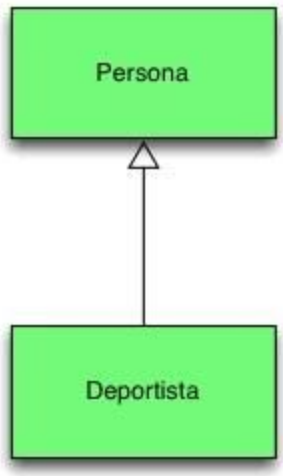
WildCard

Para finalizar con los tipos genéricos, nos falta ver el uso del carácter "?" al que se le suele denominar wildcard . El cual nos va a permitir resolver el siguiente problema que tenemos con las listas.

Vamos a suponer que tenemos dos clases, la clase Persona y la clase Deportista.

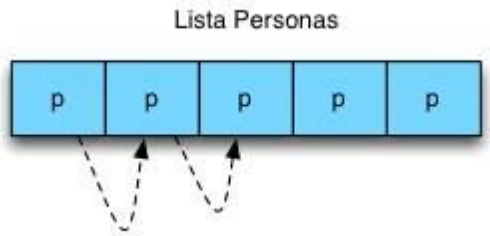


Podemos crearnos una lista de Personas la cual tenemos que recorrer e imprimir por pantalla.

```
public class Main {

    public static void main(){
        List<Persona> listaPersonas=new ArrayList<Persona>();
        listaPersonas.add(new Persona("pepe"));
        listaPersonas.add(new Persona("maria"));
        imprimir(listaPersonas);
    }
    public static void imprimir(List<Persona> lista) {
        for(Persona p:lista) {
            System.out.println(p.getNombre());
        }
    }
}
```

El programa simplemente crea una lista de Personas la llena y la recorre.



Ahora bien, que pasaría si realizáramos una modificación a la lista y esta lista fuera de Deportistas. El código quedaría de la siguiente forma:

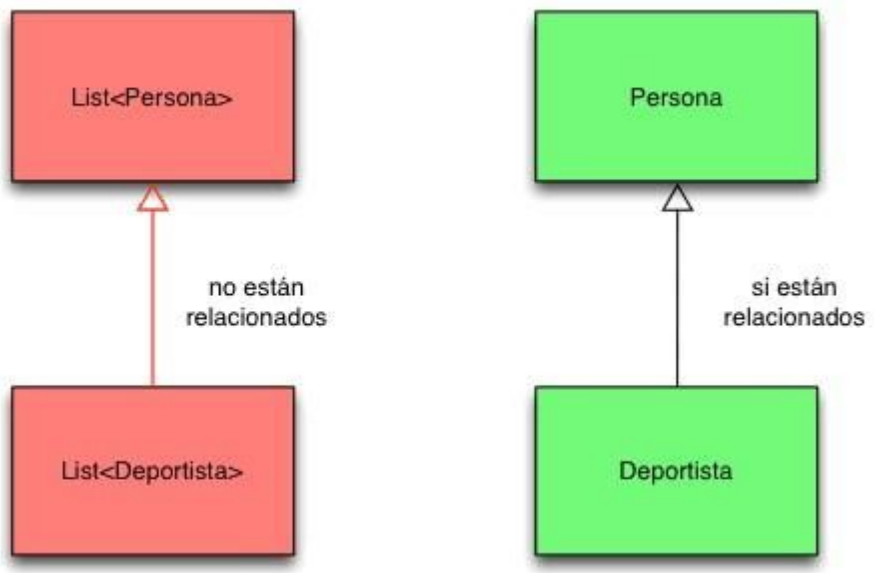
```
public class Main {

    public static void main(){
        List<Deportista> listaPersonas=new ArrayList<Deportista>();
        listaPersonas.add(new Deportista("pepe","futbol"));
        listaPersonas.add(new Deportista("maria","tenis"));
        imprimir(listaPersonas);
    }
    public static void imprimir(List<Persona> lista) {
        for(Persona p:lista) {
            System.out.println(p.getNombre());
        }
    }
}
```

Lamentablemente este código no compila y da el siguiente error:

The method imprimir(List<Persona>) in the type Principal is not applicable for the arguments (List<Deportista>)

Parece extraño ya que estamos pasando una lista de Deportista que también son Personas. Sin embargo tenemos que darnos cuenta de lo siguiente. Una lista de Personas y una lista de Deportistas son dos tipos diferentes y no están relacionados. Aunque los objetos que ellos almacenen si lo estén.



Al tratarse de dos tipos de objetos que realmente no están relacionados por herencia no nos podemos apoyar en el polimorfismo para gestionarles de forma común.

El Wildcard(?) aporta una solución y nos ayuda para decirle a Java que cuando usemos un tipo genérico se puede aplicar cualquier tipo al parámetro lista

```
public static void imprimir(List<?> lista) {

for(Persona p:lista) {
System.out.println(p.getNombre());
}

}
```

De esta forma añadimos flexibilidad, ya que podemos pasar cualquier tipo como parámetro genérico. Sin embargo tenemos todavía un problema al recorrer la lista de Personas, estamos obligando a que la clase sea de tipo Persona cosa que el Wildcard no obliga. Así que seguimos con problemas de compilación. Para evitar esto podemos añadir al Wildcard una restricción.

```
public static void imprimir(List<? extends Persona> lista) {
   for(Persona p:lista)
{
 System.out.println(p.getNombre());
 }

 }
```

Ahora sí que podremos trabajar tranquilos con los genéricos y los tipos.