Abstracción II

¡Comencemos!

¿Qué aprenderás?

- Comprender las clases abstractas para generar herencia.
- Implementar polimorfismo para los principios de POO mediante herencia.

Introducción

En este capítulo seguiremos viendo aquellos conceptos de abstracción que son utilizados por desarrolladores/as de todo el mundo. Por ejemplo, veremos qué son las clases abstractas, el polimorfismo aplicado y algunos principios que son útiles en la Programación Orientada a Objetos.

¡Vamos con todo!

Las clases abstractas

Las clases abstractas no son más que clases que no pueden instanciarse. De ahí el concepto de abstracto. Según el filósofo José Ortega y Gasset podemos entender por "sustantivo abstracto" a aquella palabra que nombra un objeto que no es independiente, es decir, que siempre necesita de otro elemento en el que apoyarse para "ser". Esto significa que dichos sustantivos, al no referirse a un elemento concreto, hacen referencia a objetos que no se pueden percibir con los sentidos, sino solo imaginar.

El significado de que no puedan instanciarse radica en que son clases que permiten categorizar a otras. Por ejemplo, la clase `Animal` que posee una herencia, debiera ser abstracta, ya que la palabra Animal la asociamos al concepto de animal como tal, pero no es algo tangible. En la imagen 1 se puede apreciar un ejemplo que será la guía para el ejercicio guiado y entender estos conceptos.

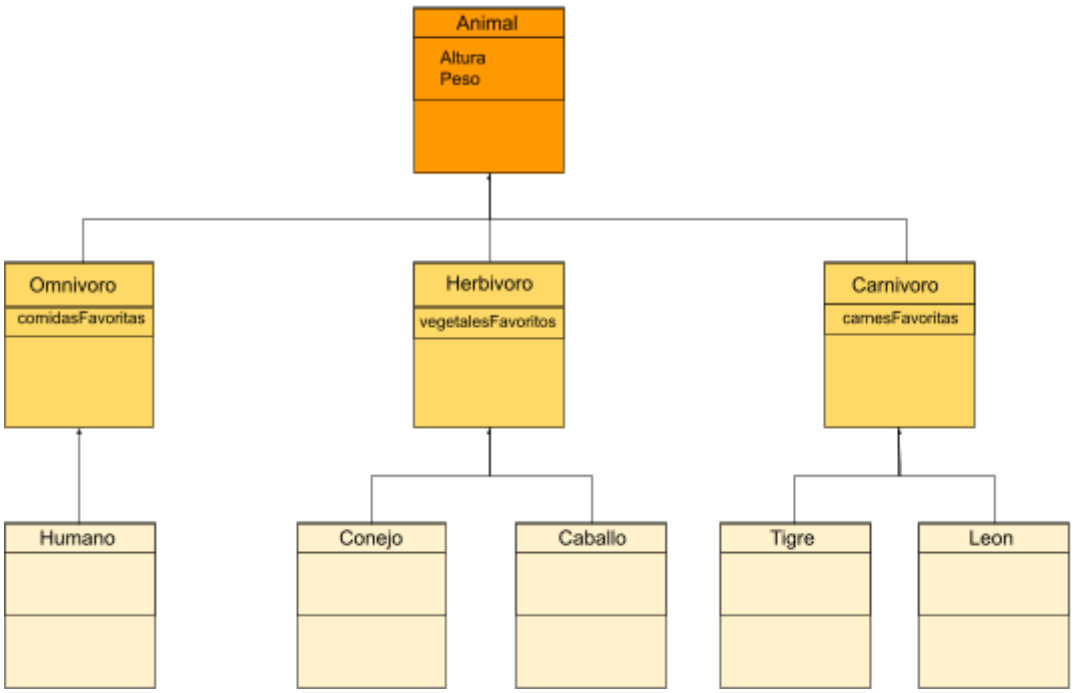

Imagen 1. Ejemplo de clase abstracta clase Animal.
Fuente: Desafío Latam

En la imagen se puede apreciar que la herencia se cumple mediante la superclase `Animal` y las subclases `Omnivoro`, `Herbivoro` y `Carnivoro`. A su vez existe otra herencia de otras subclases de cada categoría de alimentación del animal.

Ejercicio Guiado: Animales

Paso 1: Crear una clase llamada Animal que contiene la palabra reservada `abstract` antes de la palabra `class`. Esto significa que la clase **no puede ser instanciada,** generamos los getter and setter correspondientes de las variables altura y peso**.**

```
public abstract class Animal {
 private int altura;
 private int peso;

 public int getAltura() {
 return altura;
 }
 public void setAltura(int altura) {
 this.altura = altura;
 }
 public int getPeso() {
 return peso;
 }
 public void setPeso(int peso) {
 this.peso = peso;
 }
```

Esta clase puede ser la superclase de otras, como se ve en el diagrama, las cuales pueden o no ser abstractas al igual que `Animal`.

Paso 2: Terminar el árbol de carnívoros, creando la clase **abstracta** `Carnivoro`.

```
import java.util.List;

public abstract class Carnivoro extends Animal{
    List<String> carnesFavoritas;

public List<String> getCarnesFavoritas() {
      return carnesFavoritas;
   }
public void setCarnesFavoritas(List<String> carnesFavoritas) {
this.carnesFavoritas = carnesFavoritas;
  }
}
```

Paso 3: Crear la clase `Tigre` que extiende de `Carnivoro`.

```
//Tigre
public class Tigre extends Carnivoro{
}
```

Paso 4: Crear la clase `Leon` que extiende de `Carnivoro`.

```
//Leon
public class Leon extends Carnivoro{
}
```

`Tigre` y `Leon` podrán tener todos los atributos de `Carnivoro` y `Animal` .

Paso 5: Agregar un método main para crear esta lista polimórfica con clases abstractas.

```
public class Main {
  public static void main(String[] args) {
    Leon leon = new Leon();
    Tigre tigre = new Tigre();
     ArrayList<Animal> listaAnimales = new ArrayList<>();
     ArrayList<Carnivoro> listaCarnivoros = new ArrayList<>();
     listaAnimales.add(leon);
     listaAnimales.add(tigre);
     listaCarnivoros.add(leon);
     listaCarnivoros.add(tigre);
     System.out.println(listaAnimales);
     System.out.println(listaCarnivoros);
   }
}

    ---------------------------
    Impresión en pantalla:
    [desafio1.Leon@58372a00, desafio1.Tigre@4dd8dc3]
    [desafio1.Leon@58372a00, desafio1.Tigre@4dd8dc3]
```

Podemos agregar instancias de las subclases sin problemas a la lista, es decir, se puede utilizar polimorfismo y a la vez ayudamos a mantener un código ordenado, impidiendo que otros desarrolladores creen instancias de `Animal` o `Carnivoro`.

Otra ventaja de esto, es que podemos escribir el código en la superclase de un método o un atributo y utilizarlos en todas las subclases.

El concepto de abstracción en POO es el conjunto de todo lo aprendido previamente, ya que la abstracción es la realización de polimorfismo en busca de características comunes en objetos, todo esto con la finalidad de trabajarlos dentro de un mismo contexto.

Imagen 2. Principio de Abstracción.
Fuente: Desafío Latam