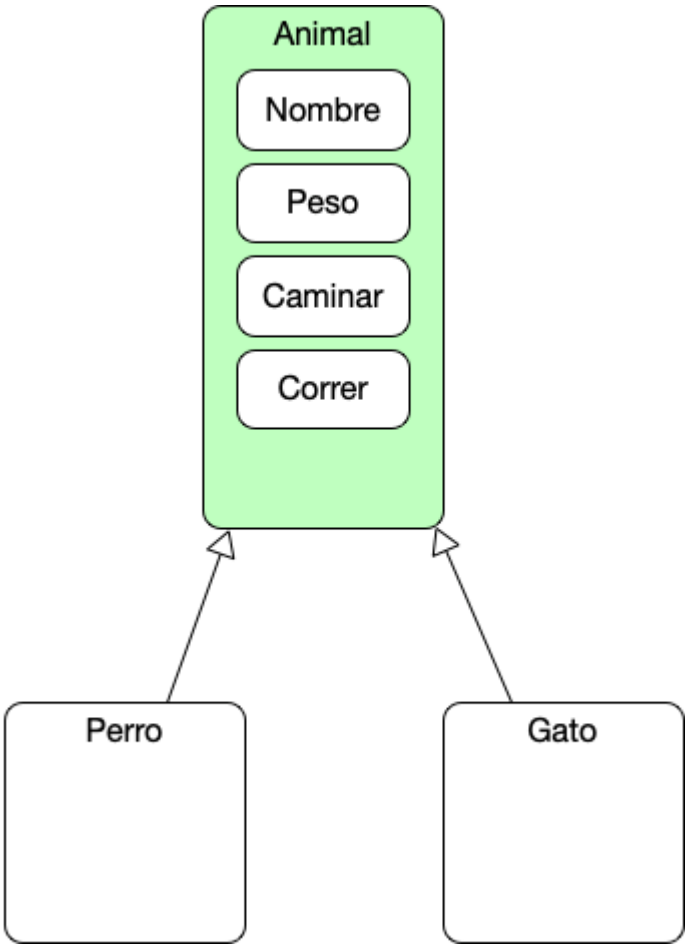
Clases Abstractas vs Interfaces. Esta es una de las preguntas más clásicas de programación orientada a objeto . ¿Qué diferencia existe entre ambas?. La realidad es que son dos conceptos cercanos pero no idénticos. Una clase Abstracta sirve como plantilla a nivel de comportamiento y propiedades sobre una Jerarquía de clases completa. Uno de los ejemplos más habituales es la clase Animal que puede tener la propiedad peso , nombre y los métodos correr y andar. La peculiaridad de la clase Animal es que no puede instanciar o crear un objeto de ella ya que lo que existe es un Perro un Gato o un Leon pero no el concepto de Animal propiamente dicho no existe y no tiene sentido que haya objetos de este tipo instanciados.



Hasta aquí todo correcto. Una clase abstract no se puede instanciar . Sin embargo si puede disponer de propiedades y funcionalidad . Es decir el método caminar puede implementar código :

```
public void caminar() {
System.out.println("el animal camina");

}
```

Java Clases Abstractas

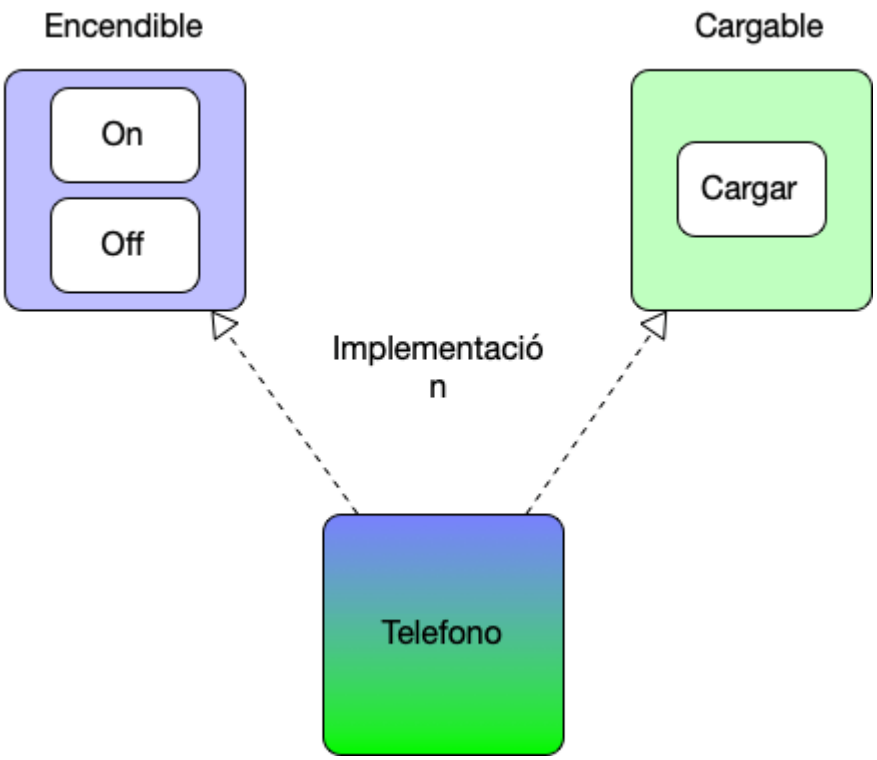
Las clases abstractas también en muchos casos disponen de métodos abstractos . Métodos que no implementan código y que las clases hijas están obligadas a implementar.

```
public abstract void correr();
```

Por ejemplo en este caso podría ser el método correr el cual puede no tener sentido tener declarado en Animal y obligar a las clases hijas a implementarlo salvo que estas mismas sean clases abstractas.

Clases Abstractas vs Interfaces y herencia

Una de las diferencias fundamentales entre una clase Abstracta y un interfaces es que una clase puede implementar varios interfaces pero solo puede tener una clase Padre sea abstracta o no . Por lo tanto la flexibilidad a nivel de que clases implementan que interfaces es mucho más abierta en estes últimos. Por ejemplo un Teléfono movil puede implementar el interface Encendible con los métodos on y off y a la vez el interface cargable con el método cargar.



Interfaces y métodos

Muchas personas me comentan que los interfaces ademas solo pueden disponer de métodos vacíos sin código ya que definen un esqueleto que otras clases implementarán. Lamentablemente esto ya no es cierto a partir de Java 8 ya que los interfaces pueden implementar métodos por defecto y métodos estáticos.

Clases abstractas vs Interfaces y propiedades

Otra de las diferencias fundamentales es que las clases Abstractas suelen contener bastantes propiedades . Algo de lo cual los interfaces carecen. Eso sí recordar que los interfaces pueden tener variables finales estáticas declaradas. Es decir constantes .

Diferencias fundamentales

Desde mi punto de vista las clases abstractas sirven para reutilizar código en jerarquías de herencia y generar un nivel mayor de encapsulación en ellas. Por el otro lado los interfaces

aportan flexibilidad y extensibilidad a nuestro sistema ya que pueden implementarse de forma multiple y únicamente declaran la funcionalidad y no la implementan

Otros artículos relacionados

- Java Herencia Multiple y sus opciones
- Java Herencia y sus limitaciones
- Java modificadores acceso y curiosidades
- Clases Abstractas