Juani Sarmiento

Trabajo Práctico Número 1 de Java

¿Qué es un TAD?

Un TAD (Tipo Abstracto de Datos) es un modelo lógico o conceptual que define una estructura de datos junto con las operaciones que se pueden realizar sobre ella, sin especificar su implementación. Se enfoca en qué hace una estructura de datos, no en cómo lo hace.

¿Qué es un objeto o instancia? Dé 3 ejemplos de objetos.

Un objeto es una instancia de una clase, es decir, una entidad que combina atributos (datos) y métodos (comportamientos). Representa un elemento específico del mundo real con características propias.

Ejemplos:

Objeto: Placa de video con marca "Nvidia", modelo "4070", y color "negro".

Instancia:

Marca, Modelo, Vram

Objeto: Persona

Instancia:

Nombre, Edad, Correo

Objeto: Bicicleta

Instancia:

Tipo, Rodado, Color

¿Dónde se produce el encapsulamiento?

El encapsulamiento se produce dentro de una clase, donde sus atributos y métodos pueden estar restringidos en su acceso mediante modificadores de visibilidad (privado, protegido, público). Esto protege la información y evita accesos o modificaciones indebidas.

¿Cuáles son las semejanzas y las diferencias entre atributos y métodos?

Semejanzas:

Ambos pertenecen a una clase y están relacionados con los objetos que la clase representa, además los dos tienen un calificador de acceso y un tipo de dato

Diferencias:

Los atributos representan las características o propiedades de un objeto (ejemplo: nombre, edad, color).

Los métodos son las acciones o comportamientos que puede realizar el objeto (ejemplo: caminar(), hablar(), sumar()).

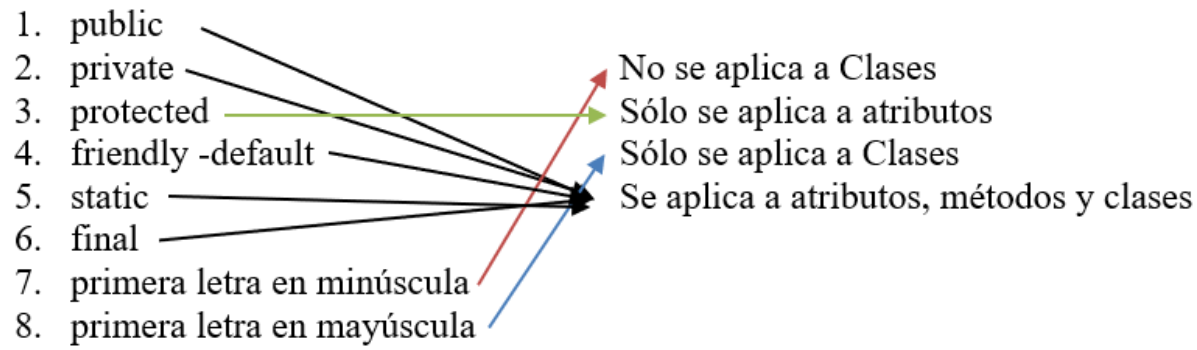
¿Qué es UML? ¿Y cómo se representa una clase en ese lenguaje?

UML (Unified Modeling Language) es un lenguaje de modelado que se usa para representar el diseño de sistemas orientados a objetos mediante diagramas.

Una clase en UML se representa con un rectángulo dividido en tres secciones:

1. **Nombre de la clase (parte superior).**
2. **Atributos (parte media).**
3. **Métodos (parte inferior).**

Una con flechas los calificadores de acceso con sus correspondientes



Realizando abstracción, escriba atributos y sus tipos, intentando que sean diferentes para cada ámbito de aplicación.

Clases	Ámbito en el que se aplica	Atributos
Comida	Para un comercio de venta	- Precio (float) - Fecha de vencimiento (String) - Stock disponible (int)
	Para el experto que lo prepara	- Ingredientes (Lista de Strings) - Tiempo de preparación (int) - Temperatura de cocción (float)
	Para el cliente que lo compra	- Sabor (String) - Tamaño de la porción (String) - Valor nutricional (String)
	Para la empresa que recicla los sobrantes	- Cantidad de desechos generados (int) - Tipo de material reciclable (String) - Método de descomposición (String)

1. **¿Qué es el bytecode?**
   El bytecode es el código intermedio generado por el compilador de Java (.class) después de traducir el código fuente (.java). No es código máquina específico de ningún procesador, sino instrucciones que la Java Virtual Machine (JVM) puede interpretar y ejecutar en cualquier sistema operativo que tenga una JVM instalada.

2. **¿Qué función cumple la JVM?**
   La JVM (Java Virtual Machine) es la encargada de interpretar y ejecutar el bytecode en cualquier sistema operativo, proporcionando la independencia de plataforma característica de Java. Además, gestiona la memoria, el control de hilos y el manejo de excepciones, asegurando la seguridad y eficiencia de la ejecución del programa.

3. **¿Cuándo se ejecuta el recolector de basura de Java?**
   El Garbage Collector (GC) de Java se ejecuta automáticamente cuando la JVM detecta que hay objetos en memoria que ya no son accesibles o necesarios. Sin embargo, su ejecución no está garantizada en un momento exacto, ya que depende de la gestión de memoria de la JVM. También se puede solicitar su ejecución manualmente con `System.gc()`, aunque no se garantiza que ocurra inmediatamente.

Describa los calificadores de acceso de los miembros de esta clase. Desarrolle el código de los métodos de set y get para el atributo que corresponda.

```java
public class Cajero_Automatico {  no usages
    public static final double efectivo=0;  no usages
    private boolean acepta;  2 usages
    private String devuelve;  2 usages

    // Método GET para 'acepta'
    public boolean isAcepta() {  no usages
        return acepta;
    }

    // Método SET para 'acepta'
    public void setAcepta(boolean acepta) {  no usages
        this.acepta = acepta;
    }

    // Método GET para 'devuelve'
    public String getDevuelve() {  no usages
        return devuelve;
    }

    // Método SET para 'devuelve'
    public void setDevuelve(String devuelve) {  no usages
        this.devuelve = devuelve;
    }
}
```

Indique verdadero (V) o falso (F):

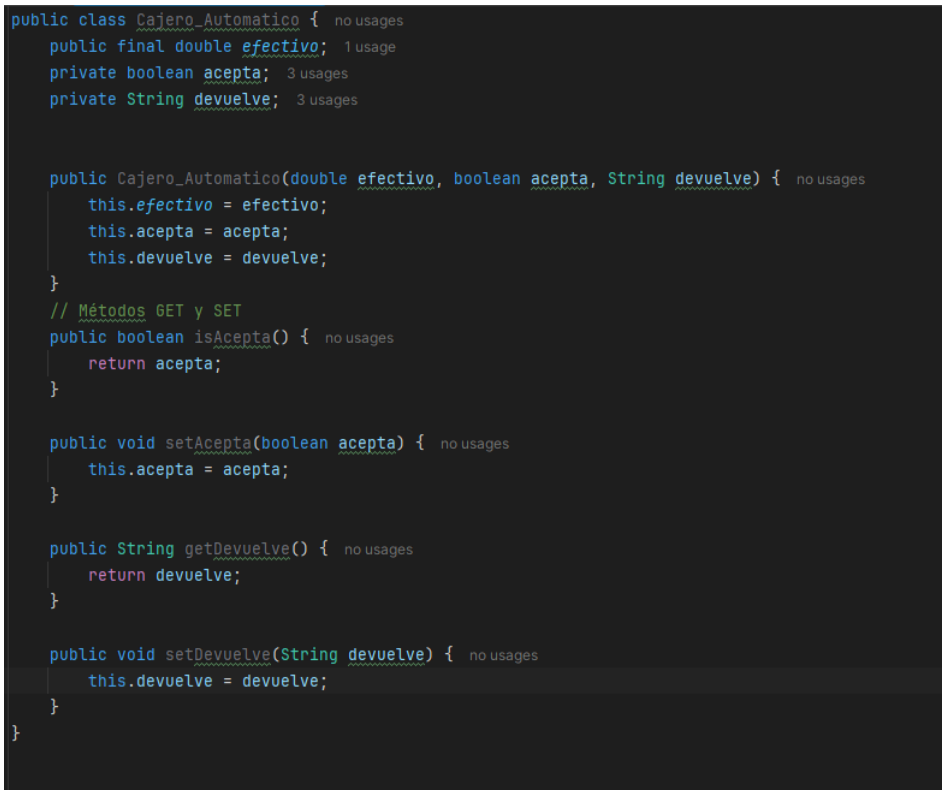
1. **Falso** – Un constructor no es el método principal para ejecutar el programa; el método principal es `main()`.
2. **Verdadero**
3. **Falso** – Un constructor no devuelve el valor de un atributo privado; su propósito es inicializar objetos, no retornar datos.
4. **Falso** – Un constructor no lleva una sentencia `return`, ya que no devuelve valores explícitamente.
5. **Verdadero**
6. **Verdadero**

7. **Falso** – Aunque el nombre del constructor debe coincidir con el de la clase, la convención en Java es que las clases se escriben con mayúscula
8. **Falso** – Un constructor no puede ser `static`, ya que trabaja con instancias de la clase.
9. **Verdadero**

Dados las siguientes opciones determine cuál es un constructor.

Auto(): void ❌ → Tiene tipo de retorno (`void`), por lo que no es un constructor.
Persona(boolean trabaja) : int ❌ → Tiene un tipo de retorno (`int`), por lo que no es un constructor.
Paciente(int doc, int edad) ✅ → No tiene tipo de retorno y su nombre parece ser el de una clase, **es un constructor**.
calcularSuma() : void ❌ → Tiene tipo de retorno (`void`), por lo que no es un constructor.
determinarFeriados() : int ❌ → Tiene tipo de retorno (`int`), por lo que no es un constructor.
Juego(int puntaje) ✅ → No tiene tipo de retorno y su nombre parece ser el de una clase, **es un constructor**.

Defina para la clase Cajero_automático el constructor que recibe valores para todos sus atributos



```
public class Cajero_Automatico {  no usages
    public final double efectivo;  1 usage
    private boolean acepta;  3 usages
    private String devuelve;  3 usages


    public Cajero_Automatico(double efectivo, boolean acepta, String devuelve) {  no usages
        this.efectivo = efectivo;
        this.acepta = acepta;
        this.devuelve = devuelve;
    }
    // Métodos GET y SET
    public boolean isAcepta() {  no usages
        return acepta;
    }

    public void setAcepta(boolean acepta) {  no usages
        this.acepta = acepta;
    }

    public String getDevuelve() {  no usages
        return devuelve;
    }

    public void setDevuelve(String devuelve) {  no usages
        this.devuelve = devuelve;
    }
}
```