**Alumno:** **Rodolfo Nicolás Velasco Fessler**

Mi Repositorio de GitHub: <https://github.com/RodolfoVelasco1/Programacion2-Com3/>

TP1: <https://github.com/RodolfoVelasco1/Programacion2-Com3/tree/master/src/TP1>

**Trabajo Práctico 1**

1. **¿Qué es un TAD?**

Un Tipo de Dato Abstracto (TDA) es un modelo que define valores y las operaciones que se pueden realizar sobre ellos. Y se denomina abstracto ya que la intención es que quien lo utiliza, no necesita conocer los detalles de la representación interna o bien el cómo están implementadas las operaciones.

1. **¿Dónde se produce el encapsulamiento?**

El encapsulamiento se produce en las clases. El encapsulamiento permite ocultar los detalles de implementación de los atributos y métodos de la clase, de modo que solo se expone la interfaz pública de la clase.

Existen tres niveles de acceso para el encapsulamiento:

-Público (Public): Todos pueden acceder a los datos o métodos de una clase que se definen con este nivel.

-Protegido (Protected): Solamente son accesibles dentro de su clase y por subclases.

-Privado (Private): Se puede declarar miembros accesibles sólo para la propia clase.

1. **¿Cuáles son las semejanzas y las diferencias entre funciones, procedimientos y métodos?**

Similitudes:

Las funciones, los métodos y los procedimientos son un conjunto de líneas de código (o instrucciones), encapsulados en un bloque, estos suelen recibir parámetros, cuyo valor utilizarán para efectuar las operaciones.

Diferencias:

Los métodos y las funciones son similares, pero el método está asociado a un objeto, pertenece a un objeto o clase, mientras que una función existe por sí sola, sin necesidad de un objeto para ser usada.

A diferencia de los métodos y funciones, los procedimientos no retornan ningún valor. En el contexto de Java un procedimiento es básicamente un método cuyo tipo de retorno es void.

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

UML, o Lenguaje de Modelado Unificado, es un lenguaje gráfico para el diseño de software. Permite representar los diferentes elementos de un sistema software, como clases, objetos, relaciones, interfaces, etc.

Las clases en UML se representan como un rectángulo con tres compartimentos: Nombre de la clase, Atributos y Métodos.

5. Marcar con cruz.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | No se aplica a las clases | Solo se aplica a los atributos | Solo se aplica a las clases | Se aplica a atributos, metodos y clases |
| Public |  |  |  | X |
| Private | X | X |  |  |
| Protected | X | X |  |  |
| Static |  |  |  | X |
| Final |  |  |  | X |
| Primera letra en minúscula | X |  |  |  |
| Primera letra en mayúscula |  |  | X |  |

1. **Verdadero o falso**

- Un constructor…

o Es el método principal para ejecutar un programa.

**Falso.** El método principal para ejecutar un programa es el método main.

o Crea instancias.

**Verdadero.**

o Devuelve el valor de un atributo privado.

**Falso.** Un constructor no puede devolver el valor de un atributo privado.

o Tiene sentencia return.

**Falso**. Un constructor no tiene una sentencia return.

o Siempre existe uno por defecto, sin parámetros ni inicializaciones de atributos.

**Verdadero.** Si no se define ningún constructor, Java crea un constructor por defecto sin parámetros.

o Se puede sobreescribir.

**Falso.** Un constructor no se puede sobreescribir.

o Se puede sobrecargar.

**Verdadero.**

o Su nombre se escribe con mayúscula.

**Falso.** El nombre de un constructor se escribe con la misma letra que la clase.

o Su calificador de acceso es static.

**Falso.** El calificador de acceso de un constructor no puede ser static.

o Su tipo de devolución no se indica y corresponde a la clase.

**Verdadero.**

- Un método…

o Puede tener múltiples parámetros con el mismo nombre, siempre y cuando tengan tipos

diferentes.

**Falso.** Un método no puede tener dos parámetros con el mismo nombre, incluso si tienen tipos diferentes.

o Puede sobrecargarse.

**Verdadero.**

o Puede sobreescribirse.

**Verdadero.**

o Puede ser static.

**Verdadero.**

o Puede ser tanto public como protected, pero no private.

Falso. Un método puede ser public, protected, default o private.

o Un método puede tener un modificador de acceso final.

**Verdadero.**

1. **Calificadores de acceso. Completa.**

a- Se necesita que cualquiera pueda acceder al color de un vehículo. Entonces, declaro color como:

**public String color;**

b- Se necesita que color se pueda acceder a través no sólo de vehículo, sí no ahora también de

Buses, y como todos sabemos un bus es un tipo de vehículo, entonces también deberá tener

acceso a color. Entonces, declaro color como:

**protected String color;**

c- Se necesita que color se pueda acceder solamente para vehículo. Entonces, declaro color como:

**private String color;**

Ejercicios 8, 9 y 10: <https://github.com/RodolfoVelasco1/Programacion2-Com3/tree/master/src/TP1>