PROGRAMACIÓN II Trabajo Práctico 2:

Programación Estructurada

MAURO ZAVATTI

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.

```
package tp.pkg2.prog.estructurada.zavatti;


import java.util.Scanner;


public class Ejercicio1 {


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


     System.out.print("Ingrese un año: ");
     int anio = input.nextInt();


     if ((anio % 4 == 0 && anio % 100 != 0) || (anio % 400 == 0)) {
        System.out.println("El año " + anio + " es bisiesto.");
```

```
    } else {
        System.out.println("El año " + anio + " 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

```
package tp.pkg2.prog.estructurada.zavatti;

import java.util.Scanner;


public class Ejercicio2 {
  public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);


    System.out.print("Ingrese el primer número: ");
      int num1 = scan.nextInt();

    System.out.print("Ingrese el segundo número: ");
```

```
        int num2 = scan.nextInt();

    System.out.print("Ingrese el tercer número: ");
        int num3 = scan.nextInt();

    int mayor;

    if (num1 >= num2 && num1 >= num3) {
        mayor = num1;
    } else if (num2 >= num1 && num2 >= num3) {
        mayor = num2;
    } else {
        mayor = num3;
    }

    System.out.println("El mayor es: " + mayor);

  }

}
```

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.

```
package tp.pkg2.prog.estructurada.zavatti;
import java.util.Scanner;
public class Ejercicio3 {
    public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    int edad;
    String etapaVida = "";

    // Le pedimos al usuario que ingrese su edad.
    System.out.println("Ingrese su edad:");
    edad = Integer.parseInt(input.nextLine());

    if(edad < 12) {
       etapaVida = "Niño";
    } else if (edad >= 12 && edad <= 17) {
       etapaVida = "Adolescente";
    } else if (edad >= 18 && edad <= 59) {
       etapaVida = "Adulto";
    } else if (edad >= 60 ) {
       etapaVida = "Adulto mayor";
    }
```

```
        System.out.println("Eres un " + etapaVida);
    }
}
```

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 tp.pkg2.prog.estructurada.zavatti;

import java.util.Scanner;

public class Ejercicio4 {

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

```
System.out.print("Ingrese el precio del producto: ");
if (!input.hasNextDouble()) {
   System.out.println("Precio inválido.");
   input.close();
   return;
}
double precio = input.nextDouble();

if (precio < 0) {
   System.out.println("El precio no puede ser negativo.");
   input.close();
   return;
}

System.out.print("Ingrese la categoría del producto (A, B o C): ");
String categoria = input.next().trim().toUpperCase();

double descuento;
switch (categoria) {
   case "A":
      descuento = 0.10;
      break;
   case "B":
      descuento = 0.15;
      break;
   case "C":
      descuento = 0.20;
      break;
```

```
            default:
                System.out.println("Categoría no válida. Use A, B o C.");
                input.close();
                return;
        }

        double montoDescuento = precio * descuento;
        double precioFinal = precio - montoDescuento;

        System.out.printf("Precio original: %.2f%n", precio);
        System.out.printf("Descuento aplicado: %d%%%n", (int) (descuento * 100));
        System.out.printf("Precio final: %.2f%n", precioFinal);

        input.close();
    }
}
```

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

```
package tp.pkg2.prog.estructurada.zavatti;

import java.util.Scanner;

public class Ejercicio5 {

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

      int numero;
      int sumaPares = 0;

      System.out.print("Ingrese un número (0 para terminar): ");
      numero = input.nextInt();

      // mientras no se ingrese 0 sigue pidiendo números
      while (numero != 0) {
         if (numero % 2 == 0) {  // si es par
            sumaPares += numero;
         }

         System.out.print("Ingrese un número (0 para terminar): ");
         numero = input.nextInt();
      }

      System.out.println("La suma de los números pares es: " + sumaPares);

      input.close();
```

```
    }
}
```

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.

```
package tp.pkg2.prog.estructurada.zavatti;

import java.util.Scanner;

public class Ejercicio6 {

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

        int positivos = 0;
        int negativos = 0;
        int ceros = 0;

        for (int i = 1; i <= 10; i++) {
            System.out.print("Ingrese el número " + i + ": ");
            int numero = input.nextInt();

            if (numero > 0) {
                positivos++;
            } else if (numero < 0) {
                negativos++;
            } else {
```

```
            ceros++;
        }
    }

    System.out.println("Resultados:");
    System.out.println("Positivos: " + positivos);
    System.out.println("Negativos: " + negativos);
    System.out.println("Ceros: " + ceros);



  }
}
```

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.

```
package tp.pkg2.prog.estructurada.zavatti;

import java.util.Scanner;

public class Ejercicio7 {

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

    do {
      System.out.print("Ingrese una nota (0-10): ");
      nota = input.nextInt();
```

```
        if (nota < 0 || nota > 10) {
            System.out.println("Error: Nota inválida. Ingrese una nota entre 0 y 10.");
        }

    } while (nota < 0 || nota > 10);

    System.out.println("Nota guardada correctamente: " + nota);

  }
}
```

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

```
package tp.pkg2.prog.estructurada.zavatti;


import java.util.Scanner;
```

```
public class Ejercicio8 {

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

        System.out.print("Ingrese el precio base del producto: ");
        double precioBase = input.nextDouble();

        System.out.print("Ingrese el impuesto en porcentaje (Ejemplo: 10 para 10%): ");
        double impuesto = input.nextDouble();

        System.out.print("Ingrese el descuento en porcentaje (Ejemplo: 5 para 5%): ");
        double descuento = input.nextDouble();

        double precioFinal = calcularPrecioFinal(precioBase, impuesto, descuento);

        System.out.println("El precio final del producto es: " + precioFinal);
    }

    // Método que calcula el precio final con impuesto y descuento
    public static double calcularPrecioFinal(double precioBase, double impuesto, double descuento) {
        double impuestoAplicado = precioBase * (impuesto / 100);
        double descuentoAplicado = precioBase * (descuento / 100);
        return precioBase + impuestoAplicado - descuentoAplicado;
    }
}
```

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.

```
package tp.pkg2.prog.estructurada.zavatti;


import java.util.Scanner;


public class Ejercicio9 {


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


    System.out.print("Ingrese el precio del producto: ");
    double precioProducto = input.nextDouble();


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


    System.out.print("Ingrese la zona de envío (Nacional/Internacional): ");
    String zona = input.next().toLowerCase(); // lo paso a minúscula


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

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

    // Método a: calcular costo de envío
    public static double calcularCostoEnvio(double peso, String zona) {
        double costoEnvio = 0;
        if(zona.equals("nacional")){   //aca tambien se podria usar un switch en lugar de un if / else
            costoEnvio = peso * 5;
        } else if (zona.equals("internacional")){
            costoEnvio = peso * 10;
        }

        return costoEnvio;
    }

    // Método b: calcular total de compra
    public static double calcularTotalCompra(double precioProducto, double costoEnvio) {
        return precioProducto + costoEnvio;
    }
}
```

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.

```
package tp.pkg2.prog.estructurada.zavatti;


import java.util.Scanner;


public class Ejercicio10 {


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


      // Pedir datos al usuario
      System.out.print("Ingrese el stock actual del producto: ");
      int stockActual = input.nextInt();


      System.out.print("Ingrese la cantidad vendida: ");
      int cantidadVendida = input.nextInt();


      System.out.print("Ingrese la cantidad recibida: ");
      int cantidadRecibida = input.nextInt();


      // Calcular nuevo stock usando el método
      int nuevoStock = actualizarStock(stockActual, cantidadVendida,
cantidadRecibida);


      // Mostrar resultado
      System.out.println("El nuevo stock del producto es: " + nuevoStock);
   }
```

```
  // Método que actualiza el stock
  public static int actualizarStock(int stockActual, int cantidadVendida, int
cantidadRecibida) {
      return stockActual - cantidadVendida + cantidadRecibida;
  }
}
```

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.

```
package tp.pkg2.prog.estructurada.zavatti;

import java.util.Scanner;

public class Ejercicio11 {

  // Variable global (estática)
  static double DESCUENTO_ESPECIAL = 0.10; // 10%

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

      System.out.print("Ingrese el precio del producto: ");
      double precio = input.nextDouble();
```

```
        // Llamar al método
        calcularDescuentoEspecial(precio);
    }

    // Método que calcula el descuento usando la variable global
    public static void calcularDescuentoEspecial(double precio) {
        // Variable local
        double descuentoAplicado = precio * DESCUENTO_ESPECIAL;

        double precioFinal = precio - descuentoAplicado;

        System.out.println("El descuento especial aplicado es: " +
descuentoAplicado);
        System.out.println("El precio final con descuento es: " + precioFinal);
    }
}
```

Arrays y Recursividad:

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

Crea un programa que:

a. Declare e inicialice un array con los precios de algunos productos.

b. Muestre los valores originales de los precios.

c. Modifique el precio de un producto específico.

d. Muestre los valores modificados.

```
package tp.pkg2.prog.estructurada.zavatti;

public class Ejercicio12 {

    public static void main(String[] args) {
```

```
        // a. Declarar e inicializar un array con precios
        double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};

        // b. Mostrar los valores originales
        System.out.println("Precios originales:");
        for (double precio : precios) {
            System.out.println("Precio: $" + precio);
        }

        // c. Modificar el precio de un producto específico
        // Ejemplo: cambiar el tercer producto (índice 2)
        precios[2] = 129.99;

        // d. Mostrar los valores modificados
        System.out.println("Precios modificados:");
        for (double precio : precios) {
            System.out.println("Precio: $" + precio);
        }
    }
}
```

13. Impresión recursiva de arrays antes y después de modificar un elemento.

Crea un programa que:

a. Declare e inicialice un array con los precios de algunos productos.

b. Use una función recursiva para mostrar los precios originales.

c. Modifique el precio de un producto específico.

d. Use otra función recursiva para mostrar los valores modificados.

```
package tp.pkg2.prog.estructurada.zavatti;

public class Ejercicio13 {

    public static void main(String[] args) {
        // a. Declarar e inicializar un array con precios
        double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};

        // b. Mostrar precios originales usando recursión
        System.out.println("Precios originales:");
        imprimirArrayRecursivo(precios, 0);

        // c. Modificar un precio específico (ej: el tercer elemento -> índice 2)
        precios[2] = 129.99;

        // d. Mostrar precios modificados usando recursión
        System.out.println("Precios modificados:");
        imprimirArrayRecursivo(precios, 0);
    }

    // Función recursiva para mostrar precios
    public static void imprimirArrayRecursivo(double[] array, int indice) {
        // Caso base: si llegamos al final, terminamos
        if (indice >= array.length) {
            return;
        }
        // Mostrar el valor en la posición actual
        System.out.println("Precio: $" + array[indice]);
```

```
        // Llamada recursiva al siguiente índice
        imprimirArrayRecursivo(array, indice + 1);
    }
}
```