

## PROGRAMACIÓN II

### Trabajo Práctico 2: Programación Estructurada

#### Caso Práctico

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

#### Estructuras Condicionales:

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

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

Ejemplo de entrada/salida:

Ingrese un año: 2024

El año 2024 es bisiesto.

Ingrese un año: 1900

El año 1900 no es bisiesto.

```
--- exec:3.1.0:exec (default-cli) @ proyecto_netbeans ---
EJ1
Ingrese un año:
2024
El año 2024 es bisiesto.

--- exec:3.1.0:exec (default-cli) @ proyecto_netbeans ---
EJ1
Ingrese un año:
1900
El año 1900 no es 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

Boyer Rodríguez Matías Ruben

DNI 43.901.799

matiasboyer7@gmail.com

```
--- exec:3.1.0:exec (default-cli) @ proyecto_netbeans ---
EJ2
Ingrese el primer número:
8
Ingrese el segundo número:
12
Ingrese el tercer número:
5
El mayor es: 12
```

### 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.

```
EJ3
Ingrese su edad:
25
Usted es un ADULTO.

EJ3
Ingrese su edad:
10
Usted es un NIÑO.
```

### 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

Boyer Rodríguez Matías Ruben

DNI 43.901.799

matiasboyer7@gmail.com

```
EJ4
Ingrese el precio del producto: 1000
Ingrese la categoría del producto (A/B/C): B
Precio original: 1000.0
Descuento aplicado: 15
Precio final: 850.0
```

## 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:

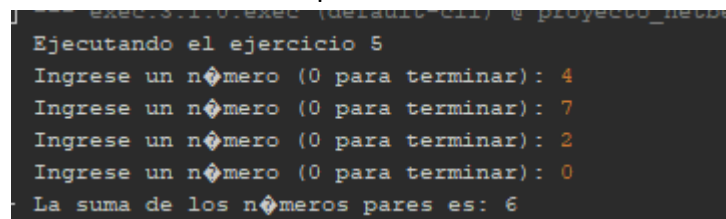
Ingrese un número (0 para terminar): 4

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



```

Ejecutando el ejercicio 5
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

```

### 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

```

Ingrese 10 números
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

```

## 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.

```

Ejecutando el ejercicio 7
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.

```

### 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.

Ejemplo de entrada/salida:

Ingrese el precio base del producto: 100

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

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

El precio final del producto es: 105.0

```
public static double calcularPrecioFinal(double preciobase, double impuesto, double descuento) {  
    return preciobase + (preciobase * (impuesto/100)) - (preciobase * (descuento/100));  
}
```

```
Ejecutando el ejercicio 8  
Ingrese el precio base del producto: 100  
Ingrese el impuesto en porcentaje (Ej 10 para 10%): 10  
Ingrese el descuento en porcentaje (Ej 10 para 10%): 5  
El precio final del producto es: 105.0
```

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

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

Nacional: \$5 por kg

Internacional: \$10 por kg

```
public static double calcularCostoEnvio(double peso, String zona) {  
    if (zona.equals("Nacional")) return peso * 5.0;  
    else if (zona.equals("Internacional")) return peso * 10.0;  
    else {  
        System.out.println("No existe la zona. Calculando costo de envío a $10 x kg");  
        return peso * 10.0;  
    }  
}
```

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

```
public static double calcularTotalCompra(double precioProducto, double costoEnvio) {  
    return precioProducto + costoEnvio;  
}
```

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

```
Ejecutando el ejercicio 9  
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
```

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:

$$\text{NuevoStock} = \text{StockActual} - \text{CantidadVendida} + \text{CantidadRecibida}$$

$$\text{NuevoStock} = \text{CantidadVendida} + \text{CantidadRecibida}$$

```
1 public static int actualizarStock(int stockActual, int cantidadVendida, int cantidadRecibida) {  
    return stockActual - 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

```
Ejecutando el ejercicio 10  
Ingrese el stock actual del producto: 50  
Ingrese la cantidad vendida: 20  
Ingrese la cantidad recibida: 30  
El nuevo stock del producto es: 60
```

11. Cálculo de descuento especial usando variable global.

Declara una variable global `descuentoEspecial`: = 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.

```
static final double descuentoEspecial = 0.10;  
public static double calcularDescuentoEspecial(double precio) {  
    double descuentoAplicado = precio * descuentoEspecial;  
    return descuentoAplicado;  
}
```

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

```
Ejecutando el ejercicio 11  
Ingrese el precio del producto: 200  
El descuento especial aplicado es: 20.0  
El precio final con descuento es: 180.0
```



### 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

```
Ejecutando el ejercicio 12
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
```

Boyer Rodríguez Matías Ruben

DNI 43.901.799

matiasboyer7@gmail.com

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

```
public static void ImprimirPreciosRecursivo(int idx, double[] arr)
{
    if(idx >= arr.length) return;
    System.out.println("Precio: $" + arr[idx]);
    ImprimirPreciosRecursivo(idx + 1, arr);
}
```

```
Ejecutando el ejercicio 13
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
```