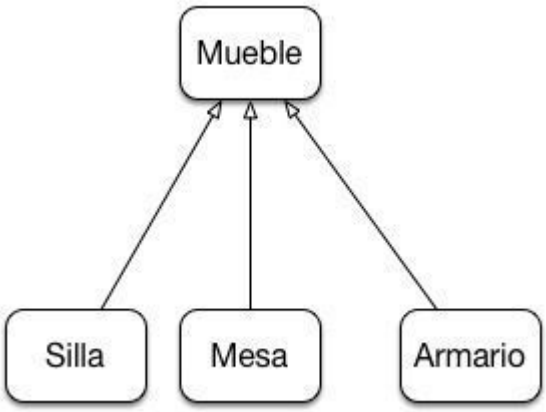
¿Qué son los Java Mixins? . El concepto de Mixin esta asociado a la reutilización de código. Un mixin es una clase que contiene código para que otras clases lo reutilicen pero que no se apoya en la herencia para hacerlo. Dicho así suena un poco extraño pero vamos a ver un ejemplo sencillo que nos aclare el tema. Supongamos que tenemos una jerarquía de clases orientadas a muebles.



Tenemos la clase padre Mueble y varias clases hijas que heredan sus propiedades. Hasta aquí todo es normal .

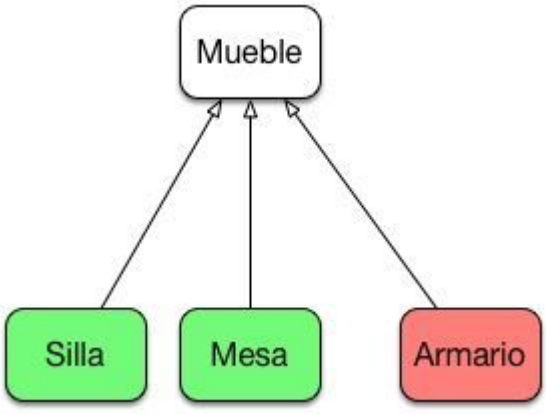
```
package com.arquitecturajava;

public class Mueble {

	private String color;

	public String getColor() {
		return color;
	}

	public void setColor(String color) {
		this.color = color;
	}
}
```

```
package com.arquitecturajava;

public class Armario extends Mueble {

}
```

```
package com.arquitecturajava;

public class Mesa extends Mueble{

}
```

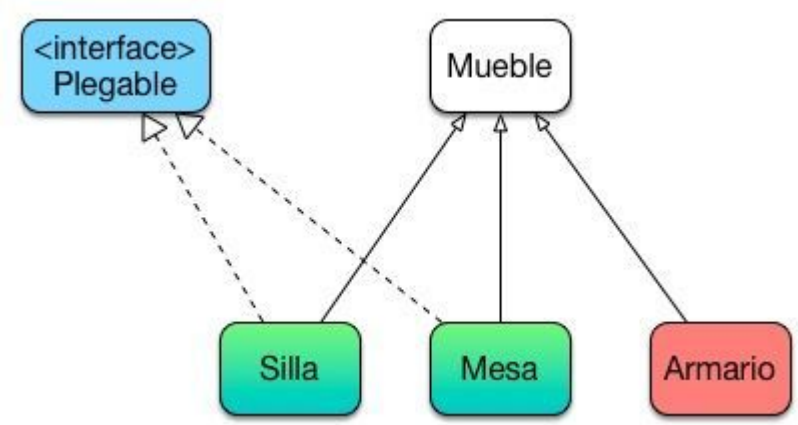
```
package com.arquitecturajava;

public class Silla  extends Mueble{

}
```

En este caso solo hemos añadido una propiedad a la clase Mueble y todas las clases hijas la heredan. Ahora bien ¿que pasaría si queremos añadir la funcionalidad de plegar un mueble?. Es aquí donde tenemos un problema ya que la Silla y la Mesa se pueden plegar pero no así el Armario. ¿Cómo solventamos este problema?.



Java Mixins al rescate

A partir de Java 8 , usar Java Mixins es muy sencillo . Recordemos que los interfaces soportan default methods , es decir métodos con implementación propia. Por lo tanto nos basta con construir un interface que implemente el método plegar.



Vamos a ver el código de este interface:

```
package com.arquitecturajava;

public interface Plegable {

        public  default void plegar() {

                System.out.println("el mueble se pliega");
        }
}
```

Modificamos nuestras clases para que implementen el interface de forma sencilla:

```
package com.arquitecturajava;

public class Silla  extends Mueble implements Plegable{
```

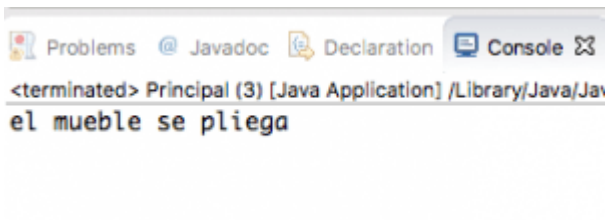
```
}
```

```
package com.arquitecturajava;

public class Mesa  extends Mueble implements Plegable{

}
```

Podemos probar el código desde un programa main y ver como la clase Silla y Mesa implementa la nueva funcionalidad y disponen del metodo plegar sin tener que modificar la Jerarquía de clases:

```
package com.arquitecturajava;

public class Principal {

	public static void main(String[] args) {

		Mesa m= new Mesa();
		m.plegar();

	}

}
```

El resultado imprimira por la consola:



Conclusiones

Hemos usado un Java Mixin para reutilizar funcionalidad dentro de una jerarquía de clases apoyandonos en default methods de Java 8

Otros artículos relacionados

- JQuery WrappedSet
- Java Strategy Pattern herencia vs composición
- Java 8 default methods y extensibilidad
- Java 8 interface static methods y reutilizacion