Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет» Кафедра ИИТ

Лабораторная работа №11

По дисциплине «СПП» за 6 семестр

Выполнил:

Студент группы ПО-3

Ковалёва А. И.

Проверил:

Крощенко А. А.

Брест 2021

**Вариант 12**

**Цель**: освоить приемы тестирования кода на примере использования библиотеки JUnit.

**Задание 1**

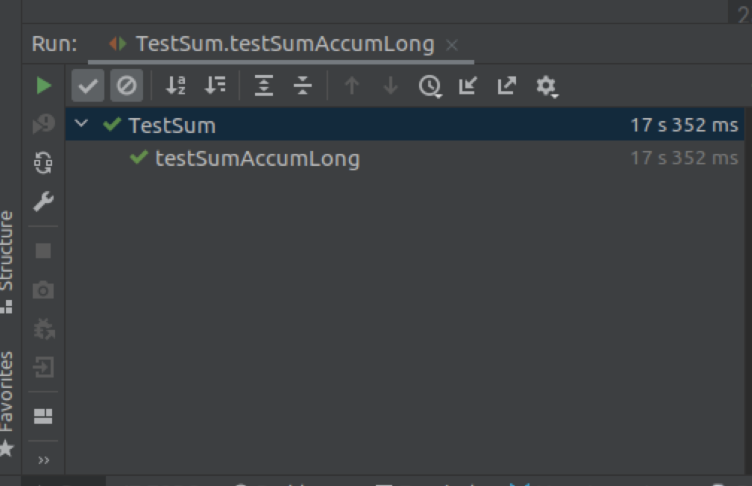
* Создаете новый класс и скопируйте код класса Sum;
* Создаете тестовый класс SumTest;
* Напишите тест к методу Sum.accum и проверьте его исполнение. Тест должен проверять рабо- тоспособность функции accum.

**Текст программы:**

**Sum**  
  
**public class** Sum {  
 **public static int** accum ( **int** ... values ) {  
 **int** result = 0;  
 **for** ( **int** i = 0; i < values . **length** ; i ++) {  
 result += values [ i ];  
 }  
 **return** result ;  
 }  
  
 **public static long** accumLong ( **int** ... values ) {  
 **long** result = 0;  
 **for** ( **int** i = 0; i < values . **length** ; i ++) {  
 result += values [ i ];  
 }  
 **return** result ;  
 }  
}

**TestSum**

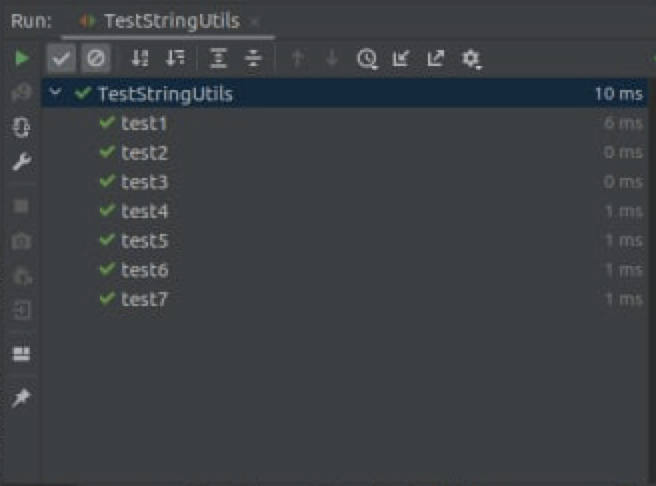
**import** org . junit .\*; *// Импорт всех основных классов и аннотаций JUnit***import** java.math.BigInteger;  
  
**import static** org . junit . Assert .\*;  
  
**public class** TestSum {  
  
 @Test  
 **public void** testSumAccumLong(){  
  
 **int** arr[] = **new int**[Integer.***MAX\_VALUE***/9];  
 BigInteger sum = BigInteger.***ZERO*** ;  
 **for** (**int** i=0; i< arr.**length**;i++) {  
 arr[i] = (**int**) (Integer.***MAX\_VALUE***);  
 sum = sum.add(BigInteger.*valueOf*(arr[i]));  
 }  
 *assertEquals*(sum.longValue(), Sum.*accumLong*(arr));  
 }  
 @Test  
 **public void** testSumAccum(){  
  
 *assertEquals*(1+2+3+4+5+6, Sum.*accum*(1,2,3,4,5,6));  
 }  
  
 }



**Задание 2**

Написать тесты к методу, а затем реализовать сам метод по заданной спецификации.

**import** org . junit .\*;  
**import static** org . junit . Assert .\*;  
  
**public class** TestStringUtils {  
*// Спецификация метода:  
//  
// 1 keep (null , null ) = NullPointerException  
// 2 keep (null , \*) = null  
// 3 keep ("", \*) = ""  
// 4 keep (\* , null ) = ""  
// 5 keep (\* , "") = ""  
// 6 keep (" hello ", "hl") = " hll "  
// 7 keep (" hello ", "le") = " ell "* @Test (expected = NullPointerException.**class** )  
 **public void** test1(){  
 *assertEquals*(**""**, StringUtils.*keep*(**null** , **null** ));  
 }  
  
 @Test  
 **public void** test2(){  
 *assertEquals*(**null**,  
 StringUtils.*keep*(**null** , **"\*"**));  
 }  
  
 @Test  
 **public void** test3(){  
 *assertEquals*(**""**, StringUtils.*keep*(**""**, **"\*"**));  
 }  
  
 @Test  
 **public void** test4(){  
 *assertEquals*(**""**, StringUtils.*keep*(**"\*"**, **null** ));  
 }  
  
 @Test  
 **public void** test5(){  
 *assertEquals*(**""**, StringUtils.*keep*(**"\*"** , **""**));  
 }  
  
 @Test  
 **public void** test6(){  
 *assertEquals*(**" hll "**, StringUtils.*keep*(**" hello "**, **"hl"**));  
 }  
  
 @Test  
 **public void** test7(){  
 *assertEquals*(**" ell "**, StringUtils.*keep*(**" hello "**, **"le"**));  
 }  
}

****

**Задание 3**

Поиск ошибок, отладка и тестирование классов.

**Queue**

**package** queue;  
  
**import** java.util.NoSuchElementException;**public class** Queue<Item> {  
 **private int N**; *// number of elements on queue* **private** Node **first**; *// beginning of queue* **private** Node **last**; *// end of queue  
  
 // helper linked list class* **private class** Node {  
 **private** Item **item**;  
 **private** Node **next**;  
 }  
**public** Queue() {  
 **first** = **null**;  
 **last** = **null**;  
 **N** = 0;  
 **assert** check();  
 }  
  
 */\*\*  
 \* Is the queue empty?  
 \*/* **public boolean** isEmpty() {  
 **return first** == **null**; *// was return first != null;* }  
  
 */\*\*  
 \* Return the number of items in the queue.  
 \*/* **public int** size() {  
 **return N**;  
 }  
  
 */\*\*  
 \* Return the item least recently added to the queue.  
 \*  
 \** ***@throws*** *java.util.NoSuchElementException if queue is empty.  
 \*/* **public** Item peek() {  
 *//* ***FIXME throw exception if queue is Empty.*** *//* ***TODO implement method* if**(isEmpty()){  
 **throw new** NoSuchElementException();  
 }  
  
 **return first**.**item**;  
 }  
  
 */\*\*  
 \* Add the item to the queue.  
 \*/* **public void** enqueue(Item item) {  
  
 Node oldlast = **last**;  
 ++**N**;  
 **last** = **new** Node();  
 **last**.**item** = item;  
 **last**.**next** = **null**;  
  
  
 **if** (isEmpty()) {  
 **first** = **last**;  
 } **else** {  
 oldlast.**next** = **last**;  
 }  
  
 **assert** check();  
 }  
  
 */\*\*  
 \* Remove and return the item on the queue least recently added.  
 \*  
 \** ***@throws*** *java.util.NoSuchElementException if queue is empty.  
 \** ***@return*** *\*/* **public** Item dequeue() {  
 *//* ***FIXME throw exception if queue is Empty.* if**(isEmpty()){  
 **throw new** NoSuchElementException();  
 }  
 Item item = **first**.**item**;  
 **first** = **first**.**next**;  
 --**N**;  
 **if** (isEmpty()) {  
 **last** = **null**; *// to avoid loitering* }  
 **assert** check();  
 **return** item;  
 }  
  
 */\*\*  
 \* Return string representation.  
 \*/* **public** String toString() {  
 StringBuilder s = **new** StringBuilder();  
 **for** (Node x = **first**; x != **null**; x = x.**next**) {  
 s.append(x.**item** + **" "**);  
 }  
  
 **return** s.toString();  
 }  
  
 *// check internal invariants* **private boolean** check() {  
 **if** (**N** == 0) {  
 **if** (**first** != **null**) {  
 **return false**;  
 }  
 **if** (**last** != **null**) {  
 **return false**;  
 }  
 } **else if** (**N** == 1) {  
 **if** (**first** == **null** || **last** == **null**) {  
 **return false**;  
 }  
 **if** (**first** != **last**) {  
 **return false**;  
 }  
 **if** (**first**.**next** != **null**) {  
 **return false**;  
 }  
 } **else** {  
 **if** (**first** == **last**) {  
 System.***out***.println((**first** == **last**) + **"first == last"**);  
 **return false**;  
 }  
 **if** (**first**.**next** == **null**) {  
 **return false**;  
 }  
 **if** (**last**.**next** != **null**) {  
 **return false**;  
 }  
  
 *// check internal consistency of instance variable N* **int** numberOfNodes = 0;  
 **for** (Node x = **first**; x != **null**; x = x.**next**) {  
 numberOfNodes++;  
 }  
 **if** (numberOfNodes != **N**) {  
 **return false**;  
 }  
 *// check internal consistency of instance variable last* Node lastNode = **first**;  
 **while** (lastNode.**next** != **null**) {  
 lastNode = lastNode.**next**;  
 }  
 **if** (**last** != lastNode) {  
 **return false**;  
 }  
 }  
  
 **return true**;  
 }  
}

**QueueClient**

**package** queue;  
  
**import** java.util.Scanner;  
  
**public class** QueueClient {  
  
 */\*\*  
 \* A test client.  
 \*/* **public static void** main(String[] args) {  
 Queue<String> q = **new** Queue<String>();  
  
 Scanner scanner = **new** Scanner(System.***in***);  
  
 **while** (scanner.hasNext()) {  
 String item = scanner.next();  
 **if** (!item.equals(**"-"**)) {  
 q.enqueue(item);  
 } **else if** (!q.isEmpty()) {  
 System.***out***.println(q.dequeue() + **" "**);  
 }  
 }  
 System.***out***.println(q.size());  
 }  
}

**QueueTest1**

**package** queue;  
  
**import** org . junit .\*; *// Импорт всех основных классов и аннотаций JUnit***import** java.util.NoSuchElementException;  
  
**import static** org . junit . Assert .\*;  
  
**public class** QueueTest1 {  
 Queue<String> **stringQueue**;  
  
 @Before  
 **public void** setUpBeforTest () {  
 **stringQueue** = **new** Queue<String>();  
 }  
  
 @After  
 **public void** setUpAfterTest () {  
  
 **stringQueue** = **null**;  
 }  
  
 @Test  
 **public void** testIsEmpty() {  
 *assertEquals*(**true**, **stringQueue**.isEmpty());  
 }  
  
  
  
 @Test (expected = NoSuchElementException. **class** )  
 **public void** testPeekNoSuchElementException(){  
 **stringQueue**.peek();  
 }  
  
 @Test (expected = NoSuchElementException. **class** )  
 **public void** testDequeue(){  
 **stringQueue**.dequeue();  
 }  
}

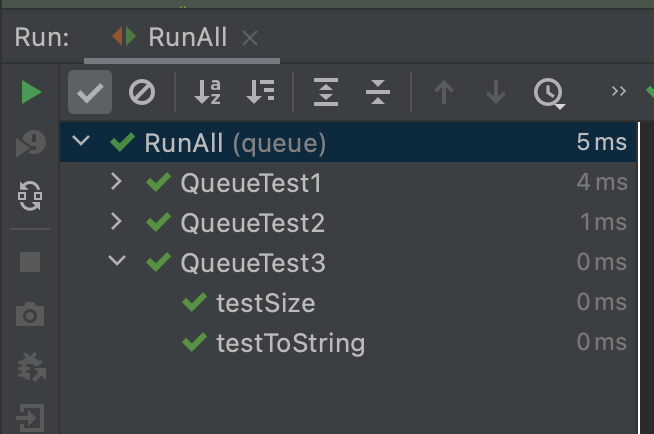
**QueueTest2**

**package** queue;  
  
**import** org . junit .\*; **import** java.util.NoSuchElementException;  
  
**import static** org . junit . Assert .\*;  
  
**public class** QueueTest2 {  
 Queue<String> **stringQueue**;  
  
 @Before  
 **public void** setUpBeforTest () {  
 **stringQueue** = **new** Queue<String>();  
 **stringQueue**.enqueue(**"AAA"**);  
 }  
  
 @After  
 **public void** setUpAfterTest () {  
  
 **stringQueue** = **null**;  
 }  
  
 @Test  
 **public void** testPeek() {  
 *assertEquals*(**"AAA"**, **stringQueue**.peek());  
 }  
  
 @Test  
 **public void** testEnqueue() {  
 *assertEquals*(**"AAA"**, **stringQueue**.peek());  
 }  
}

**QueueTest3**

**package** queue;  
  
**import** org . junit .\*; */***import** java.util.NoSuchElementException;  
  
**import static** org . junit . Assert .\*;  
  
**public class** QueueTest3 {  
 Queue<String> **stringQueue**;  
  
 @Before  
 **public void** setUpBeforTest () {  
 **stringQueue** = **new** Queue<String>();  
 **stringQueue**.enqueue(**"AAA"**);  
 **stringQueue**.enqueue(**"AAA"**);  
 **stringQueue**.enqueue(**"AAA"**);  
 }  
  
 @After  
 **public void** setUpAfterTest () {  
  
 **stringQueue** = **null**;  
 }  
  
 @Test  
 **public void** testSize() {  
 *assertEquals*(3, **stringQueue**.size());  
 }  
  
 @Test  
 **public void** testToString() {  
 *assertEquals*(**"AAA AAA AAA "**, **stringQueue**.toString());  
 }  
}

**RunAll**

**package** queue;  
  
**import** org . junit . runner . RunWith ;  
**import** org . junit . runners . Suite ;  
**import** org . junit . runners . Suite . SuiteClasses ;  
@RunWith ( Suite . **class** ) *// Запустить класс как тестовый набор*@SuiteClasses ({ *// Список тестовых классов в наборе для запуска* QueueTest1 .**class** ,  
 QueueTest2 .**class** ,  
 QueueTest3 .**class** ,  
})  
**class** RunAll {  
  
}

**Вывод:** освоила приемы тестирования кода на примере использования библиотеки JUnit.