     } else if (age <= 59) {
105         System.out.println("Eres un Adulto");
106     } else {
107         System.out.println("Eres un Adulto mayor");
108     }
109 }
```

Ejecicios.Funciones > ejercicio2 > a >

Output x

netbeans - /Users/belenyardebuller/code/netbeans x TP2 (run) #2 x

run:  
Ingrese su edad: 25  
Eres un Adulto  
BUILD SUCCESSFUL (total time: 9 seconds)

#### 4. Calculadora de Descuento según categoría.

```
112     * Escribe un programa que solicite al usuario el precio de un producto y su
113     * categoría (A, B o C). Luego, aplique los siguientes descuentos: Categoría
114     * A: 10% de descuento Categoría B: 15% de descuento Categoría C: 20% de
115     * descuento El programa debe mostrar el precio original, el descuento
116     * aplicado y el precio final
117     *
118     */
119 public static void ejercicio4() {
120     String appliedDiscount = "";
121     double productPriceWithDiscount = 0;
122
123     System.out.print("Ingrese el precio del producto: ");
124     double productPrice = Double.parseDouble(scanner.nextLine());
125
126     System.out.print("Ingrese la categoría del producto (A, B o C): ");
127     String productCategory = scanner.nextLine();
128
129     switch (productCategory) {
130         case "A":
131             appliedDiscount = "10%";
132             productPriceWithDiscount = productPrice - (productPrice * 0.10);
133             break;
134
135         case "B":
136             appliedDiscount = "15%";
137             productPriceWithDiscount = productPrice - (productPrice * 0.15);
138             break;
139
140         case "C":
141             appliedDiscount = "20%";
142             productPriceWithDiscount = productPrice - (productPrice * 0.20);
143             break;
144
145         default:
146             System.out.println("Error. Categoría inválida.");
147             return;
148     }
149
150     System.out.println("Descuento aplicado: " + appliedDiscount);
151     System.out.println("Precio final: " + productPriceWithDiscount);
152
153 }
```

Output x

```
netbeans - /Users/belenyardebuller/code/netbeans x TP2 (run) #2 x
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: 10 seconds)
```

## Estructuras de Repetición

### 5. Suma de Números Pares (while).

The screenshot shows the NetBeans IDE interface. The top part displays a Java code editor with the following content:

```
155  */
156  * Escribe un programa que solicite números al usuario y sume solo los
157  * números pares. El ciclo debe continuar hasta que el usuario ingrese el
158  * número 0, momento en el que se debe mostrar la suma total de los pares
159  * ingresados.
160  */
161 public static void ejercicio5() {
162     int userInput = -1;
163     int sumaPares = 0;
164
165     while (userInput != 0) {
166         userInput = returnValidInteger("Ingrese un número (0 para terminar): ");
167         if (userInput != 0 && userInput % 2 == 0) {
168             sumaPares += userInput;
169         }
170     }
171
172     System.out.println("La suma de los números pares es: " + sumaPares);
173 }
```

The bottom part shows the 'Output' window with the following log:

```
run:
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
BUILD SUCCESSFUL (total time: 13 seconds)
```

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

The screenshot shows the NetBeans IDE interface. On the left is the code editor with Java code for Exercise 6. On the right is the Output window showing the program's execution and results.

**Code Editor Content (Exercise 6):**

```
175  /**
176   * Escribe un programa que pida al usuario ingresar 10 números enteros y
177   * cuente cuántos son positivos, negativos y cuántos son ceros.
178   */
179 public static void ejercicio6() {
180     Scanner scan = new Scanner(System.in);
181
182     int contadorPositivos = 0;
183     int contadorNegativos = 0;
184     int contadorCeros = 0;
185     int totalNumeros = 10;
186     int num;
187
188     for (int j = 0; j < totalNumeros; j++) {
189         System.out.print("Ingrese un número: ");
190         num = scan.nextInt();
191
192         if (num > 0) {
193             contadorPositivos++;
194         } else if (num < 0) {
195             contadorNegativos++;
196         } else {
197             contadorCeros++;
198         }
199     }
200
201     System.out.println("Positivos: " + contadorPositivos);
202     System.out.println("Negativos: " + contadorNegativos);
203     System.out.println("Ceros: " + contadorCeros);
204 }
```

**Output Window Content:**

```
run:
0 Ingrese un número: -5
1 Ingrese un número: 3
2 Ingrese un número: 0
3 Ingrese un número: -1
4 Ingrese un número: 6
5 Ingrese un número: 0
6 Ingrese un número: 9
7 Ingrese un número: -3
8 Ingrese un número: 4
9 Ingrese un número: -8
Positivos: 4
Negativos: 4
Ceros: 2
BUILD SUCCESSFUL (total time: 24 seconds)
```

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

The screenshot shows the NetBeans IDE interface. On the left is the code editor with the following Java code:

```
207  */
208  * Escribe un programa que solicite al usuario una nota entre 0 y 10. Si el
209  * usuario ingresa un número fuera de este rango, debe seguir pidiéndole la
210  * nota hasta que ingrese un valor válido.
211  */
212 public static void ejercicio7() {
213     int grade;
214
215     do {
216         grade = returnValidInteger("Ingrese una nota (0-10): ");
217         if (grade < 0 || grade > 10) {
218             System.out.println("Error: Nota inválida. Ingrese una nota entre 0 y 10");
219         }
220     } while (grade < 0 || grade > 10);
221
222     System.out.println("Nota guardada correctamente.");
223 }
```

Below the code editor is the 'Output' window, which displays the following run log:

```
run:
Ingresé una nota (0-10): 15
Error: Nota inválida. Ingresé una nota entre 0 y 10
Ingresé una nota (0-10): -2
Error: Nota inválida. Ingresé una nota entre 0 y 10
Ingresé una nota (0-10): 8
Nota guardada correctamente.

BUILD SUCCESSFUL (total time: 15 seconds)
```

## 8. Cálculo del Precio Final con impuesto y descuento.

The screenshot shows the NetBeans IDE interface with the code editor containing the following Java code:

```
/** 
 * Ejercicio 8: 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)
 *
 * @param precioBase
 * @param impuesto
 * @param descuento
 * @return
 */
public static double calcularPrecioFinal(double precioBase, double impuesto, double descuento) {
    double impuestoDecimal = impuesto / 100.0;
    double descuentoDecimal = descuento / 100.0;
    return (precioBase + (precioBase * impuestoDecimal) - (precioBase * descuentoDecimal));
}
```

```
16 public static void main(String[] args) {
17     // Acá se ejecuta cada función que resuelve los ejercicios:
18
19     // Funciones.ejercicio1();
20
21     // Funciones.ejercicio2();
22
23     // Funciones.ejercicio3();
24
25     // Funciones.ejercicio4();
26
27     // Funciones.ejercicio5();
28
29     // Funciones.ejercicio6();
30
31     // Funciones.ejercicio7();
32
33     // Ejercicio 8: Desde main(), solicita el precio base del producto, el porcentaje de
34     // impuesto y el porcentaje de descuento, llama al método y muestra el precio
35     // final.
36     double precioBase = Funciones.returnValidDouble("Ingrese el precio base del producto: ");
37     double impuesto = Funciones.returnValidDouble("Ingrese el impuesto en porcentaje (Ejemplo: 10 para 10%): ");
38     double descuento = Funciones.returnValidDouble("Ingrese el descuento en porcentaje (Ejemplo: 5 para 5%): ");
39     System.out.println("El precio final del producto es: " + Funciones.calcularPrecioFinal(precioBase, impuesto, descuento));
```

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

```
243 // Ejercicio 9: Composición de funciones para calcular costo de envío y total de compra.
244 /**
245 * 9a: calcularCostoEnvio(double peso, String zona): Calcula el costo de
246 * envío basado en la zona de envío (Nacional o Internacional) y el peso del
247 * paquete. Nacional: $5 por kg Internacional: $10 por kg
248 *
249 * @param peso
250 * @param zona
251 * @return
252 */
253 public static double calcularCostoEnvio(double peso, String zona) {
254     double costoPorKg = 0;
255
256     if (zona.equalsIgnoreCase("Nacional")) {
257         costoPorKg = 5;
258     } else if (zona.equalsIgnoreCase("Internacional")) {
259         costoPorKg = 10;
260     } else {
261         System.out.println("Zona inválida, se asigna costo 0.");
262     }
263
264     return peso * costoPorKg;
265 }
266
267 /**
268 * 9b: calcularTotalCompra(double precioProducto, double costoEnvio): Usa
269 * calcularCostoEnvio para sumar el costo del producto con el costo de
270 * envío.
271 *
272 * @param precioProducto
273 * @param CostoEnvio
274 * @return
275 */
276 public static double calcularTotalCompra(double precioProducto, double costoEnvio) {
277     return precioProducto + costoEnvio;
278 }
```

```

41 // Ejercicio 9: Desde main(), solicita el peso del paquete, la zona de envío y el precio
42 // del producto. Luego, muestra el total a pagar.
43 double precioProducto = Funciones.returnValidDouble("Ingrese el precio del producto: ");
44 double peso = Funciones.returnValidDouble("Ingrese el peso del paquete en kg: ");
45 System.out.print("Ingrese la zona de envío (Nacional/Internacional): ");
46 String zona = Funciones.scanner.nextLine();
47 double costoEnvio = Funciones.calcularCostoEnvio(peso, zona);
48 double totalCompra = Funciones.calcularTotalCompra(precioProducto, costoEnvio);
49 System.out.println("El costo de envío es: " + costoEnvio);
50 System.out.println("El total a pagar es: " + totalCompra);
51

```

## Output x

```

▶ netbeans - /Users/belenyardebuller/code/netbeans × TP2 (run) #2 ×
▶ run:
Ingresé el precio del producto: 50
Ingresé el peso del paquete en kg: 2
Ingresé la zona de envío (Nacional/Internacional): Nacional
El costo de envío es: 10.0
El total a pagar es: 60.0
BUILD SUCCESSFUL (total time: 14 seconds)

```

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

```

280 /**
281 * Ejercicio 10: Actualización de stock a partir de venta y recepción de
282 * productos. Crea un método actualizarStock(int stockActual, int
283 * cantidadVendida, int cantidadRecibida), que calcule el nuevo stock
284 * después de una venta y recepción de productos: NuevoStock = StockActual -
285 * CantidadVendida + CantidadRecibida
286 *
287 * @param stockActual
288 * @param cantidadVendida
289 * @param cantidadRecibida
290 * @return
291 */
292 public static int actualizarStock(int stockActual, int cantidadVendida, int cantidadRecibida) {
293     return stockActual - cantidadVendida + cantidadRecibida;
294 }

```

```

52 // Ejercicio 10: Desde main(), solicita al usuario el stock actual, la cantidad vendida y la
53 // cantidad recibida, y muestra el stock actualizado.
54 int stockActual = Funciones.returnValidInteger("Ingrese el stock actual del producto: ");
55 int cantidadVendida = Funciones.returnValidInteger("Ingrese la cantidad vendida: ");
56 int cantidadRecibida = Funciones.returnValidInteger("Ingrese la cantidad recibida: ");
57 int nuevoStock = Funciones.actualizarStock(stockActual, cantidadVendida, cantidadRecibida);
58 System.out.println("El nuevo stock del producto es: " + nuevoStock);
59

```

## Output x

```

▶ netbeans - /Users/belenyardebuller/code/netbeans × TP2 (run) #2 ×
▶ run:
Ingresé el stock actual del producto: 50
Ingresé la cantidad vendida: 20
Ingresé la cantidad recibida: 30
El nuevo stock del producto es: 60
BUILD SUCCESSFUL (total time: 14 seconds)

```

**11. Cálculo de descuento especial usando variable global.**

```
296 // Ejercicio 11
297 static final double DESCUENTO_ESPECIAL = 0.10;
298
299 /**
300 * 11. Cálculo de descuento especial usando variable global. Declara una
301 * variable global Ejemplo de entrada/salida: = 0.10. Luego, crea un método
302 * calcularDescuentoEspecial(double precio) que use la variable global para
303 * calcular el descuento especial del 10%.
304 *
305 * @param precio
306 */
307 public static void calcularDescuentoEspecial(double precio) {
308     double descuentoAplicado = precio * DESCUENTO_ESPECIAL;
309     double precioFinal = precio - descuentoAplicado;
310
311     System.out.println("El descuento especial aplicado es: " + descuentoAplicado);
312     System.out.println("El precio final con descuento es: " + precioFinal);
313 }
```

```
60 // Ejercicio 11:
61 int precio = Funciones.returnValidInteger("Ingrese el precio del producto: ");
62 Funciones.calcularDescuentoEspecial(precio);
63
```

Output x

```
▶ netbeans - /Users/belenyardebuller/code/netbeans × TP2 (run) #2 ×
▶ 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)
```

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

The screenshot shows the NetBeans IDE interface. The code editor window displays Java code for modifying an array of prices. The output window shows the original and modified price lists.

```
315  */
316  * 12. Modificación de un array de precios y visualización de resultados. Crea
317  * un programa que: a. Declare e inicialice un array con los precios de
318  * algunos productos. b. Muestre los valores originales de los precios. c.
319  * Modifique el precio de un producto específico. d. Muestre los valores
320  * modificados.
321 */
322 public static void ejercicio12() {
323     // a. Declarar e inicializar un array con los precios de algunos productos
324     double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};
325
326     // b. Mostrar los valores originales de los precios
327     System.out.println("Precios originales:");
328     for (double precio : precios) {
329         System.out.println("Precio: $" + precio);
330     }
331
332     // c. Modificar el precio de un producto específico
333     precios[2] = 129.99;
334
335     // d. Mostrar los valores modificados
336     System.out.println("Precios modificados:");
337     for (double precio : precios) {
338         System.out.println("Precio: $" + precio);
339     }
340 }
```

Output x

netbeans - /Users/belenyardebuller/code/netbeans x TP2 (run) #2 x

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.

The screenshot shows the NetBeans IDE interface. The top part displays a Java code editor with line numbers from 342 to 372. The code implements a recursive function to print array elements and modify specific values. The bottom part shows the 'Output' window with the run log, which includes the original array values and the modified array values after the third element is changed to 129.99.

```
342  /** 13. Impresión recursiva de arrays antes y después de modificar un elemento.
343  * Crea un programa que: a. Declare e inicialice un array con los precios de
344  * algunos productos. b. Use una función recursiva para mostrar los precios
345  * originales. c. Modifique el precio de un producto específico. d. Use otra
346  * función recursiva para mostrar los valores modificados.
347  *
348  */
349  public static void ejercicio13() {
350      // a. Declarar e inicializar el array
351      double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};
352
353      System.out.println("Precios originales:");
354      mostrarArrayRecursivo(precios, 0);
355
356      // c. Modificar un valor específico
357      precios[2] = 129.99;
358
359      System.out.println("\nPrecios modificados:");
360      mostrarArrayRecursivo(precios, 0);
361  }
362
363  // b/d. Función recursiva para mostrar los precios
364  public static void mostrarArrayRecursivo(double[] array, int indice) {
365      if (indice < array.length) {
366          System.out.println("Precio: $" + array[indice]);
367          mostrarArrayRecursivo(array, indice + 1); // llamada recursiva al siguiente índice
368      }
369      // caso base: cuando indice == array.length, se detiene la recursión
370  }
371
372 }
```

Output x

▶ netbeans - /Users/belenyardebuller/code/netbeans x TP2 (run) #2 x

▶ 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)