* Don’t forget to set your Eclipse workspace and working set.
* **You must submit the JAR file, exported (with source code), from your Eclipse project.**
* **You must check your JAR file to make sure all the source files (.java files) are present. It can be opened with file compression programs such as 7-zip or Winrar.**
* **Failure to export properly will result in your work not getting marked.**

**To submit:**

* **Export your project to a JAR file, with source code.**
* **Name your JAR file ID\_Week14\_Q1.jar. For example, 6623110021\_Week14\_Q1.jar**
* **Submit the JAR file on MyCourseville.**

1. (5 marks) You are given code for minheap (small values are important) and JUnit to test your code.

* Assume that:
  + before “add” method is called, the heap has its largest data stored in position (size-1).
  + Method “pop” is never called.

We want to modify the code of the method **“add”** of this heap such that:

* each time it is called, it adds new value to the heap (as done for normal heap).
* Then it modifies the heap so that largest value is at position (size-1).
* The modified heap must remain a heap.
* You are **NOT allowed** to add new fields to Heap class. **If you do, you get 0 point**.
* You are **Not allowed** to modify any files apart from **Heap.java**. No other files should be added to the project. Otherwise the grader will not mark your file.
* You are allowed to write your own methods to help with this question.
* There are 5 test cases, each test case is 1 mark. The marking test cases will be different from the ones given to you.

The following example illustrates how you **MUST** implement “add” in this question when adding -2 to the following heap (**If you do not use this algorithm, you get 0 point**):

If the last position does not contain the largest data, swap its data with the largest data.



Add -2.

First, add and percolate normally.



Then percolate again as necessary.