Learning Objectives:

- Implement a priority queue
- Select the appropriate data reprentation to produce the desired results regarding performance and memory usage
- Build experience in the use of jUnit tests

Turning In:

Make sure to use the same names and signatures that are specified in the instructions.
Submit a jar file that **includes all the java files** via Canvas.

Description:

a) Write a **generic** data type (a class) called **`MinMaxPQ`**
b) Write **jUnit tests** for all public methods of MinMax
c) Measure the **performance**

Specification:

a) Class MinMaxPQ:

Write a **generic** data type (a class) called **`MinMaxPQ`**.
It needs to be able to store elements of any reference type as long as the type is comparable (implements Comparable<T>)
Class `MinMaxPQ` allows you to add elements and remove either the largest or the smallest element (you can think of it as a priority queue that allows you to remove elements from both ends)

Class **`MinMaxPQ`** has a default constructor and exactly **6 public class members**:

1. **`add ..`** adds an element of type T (no return type)
   throws a `NullPointerException` if an attempt is made to add a **`null`** value to MinMax
2. **`min ..`** returns the smalles element
3. **`removeMin ..`** removes the smalles element and returns it
4. **`max ..`** returns the largest element
5. **`removeMax ..`** removes the largest element and returns it
   Methods 2 to 5 throw a `NoSuchElementException` if an attempt is made to remove from an empty **`MinMaxPQ`**
6. **`isEmpty()..`** returns **`true`** if the **`MinMaxPQ`** is empty, **`false`** otherwise

All six public methods have to be supported in constant or logarithmic amortized time
Memory usage has to be proportional to N.

b) jUnit Tests:

Write jUnit tests for each of the 6 public methods.
Create separate test cases for different equivalence partitions.
Keep in mind that it is the job of unit tests to uncover defects in the program.
If I run your unit tests with a MinMaxPQ class, that does not work properly, your unit tests need to fail.

c) Measure Performance:

Write test code that measures the performance of the 6 methods and that prints out the table(s) with the test results.
Make sure that the results are clearly labeled and that the tests are set up so that the measurements lead to conclusive results regarding the performance of the methods.