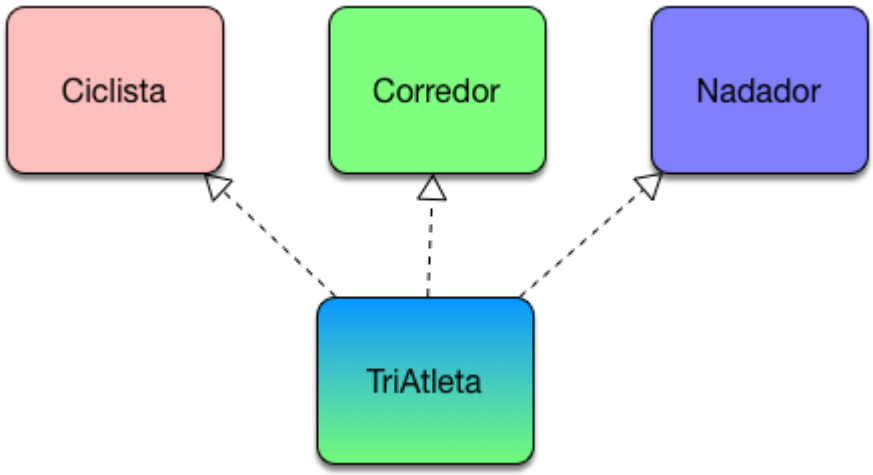
El soporte de Java Multiple Inheritance ha sido siempre una de las características que mas se echaban en falta en Java. La razón que siempre se ha esgrimido es que la herencia múltiple es algo que complica el código y la vida al desarrollador. Sin embargo hay que reconocer que en más de una ocasión nos puede ayudar a aumentar la reutilización. A partir de Java 8 los interfaces aportan soporte para la herencia múltiple utilizando default methods. Vamos a construir una jerarquía de interfaces y clases para herencia multiple.



En este caso hemos definido 4 conceptos , Nadador, Corredor ,Ciclista y TriAtleta. Este último heredara la funcionalidad de los otros .

```
package com.arquitecturajava;

public interface Ciclista {

	public default void haceBici() {
		System.out.println("la persona anda en bici");
	}
}
```

```
package com.arquitecturajava;

public interface Corredor {

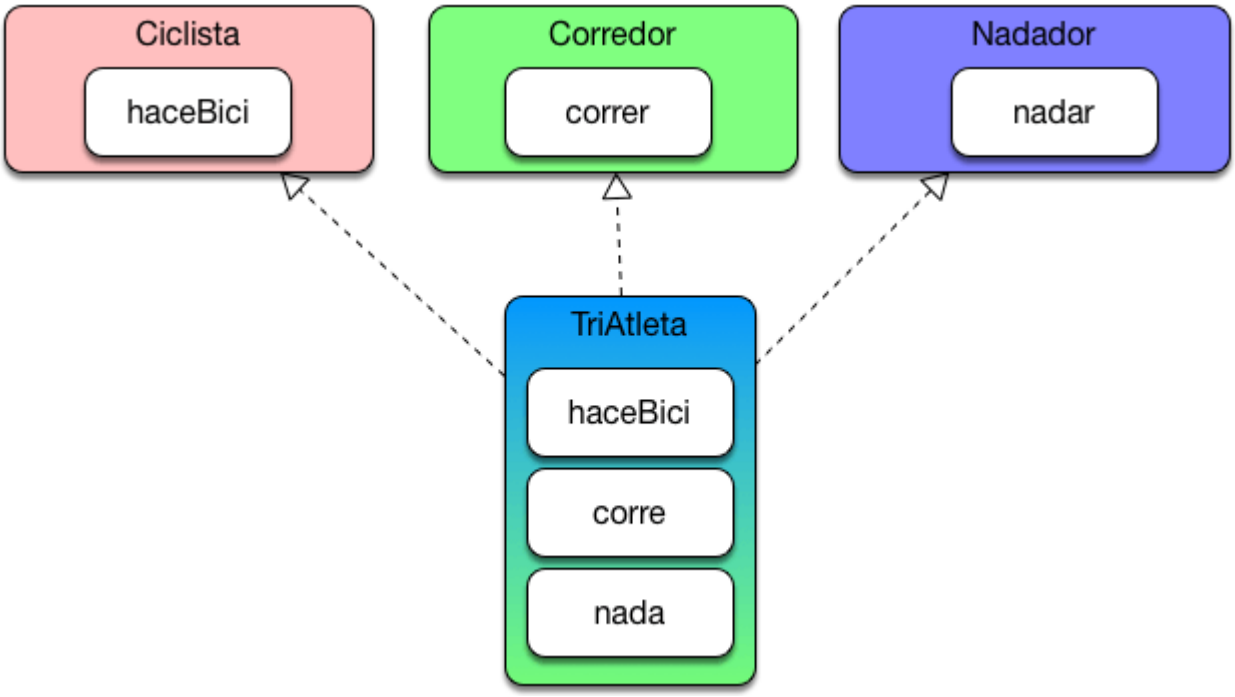
	public default void correr() {
		System.out.println("la persona corre");
	}
}
```

```
 package com.arquitecturajava;

public interface Nadador {

	public default void nadar() {
		System.out.println("la persona nada");
	}
}
```

Como podemos ver todos los conceptos de los que queremos extender los hemos definido como interfaces . A partir de Java 8 los interfaces soportan default methods .



Java Multiple Inheritance e interfaces

Hemos usado estos métodos para implementar la funcionalidad necesaria. Nos queda por construir la clase que implemente todos los interfaces:

```
package com.arquitecturajava;

public class TriAtleta implements Ciclista,Corredor,Nadador{

}
```

Una vez hecho esto construimos el programa principal y ejecutamos cada uno de los métodos que el Triatleta soporta.

```
package com.arquitecturajava;

public class Principal {

	public static void main(String[] args) {
		TriAtleta t= new TriAtleta();
		t.correr();
		t.nadar();
		t.haceBici();

	}

}
```

El resultado se ejecuta en la consola:



Acabamos de implementar una solución de herencia múltiple basada en el manejo de interfaces.

Otros artículos relacionados:

1. Java 8 interface static methods y reutilizacion
2. El concepto de Java 8 reference method
3. Utilizando Java 8 Predicate
4. Novedades de Java 8 Collections y Listas
5. Java default method Oracle