Classes contra Interfícies a Java

Classes contra Interfícies

Les **classes** són plantilles d'objectes.

- Defineixen el tipus d'objectes de dades que contindrà i els mètodes per a operar amb aqueixes dades.
- Un exemplar d'aqueixa classe és la plantilla farcida amb les dades i les anomenades als mètodes
  - l'exemplar és l'**objecte**.
- Podem crear molts exemplars de la mateixa classe, creant molts objectes d'aquest tipus. Cada objecte té les seues pròpies dades.
- No hem d'escriure una classe des del principi. Si ja existeix una classe que té característiques similars a la que volem crear, podem utilitzar la paraula clau extends per a heretar camps i mètodes des d'una altra classe. Després afegim més camps i mètodes per a crear una classe més rica que la classe pare. Però no hem de conformar-nos amb els camps o mètodes heretats. Podem sobreescriure mètodes, o ocultar dades de la classe pare per a complir amb les nostres necessitats.

Això és cert per a classes concretes, però no per a les interfícies.

Les **interfícies** declaren mètodes, i possiblement camps estàtics, però no defineixen mètodes.

- En el seu lloc, les interfícies només declaren els mètodes sense cap instrucció dins d'ells.
- En altres paraules, les interfícies són com a plantilles que sempre seran plantilles. No podem exemplificar una interfície per a crear un objecte.
- Són com a classes sense implementació.

Llavors per què existeixen?

Serveixen per a un propòsit: per a forçar al desenvolupador a proporcionar aqueixos mètodes, amb detalls, en la classe que implementa la interfície. En altres paraules, implementar una interfície significa que estem fent la promesa d'usar uns certs mètodes, però nosaltres, els desenvolupadors, definim els detalls d'aqueixos mètodes.

Per què és això útil o necessari?

Suposem que tenim un equip de desenvolupadors que estan creant classes per a fer diferents tipus d'objectes animals. Tots aqueixos animals tindran dues coses en comú:

- Usen alguna forma de locomoció.
- Mengen alguna classe de menjar.

La diferència entre els animals està en com es mouen i quins i com mengen. En altres paraules, cadascun necessita tindre els mètodes locomotion() i eat(), però cada individu, cada classe animal, defineix els detalls d'aqueixos mètodes separadament basant-se en les necessitats de les espècies d'animals.
Dissenyem una interfície per a assegurar-nos que cada objecte animal fa unes certes coses, i el nostre equip desenvolupa classes com aquesta:

Animal Interface

```
public interface Animal
 {
    public void locomotion();
    public void eat();
 }
```

Class Shark

```
public class Shark implements Animal
 {
    public void locomotion()
    {
      System.out.println("I swim.");
    }
    public void eat()
    {
      System.out.println("I hunt for seals.");
    }
 }
```

Class Dog

```
public class Dog implements Animal
 {
   public void locomotion()
    {
      System.out.println("I run on four legs.");
    }
   public void eat()
    {
      System.out.println("I eat kibble.");
    }
 }
```

```
public class  AnimalTest
 {
   public static void main(String[] arg)
     {
       Shark shark = new Shark();
       shark.locomotion();
       shark.eat();
       Dog dog = new dog();
       dog.locomotion();
       dog.eat();
     }
 }
```