

Práctico 2: Programación Estructurada

Objetivo:

Desarrollar habilidades en programación estructurada en Java, abordando desde conceptos básicos como operadores y estructuras de control hasta temas avanzados como funciones, recursividad y estructuras de datos. Se busca fortalecer la capacidad de análisis y solución de problemas mediante un enfoque práctico,

Resultados de aprendizaje:

1. **Utilizar operadores y estructuras condicionales para la toma de decisiones:** El estudiante podrá aplicar operadores aritméticos, lógicos y relacionales en conjunto con estructuras condicionales para implementar lógica de control en programas, asegurando claridad y eficiencia en la toma de decisiones.
2. **Implementar estructuras de repetición para el procesamiento de datos:** El estudiante será capaz de diseñar y desarrollar programas que utilicen ciclos de iteración eficientes, aplicando estructuras como for, while y do-while para automatizar tareas y manejar grandes volúmenes de datos.
3. **Desarrollar programas modulares mediante funciones:** El estudiante podrá crear y organizar código en funciones reutilizables, aplicando principios de modularidad para mejorar la mantenibilidad, escalabilidad y legibilidad de sus programas.
4. **Aplicar recursividad y estructuras de datos básicas para la resolución de problemas:** El estudiante será capaz de diseñar soluciones algorítmicas eficientes utilizando recursividad y estructuras de datos como listas, pilas y colas, comprendiendo su impacto en el rendimiento y optimización del código.

Ejercicios:

Ejercicio de estructuras de programación condicional:

Ejercicio 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 también 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.

Ejercicio 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

Ejercicio 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.

Ejercicio 4: Calculadora de Descuento

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

Ejercicios de Estructuras de Repetición

Ejercicio 5: Suma de Números Pares (Ciclo 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

Ejercicio 6: Contador de Números Positivos y Negativos (Ciclo 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

Ejercicio 7: Validación de Entrada (Ciclo 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.

Ejercicios de Funciones:

Aquí tienes los ejercicios aplicados a la **gestión de productos para e-commerce**.

Ejercicio 8: Cálculo del Precio Final de un Producto (Funciones en Java)

Crea un método `calcularPrecioFinal(double precioBase, double impuesto, double descuento)` que calcule el precio final de un producto en un e-commerce. La fórmula es:

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

`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

Ejercicio 9: Composición de Funciones - Cálculo de Envío

Crea los siguientes métodos:

- `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
- `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.

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

Ejercicio 10: Función con Varios Parámetros - Gestión de Stock

Crea un método `actualizarStock(int stockActual, int cantidadVendida, int cantidadRecibida)`, que calcule el nuevo stock después de una venta y recepción de productos:

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

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

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

Ejercicio 11: Uso de Variables Locales y Globales - Cálculo de Descuento Especial

Declara una **variable global** `DESCUENTO_ESPECIAL = 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.

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

Ejercicio 12: Lista de Precios de Productos con Modificación

Descripción

Crea un programa que:

1. **Declare e inicialice un array** con los precios de algunos productos.
2. **Muestre los valores originales** de los precios.
3. **Modifique el precio de un producto específico.**

4. 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

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.

Ejercicio 13:

Ejercicio Simple: Lista de Precios con Recursividad

Descripción

Crea un programa que:

1. Declare e inicialice un array con los precios de algunos productos.
2. Use una función recursiva para mostrar los precios originales.
3. Modifique el precio de un producto específico.
4. 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

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.