Priority Queue

Overview

For this assignment you will be implementing a heap using an array and a min and max priority queue
Due: April 25th, 2014 @ 5pm

Files to complete

You are expected to write an implementation for each of the interfaces/abstract classes listed in the classes presented in the **config** package provided. As with the last assignment, you must specify which implementation you would like us to grade in this file.

In addition to this, you **MUST** include a String field in the **config.Configuration** file with your UMass student ID number. This is the 8 digit value found on your student ID card and is used to identify you in moodle. This will allow us to get your grade and feedback to you quickly. We recommend that you enter this ID value immediately after importing the project.

It will almost certainly be useful for you to write additional class files. Any class file you write that is used by your solution **MUST** be in the provided **src** folder. When we test your assignment, all files not included in the **src** folder will be ignored.

Note: When you submit your solution, be sure to remove **ALL** compilation errors from your project. Any compilation errors in your project may cause the autograder to fail and you will receive a zero for your submission.

Test files

In the test folder, you are provided with several JUnit test cases that will help you keep on track while completing this assignment. We recommend you run the tests often and use them as a checklist of things to do next. You are not allowed to remove anything from these files. If you have errors in these files, it means the structure of the files found in the **src** folder have been altered in a way that will cause your submission to lose points. We highly recommend that you add new @Test cases to these files. However, before submitting, make sure that your program compiles with the original test folder provided.

Support Code API

The Support Code's comments have been generated into a nicely formatted API that can be found here:
http://people.cs.umass.edu/~jcollard/cs187/s14/assignments/9/doc/

It is highly recommended that you spend some time simply reading over the comments in each of the interfaces and classes provided.

Part One: Importing Project into Eclipse

Begin by downloading the provided project from Moodle and importing it into your workspace.

You should now have a project called **priority-queue-student.** It is very important that you not rename this project as it is used during the grading process. If the project is renamed, your assignment may not be graded.

By default, your project should have no errors and contain the following root items:
src - The source folder where all code you are submitting must go. You can change anything you want in this folder, you can add new files, etc...
support - This folder contains support code that we encourage you to use (and must be used to pass certain tests). You are not allowed to change or add anything in this folder
test - The test folder where all of the public unit tests are available
JUnit 4 - A library that is used to run the test programs
JRE System Library - This is what allows java to run

If you are missing any of the above or errors are present in the project, seek help immediately so you can get started on the project right away.

Once you have your project imported, add your 8 digit student ID number to the **config.Configuration** class.

Part Two: Extend and Implement AbstractArrayHeap

For this part of the assignment, you will implement a Heap. A Heap is a Binary Tree with the property that for each node in the heap, its children's priorities are less than or equal to the nodes priority. In addition to this, a Heap has the property that it is complete. One way to maintain the completeness property is to use an Array as our underlying data structure (thus the name Abstract**Array**Heap). For more information on implementing a binary tree using an array, see the wikipedia article on Binary Heaps. This link will be useful when implementing getLeftChildOf, getRightChildOf, and getParentOf.

Start this part of the assignment by opening up the AbstractArrayHeap class and reviewing what is already provided for you and what you have to implement.

Something you should notice right away is that our add function is a little bit different than we are used to seeing in this course. It takes in two arguments: a priority and a value. This allows us to insert two elements with the same value but different priorities. After receiving these inputs, we transform them into an Entry<P,V> and use this to store our element. This is commonly referred to as a Priority Value Pair.

Another thing you will probably notice is that our declaration for P doesn't state that it must be Comparable. So… how are we supposed to maintain an order? Take a look at the constructor. It takes in a Comparator<P>. This allows us to provide arbitrary ordering for Priorities. If we want a Max Heap, we simply pass in a Comparator that says larger values come first.

How does bubble up work:
By now, you should have noticed that the add method simply simply creates an entry and adds it to the end of the array. Then, a call to bubbleUp on the newly added index is called. The bubbleUp method will cause the new entry to float up the heap until it is in a location such that the properties of a heap are maintained. Related wikipedia article: http://en.wikipedia.org/wiki/Binary_heap#Insert

How does bubble down work:
By now, you should have noticed that the remove method swaps the bottom most node that is furthest to the right with the node at the top of the heap. It then removes that node. This allows us to maintain the property that our heap is complete. Unfortunately, we no longer have the property that our nodes are properly placed in the heap. After this swap, bubble down is called on the top most node where it will sink into the heap until it finds an appropriate location such that the heap properties are maintained. Related wikipedia article: http://en.wikipedia.org/wiki/Binary_heap#Delete

Part Three: Write a MinQueue and MaxQueue

Utilizing the ArrayHeap implemented in Part 2, create a two classes: MinQueue<V> and MaxQueue<V>. Each of these should be a PriorityQueue<V> that use Integer priorities. The MinQueue is implemented such that lower integer values have the highest priority. The MaxQueue is implemented such that higher integer values have the highest priority.

To do this part, you will need to write a Comparator<Integer> class. An example Comparator has been provided in the support folder: support.StringLengthComparator. This Comparator orders Strings such that shorter Strings have lower priority than longer Strings. You may also find the following link useful: http://docs.oracle.com/javase/7/docs/api/java/util/Comparator.html

Part Four: Export and Submit

When you have finished and are ready to submit, export the entire project. Be sure that the project is named **priority-queue-student**. Save the exported file with the zip extension (any name is fine).

Log into Moodle and submit the exported zip file.