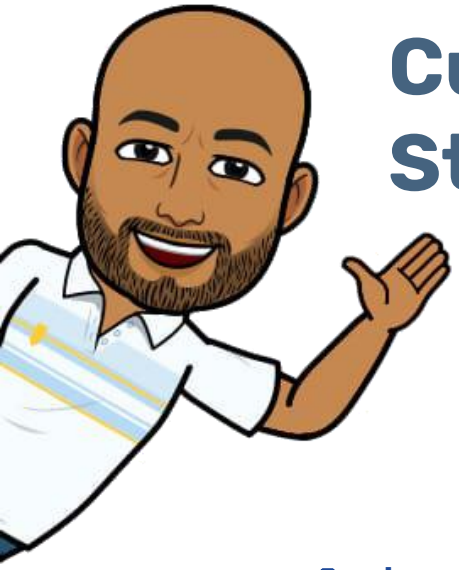

Curso de Java Standard

Ing. Octavio Robleto

octavio.robleto@gmail.com

https://octaviorobleto.com

Introducción

Es el proceso mediante el cual se procede a seleccionar los aspectos mas relevantes y genéricos de un grupo de objetos e identificar los comportamientos mas comunes.

Es uno de los pilares mas importantes y claves en la POO y aunque no lo habíamos mencionado ya hemos realizado parte de este proceso anteriormente, por ejemplo en la clase Auto.

Cuando creamos un programa lo primero que hacemos es analizar las clases que vamos a desarrollar para crear nuestros objetos y de esa forma al pensar en la jerarquía de Herencia creamos una clase con los atributos y métodos esenciales y así tener algo como una “Plantilla Base” para aprovecharla.

Abstracción

En este proceso podemos observar que hemos abstraído tanto un objeto y al momento de plasmarlo o codificarlo en una clase vemos que seria incongruente instanciar uno con solo estas características y comportamientos por lo que recurriríamos a crear unas clases hijas mas especificas que si terminen de cumplir con los requerimientos de nuestro software.

Para evitar instanciar un objeto tan genérico recurrimos a la palabra reservada **abstract** después del modificador de acceso y antes de la palabra reservada **class**.

```
public abstract class Auto
```

Métodos Abstractos

Además de impedir que una clase abstracta sea instanciada también podemos también crear métodos abstractos.

Un método abstracto es aquel que solo posee firma (modificador de acceso, tipo de retorno, identificador y parámetros) y no posee el cuerpo (algoritmo), esto completa la idea de la abstracción ya que en muchos momentos vamos a querer que nuestras clases hijas implementen un mismo método pero según sea el caso se comporte diferente.

```
public abstract void vender();
```

Solo las clases abstractas pueden poseer métodos abstractos, si codificamos un método abstracto y la clase no es abstracta nos dará un error en tiempo de compilación.

Y que ganamos con esto? La ventaja es que las clases que hereden de una clase abstracta con métodos abstractos el JDK obligará a implementar los métodos en dichas clases (apoyándonos en el concepto de sobreescritura de miembros) siempre y cuando las clases hijas no sean abstractas.