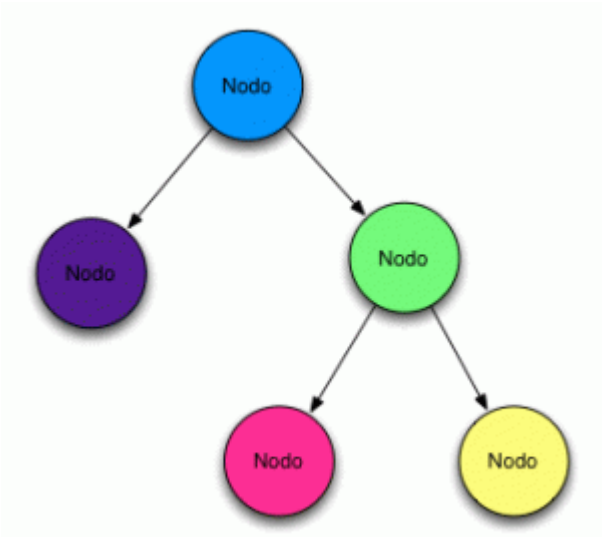
En muchas ocasiones cuando utilizamos Java Generics nos es suficiente con utilizar el framework de Colecciones (ArrayList,HashSet etc) para solventar nuestros problemas . Sin embargo en algunos casos esto no es posible ya que ninguno de los elementos encajan con lo que estamos buscando. En estos casos debemos construir un Java Custom Generics que se encargue de solventar el problema al que nos enfrentamos. Un ejemplo clásico de esto es el concepto de Nodo. Este concepto es conocido por todos ya que HTML lo usa a traves de su famoso DOM (Document Object Model). Recordemos que un Nodo es un elemento que esta relacionado con otro elemento de su mismo tipo de forma jerárquica.



Nodos y Java Custom Generics

Vamos a ver como construimos una clase genérica que gestione el concepto de Nodo.

```
package com.arquitecturava;
```

```
public class Nodo&amp;lt;T&amp;gt; {
 private T dato;
 private Nodo&amp;lt;T&amp;gt; siguiente;
 private Nodo&amp;lt;T&amp;gt; anterior;

 public Nodo(T dato) {
 this.dato = dato;
 }

 public void setDato(T dato) {

 this.dato = dato;
 }

 public T getDato() {

 return this.dato;
 }

 public void setSiguiente(Nodo&amp;lt;T&amp;gt; siguiente) {

 this.siguiente = siguiente;

 }

 public Nodo&amp;lt;T&amp;gt; getSiguiente() {

 return this.siguiente;
 }

 public Nodo&amp;lt;T&amp;gt; getAnterior() {
```

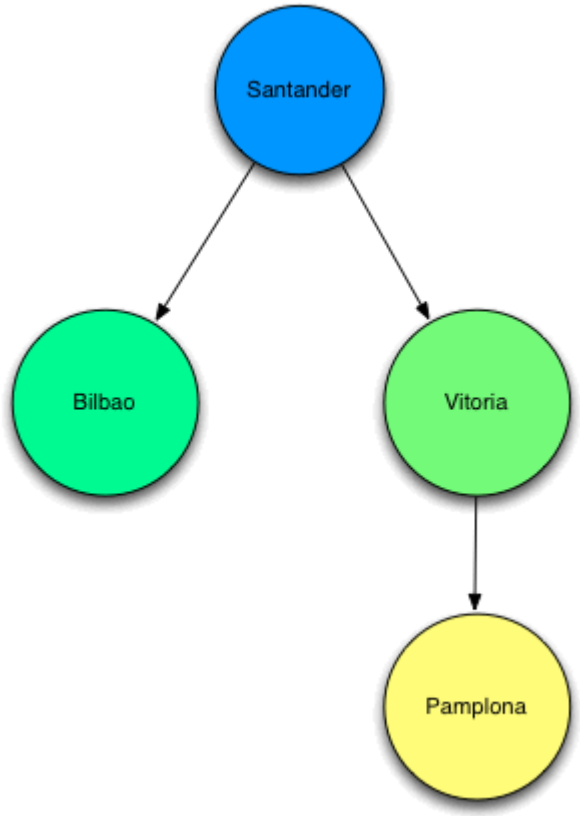
```
 return anterior;
 }

 public void setAnterior(Nodo&amp;lt;T&amp;gt; anterior) {
 this.anterior = anterior;
 }
}
```

Como podemos ver la clase Nodo dispone de un enlace con el Nodo siguiente y con el Nodo anterior. Se trata ademas de una clase genérica así pues podemos asignar cualquier tipo de objeto. Vamos a usar el concepto de Ciudad para construir una estructura de Nodos.

```
package com.arquitecturava;

public class Ciudad {

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

 public String getNombre() {
 return nombre;
 }

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

 private String nombre;
```

```
}
```

Una vez tenemos el concepto de Ciudad vamos a crearnos la siguiente estructura de nodos.



Vamos a verlo en código:

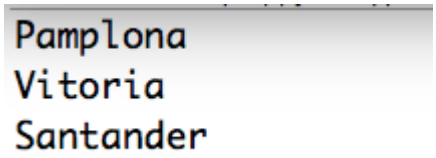
```
public static void main(String[] args) {

 Ciudad c1= new Ciudad("Santander");

 Ciudad c2= new Ciudad("Bilbao");
```

```
 Ciudad c3= new Ciudad("Vitoria");

 Ciudad c4= new Ciudad("Pamplona");

 Nodo&amp;lt;Ciudad&amp;gt; nodo1= new Nodo&amp;lt;Ciudad&amp;gt;(c1);
 Nodo&amp;lt;Ciudad&amp;gt; nodo2= new Nodo&amp;lt;Ciudad&amp;gt;(c2);
 Nodo&amp;lt;Ciudad&amp;gt; nodo3= new Nodo&amp;lt;Ciudad&amp;gt;(c3);
 Nodo&amp;lt;Ciudad&amp;gt; nodo4= new Nodo&amp;lt;Ciudad&amp;gt;(c4);

 nodo1.setSiguiente(nodo2);
 nodo2.setAnterior(nodo1);
 nodo2.setSiguiente(nodo4);
 nodo3.setAnterior(nodo1);
 nodo4.setAnterior(nodo3);
 viaje(nodo4);
 }
```

Acabamos de crear una estructura de Nodos nos queda ver el código de la función viaje que de forma recursiva nos mostrará como los nodos están relacionados para un viaje entre Santander y Pamplona.

```
private static void viaje (Nodo&amp;lt;Ciudad&amp;gt; nodo) {
 System.out.println(nodo.getDato().getNombre());
 if (nodo.getAnterior()!=null) {
 viaje(nodo.getAnterior());
 }

 }
```

El resultado es el siguiente :



Los Java Custom Generics pueden sernos muy útiles en muchos casos en los cuales las estructuras clásicas no son capaces de aportar una solución.

Otros artículos relacionados:

Java Generics

Java Generics WildCards

Java Set y List