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) {
      // Declaramos input para leer los datos
      Scanner input = new Scanner(System.in);
      int anio;


      // Le pedimos el valor al usuario
      System.out.println("Ingrese un año para verficar si es bisiesto");
```

```
        anio = Integer.parseInt(input.nextLine());

        // Verificamos en el condicional para anio bisiesto.
        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) {
    // Declaramos input para leer los datos
    Scanner input = new Scanner(System.in);
```

```
        // Declaramos variables
        int numUno, numDos, numTres, mayor;

        // Le pedimos los numeros al usuario
        System.out.println("Ingrese 3 numeros");
        System.out.println("Primer numero: ");
        numUno = Integer.parseInt(input.nextLine());
        System.out.println("Segundo numero: ");
        numDos = Integer.parseInt(input.nextLine());
        System.out.println("Tercer numero: ");
        numTres = Integer.parseInt(input.nextLine());

        mayor = numUno; // Iniciliamos mayor igualandolo a numUno

        if (numDos > mayor) { // Vericamos si numDos es mayor se guarda su valor
           mayor = numDos;
        }
        if (numTres > mayor) { // En el caso que num3 sea mayor se guarda su valor
           mayor = numTres;
        }

        // Mostramos al mayor
        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) {

    // Declaramos input para leer los datos

    Scanner input = new Scanner(System.in);


    // Declaramos las varibales donde vamos a guardar el precio y la categoria(A, B o C)

    int precio;

    double precioConDescuento;

    char categoria;


    // Pedimos el precio al usuario

    System.out.println("Ingrese el precio del producto");

    precio = Integer.parseInt(input.nextLine());


    // Pedimos la categoria al usuario

    System.out.println("Ingrese la categoria del producto (A, B o C)");

    categoria = input.nextLine().charAt(0);


    // Utilizamos un switch para manejar las condiciones segun la categoria ingresada

    switch (categoria) {

      case 'A':

      case 'a':

        precioConDescuento = precio -(precio * 0.10);

        System.out.print("Descuento aplicado 10%");

        System.out.println("");
```

```
            System.out.print("Precio final: $" + precioConDescuento);
            break;
        case 'B':
        case 'b':
            precioConDescuento = precio -(precio * 0.15);
            System.out.print("Descuento aplicado 15%");
            System.out.println("");
            System.out.print("Precio final: $" + precioConDescuento);
            break;
        case 'C':
        case 'c':
            precioConDescuento = precio -(precio * 0.20);
            System.out.print("Descuento aplicado 20%");
            System.out.println("");
            System.out.print("Precio final: $" + precioConDescuento);
            break;
        default:
            System.out.println("No ingreso una categoria valida");
            break;
    }
  }
}
```

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);


     // Declaramos num e inicilizamos en 1 para entrar al menos una vez al ciclo
     int num = 1;


     // Declaramos sumaPares e inicilizamos en 0
     int sumaPares = 0;


     while(num != 0){
       System.out.println("Ingrese un numero(0 para terminar)");
```

```
        num = Integer.parseInt(input.nextLine());


        // Si el resto de la division entre el numero y 2 nos da 0 entonces el numero es par
        if(num % 2 == 0){
           sumaPares = sumaPares + num; // Acumulamos la suma de numeros pares
        }
     }

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


  }
}
```

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);


     // Declaramos e iniciliamos contadores en 0, y declaramos num para alojar el numero del usuario
     int contadorPositivos = 0, contadorNegativos = 0, contadorCeros = 0, num;
```

```
        for (int i = 1; i <= 10; i++) {
            System.out.print("Ingrese el numero " + i + ": ");
            num = Integer.parseInt(input.nextLine());

            // Verificamos si es igual a 0, menor a 0 o mayor a 0
            if(num > 0) {
                contadorPositivos++;
            } else if (num < 0) {
                contadorNegativos++;
            } else if (num == 0){
                contadorCeros++;
            }
        }

        System.out.println("Resultados");
        System.out.print("Positivos: " + contadorPositivos + "\n");
        System.out.print("Negativos: " + contadorNegativos + "\n");
        System.out.println("Ceros: " + contadorCeros);
    }
}
```

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.println("Ingrese una nota(0-10)");
            nota = Integer.parseInt(input.nextLine());


            // En el caso que la nota sea menor que 0 o mayor a 10 lanza un mensaje de
error sino lanza un mensaje de éxito
            if(nota < 0 || nota > 10) {
                System.out.println("ERROR: nota invalida. Ingrese una nota entre 0 y 10 ");
            } else{
                System.out.println("Nota guardada correctamente");
            }
        }while(nota < 0 || nota > 10);
    }
}
```

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);

      // Declaramos la varibales para guardar los datos ingresados por el usuario
      double precio, impuesto, descuento;


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

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



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


         // Alojamos el retorno del metodo en una variable
        double precioFinal = calcularPrecioFinal(precio, impuesto, descuento);


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



    /**
     * Calcula el precio final de un producto aplicando un impuesto y un descuento.
     *
     * @param precioBase El precio base del producto (sin impuesto ni descuento).
     * @param impuesto El porcentaje de impuesto a aplicar (por ejemplo, 21 para 21%).
     * @param descuento El porcentaje de descuento a aplicar (por ejemplo, 15 para 15%).
     * @return El precio final del producto con el impuesto sumado y el descuento restado.
     */
```

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

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);

      // Declaramos variables para guardar los datos que pedimos al usuario
      double peso, precioProducto;
      String zona;

      System.out.println("Ingrese el precio del producto");
      precioProducto = input.nextDouble();
```

```
        System.out.println("Ingrese el peso del paquete");
        peso = input.nextDouble();
        input.nextLine(); // Consumimos el Enter que quedó pendiente

        System.out.println("Ingrese la zona de envio(Nacional/Internacional)");
        zona = input.nextLine();

        // Llamamos al metodo para calcular costo de envio y guardamos el retorno en una variable
        double costoEnvio = calcularCostoEnvio(peso, zona);

        // Llamamos al metodo para calcular el precio total de la compra y guardamos el retorno en una variable
        double precioFinal = calcularTotalCompra(precioProducto, costoEnvio);

        // Mostramos los resultados
        System.out.println("El costo de envio es: " + costoEnvio);
        System.out.println("El total a pagar es: " + precioFinal);
    }

    /**
     * Calcula el costo de envío en función del peso y la zona de destino.
     *
     * @param peso Peso del paquete en kilogramos.
     * @param zona Zona de envío: puede ser "Nacional" o "Internacional" (no
     * distingue mayúsculas).
     * @return El costo de envío calculado: $5 por kilo para envíos nacionales,
     * $10 por kilo para internacionales.
```

```
    */
    public static double calcularCostoEnvio(double peso, String zona){
        double costoEnvio = 0;
        if(zona.equalsIgnoreCase("Nacional")){
            costoEnvio = peso * 5;
        } else if (zona.equalsIgnoreCase("Internacional")){
            costoEnvio = peso * 10;
        }

        return costoEnvio;
    }

    /**
     * Calcula el total de la compra sumando el precio del producto y el costo
     * de envío.
     *
     * @param precioProducto Precio base del producto sin costos adicionales.
     * @param costoEnvio Costo de envío previamente calculado.
     * @return El monto total a pagar por la 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) {
      // Declaramos input para leer los datos
      Scanner input = new Scanner(System.in);

      // Declaramos las variables para alojar los datos ingresados por el usuario
      int stockActual, cantidadVendida, cantidadRecibida;

      System.out.println("Ingrese el stock actual del producto");
      stockActual = Integer.parseInt(input.nextLine());

      System.out.println("Ingrese la cantidad vendida");
      cantidadVendida = Integer.parseInt(input.nextLine());

      System.out.println("Ingrese la cantidad recibida");
      cantidadRecibida = Integer.parseInt(input.nextLine());
```

```
    // Llamamos al metodo actualizarStock, y alojamos el valor de retorno en una variable
    int nuevoStock = actualizarStock(stockActual, cantidadVendida, cantidadRecibida);


    System.out.println("El nuevo stock del producto es: " + nuevoStock);
  }
  /**
   * Actualiza el stock disponible de un producto según la cantidad vendida y
   * la cantidad recibida.
   *
   * @param stockActual Cantidad actual en stock antes de la operación.
   * @param cantidadVendida Cantidad de unidades vendidas (se restan del
   * stock).
   * @param cantidadRecibida Cantidad de unidades recibidas o repuestas (se
   * suman al stock).
   * @return El nuevo stock disponible después de aplicar ventas y
   * reposiciones.
   */
  public static int actualizarStock(int stockActual, int cantidadVendida, int cantidadRecibida){
    int nuevoStock = (stockActual - cantidadVendida) + cantidadRecibida;
    return nuevoStock;
  }
}
```

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 {

    // Declaramos la variable global
  public static final double DESCUENTO_ESPECIAL = 0.10;


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


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


    calcularDescuentoEspecial(precio);
  }


  /**
   * Calcula y muestra el descuento especial aplicado a un producto, junto con
   * el precio final.
   *
   * Usa la constante global DESCUENTO_ESPECIAL (10%) para calcular el
   * descuento. Imprime por pantalla: - El monto del descuento aplicado. - El
   * precio final del producto después de aplicar el descuento.
   *
```

```
     * @param precio El precio original del producto antes de aplicar el
     * descuento.
     */
    public static void calcularDescuentoEspecial(double precio){
        double descuentoAplicado = precio * DESCUENTO_ESPECIAL;
        double precioFinal = precio - descuentoAplicado;

        System.out.println("El descuento aplicado es de: " + descuentoAplicado);
        System.out.println("El precio final del producto 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) {
        // Declaramos el array de precios y le asignamos valores a cada posicion
        double[] precioProducto = {199.99, 299.5, 149.75, 399.0, 89.99};

        // Mostramos los precios originales recorriendo el array con un for-each
```

```
        System.out.println("Precios originales: ");
        for(double precio : precioProducto){
            System.out.println("Precio: " + precio);
        }

        //  Modificamos el precio de la posicion 4
        precioProducto[4] = 129.99;

        // Mostramos los precios modificados recorriendo el array con un for-each
        System.out.println("Precios modificados: ");
        for(double precio : precioProducto){
            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) {
    // Declaramos e inicializamos el array
    double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};
```

```
    // Mostramos los precios originales
    System.out.println("Precios originales:");
    mostrarPreciosRecursivo(precios, 0);

    // Modificamos el precio de la posicion 2
    precios[2] = 129.99;

    // Mostramos los precios modificados
    System.out.println("Precios modificados:");
    mostrarPreciosRecursivo(precios, 0);
}

/**
 * Muestra los precios de un arreglo de precios de manera recursiva.
 *
 * Esta función imprime los precios contenidos en el arreglo `precios`, uno
 * por uno, en la consola. La recursividad se utiliza para iterar sobre cada
 * elemento del arreglo.
 *
 * @param precios Un arreglo de tipo `double` que contiene los precios de
 * los productos.
 * @param indice Un índice entero que indica la posición actual en el
 * arreglo. Debe ser 0 cuando se llama a la función por primera vez.
 */
public static void mostrarPreciosRecursivo(double[] precios, int indice) {
    // Si el índice es menor que la longitud del arreglo, continúa el proceso.
    if (indice < precios.length) {
```

```
        // Imprime el precio en la posición actual del arreglo.
        System.out.println("Precio: $" + precios[indice]);
        // Llama recursivamente a la función para imprimir el siguiente precio.
        mostrarPreciosRecursivo(precios, indice + 1);
    }
  }
}
```