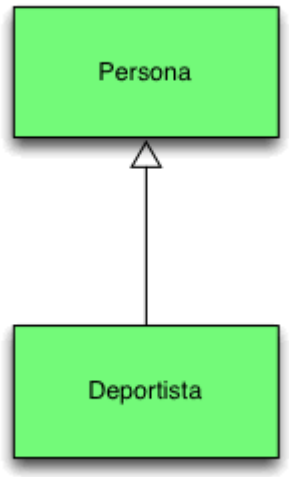
En este artículo vamos a cubrir una de las dudas más habituales que se producen con el manejo Java Generics, el uso del caracter “?” al que se le suele denominar wildcard . Vamos a suponer un ejemplo sencillo en el que tenemos dos clases. La clase Persona y la clase Deportista.



Podemos crearnos una lista de Personas la cual tenemos que recorrer e imprimir por pantalla. El programa Java sería así de sencillo :

```
package com.arquitecturajava;

import java.util.ArrayList;
import java.util.List;

public class PrincipalDeportista {
public static void main(String[] args) {

List&amp;lt;Persona&amp;gt; listaPersonas=new
```

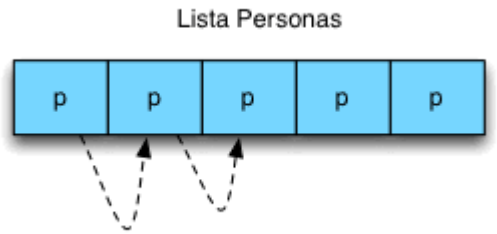
```
ArrayList&amp;lt;Persona&amp;gt;();
listaPersonas.add(new Persona("pepe"));
listaPersonas.add(new Persona("maria"));
imprimir(listaPersonas);
}
public static void imprimir(List&amp;lt;Persona&amp;gt; lista) {

for(Persona p:lista) {

System.out.println(p.getNombre());
}

}
}
```

Mostramos tambien la clase:

```
package com.arquitecturajava;

public class Persona {

&amp;amp;nbsp;

private String nombre;

public String getNombre() {
return nombre;
}
```

```
public void setNombre(String nombre) {
this.nombre = nombre;
}

public Persona(String nombre) {
super();
this.nombre = nombre;
}

}
```

El programa simplemente construye una lista de Personas la rellena y luego la recorre.



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

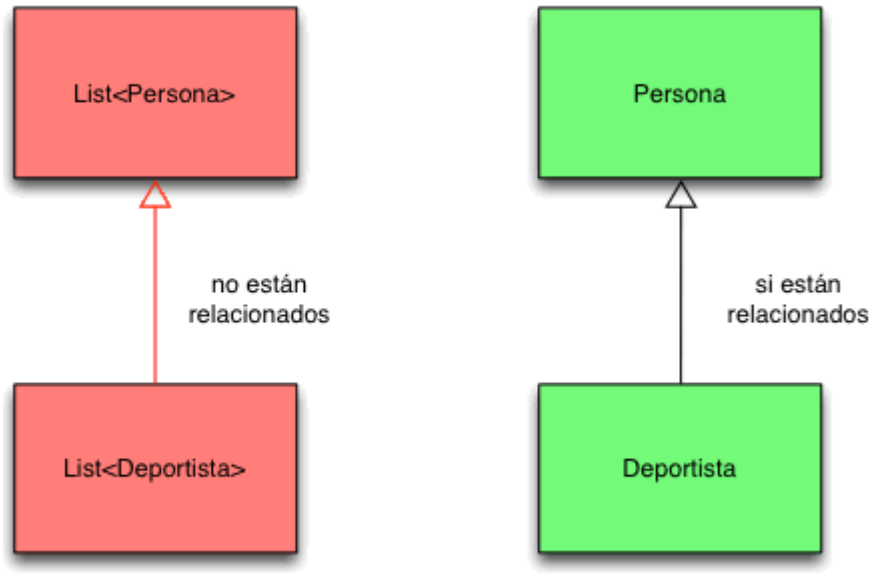
```
package com.arquitecturajava;

import java.util.ArrayList;
import java.util.List;

public class Principal {
```

```
public static void main(String[] args) {

List&amp;lt;Deportista&amp;gt; listaPersonas=new
ArrayList&amp;lt;Deportista&amp;gt;();
listaPersonas.add(new Deportista("pepe","futbol"));
listaPersonas.add(new Deportista("maria","tenis"));
imprimir(listaPersonas);
}
public static void imprimir(List&amp;lt;Persona&amp;gt; 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.



Así pues 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. Vamos a ver una solución.

Java Generics y Wildcard (?)

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

vamos a verlo:

```
public static void imprimir(List&amp;lt;?&amp;gt; lista) {

for(Persona p:lista) {

System.out.println(p.getNombre());
}
```

```
}
```

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

```
public static void imprimir(List&amp;lt;? extends Persona&amp;gt;
lista) {

 for(Persona p:lista) {

 System.out.println(p.getNombre());
 }

 }
```

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

Otros artículos :

Java Generics