**Software Testing**

**Lab 1: Debugging**

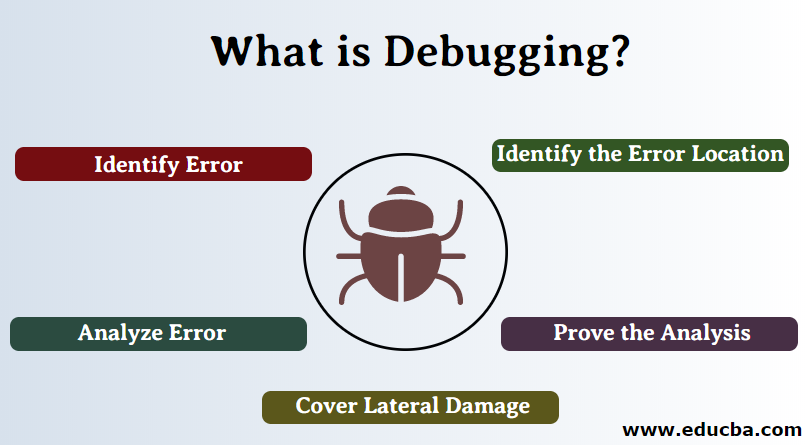
**Kodirkulov Zhengis 22MD0242**

**Telmanov Merlan 22MD0509**

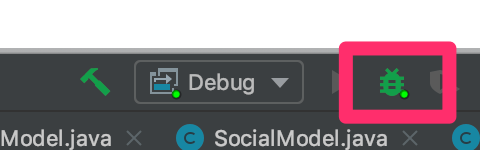
**Ospan Bekbolatabylay 22MD0373**

**Tileov Maksat 22MD0291**

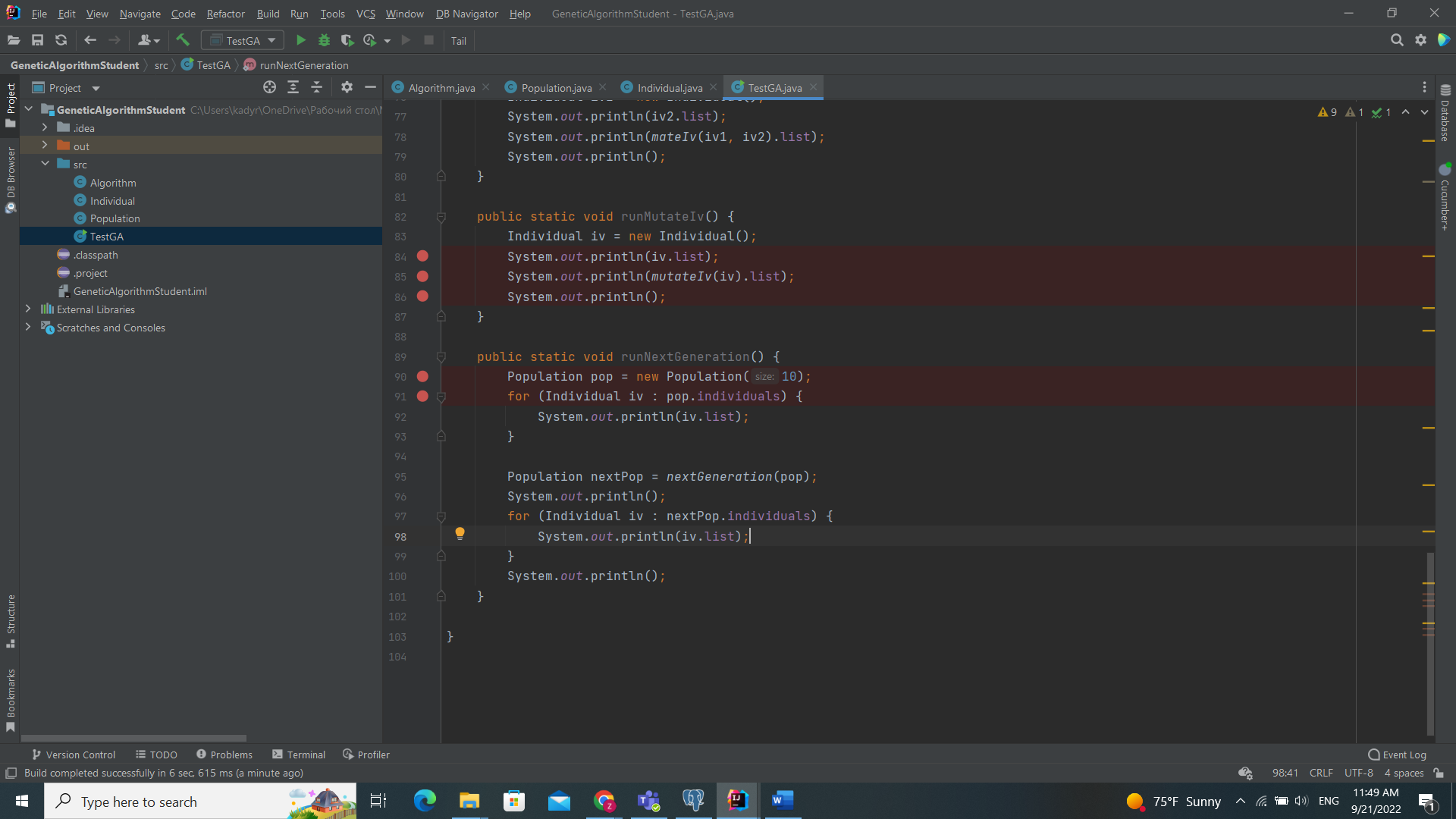
1. **Introduction**
2. **Some useful vocabulary:** 
   * Debugging



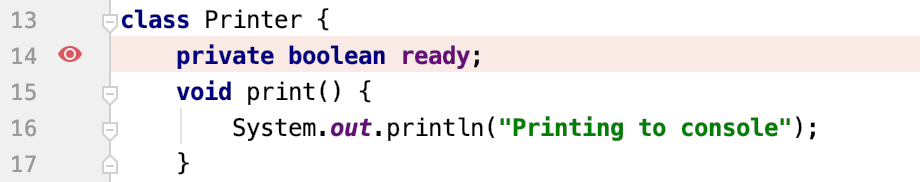
* Debugger



* Breakpoints

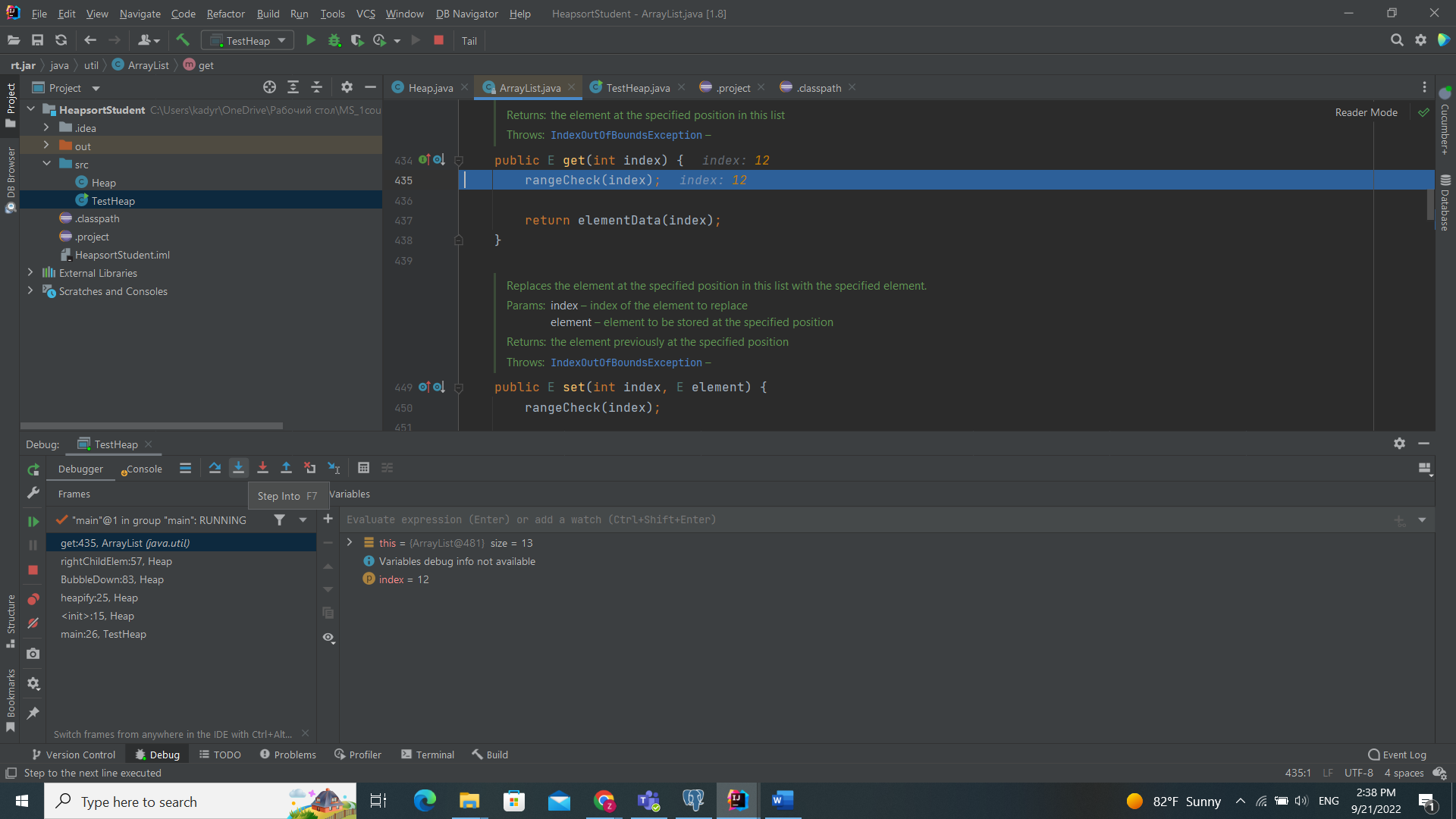


* Watchpoints

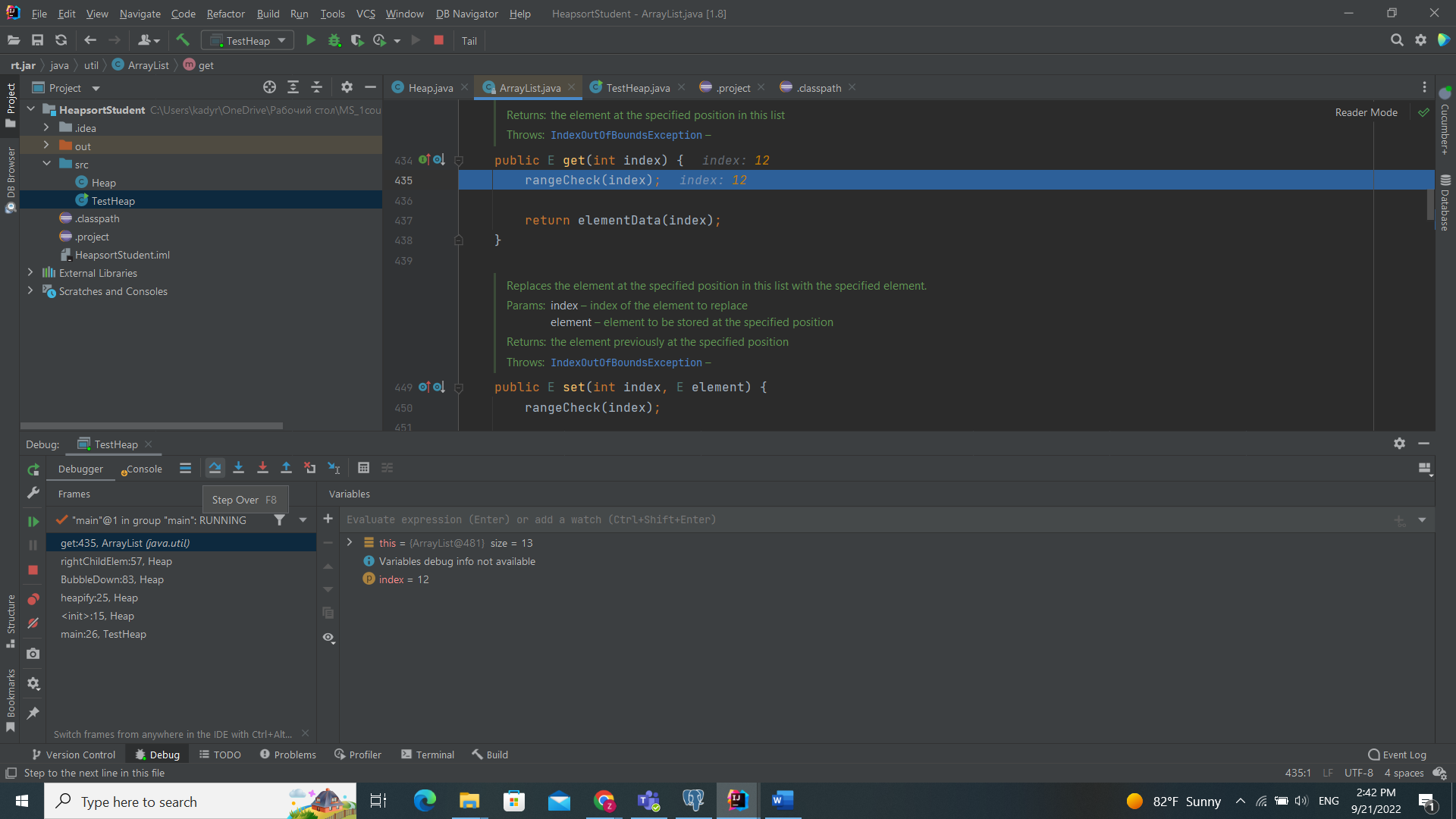


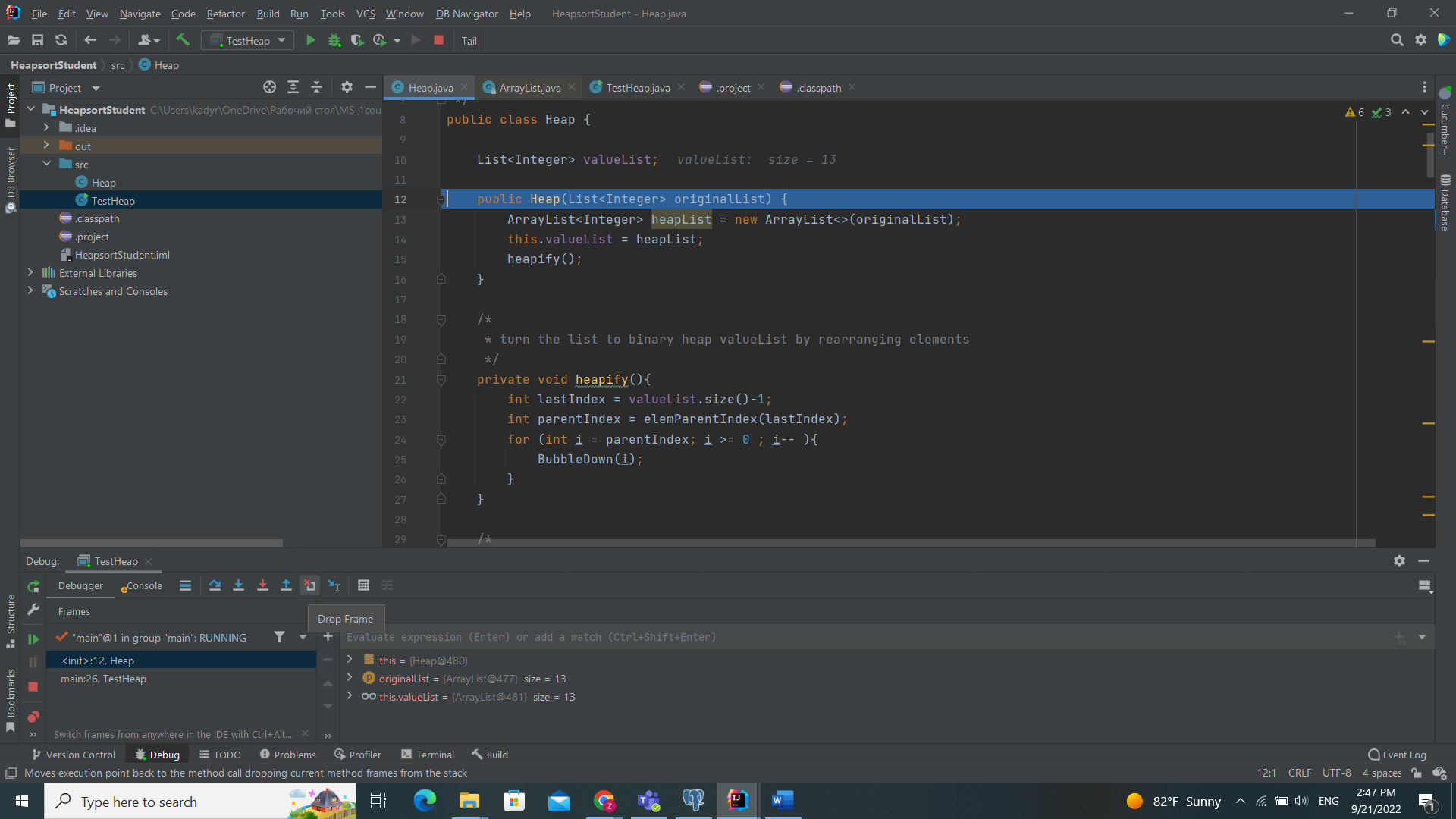
1. **Eclipse and IntelliJ buttons and meanings: Here I choose IntelliJ idea**

* **Step into change next line with function**

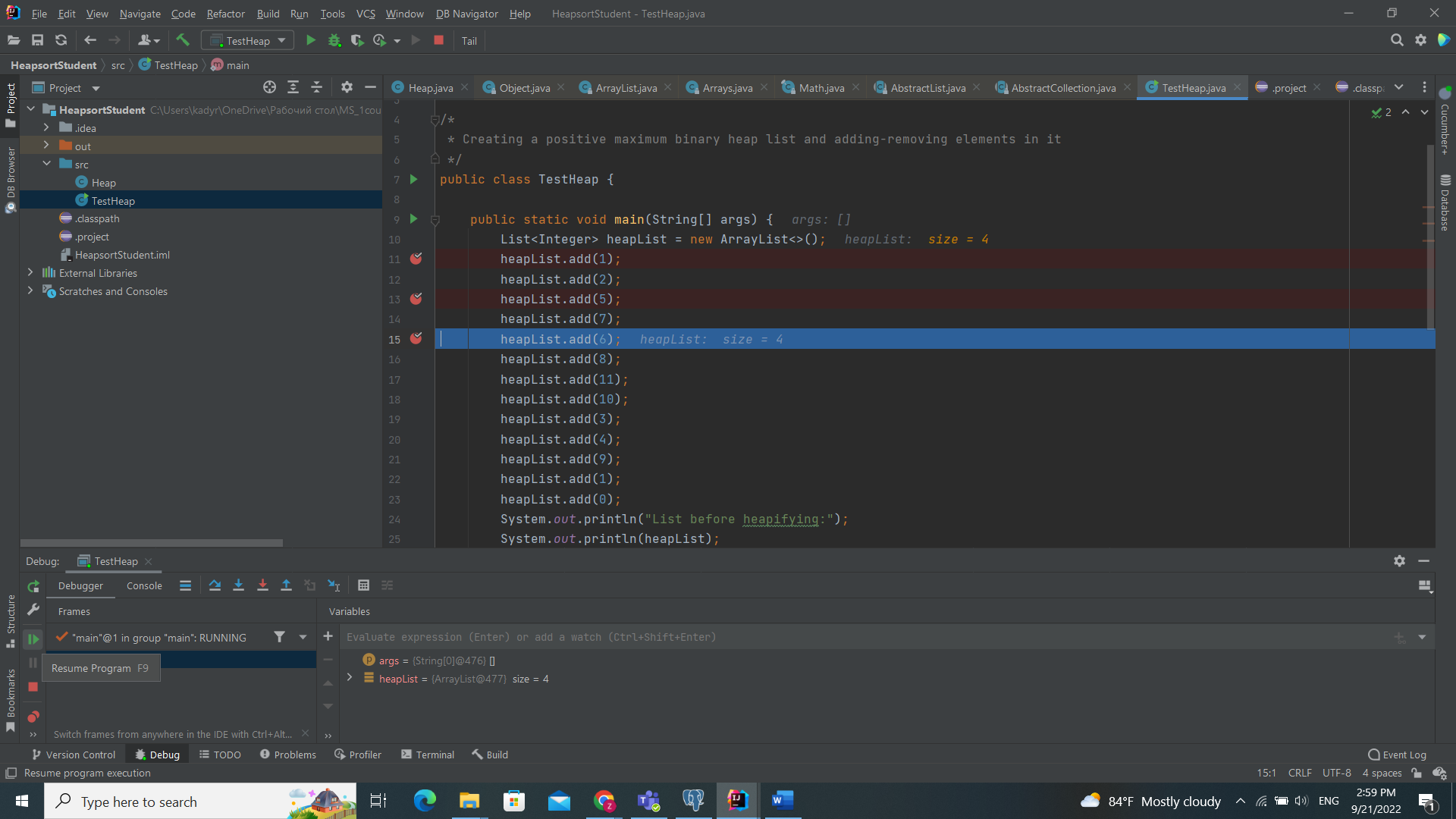


* **Step over change next line without function**



* **Drop frame - this go back function you are in**

* **Resume- change breakpoints or jump another point**



1. **Homework Tasks and Reporting**

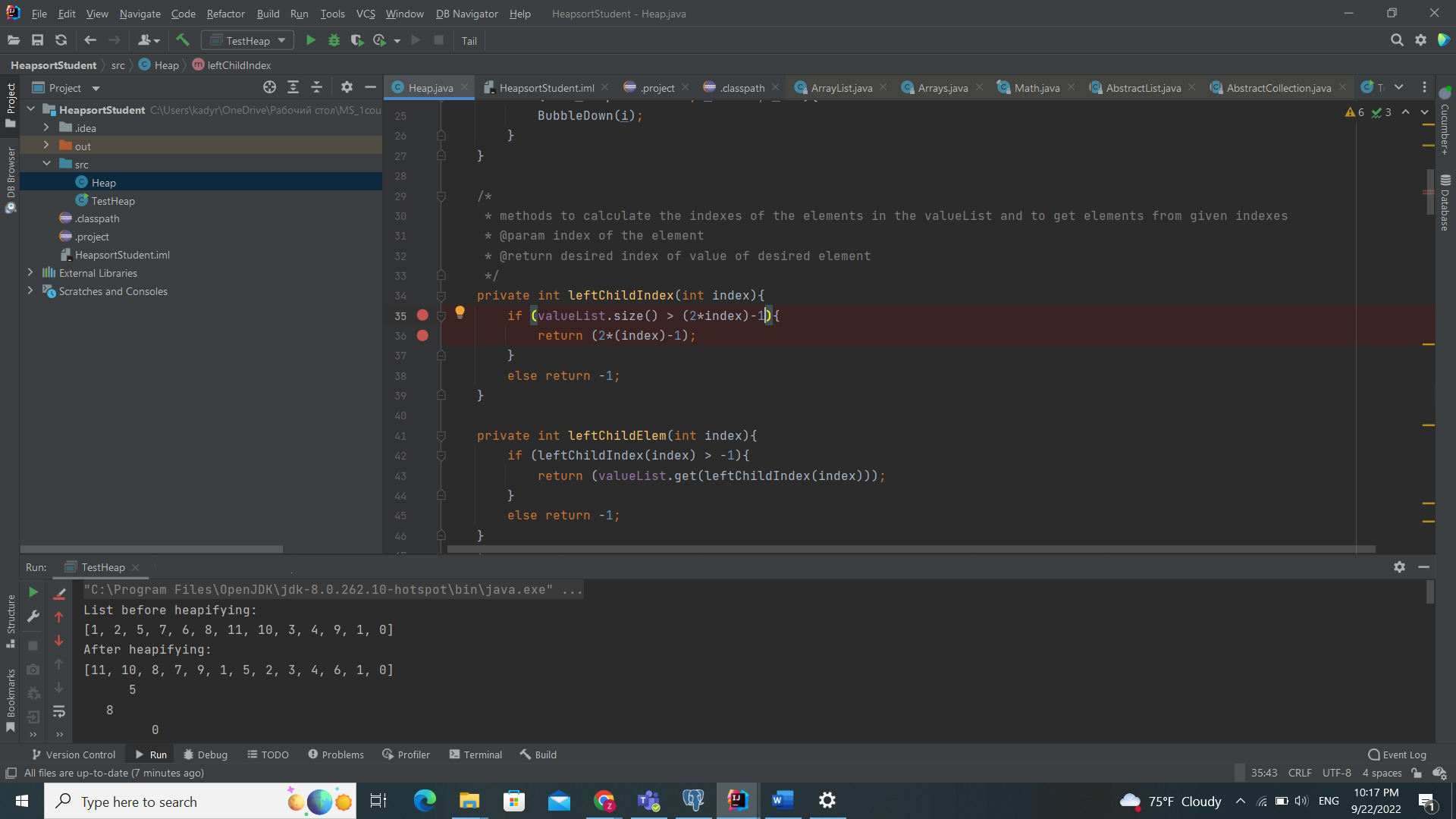
**Task 1 HeapSort**

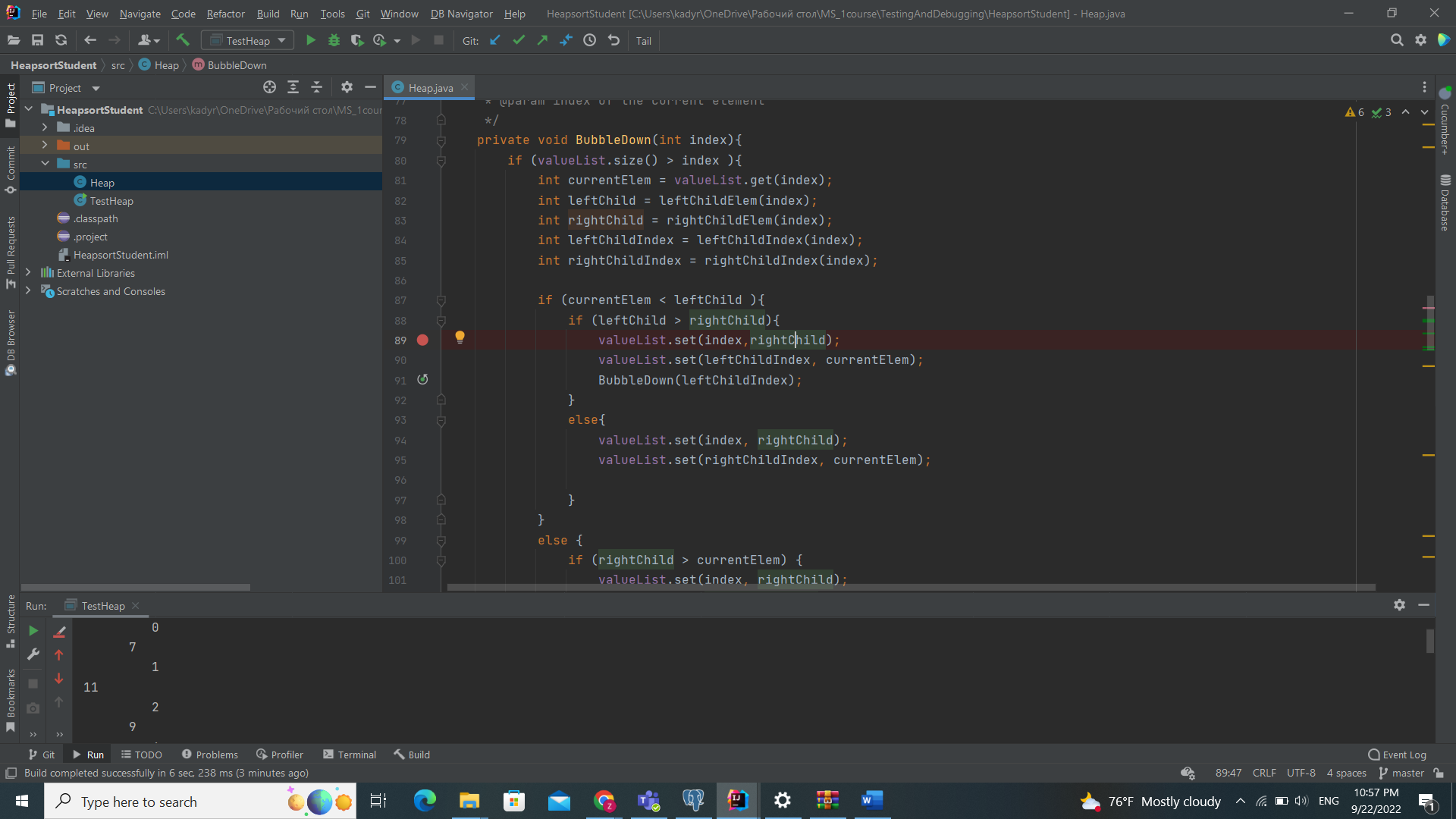
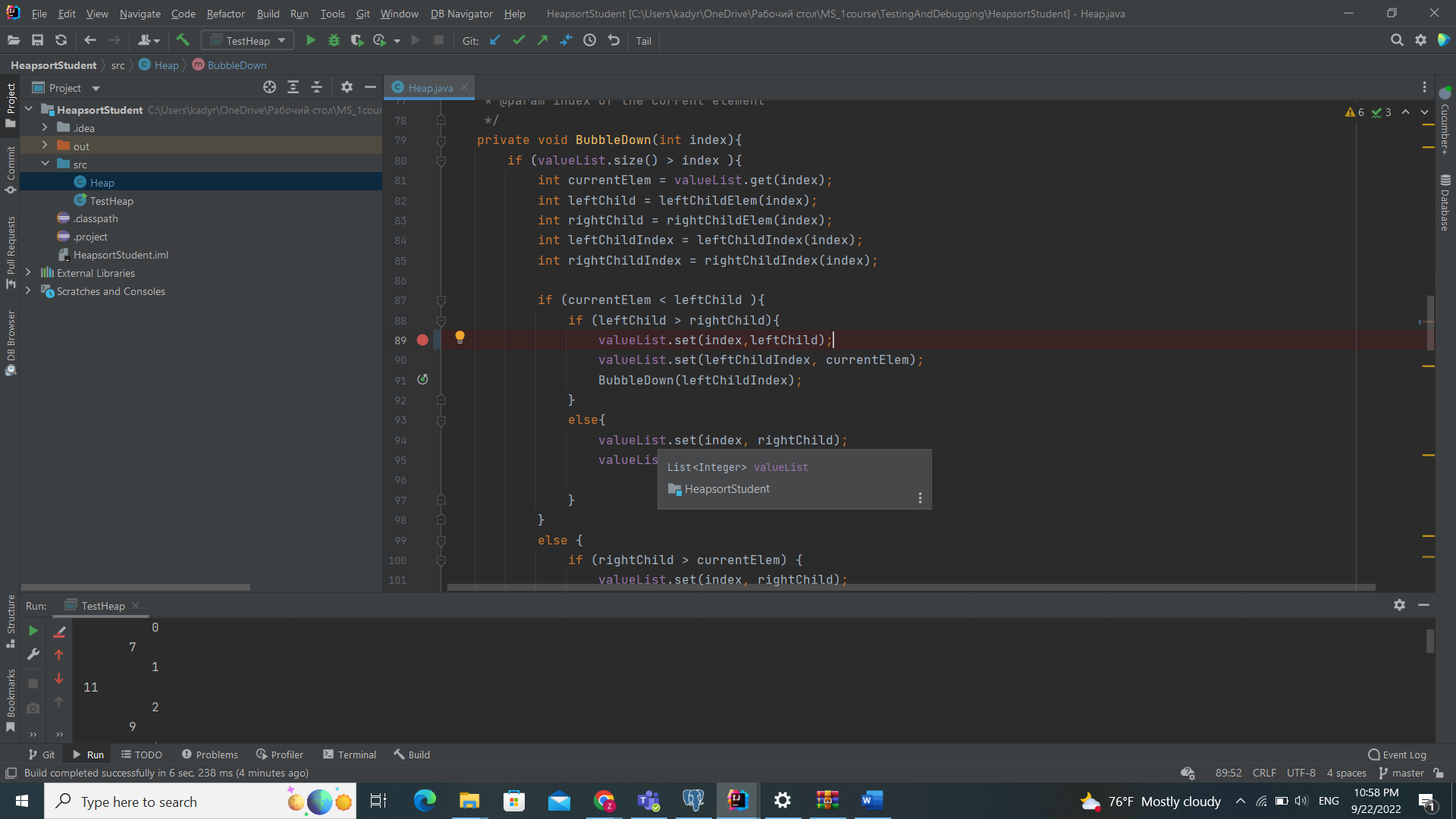
Issue report 1: Description: The program should heapify any given list of positive integers but the resulting tree (and list) does not meet the max binary heap structure

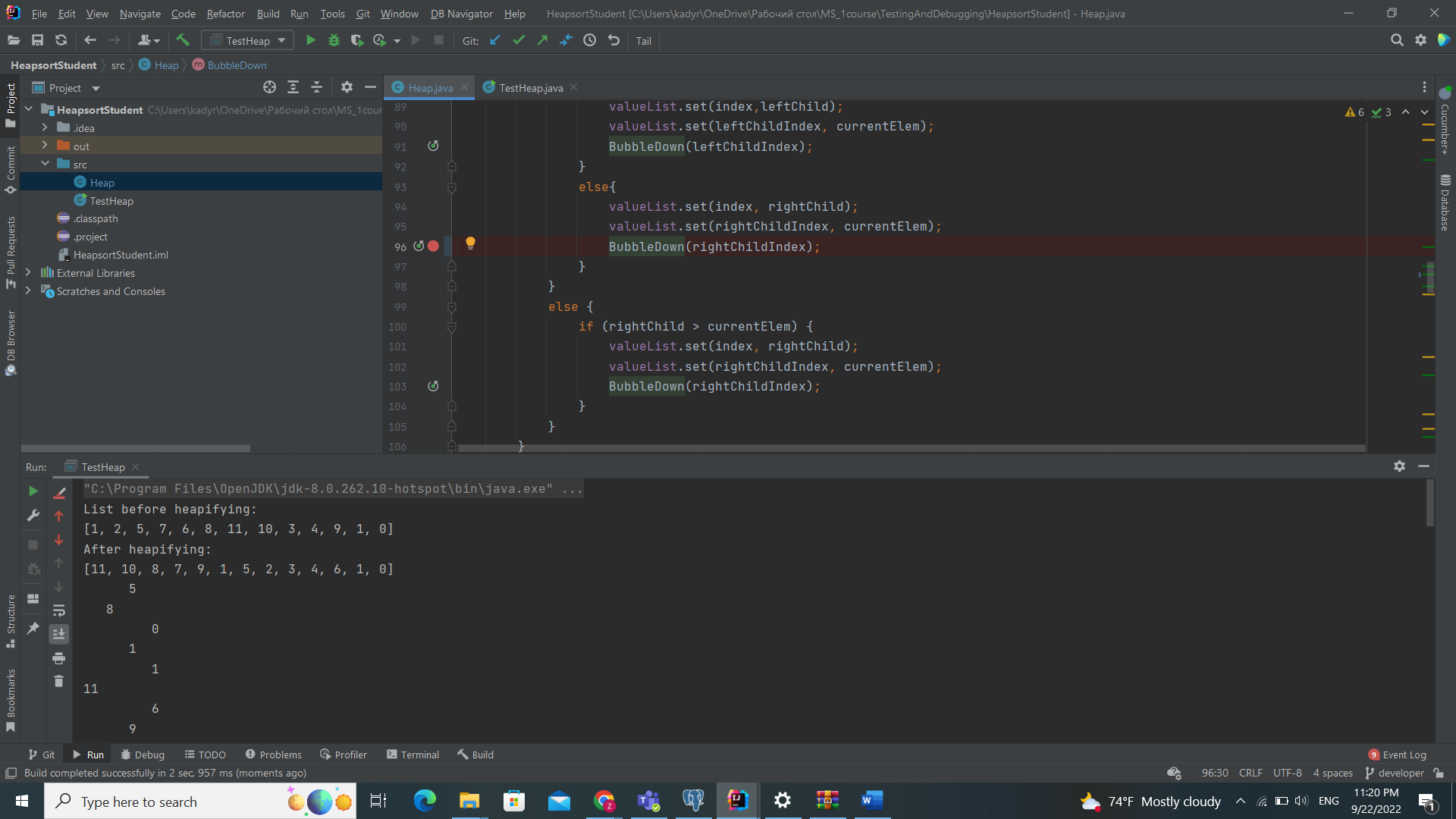
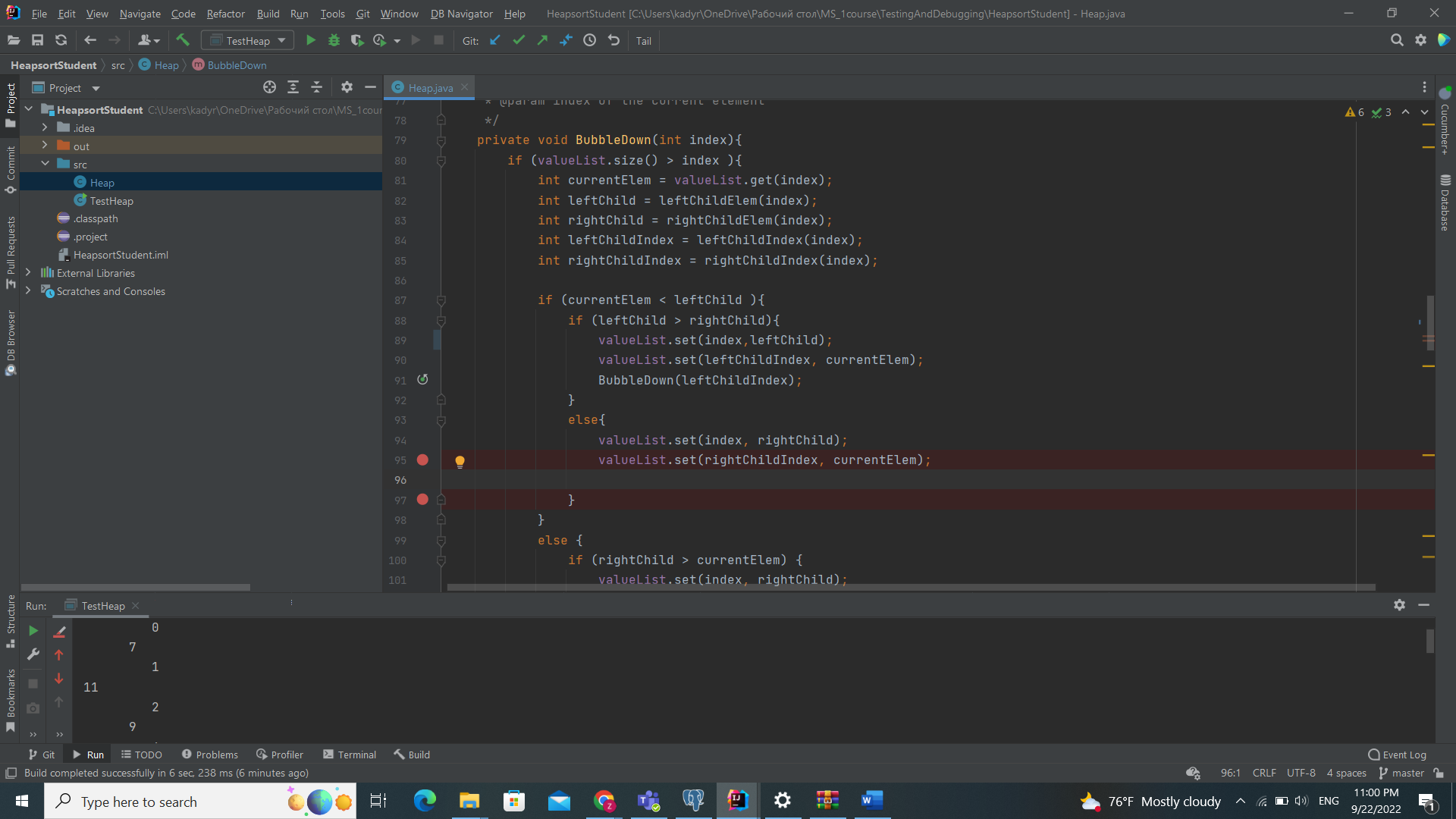
**After debugging I found tree bugs**

**First: Heap.java file 35 ,36,89 and 96 in lines I found bugs**

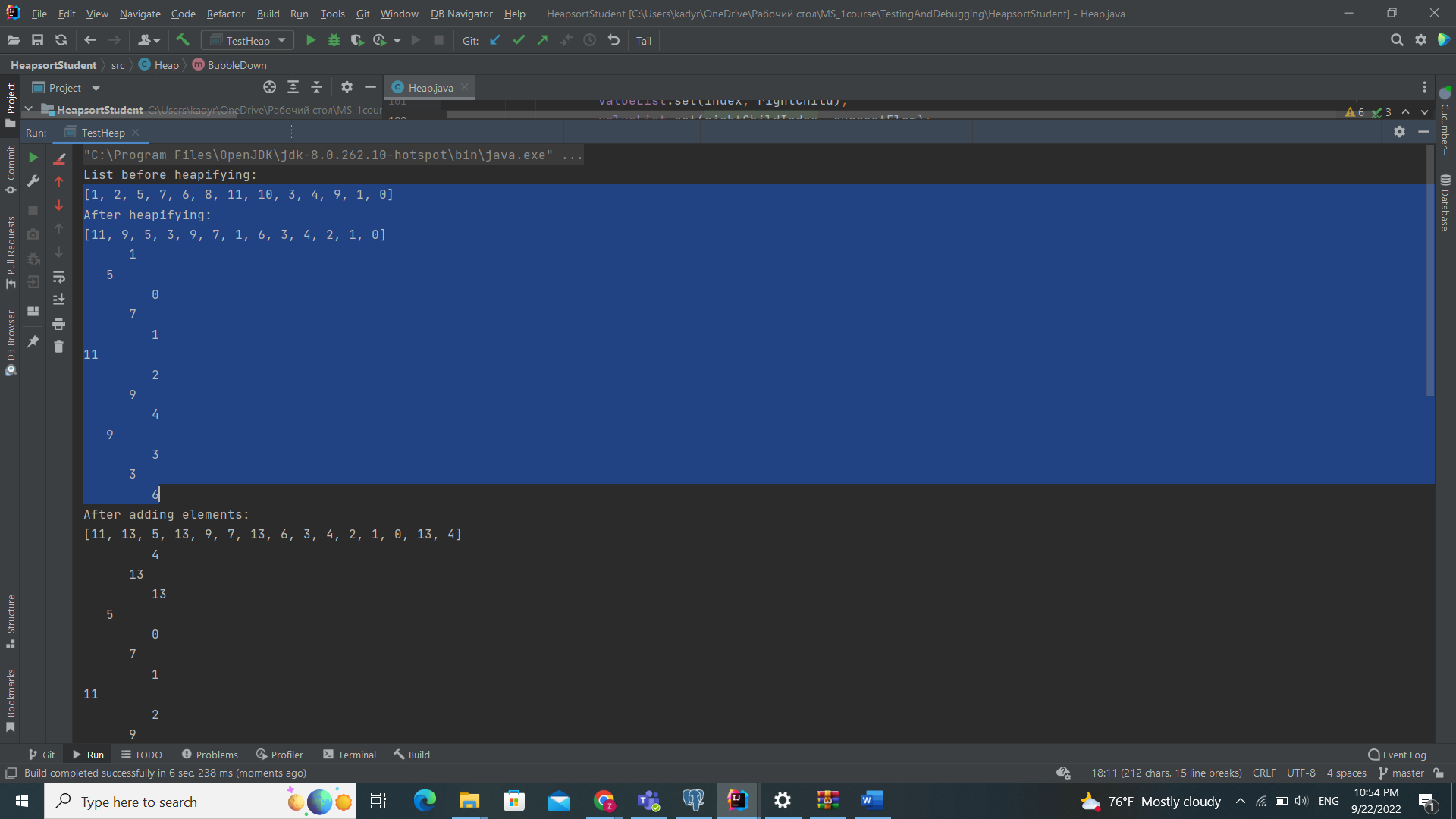
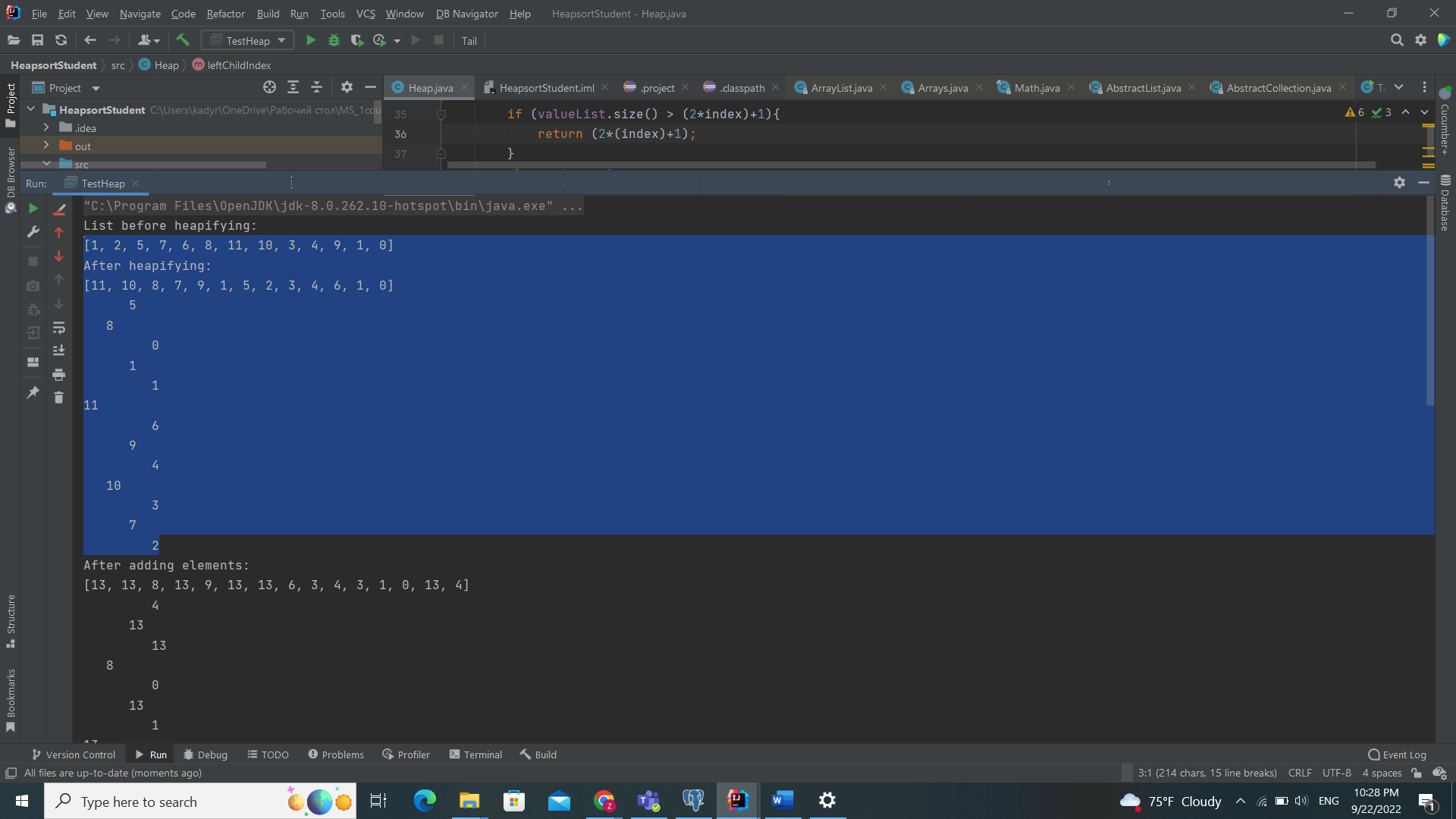
**Before: After:**

 **code**



  Here add 96 line code

**outputs**

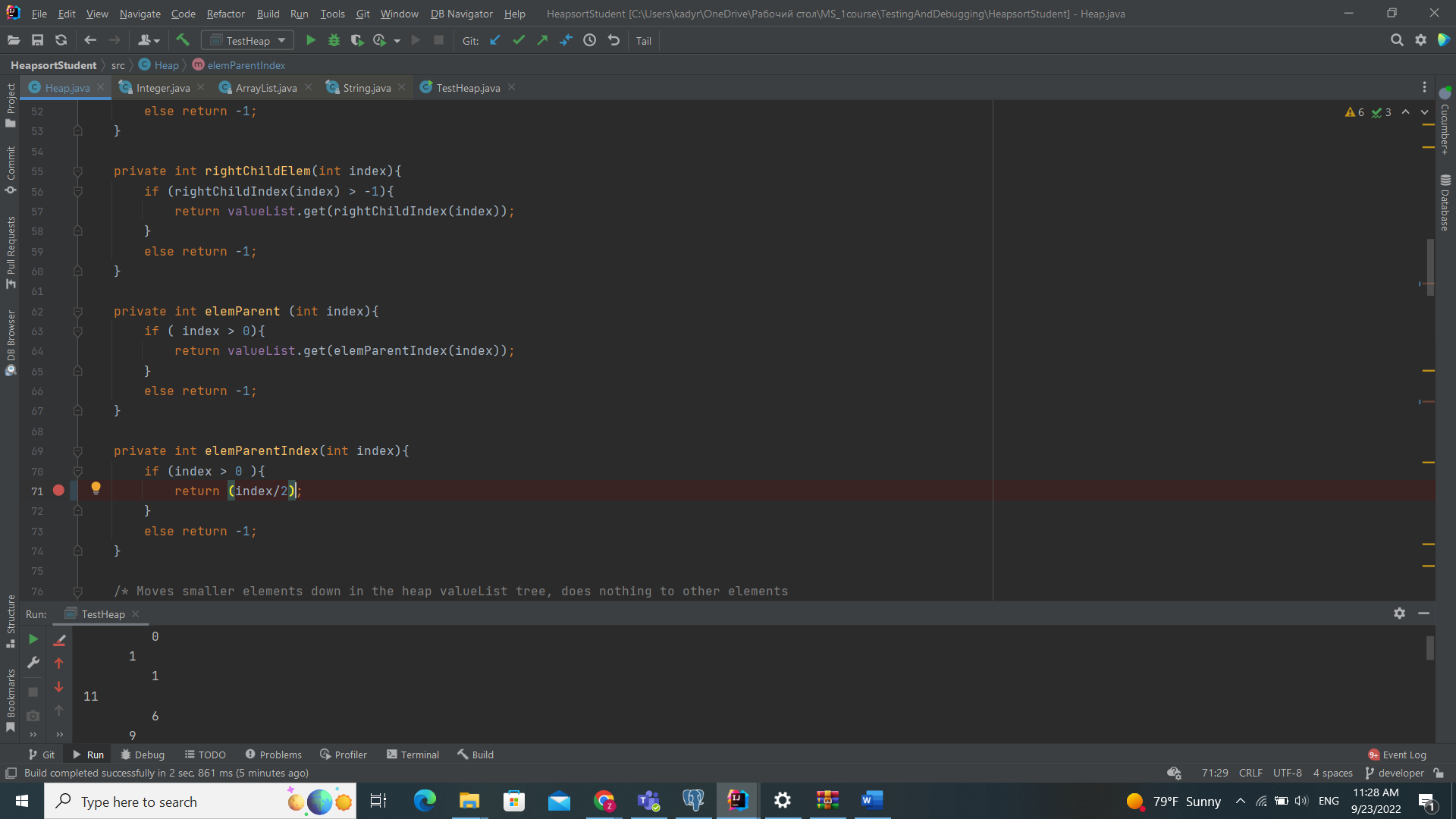


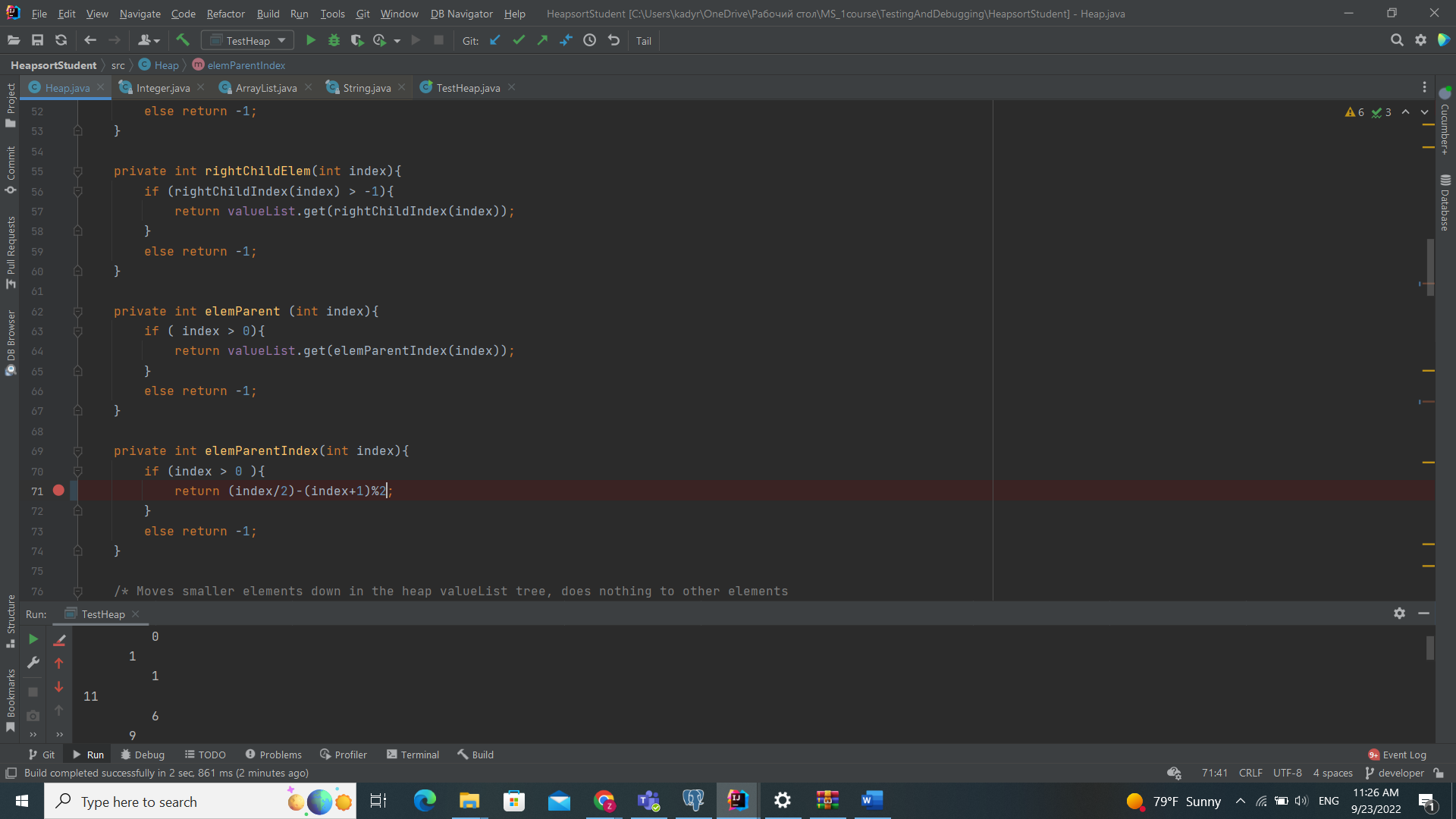
**Issue report 2:**

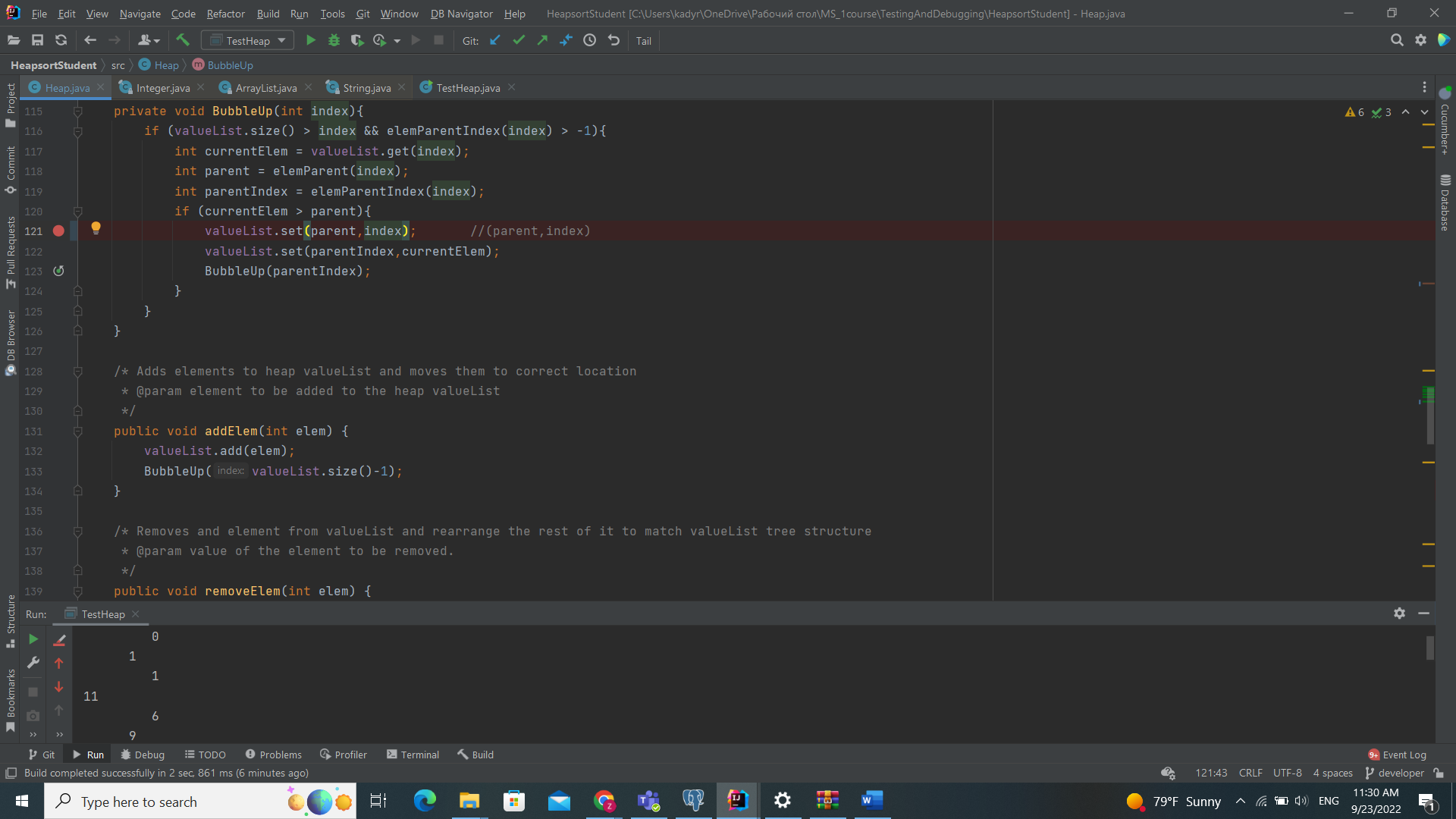
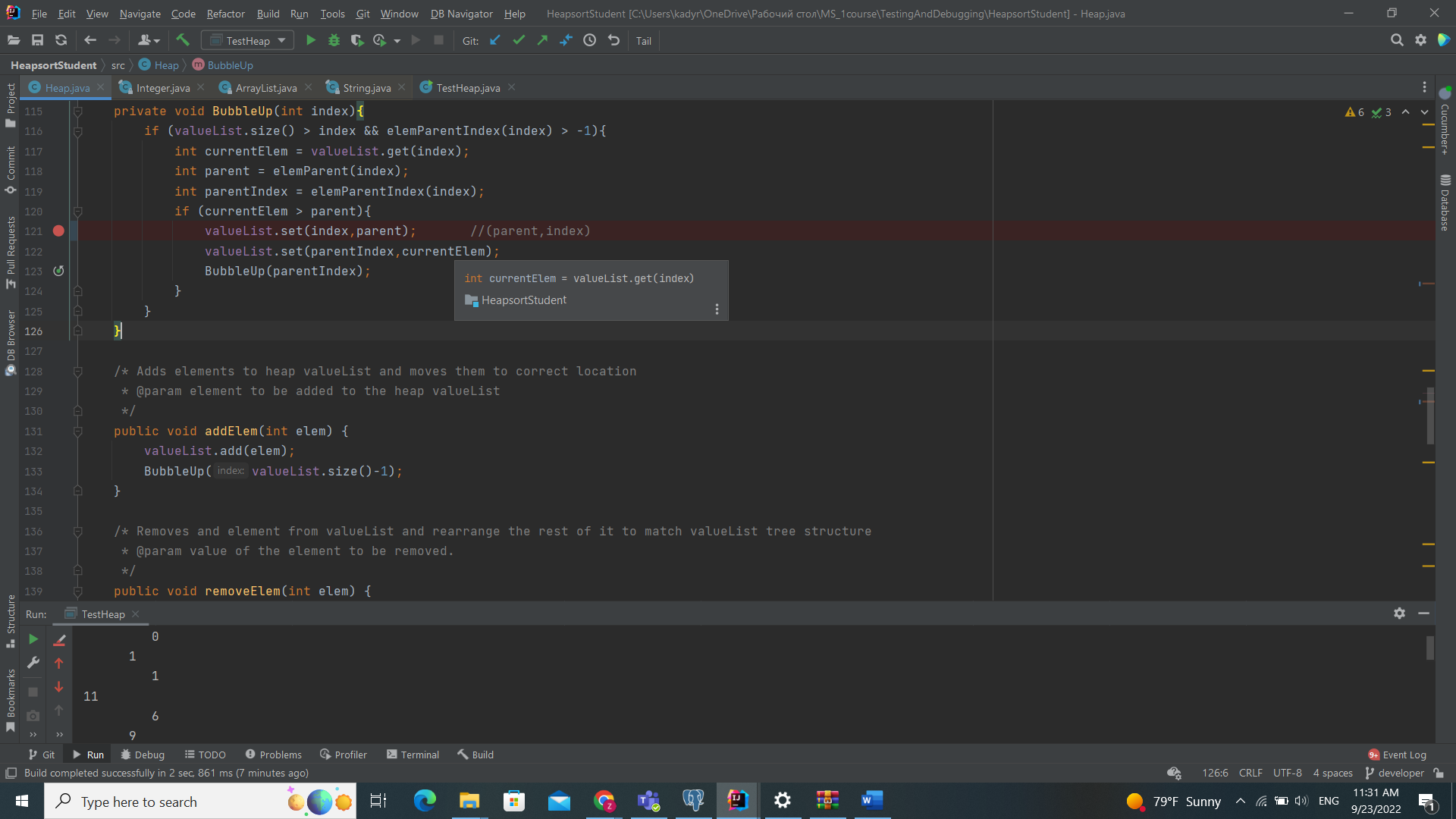
Description: When adding elements to a heap, the element should automatically go to the correct location in the heap. However, after adding elements, the resulting list (tree) does not match the expected max bin heap structure, there are different elements than what should be there (some more, some missing). Input: heapifying a list of integers - [1, 2, 5, 7, 6, 8, 11, 10, 3, 4, 9, 1, 0], then add integers 13 and 4

After testing I found two bugs in this part

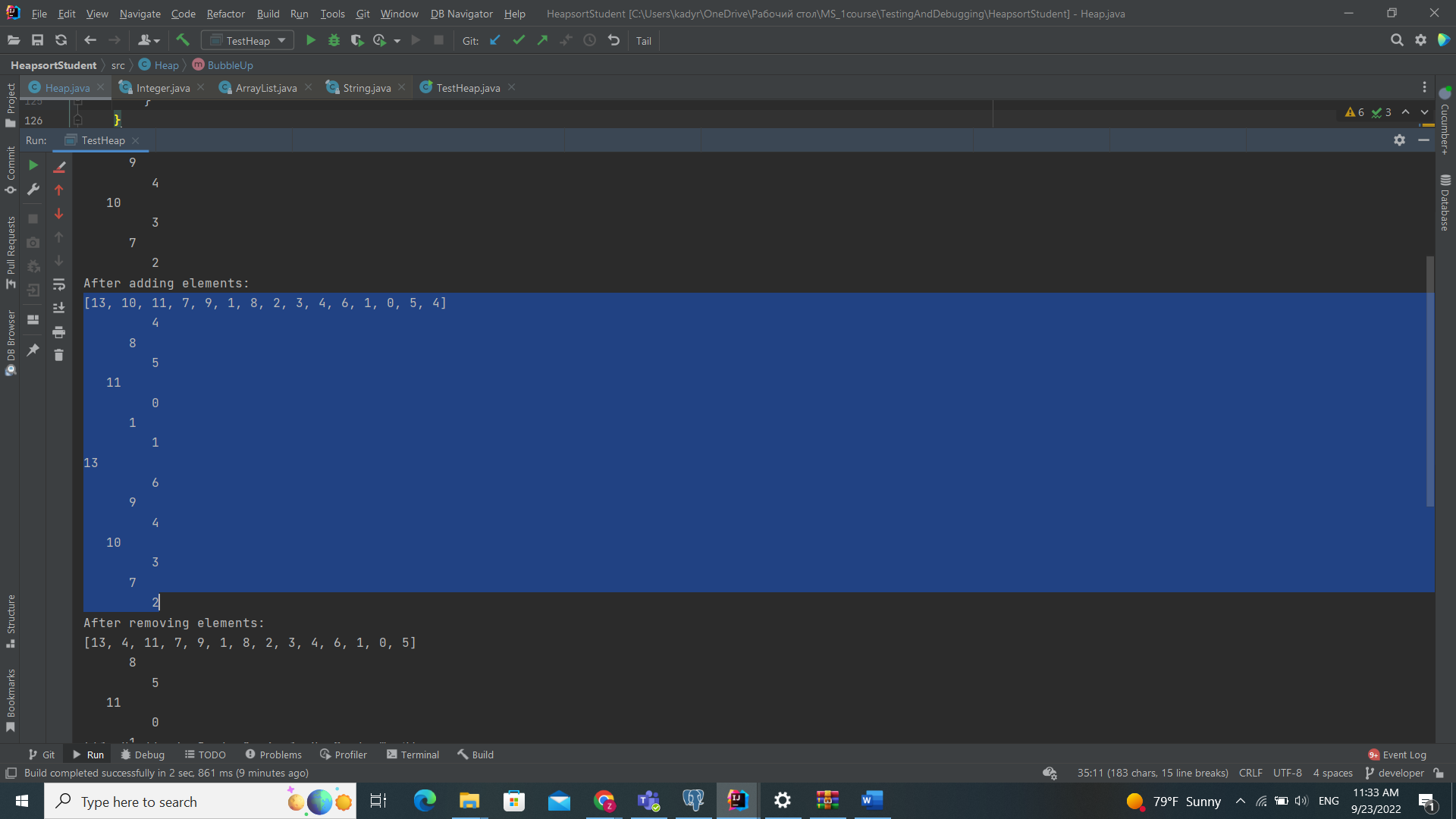
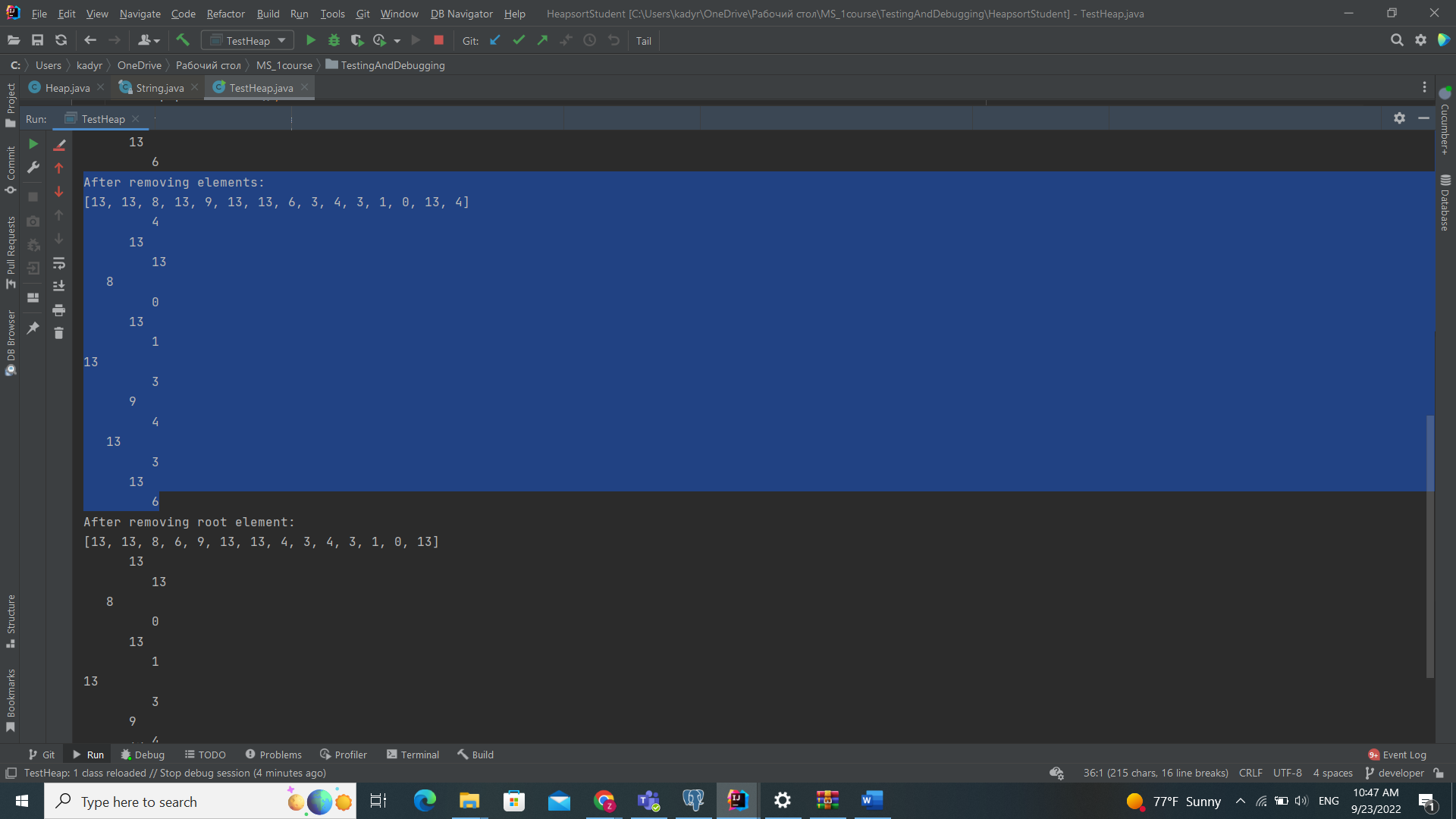
First in 71 line second 121 line

 **Before After**





**Outputs**



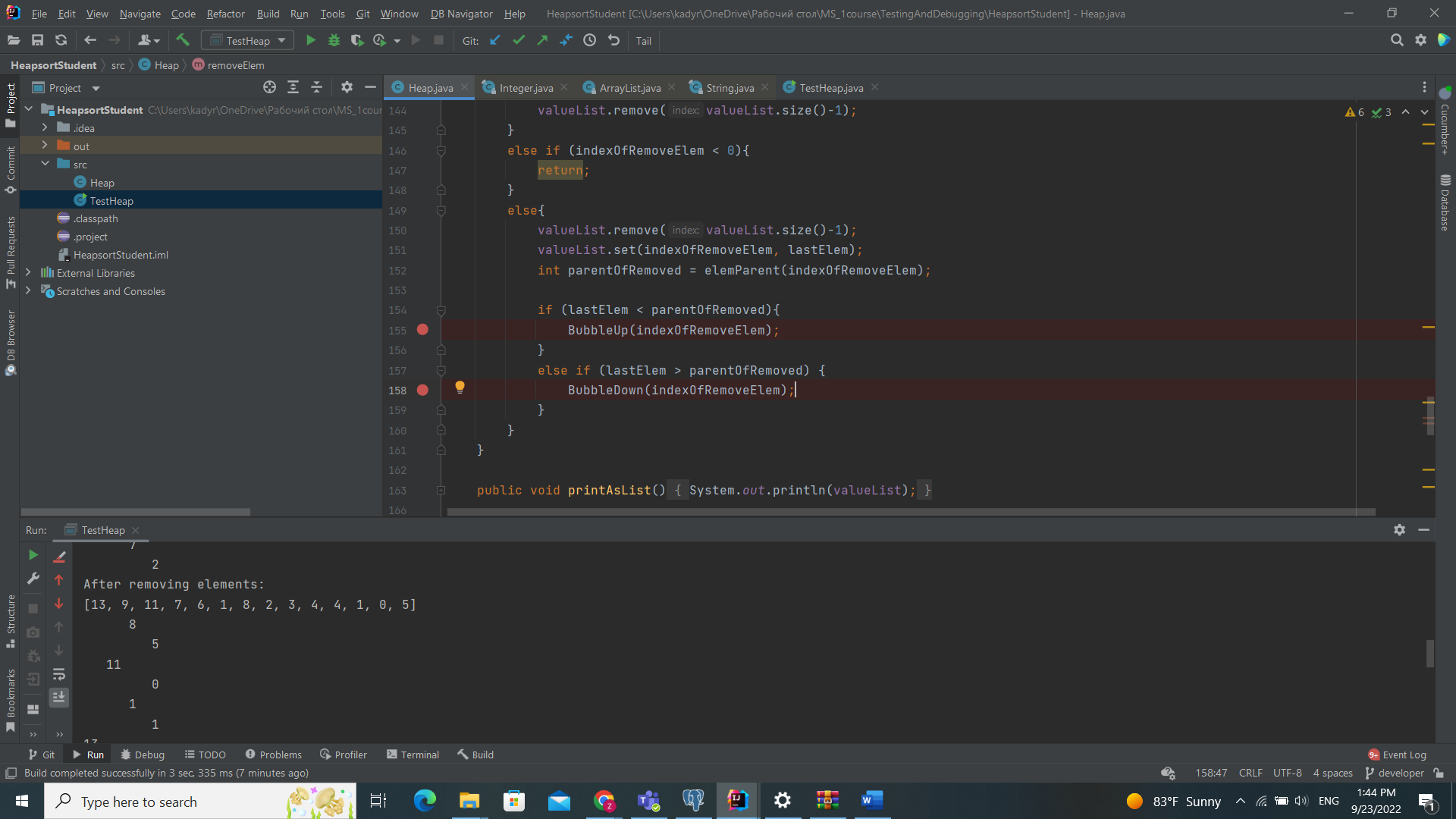
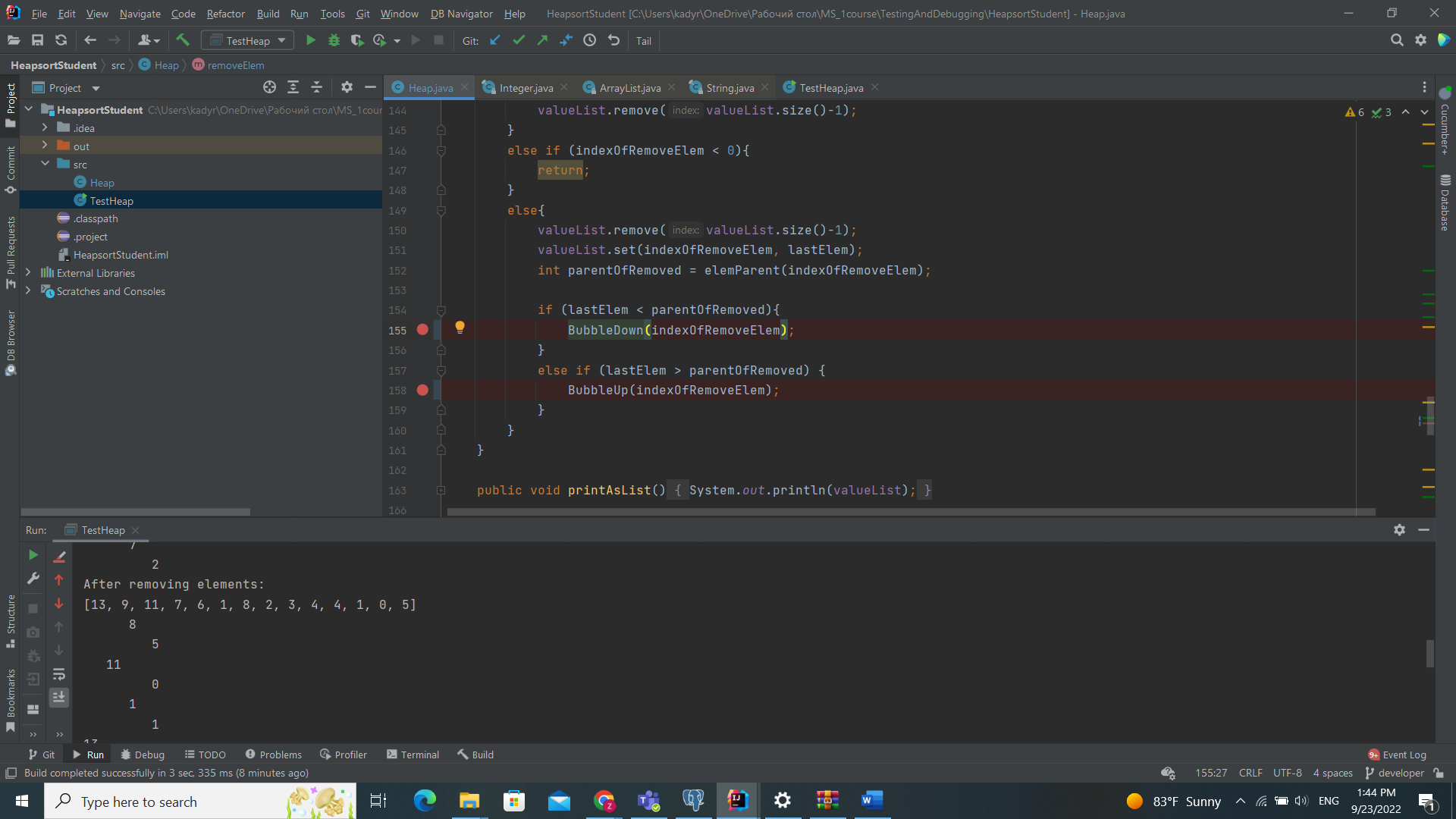
**Issue report 2:**

Description: When removing a given element (the first occurrence of that element) the element should be gone, the rest of the heap in a correctly sorted order, and the list itself one element shorter. The element is gone, but the heap is not in the correct order and the size of the list is not smaller. Input: heapifying a list of integers - [1, 2, 5, 7, 6, 8, 11, 10, 3, 4, 9, 1, 0], add elements 13 and 4, then remove element 10 and try to remove non-existing element 99.

After testing I found two bugs in this part

First in 155 line second 158 line here change method **BubbleUp to BubleDown**

**Before After**

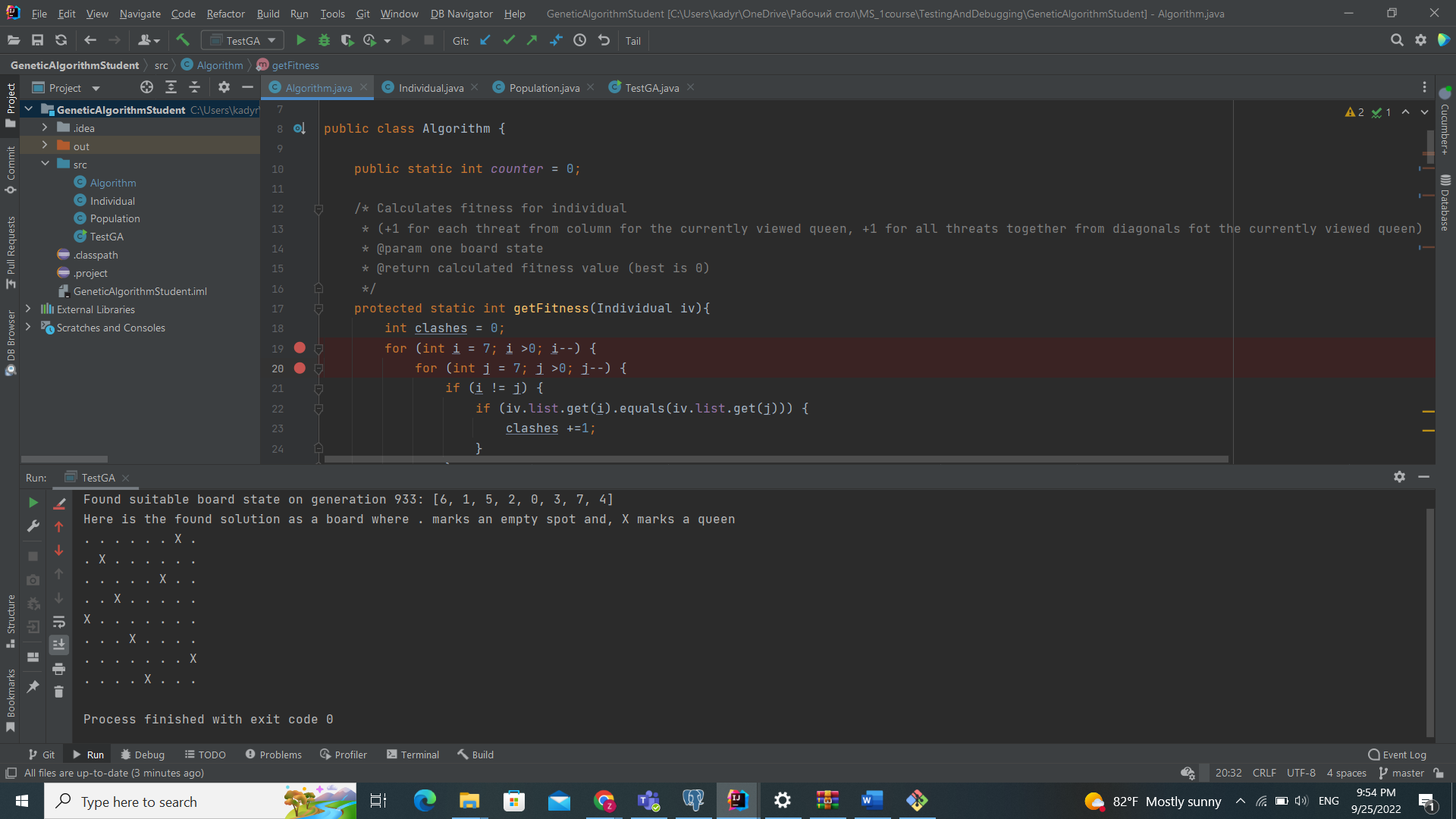
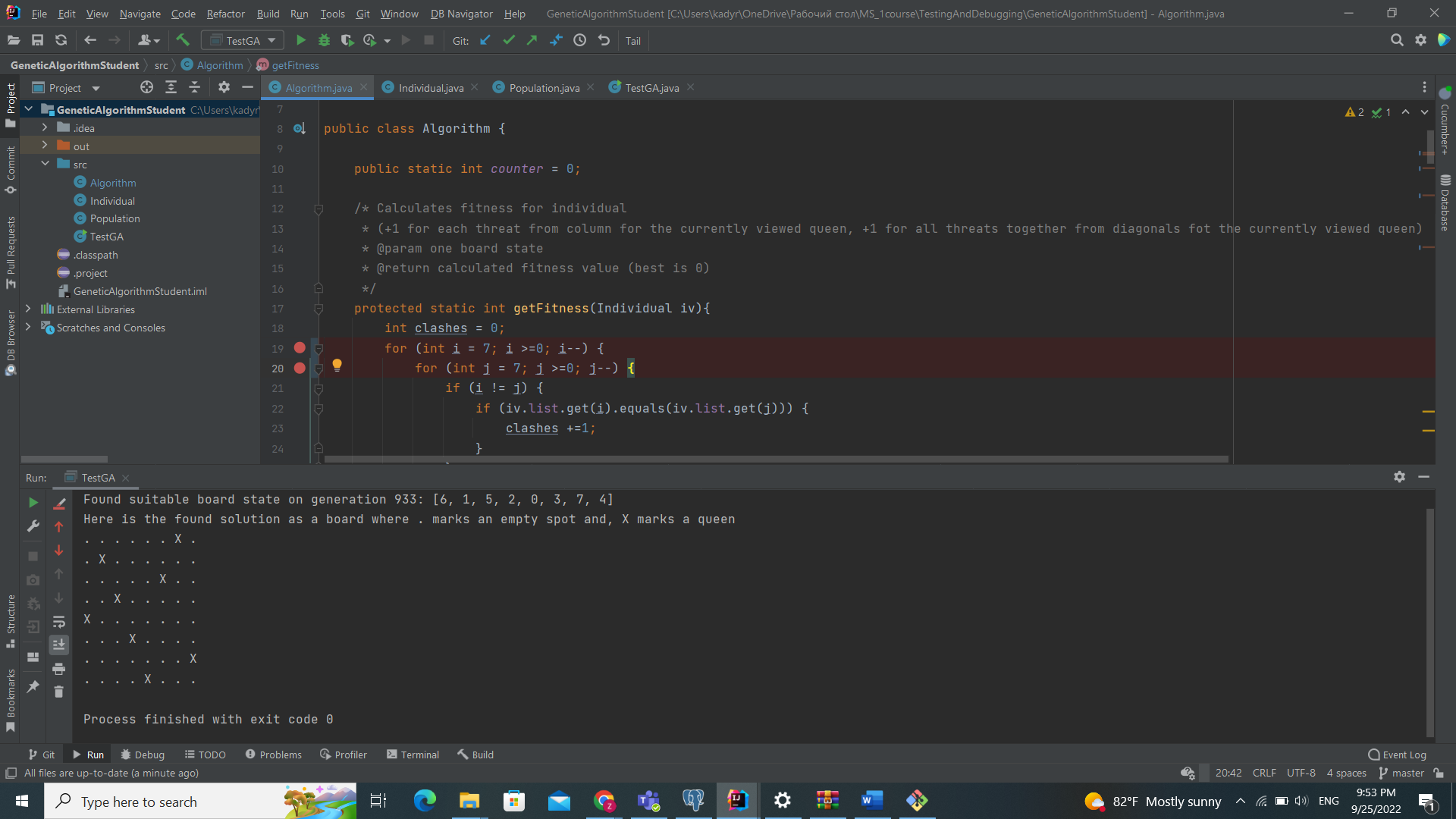


**Task 1 finished and I fix all bugs;**

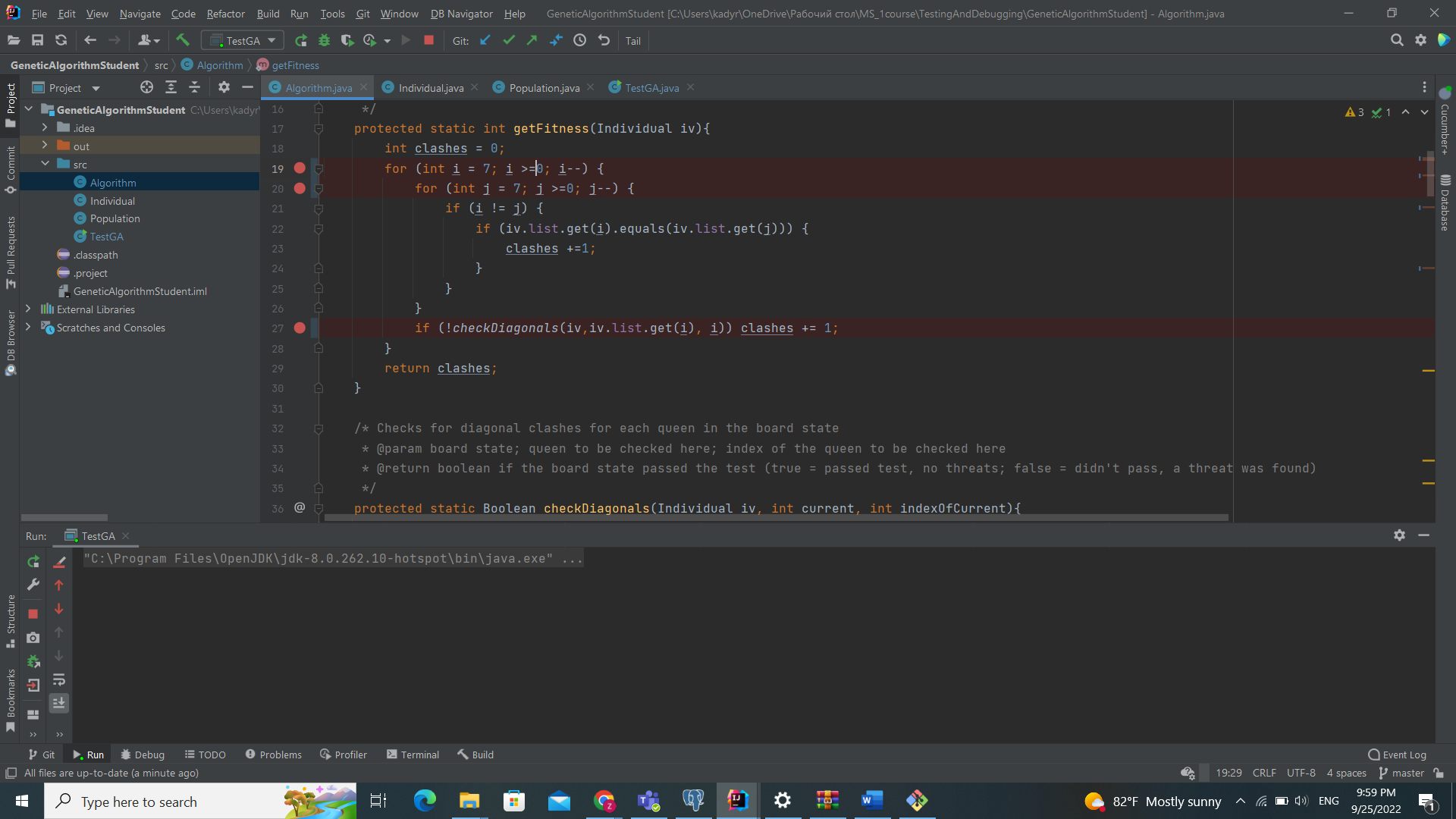
**Task 2 – Debugging “Genetic algorithm”**

**In this task we found 4 bugs there are:**

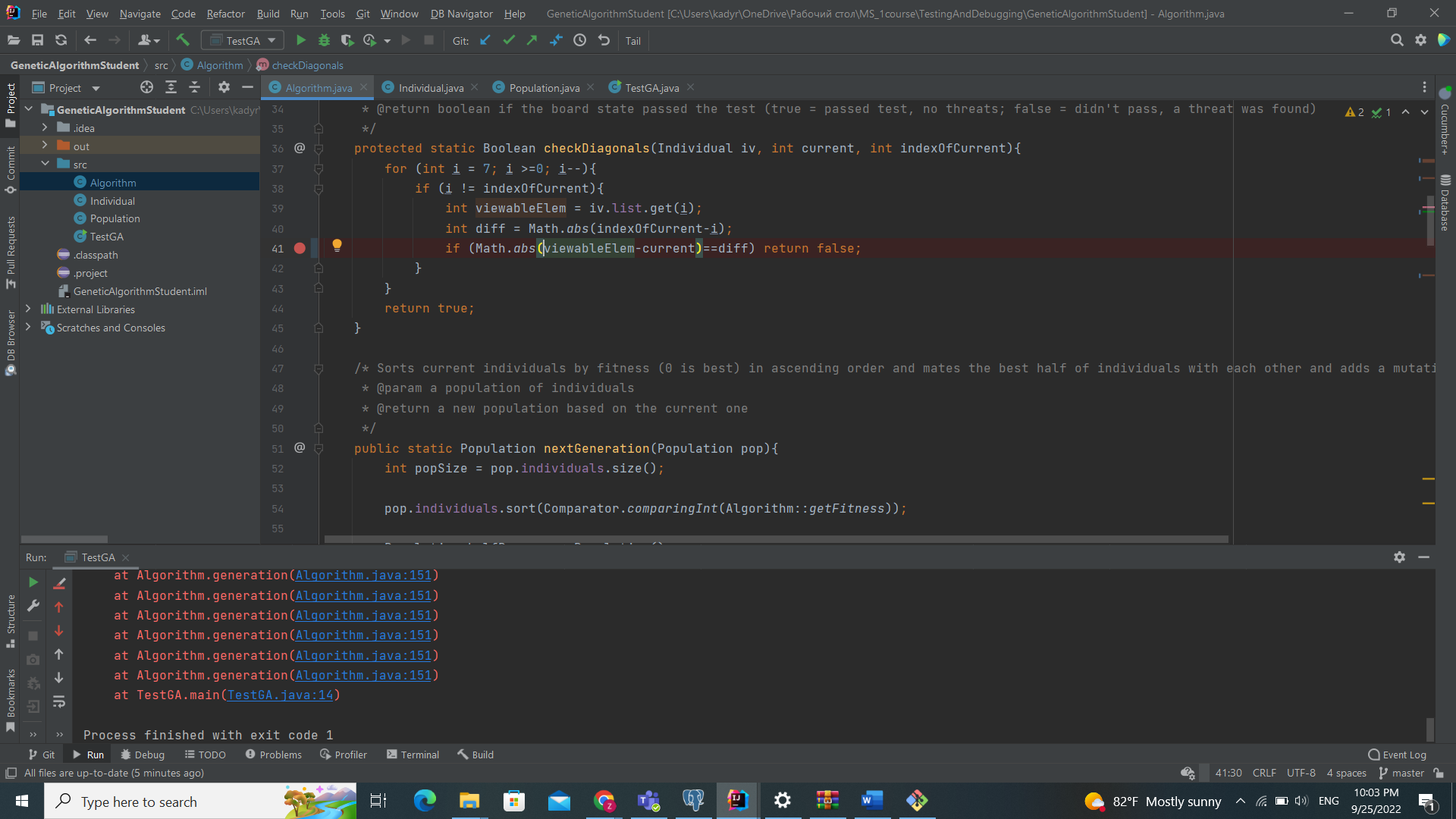
**Before: After:**



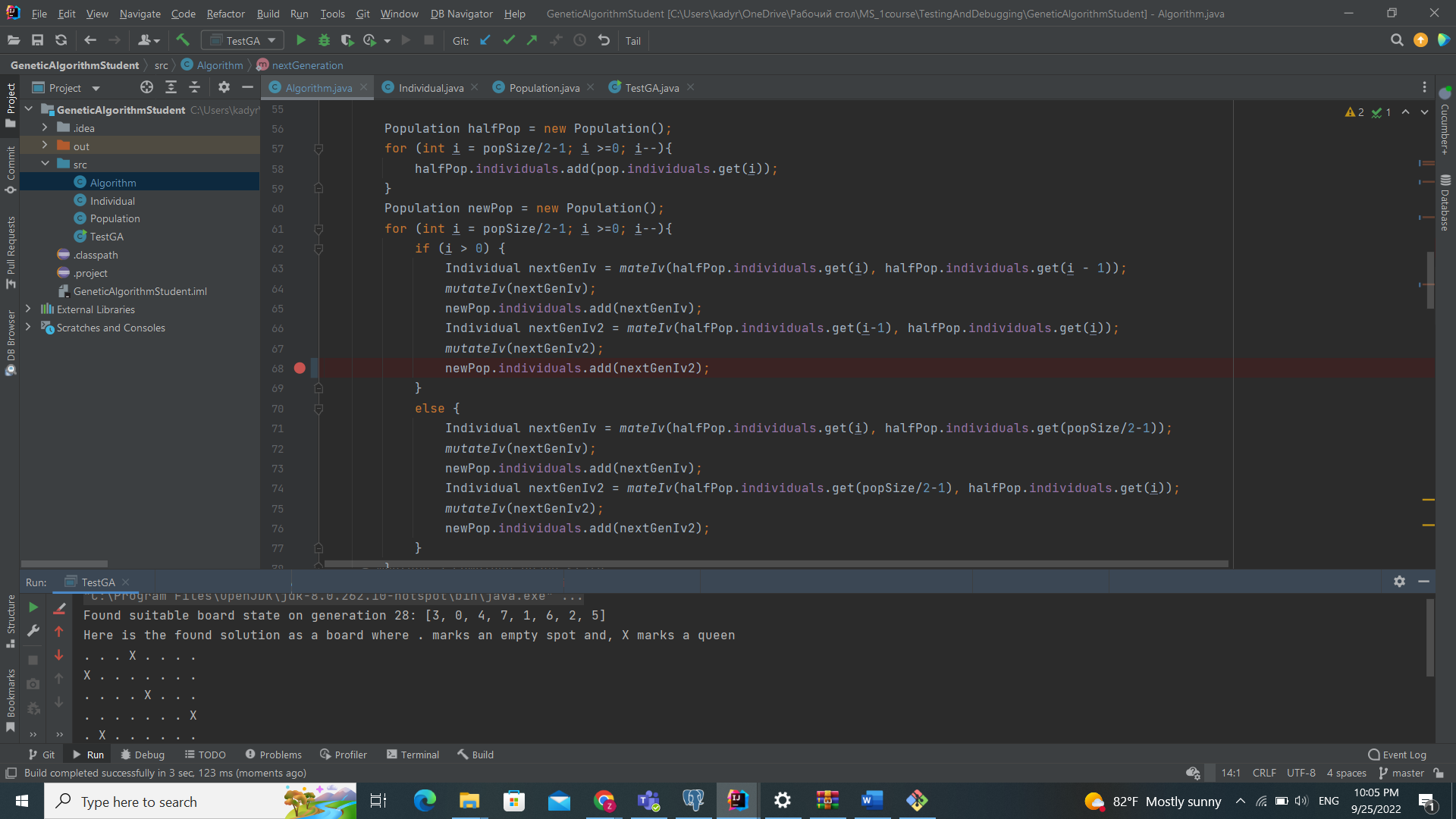
Here we put ! after that work correct



here we add Math.abs method

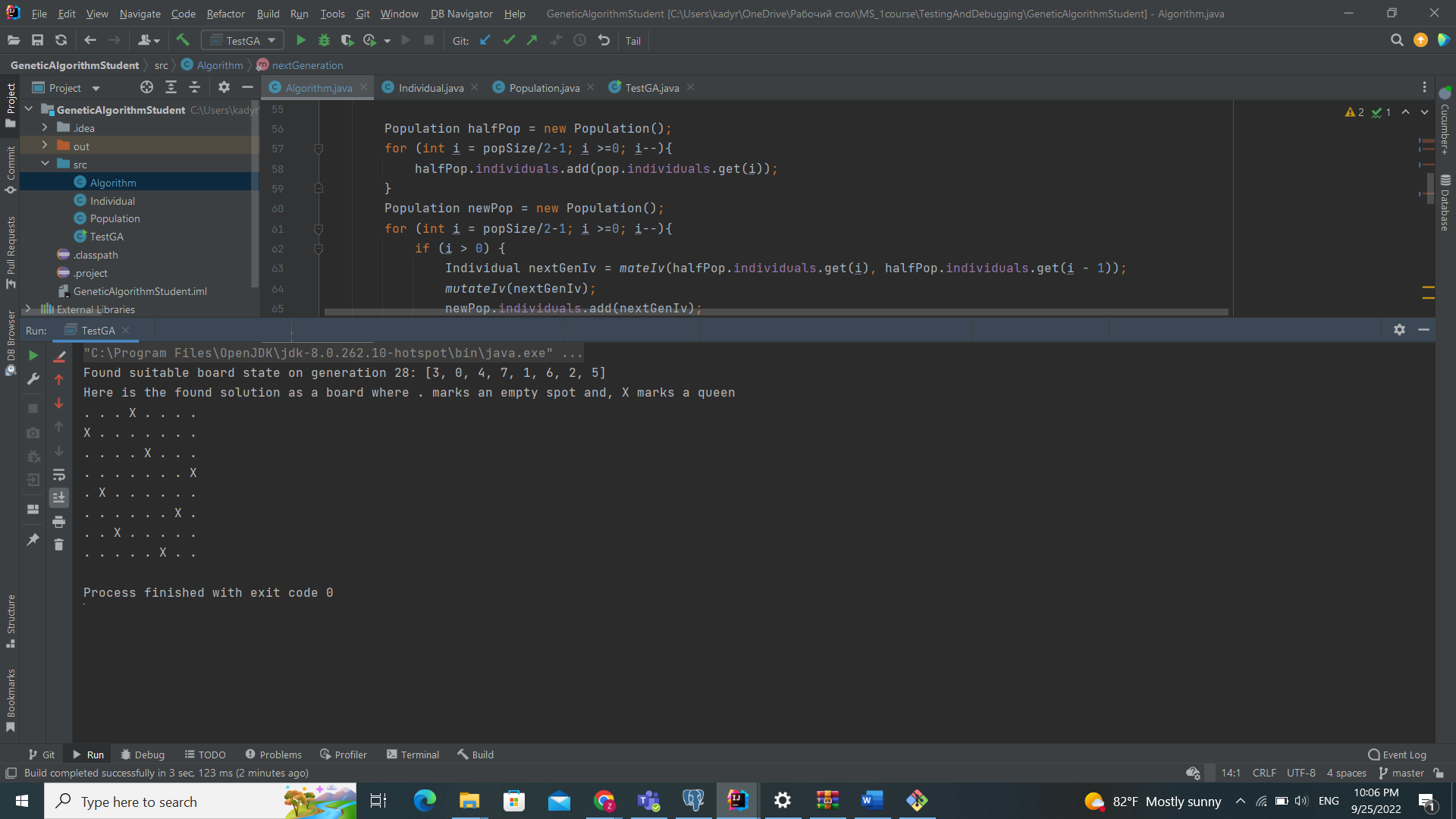


In this line chane halfPop to newPop



**Finished**

**Results**



Based on past projects using genetic algorithms, the average amount of generations should be less than 87 and the program should produce the correct output in less than 100 generations on at least 75% of the runs. However, the performance is much worse, the average amount of generations is over 100 and it only gets the solution in under 100 generations in less than 62% of the time. On very few runs, the program throws an exception of not finding a solution in under 1000 generations.

