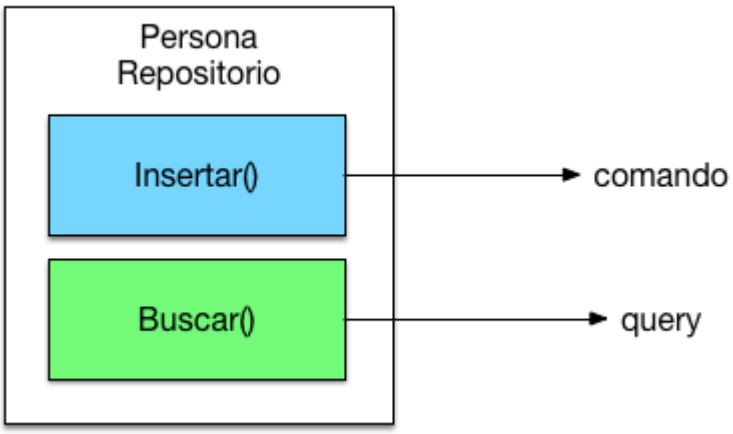
Command Query Separation o CQS es uno de los principios de diseño orientado a objeto que nos permite construir nuestras clases de forma que sean más sencillas de usar por parte de los desarrolladores. Este principio define que los métodos de una clase se deben dividir en dos grandes grupos , Consultas y Comandos. Las consultas son las encargadas de devolvernos un conjunto de datos y los Comandos son los encargados de modificar el estado del objeto. Este principio se puede ver de una forma clara cuando utilizamos por ejemplo capas de Repositorio.



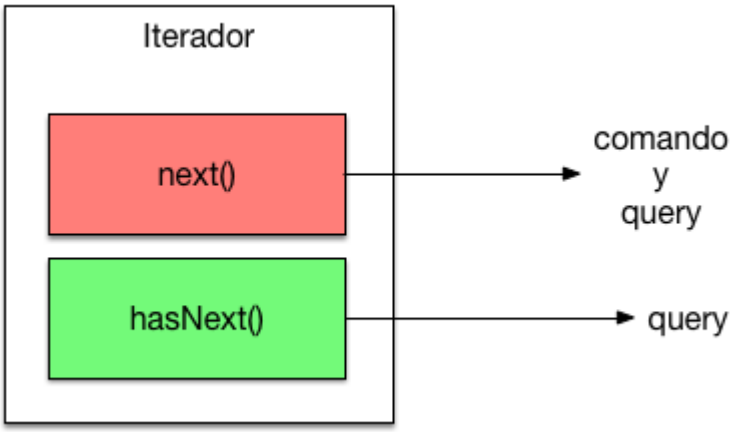
Cualquier consulta que se lance tipo busqueda devolverá un conjunto de resultados pero no modificará el estado del objeto mientras que los métodos tipo Insertar modificarán el estado del objeto pero no devolverán nada. De esta forma podremos diseñar unas clases que sean más sencillas de utilizar por parte de los desarrolladores. Cuando esto no sucede nos encontramos que las APIs que manejamos se complican y son más difíciles de entender.

Command Query Separation e Iteradores

Uno de los ejemplos clásicos de este tipo de problemática es el uso de iteradores en Java. Los iteradores se han usado siempre para recorrer una colección de elementos.

```
package com.arquitecturajava;

import java.util.ArrayList;
import java.util.Iterator;

public class PrincipalIteradores {

public static void main(String[] args) {

ArrayList<String> lista= new ArrayList<String>();
lista.add("primero");
lista.add("segundo");
lista.add("tercero");
Iterator<String> it= lista.iterator();

while(it.hasNext()) {

System.out.println(it.next());
}

}

}

 
```

Siempre le cuesta entender a los nuevos desarrolladores el funcionamiento del iterador. Esto es debido a que el método next() no cumple con el principio CQS ya que es un método de Consulta (nos devuelve un elemento) pero ademas cambia el estado del objeto actual ya que se mueve al siguiente item y por lo tanto actúa a la vez de Comando.



Apoyemonos siempre en este principio a la hora de diseñar nuestras clases de esta forma ganaremos en claridad en cuanto al diseño se refiere. Otros artículos relacionados : Java Singleton Pattern , Java FlyWeight CQR Martin Fowler