**Модульное тестирование**

Цель работы:

1. Изучение назначения и задач модульного тестирования.

2. Программная реализация тестов, производящих модульное тестирование алгоритма пирамидальной сортировки из курса лабораторных работ

*// Реализация пирамидальной сортировки на Java*public class HeapSort  
{  
 public void sort(int arr[])  
 {  
 int n = arr.length;  
  
 *// Построение кучи (перегруппируем массив)* for (int i = n / 2 - 1; i >= 0; i--)  
 heapify(arr, n, i);  
  
 *// Один за другим извлекаем элементы из кучи* for (int i=n-1; i>=0; i--)  
 {  
 *// Перемещаем текущий корень в конец* int temp = arr[0];  
 arr[0] = arr[i];  
 arr[i] = temp;  
  
 *// Вызываем процедуру heapify на уменьшенной куче* heapify(arr, i, 0);  
 }  
 }  
  
 *// Процедура для преобразования в двоичную кучу поддерева с корневым узлом i, что является  
// индексом в arr[]. n - размер кучи* void heapify(int arr[], int n, int i)  
 {  
 int largest = i; *// Инициализируем наибольший элемент как корень* int l = 2\*i + 1; *// левый = 2\*i + 1* int r = 2\*i + 2; *// правый = 2\*i + 2  
  
 // Если левый дочерний элемент больше корня* if (l < n && arr[l] > arr[largest])  
 largest = l;  
  
 *// Если правый дочерний элемент больше, чем самый большой элемент на данный момент* if (r < n && arr[r] > arr[largest])  
 largest = r;  
 *// Если самый большой элемент не корень* if (largest != i)  
 {  
 int swap = arr[i];  
 arr[i] = arr[largest];  
 arr[largest] = swap;  
  
 *// Рекурсивно преобразуем в двоичную кучу затронутое поддерево* heapify(arr, n, largest);  
 }  
 }  
  
 */\* Вспомогательная функция для вывода на экран массива размера n \*/* static void printArray(int arr[])  
 {  
 int n = arr.length;  
 for (int i=0; i<n; ++i)  
 System.*out*.print(arr[i]+" ");  
 System.*out*.println();  
 }  
  
 *// Управляющая программа* public static void main(String args[])  
 {  
 int arr[] = {12, 11, 13, 5, 6, 7};  
 int n = arr.length;  
  
 HeapSort ob = new HeapSort();  
 ob.sort(arr);  
  
 System.*out*.println("Sorted array is");  
 *printArray*(arr);  
 }  
}

Тесты:

import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.\*;  
class HeapSortTest {  
  
  
 @Test  
 void printArray()  
 {  
 int arr[] = {12, 11, 13, 5, 6, 7};  
 int n = arr.length;  
 for (int i=0; i<n; ++i)  
 System.*out*.print(arr[i]+" ");  
 System.*out*.println();  
 *assertNotNull*(arr);  
  
 }  
 @Test  
 void sort()  
 {  
 int arr[] = {12, 11, 13, 5, 6, 7};  
 int n = arr.length;  
 *assertTrue*(arr.length>0);  
  
  
 *// Построение кучи (перегруппируем массив)* for (int i = n / 2 - 1; i >= 0; i--)  
 heapify(arr, n, i);  
  
 *// Один за другим извлекаем элементы из кучи* for (int i=n-1; i>=0; i--)  
 {  
 *// Перемещаем текущий корень в конец* int temp = arr[0];  
 arr[0] = arr[i];  
 arr[i] = temp;  
  
 *// Вызываем процедуру heapify на уменьшенной куче* heapify(arr, i, 0);  
 }  
  
 }  
  
  
@Test  
 void heapify()  
 {  
 int n = 2;  
 int i = 3;  
 int arr[] = {12, 11, 13, 5, 6, 7};  
 int largest = i; *// Инициализируем наибольший элемент как корень* int l = 2\*i + 1; *// левый = 2\*i + 1* int r = 2\*i + 2; *// правый = 2\*i + 2  
 assertNotEquals*(l,r);  
 *// Если левый дочерний элемент больше корня* if (l < n && arr[l] > arr[largest])  
 largest = l;  
  
  
 *// Если правый дочерний элемент больше, чем самый большой элемент на данный момент* if (r < n && arr[r] > arr[largest])  
 largest = r;  
 *// Если самый большой элемент не корень* if (largest != i)  
 {  
 int swap = arr[i];  
 arr[i] = arr[largest];  
 arr[largest] = swap;  
  
 *// Рекурсивно преобразуем в двоичную кучу затронутое поддерево* heapify(arr, n, largest);  
 }  
 }  
  
  
  
 void heapify(int arr[], int n, int i)  
 {  
  
 int largest = i; *// Инициализируем наибольший элемент как корень* int l = 2\*i + 1; *// левый = 2\*i + 1* int r = 2\*i + 2; *// правый = 2\*i + 2  
 assertNotEquals*(l,r);  
 *// Если левый дочерний элемент больше корня* if (l < n && arr[l] > arr[largest])  
 largest = l;  
  
  
 *// Если правый дочерний элемент больше, чем самый большой элемент на данный момент* if (r < n && arr[r] > arr[largest])  
 largest = r;  
 *// Если самый большой элемент не корень* if (largest != i)  
 {  
 int swap = arr[i];  
 arr[i] = arr[largest];  
 arr[largest] = swap;  
  
 *// Рекурсивно преобразуем в двоичную кучу затронутое поддерево* heapify(arr, n, largest);  
 }  
 }  
}

Итоги:

Было проведено тестирование модуля “PrintArray”, его работа проверялась подачей на метод массива. Модуль работоспособен.

Также был добавлен тест на проверку существования массива через тест assertNotNull(arr);

В методе sort() проверялась длинна массива, на соответствие минимальному размеру

assertTrue(arr.length>0);

В методе heapify() проверялись левые и правые элементы массива, от основного наибольшего.

Тест assertNotEquals(l,r);