Examen de prácticas

Validar una Interfaznoinstanciada utilizando pruebas unitarias con un TestCase abstracto.

Se quiere realizar la prueba unitaria de una interfaz pero no tenemos forma de instanciarla aún y queremos que los resultados de esta prueba sean válidos para todas las posibles implementaciones de dicha interfaz.

Lo que tendremos que hacer para resolver el problema anterior es escribir “test cases” que capturen los requerimientos del comportamiento común de la interfaz mencionada pero que sean lo más generales posible, de tal forma que nos permitiesen probar futuras implementaciones que satisfagan los requerimientos sin replicar nunca el código de las pruebas, es decir, sin replicar jamás el código de los métodos testXXX().

Se pide programar un TestCase abstracto que, por ejemplo, declarará lo siguiente:

```
public abstract void setUp() throws Exception;
public abstract void instanciar () ;
public abstract void testAniadir() ;
public abstract void testhasNext() ;
public abstract void noelementos() ;
public abstract void eliminarnoexiste() ;
public abstract void eliminar() ;
```

Para el siguiente caso a probar: java.util.Iterator, que sirve para recorrer una colección en Java.

Un iterator sustituye a una instancia de Enumeration en el marco de trabajo Collections de Java. Los iteradores se diferencian de los enumeradores de Java en 2 aspectos principales:

los iteradores permiten al cliente que los llama retirar (remove()) elementos de la colección durante una iteración, además poseen una semántica bien definida

Se han mejorado los nombre de los métodos de la interfaz.

La siguiente interfaznoinstanciada es lo que habría que probar:

```
public interface Iterador <E> {
        public boolean hasnext();
        public <E>  E next();
        public void remove();
}
```

programando un test case concreto que extienda al TestCase anterior.

Se pide programar una clase iteradora genérica (Iteradora<E>) que implementa Iterador<E>, cuyos métodos nos sirven para escribir los “tests” concretos que nos piden en este ejercicio.

Ayuda: la siguiente interfaz es un miembro del marco de trabajo Java Collections, que puede servir como base para programar Iteradora<E>

public Interface Iterator<E>

hasNext() : este método devuleve “true” si la iteración posee más elementos

next(): devuelve el siguiente elemento (de tipo E) de la iteración

remove(): elimina el último elemento devuelto por el iterador de la colección actual
Excepciones que puede lanzar un código que implemente los métodos de la interfaz Iterator<E>:
E next(): NoSuchElementException: si la iteración no tiene ya más elementos
void remove(): UnsupportedOperationException: si la operación remove() no está soportada por este iterador
IllegalStateException: el método next() no ha sido llamado todavía, o el método remove() ya ha sido llamado, después de la última llamada del método next().