**Taller de repaso de Java**

**Tipos de datos y Clases Wrapper en Java:**

**Tipos de datos:**

**Primitivos:**

Con este tipo de datos es posible acceder a ellos directamente, algunos de estos son:

Numéricos Enteros: byte, short, int, long.

Numéricos de Punto: Flotante: float, double.

Booleanos: boolean

Caracteres: char.

**Objetos:**

a diferencia de los tipos de datos primitivos que contienen valores simples, los objetos pueden contener múltiples atributos y métodos, y permiten la representación de datos y comportamientos complejos.

**Ejercicio:**

Crea una clase llamada TiposDeDatos y dentro de ella declara una variable para cada tipo de dato primitivo en Java.

utiliza System.out.println() para imprimir el valor y el nombre del tipo de dato en la consola.

class TiposDeDatos {

byte unByte = 127;

System.out.println(“Byte = ” unByte)

short unShort = 32000;

System.out.println(“Short = ” unShort)

int unInt = 2147483647;

System.out.println(“int = ” unInt)

long unLong = 9223372036854775807L;

System.out.println(“Long = ”unLong)

float unFloat = 3.14f;

System.out.println(“Float = ”unFloat)

double unDouble = 3.14159265359;

System.out.println(“Double = ”unDouble)

boolean unBoolean = true;

System.out.println(“Boolean = ”unBoolean)

char unChar = 'A';

System.out.println(“Char = ”unChar)

}

**Clases Wrapper:**

Las **clases Wrapper** en Java son clases que encapsulan los tipos de datos primitivos en objetos. Son útiles cuando necesitamos tratar un valor primitivo como un objeto, por ejemplo, al utilizar estructuras de datos que solo funcionan con objetos (List, Set, etc.).

Las clases Wrapper correspondientes a los tipos primitivos son:

* Byte, Short, Integer, Long (para números enteros)
* Float, Double (para números de punto flotante)
* Character (para caracteres)
* Boolean (para valores booleanos)

Ejercicio clases Wrapper:

 Declara variables de tipo primitivo y luego crea objetos utilizando las clases Wrapper correspondientes.

 Usa los métodos proporcionados por las clases Wrapper para convertir estos valores a otros tipos de datos.

Clases, métodos, atributos e instancias en java

¿Qué son las Clases, atributos y métodos en Java?

En Java, una clase es una plantilla o plano a partir del cual se pueden crear objetos. Las clases definen los atributos y los comportamientos (métodos) que los objetos creados a partir de esa clase tendrán.

Los atributos son las variables que definen el estado de un objeto. Pueden ser de diferentes tipos (primitivos o de referencia). Los atributos se declaran dentro de una clase y se acceden desde sus métodos.

Los métodos son bloques de código que realizan acciones específicas. Pueden recibir parámetros y retornar valores. Los métodos definen el comportamiento de un objeto y permiten interactuar con sus atributos.

En Java, una instancia es un objeto que se crea a partir de una clase

Ejercicio 1: Creación de una Clase Simple con atributos, métodos y creación de instancia de esta.

Imaginemos que estamos creando un sistema para gestionar Personas. Cada persona tiene un nombre y una edad. Definamos una clase Persona que tendrá estos atributos.

Paso a Paso:

Define la clase Persona con dos atributos: nombre y edad.

Crea un constructor para inicializar los atributos cuando se cree un nuevo objeto.

Define un método mostrarInformacion () para imprimir el nombre y la edad de la persona.

Estructuras de control en java

Las estructuras de control son esenciales para escribir programas que puedan tomar decisiones y realizar acciones repetitivas de manera dinámica.

Estas estructuras se dividen en 2 grupos:

Estructuras condicionales:

If-else-Switch

Bucles:

for, while, do-while

Ejercicio

Combina todo lo que has aprendido en este tutorial para crear un programa interactivo.

1. Escribe un programa que:
   * Muestre un menú con las siguientes opciones:
     1. Sumar dos números.
     2. Restar dos números.
     3. Multiplicar dos números.
     4. Dividir dos números.
     5. Salir.
   * Use un switch para ejecutar la opción seleccionada.
   * Pida al usuario los números necesarios para cada operación.
   * Use un bucle do-while para seguir mostrando el menú hasta que el usuario elija la opción "Salir".

Modificadores de acceso

¿Qué son los modificadores de acceso?

Los modificadores de acceso son palabras clave en Java que determinan la visibilidad y el alcance de clases, métodos y variables. Existen cuatro tipos: public, private, protected y el modificador por defecto (sin palabra clave). En este tutorial, nos centraremos en los tres primeros.

**Actividad 1: Encapsulación**

Crea una clase llamada CuentaBancaria con un atributo privado saldo. Implementa métodos públicos para depositar dinero, retirar dinero (con una verificación de saldo suficiente) y mostrar el saldo.

**🛠️ Actividad 2: Herencia y protected**

Crea una clase base llamada Animal con un atributo protected llamado nombre. Luego, crea una subclase llamada Perro que tenga un método para imprimir el nombre del animal. ¿Puedes acceder al atributo nombre en Perro?

**🛠️ Actividad 3: Acceso a través de métodos**

Crea una clase llamada Vehiculo con un atributo privado marca y métodos públicos para establecer y obtener el valor de marca. Luego, intenta acceder a marca directamente desde otra clase. ¿Qué ocurre?

Modificador de acceso Final

¿Qué es el modificador final en Java?

El modificador final es una palabra clave en Java que tiene diferentes significados según dónde se aplique. A continuación, vamos a explorar tres formas principales de uso de final:

Variables final: Una vez que se le asigna un valor, no puede cambiar.

Métodos final: Un método final no puede ser sobrescrito por una subclase.

Clases final: Una clase final no puede ser extendida (heredada).

Actividad 1: Crea una constante

Crea una clase que tenga una variable final para almacenar el valor de PI (3.14159) y calcula el área de un círculo. Usa la fórmula:

🛠️ Actividad 2: Protege un método

Crea una clase base con un método final que no deba ser sobrescrito, como un método que imprima un mensaje estándar de bienvenida.

🛠️ Actividad 3: Diseña una clase inmutable

Usa el modificador final en combinación con variables privadas en una clase para asegurarte de que ningún dato interno de la clase pueda ser modificado después de la creación del objeto (piensa en cómo se construyen clases como String).

Constructores y sobrecarga

¿Qué es un constructor?

Un constructor en Java es un método especial que se llama automáticamente cuando se crea una instancia (u objeto) de una clase. Su función principal es inicializar los atributos del objeto. Los constructores:

No tienen un tipo de retorno (ni siquiera void).

Tienen el mismo nombre que la clase.

Pueden recibir parámetros para personalizar la inicialización.

¿Qué es la sobre carga de constructores?

La sobrecarga de constructores es la capacidad de definir múltiples constructores en la misma clase, siempre que tengan diferentes listas de parámetros. Esto permite crear objetos de diferentes maneras, según las necesidades.

Actividad 1: Clase Coche

Crea una clase llamada Coche que tenga los atributos marca, modelo y año. Implementa los siguientes constructores:

Un constructor sin parámetros que inicialice los atributos con valores predeterminados.

Un constructor que reciba solo la marca y modelo.

Un constructor que reciba la marca, modelo y año.

Luego, crea instancias de Coche utilizando cada uno de los constructores y muestra sus valores en la consola.

Herencia y polimorfismo

¿Qué es la herencia en Java?

La herencia es un mecanismo que permite a una clase (denominada subclase) heredar las propiedades y comportamientos (atributos y métodos) de otra clase (denominada superclase). Esto te ayuda a reutilizar código y a extender las funcionalidades de una clase existente.

¿Qué es el polimorfismo?

El polimorfismo significa "muchas formas" y es un principio que permite a las subclases ser tratadas como si fueran instancias de su superclase. En Java, esto significa que una referencia de tipo superclase puede apuntar a un objeto de cualquier subclase.

Actividad 1: Clase de vehículos

Crea una clase base Vehiculo con un método mover(). Luego, crea dos subclases: Coche y Bicicleta, cada una sobrescribiendo el método mover(). Crea una clase de prueba donde crees una referencia de tipo Vehiculo y la uses para instanciar objetos de Coche y Bicicleta. Llama a mover() para cada uno.

🛠️ Actividad 2: Gestión de empleados

Crea una clase base Empleado con un método calcularSalario(). Luego, crea dos subclases: EmpleadoTiempoCompleto y EmpleadoMedioTiempo, cada una con su propia implementación de calcularSalario(). Usa polimorfismo para crear una lista de Empleado que contenga objetos de ambas subclases, y recorre la lista para llamar a calcularSalario().

🛠️ Actividad 3: Zoológico

Usando la clase Animal y sus subclases (Perro, Gato, Vaca), crea un método llamado emitirSonidos(List<Animal> animales) que reciba una lista de objetos Animal. Recorre la lista y llama a hacerSonido() en cada uno, demostrando cómo el polimorfismo permite trabajar con diferentes tipos de animales en una sola colección.

Clases abstractas e interfaces

¿Qué es una clase abstracta?

Una clase abstracta es una clase que no puede ser instanciada directamente. Se utiliza para definir un modelo base o un conjunto de características comunes que deben compartir las clases que la extienden. La clase abstracta puede contener métodos abstractos (que no tienen implementación) y métodos normales (con implementación).

¿Qué es una interfaz?

Una interfaz es una forma de definir un conjunto de métodos que una clase debe implementar. A diferencia de las clases abstractas, las interfaces no contienen ninguna implementación, solo la declaración de los métodos. Las clases que implementan una interfaz deben proporcionar una implementación para todos los métodos de esa interfaz.

#### ****Actividad 1: Clase abstracta de Figura****

Crea una clase abstracta Figura que tenga un método abstracto calcularArea(). Luego, crea dos subclases: Circulo y Rectangulo, que hereden de Figura e implementen el método calcularArea() para calcular el área según su forma. Finalmente, en una clase de prueba, crea instancias de Circulo y Rectangulo, llama a calcularArea() y muestra los resultados.

**Ejercicio 2:**

Crea una nueva clase Avion que también implemente la interfaz Volador. Define el método volar() para que imprima "El avión está volando". Luego, en una clase de prueba, crea instancias de Pajaro y Avion, y llama al método volar() en ambas.

ArrayList en java

¿Qué es un ArrayList?

Un ArrayList es una estructura de datos dinámica en Java que te permite almacenar objetos de forma ordenada. A diferencia de los arrays tradicionales, un ArrayList puede crecer y reducir su tamaño automáticamente según sea necesario.

Características principales:

Capacidad dinámica: Aumenta o reduce el tamaño automáticamente.

Acceso indexado: Los elementos se acceden a través de índices, comenzando desde 0.

Sólo objetos: Los ArrayList solo almacenan objetos. Para almacenar tipos primitivos (como int o double), se utilizan sus clases envolventes (como Integer o Double).

Ejercicio práctico 1:

Crea un ArrayList llamado colores que almacene tres colores como cadenas de texto.

Imprime el segundo color.

Cambia el primer color y vuelve a imprimir la lista completa.

Elimina el último color y vuelve a imprimir la lista.

Agregación de clases en java

¿Qué es la agregación en Java?

La agregación es una relación entre dos clases en la que una clase contiene o incluye instancias de otra clase. Se utiliza cuando un objeto "tiene" otro objeto, pero ambos pueden existir independientemente. A veces, esto se describe como una relación "has-a".

Por ejemplo, una clase "Coche" puede contener un objeto de la clase "Motor", pero un motor puede existir sin estar asociado a un coche.

¿Por qué es importante la agregación?

La agregación en Java te permite construir sistemas más flexibles, modulares y reutilizables. Al separar las responsabilidades en diferentes clases, puedes reutilizar esas clases en diferentes partes de tu aplicación sin duplicar código.

Diferencias clave entre agregación y composición

Es importante diferenciar la agregación de la composición, otro concepto en POO.

Agregación: La relación "has-a" entre las clases es débil. Los objetos pueden existir independientemente. Ejemplo: Un Estudiante tiene una Escuela, pero el estudiante puede existir sin la escuela.

Composición: La relación es más fuerte, y la vida de los objetos está vinculada. Ejemplo: Un Coche tiene un Motor, pero si el coche deja de existir, también lo hace el motor.

Ejercicio práctico de agregación

Construir una pequeña aplicación que modele una biblioteca. Tendrá las clases Libro, Autor y Biblioteca. Los libros serán agregados a la biblioteca, y cada libro tendrá un autor.

Funciones de cadena

En Java, las cadenas (String) son objetos que representan una secuencia de caracteres. A diferencia de otros lenguajes, en Java las cadenas son inmutables, lo que significa que una vez creada, no se puede cambiar su valor. Cualquier operación que intente modificar una cadena genera una nueva cadena.

Ejercicio 1.

#### ****Desafío:****

Crea un programa que tome una frase como entrada y:

1. Imprima la longitud de la frase.
2. Extraiga la primera palabra de la frase.
3. Convierta toda la frase a minúsculas.
4. Reemplace todas las vocales "a" por "o".
5. Divida la frase en palabras y las imprima en líneas separadas.