educación IT
CODE YOUR FUTURE

Bootcamp Java Developer

Fase 1 - Java Analyst

Módulo 8

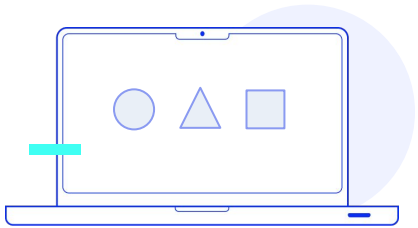
Abstracción

Introducción a abstracción

Es el proceso mediante el cual se procede a seleccionar **los aspectos más relevantes y genéricos de un grupo de objetos y a identificar los comportamientos más comunes**. Es uno de los pilares más 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 se crea un programa, lo primero que se hace es un análisis de las clases que se van a desarrollar para crear los objetos.

Al pensar en la jerarquía de herencia se crea una clase con los atributos y métodos esenciales para tener como “Plantilla Base”.



Abstracción

En este proceso, puede ocurrir que, luego de abstraer el objeto y al momento de plasmarlo o codificarlo en una clase, se descubre que sería incongruente instanciar uno con solo unas pocas características y comportamiento. En ese caso, se crean **clases *hijas* más específicas** que sí 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 es posible 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.

¿Qué se gana con esto? La ventaja es que para 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.

¡Sigamos trabajando!