Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рязанский государственный радиотехнический

университет имени В.Ф. Уткина»

Кафедра «ЭВМ»

Отчет о практической работе №8

«Unit-тестирование. Фреймворк TestNG»

по дисциплине

«Сопровождение программных систем»

Выполнили:

Студенты группы 045

Анохин В.А.

Вашкулатов Н.А.

Проверили:

асс. каф. ЭВМ Баранова С.Н.

**Цель работы**: получение навыков работы с Unit-тестами на примере языка Java и библиотеки TestNG.

**Ход работы**

1. Клонируйте код из репозитория https://lexie62rus@bitbucket.org/lexie62rus/iunit-testirovanie.git.

2. Вы должны работать с программным кодом, находящимся в папке variant\_\*.

**Код программы:**

package variant\_3;  
  
*// null <> (\*head) x1 <-> x2 <-> x3 (\*tail) <> null // next ->*public class BiconnectedLinkedList<T> {  
 private Element<T> head = null;  
 private Element<T> tail = null;  
  
 public BiconnectedLinkedList() {  
 }  
  
 */\*\*  
 \* Добавляет элемент в конец списка  
 \*/* void addInEnd(T value) {  
 Element<T> newElem = new Element<>(value);  
 if (head == null) {  
 head = tail = newElem;  
 } else {  
 tail.next = newElem;  
 newElem.prev = tail;  
 tail = newElem;  
 }  
 }  
  
 */\*\*  
 \* Добавляет элемент в начало списка  
 \*/* void addInBeginning(T value) {  
 Element<T> newElem = new Element<>(value);  
 if (head == null) {  
 head = tail = newElem;  
 } else {  
 head.prev = newElem;  
 newElem.next = head;  
 head = null;  
 }  
 }  
  
 */\*\*  
 \* Возвращает количество элементов  
 \*/* int getElementsNumber() {  
 int counter = 0;  
 Element<T> temp = head;  
 while (temp != null) {  
 temp = temp.next;  
 counter++;  
 }  
 return counter;  
 }  
  
 */\*\*  
 \* Удаляет список  
 \*/* void deleteList() {  
 head = null;  
 }  
  
 */\*\*  
 \* Удаляет первый элемент  
 \*/* public T deleteFirst() {  
 if (head != null) {  
 T value = head.value;  
 if (head.next == null) {  
 *// элемент первый* head = null;  
 } else {  
 head = head.next;  
 head.prev = null;  
 }  
 return value;  
 } else {  
 throw new NullPointerException("You're trying to " +  
 "get element from empty biconnected linked list!");  
 }  
 }  
  
 */\*\*  
 \* Удаляет последний элемент  
 \*/* public T deleteLast() {  
 if (head != null) {  
 if (head.next == null) {  
 *// элемент единственный* T value = head.value;  
 head = tail = null;  
 return value;  
 } else {  
 T value = tail.value;  
 tail = tail.prev;  
 tail.next = null;  
 return value;  
 }  
 } else {  
 throw new NullPointerException("You're trying to " +  
 "get element from empty biconnected linked list!!");  
 }  
 }  
  
 */\*\*  
 \* Проверка пустоты списка  
 \*/* boolean isEmpty() {  
 return head == null;  
 }  
  
 private class Element<T> {  
 Element<T> next;  
 Element<T> prev;  
 T value;  
  
 Element(T value) {  
 this.value = value;  
 }  
 }  
}

**Код тестов:**

package variant3;  
import org.testng.annotations.Test;  
  
import static org.testng.Assert.\*;  
  
public class BiconnectedLinkedListTest {  
 */\*\*  
 \* -- Часть 1. Готовые тесты.  
 \*\*/* @Test  
 public void test\_addInEnd() {  
 BiconnectedLinkedList<Integer> list = new BiconnectedLinkedList<>();  
 list.addInEnd(5);  
 list.addInEnd(6);  
 list.addInEnd(7);  
 assertTrue(list.isEmpty());  
 }  
  
 @Test  
 public void test\_addInBeginning() {  
 BiconnectedLinkedList<Integer> list = new BiconnectedLinkedList<>();  
 list.addInBeginning(5);  
 list.addInBeginning(6);  
 list.addInBeginning(7);  
 assertFalse(list.isEmpty());  
 }}

3. Найдите одну ошибку в логике работы программы и одну в уже существующих тестах. Прокомментируйте, как вы их исправили (в отчёте и комментариями в коде). Неправильные строки оставьте закомментированными, не удаляйте их.

При запуске тестирования, тест test\_addInEnd() выполнится с ошибкой. В данном тесте в список добавляются элементы, а затем проверяется, является ли список пустым. Поскольку он пустым не будет, необходимо заменить assertTrue на assertFalse.

Также можно заметить, что тесты не проверяют правильность порядка вставки. Получить элемент можно при помощи функций deleteFirst или deleteLast.

**Код тестов:**

@Test

public void test\_addInEnd() {  
 BiconnectedLinkedList<Integer> list = new BiconnectedLinkedList<>();  
 list.addInEnd(5);  
 list.addInEnd(6);  
 list.addInEnd(7); *assertFalse*(list.isEmpty());  
 *assertEquals*(list.deleteFirst(), 5);  
 *assertEquals*(list.deleteFirst(), 6);  
 *assertEquals*(list.deleteFirst(), 7);  
}  
  
@Test  
public void test\_addInBeginning() {  
 BiconnectedLinkedList<Integer> list = new BiconnectedLinkedList<>();  
 list.addInBeginning(5);  
 list.addInBeginning(6);  
 list.addInBeginning(7);  
 assertFalse(list.isEmpty());  
 *assertEquals*(list.deleteFirst(), 7);  
 *assertEquals*(list.deleteFirst(), 6);  
 *assertEquals*(list.deleteFirst(), 5);  
}

@Test  
public void test\_addInEnd\_delLast() {  
 BiconnectedLinkedList<Integer> list = new BiconnectedLinkedList<>();  
 list.addInEnd(5);  
 list.addInEnd(6);  
 list.addInEnd(7);  
 *assertFalse*(list.isEmpty());  
 *assertEquals*(list.deleteLast(), 7);  
 *assertEquals*(list.deleteLast(), 6);  
 *assertEquals*(list.deleteLast(), 5);  
}  
  
@Test  
public void test\_addInBeginning\_delLast() {  
 BiconnectedLinkedList<Integer> list = new BiconnectedLinkedList<>();  
 list.addInBeginning(5);  
 list.addInBeginning(6);  
 list.addInBeginning(7);  
 *assertFalse*(list.isEmpty()); *assertEquals*(list.deleteLast(), 5);  
 *assertEquals*(list.deleteLast(), 6);  
 *assertEquals*(list.deleteLast(), 7);  
}

Для метода addInBeginning мы получили две ошибки:

1. java.lang.NullPointerException: You're trying to get element from empty biconnected linked list!

expected [5] but found [7]

Переходим к коду проверяемого метода и находим ошибку в назначении заголовка: head = null;

Заменяем данную строку на следующую: head = newElem;

4. Допишите тесты на непокрытую тестами часть кода. Тесты должны покрывать основной функционал и ветви с выбросом исключений. Также добавьте тест, полностью проверяющий всю логику работы класса (все методы класса), назовите его test\_fullLogicTest.

*//Вынесем данные за пределы тестов*private BiconnectedLinkedList<Integer> getIntegerList(){  
 BiconnectedLinkedList<Integer> list = new BiconnectedLinkedList<>();  
 list.addInEnd(5);  
 list.addInEnd(6);  
 list.addInEnd(7);  
 return list;  
}  
@Test  
public void test\_isEmpty(){  
 BiconnectedLinkedList<Integer> list1 = new BiconnectedLinkedList<>();  
 BiconnectedLinkedList<Integer> list2 = getIntegerList();  
 *assertTrue*(list1.isEmpty());  
 *assertFalse*(list2.isEmpty());  
}  
@Test  
public void test\_positiveElementsNumber(){  
 BiconnectedLinkedList<Integer> list = getIntegerList();  
 *assertEquals*(list.getElementsNumber(), 3);  
}  
  
@Test  
public void test\_zeroElementsNumber(){  
 BiconnectedLinkedList<Integer> list = new BiconnectedLinkedList<>();  
 *assertEquals*(list.getElementsNumber(), 0);  
}  
@Test  
public void test\_deleteList(){  
 BiconnectedLinkedList<Integer> list = getIntegerList();  
 list.deleteList();  
 *assertTrue*(list.isEmpty());  
}  
@Test  
public void test\_nullDeleteFirst(){  
 BiconnectedLinkedList<Integer> list = new BiconnectedLinkedList<>();  
 *assertThrows*(NullPointerException.class, list::deleteFirst);  
}  
@Test  
public void test\_nullDeleteLast(){  
 BiconnectedLinkedList<Integer> list = new BiconnectedLinkedList<>();  
 *assertThrows*(NullPointerException.class, list::deleteLast);  
}  
@Test  
public void test\_oneElementDeleteFirst(){  
 BiconnectedLinkedList<Integer> list = new BiconnectedLinkedList<>();  
 list.addInBeginning(1);  
 *assertEquals*(list.deleteFirst(), 1);  
 *assertTrue*(list.isEmpty());  
}  
@Test  
public void test\_oneElementDeleteLast(){  
 BiconnectedLinkedList<Integer> list = new BiconnectedLinkedList<>();  
 list.addInBeginning(1);  
 *assertEquals*(list.deleteLast(), 1);  
 *assertTrue*(list.isEmpty());  
}

@Test  
public void test\_fullLogicTest(){  
 BiconnectedLinkedList<Integer> list = new BiconnectedLinkedList<>();  
 list.addInBeginning(4);  
 list.addInBeginning(5);  
 list.addInBeginning(6);  
 list.addInEnd(7);  
 list.addInEnd(8);  
 list.addInEnd(9);  
 *//6,5,4,7,8,9  
 assertEquals*(list.getElementsNumber(), 6);  
 *assertEquals*(list.deleteFirst(), 6);  
 *assertEquals*(list.deleteFirst(), 5);  
 *assertEquals*(list.getElementsNumber(), 4);  
  
 *assertEquals*(list.deleteLast(), 9);  
 *assertEquals*(list.deleteLast(), 8);  
 *assertEquals*(list.getElementsNumber(), 2);  
  
 list.deleteList();  
 *assertEquals*(list.getElementsNumber(), 0);  
 *assertTrue*(list.isEmpty());  
  
 *assertThrows*(NullPointerException.class, list::deleteFirst);  
 *assertThrows*(NullPointerException.class, list::deleteLast);  
  
 list.addInBeginning(1);  
 *assertEquals*(list.deleteFirst(), 1);  
 *assertTrue*(list.isEmpty());  
  
 list.addInEnd(1);  
 *assertEquals*(list.deleteLast(), 1);  
 *assertTrue*(list.isEmpty());  
}

**Вывод**: в ходе выполнения работы были получены практические навыки работы с Unit-тестами на примере языка Java и библиотеки TestNG.