Due: Oct. 30 (10/30), 13:00

Overview

This assignment consists of two parts: implementing a heap, and implementing a job scheduler.

General Notes

- *Read this homework guideline carefully.* If you do not follow the guidelines, you may receive a 0 regardless of whether your code works or not.
- Do not use any IDEs (Eclipse, IntelliJ IDEA, etc.)
  - We recommend Sublime Text (Linux/Mac/Windows), Notepad++ (Windows), or TextWrangler (Mac).
  - IDEs often create a "package" of your code, which breaks the autograder.
  - **If you know how to fix the package problem**, you can use any IDE you want. However, we will not answer any questions related to this problem since we have already recommended a solution.
- Do not change any method or class signatures. If your code changes any class or method names or signatures, you will receive an automatic 0.
- Make sure your code compiles. Non-compiling code will automatically receive a 0. If your code does not compile, it may be better to just comment out the incorrect code and return a dummy value (something like null or -1) so the rest can compile.
- To ensure that your code will be accepted by the autograder, you should submit your code on LearnUS, download it again, recompile it and check the provided test suite. This way, you know the file you are submitting is correct.

Heap

Heaps are data structures that are used in implementing priority queues. In this homework, you will implement a generic min-heap. Since each item in a heap should be comparable, the generic type `T` implements the `Comparable` interface. The methods to be implemented are:

- `Heap()`: initializes an empty heap.
- `T min()`: returns the minimum entry. Does not remove the minimum entry.
- `T removeMin()`: removes and returns the minimum entry.
- `void insert(T entry)`: inserts the entry into the heap.
- `void clear()`: removes all entries of the heap.
- `int getSize()`: returns how many entries are currently in the heap.
- `boolean isEmpty()`: returns true if the heap has no entries. Otherwise, returns false.
- `void merge(Heap<T> otherHeap)`: merges two heaps together to make a single heap. The merged heap must be in the `Heap` instance where this method is called. That is, if you call `heap1.merge(heap2)`, then `heap1` should hold the merged result. You may assume that the `otherHeap` will be discarded afterwards, so you may alter the heap structure of the `otherHeap`.

The method `merge(Heap<T> otherHeap)` should run in $O(n)$ time where n is the total number of entries in the merged heap. The methods `removeMax()` and `insert(T data)` should run in $O(\log n)$ time and all other methods should run in $O(1)$ time, where n be the total number of entries in the heap. Use the `compareTo()` method instead of an external comparator to compare two type `T` objects.

Job Scheduling

The job scheduler of an operating system schedules the execution of jobs according to their priorities. In this homework, you will implement a simplified job scheduler. Each time slice, the scheduler picks one of the jobs (implemented by the `IJob` interface) from the job pool and serves the chosen job. Any instance of the `IJob` interface requires a certain number of time slices to finish. This number is fixed upon instantiation, but may vary between instances. When an `IJob` instance is done, the scheduler removes the finished job from the job pool. Moreover, any number of new jobs can be registered to the job pool between each time slice. The methods to be implemented are:

- `Scheduler()`: initialize the scheduler with an empty job pool.
- `void register(IJob j)`: add job `j` to the job pool.
- `IJob process()`: perform the following.
  1. Retrieve the job in the job pool with the highest priority. The job with the lower patience has the higher priority. If two jobs have the same patience, then the process with the smaller process ID has the higher priority.
  2. Call `serve()` on the job.
  3. If the job is done, then remove it from the job pool and return the job. Otherwise, do not remove it from the job pool and return `null`.

Note that each call of `register(IJob j)` and `process()` should run in $O(\log n)$ time, where n is the number of jobs in the job pool. Hint: make a wrapper class for `IJob` that implements the `Comparable` interface, and then use the `Heap` you implemented in part 1. The `IJob` interface has the following methods.

- `int getPatience()`: returns the patience of the `IJob` instance. The patience of an `IJob` instance is a non-negative integer. The lower the patience, the earlier the job wants to be scheduled.
- `int getPid()`: returns the process ID. The process ID of an `IJob` instance is a non-negative integer. Unlike the patience, no two `IJob` instances share the same process ID.

- `void serve()`: serve this `IJob` instance for this time slice. This function call may alter the patience of the served `IJob` instance. May cause an error if `serve()` is called on a job that is already finished.
- `boolean isDone()`: checks if the `IJob` instance is finished.

For the sake of testing your code, you are given a sample job class in `src/base/TestJob.java` which implements the interface `IJob`. Note that we may use a different class that implements `IJob` when we grade your code.

General Directions

- Write your name and student ID number in the comment at the top of the files in src/main directory.
- Implement all of the required methods.
- You should not import anything that is not already included in the file.
- Pay careful attention to the required return types and edge cases.
- All the codes we provided can be found in src/base directory. If you are unsure what a class/method exactly does, please refer to the code.

Submission Procedure

Submit the files in the src/main directory, excluding Main.java, as a zip file. You *must* make a zip file for submission using the Gradle build tool (refer to Compiling section).

For this assignment, you should submit only the following three files:

- Heap.java
- Scheduler.java
- your_student_id_number.txt

You must rename 2023xxxxxx.txt to your actual student ID number. Inside that text file, you must include the following text. Please be sure to write all the following text including the last period.

In completing this assignment, I pledge that I have not given nor received any unauthorized assistance.

If this file is missing, you will get a 0 on the assignment. It should be named *exactly* your student id, with no other text. For example, *2023123456.txt* is correct while something like *2023123456_pa3.txt* will receive 0.

Compiling

You can test your Java code using the following command:

```
% ./gradlew -q runTestRunner
```

You can also make a zip file for submission using the following command. The zip file named with your student id (the name of the .txt file) will lie in the “build” directory:

```
% ./gradlew -q zipSubmission
```

We provide an empty Main class for testing using standard input/output:

```
% ./gradlew -q --console=plain runMain
```

Since this file (src/main/Main.java) is not for submission, you can use any package in the file.

On Windows, use `gradlew.bat` instead of `./gradlew`.

Testing

We have provided a small test suite (src/test) for you to check your code. You can test your code by compiling and running the tester program.

Note that the test suite we will use to grade your code will be much more rigorous than the one provided here (and not necessarily a superset of the provided tests). You should consider making your own test cases to check your code more thoroughly.