UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science
Faculty of Engineering, Built Environment & IT
University of Pretoria

COS212 - Data structures and algorithms

Practical 6 Specifications: Heaps

Release date: 31-03-2025 at 06:00
Due date: 04-04-2025 at 23:59
Total marks: 280

Contents

1 General Instructions

- ***Read the entire assignment thoroughly before you start coding.***
- This assignment should be completed individually; no group effort is allowed.
- **To prevent plagiarism, every submission will be inspected with the help of dedicated software.**
- Be ready to upload your assignment well before the deadline, as no extension will be granted.
- You may not import any of the built-in Java data structures. Doing so will result in a zero mark. You may only use native 1-dimensional arrays where applicable.
- If your code does not compile, you will be awarded a mark of zero. Only the output of your program will be considered for marks, but your code may be inspected for the presence or absence of certain prescribed features.
- If your code experiences a runtime error, you will be awarded a zero mark. Runtime errors are considered unsafe programming.
- Read the entire specification before you start coding.
- **Ensure your code compiles with Java 8**
- The usage of ChatGPT and other AI-Related software is strictly forbidden and will be considered as plagiarism.
- Note that plagiarism is considered a very serious offence. Plagiarism will not be tolerated, and disciplinary action will be taken against offending students. Please refer to the University of Pretoria's plagiarism page at `https://portal.cs.up.ac.za/files/departmental-guide/`.
- Please note that there is a late deadline which is 1 hour after the initial deadline, but you will lose 20% of your mark.

2 Overview

Heaps are variations of Binary Trees where each node is either the largest or smallest value in its subtree depending on whether it is a min heap or a max heap. This allows instant access to either the minimum or maximum value at all times, and as such are often used as more efficient priority queues. Since the conditions for children in heaps are less strict than Binary Search Trees, it is very easy to convert Binary Heaps to Heaps with more than two children. These are often called D-Heaps. D-Heaps are heaps where each node has D amount of children and still follows the two heap properties.

Leftist heaps are variations of Binary Heaps which allows efficient merging. This is done by ensuring that the left subtree is always as big or bigger than the right subtree. Thus, merging on the right child is more efficient. This is usually implemented bottom-up, but a top-down version also exists called Skew Heaps. These are heaps which are not always perfectly leftist but will be leftist most of the time.

3 Tasks

For this practical, you are tasked with creating three maximum heaps. All three of these inherits from an abstract class called MaxHeap. You are tasked with implementing the following UML:

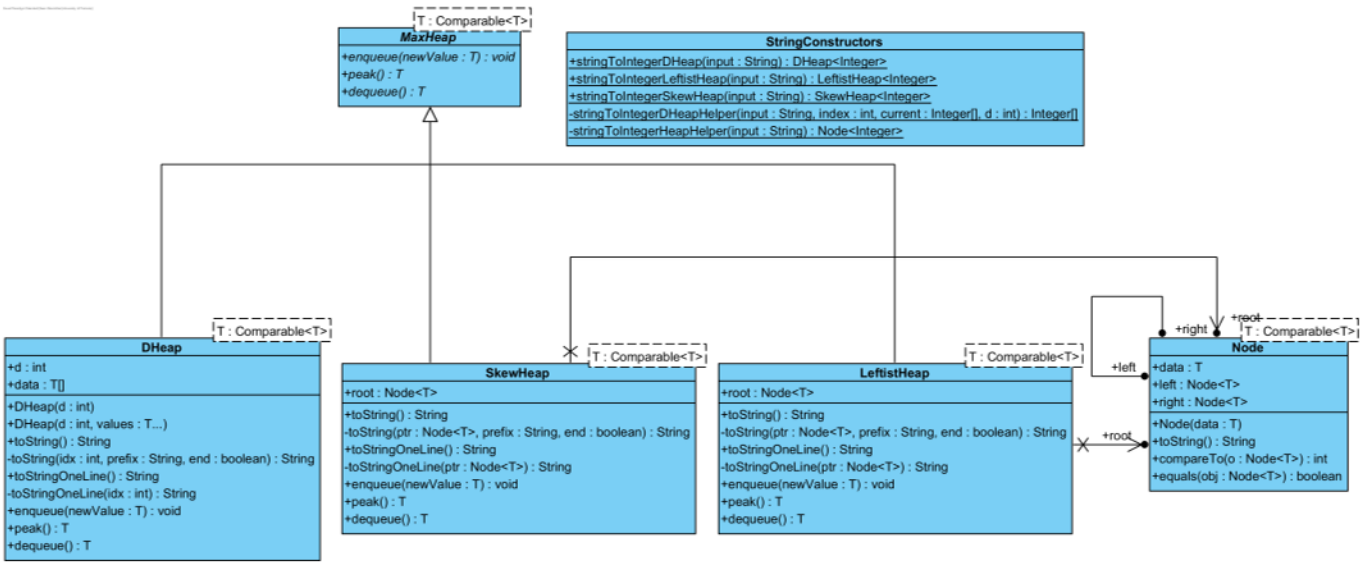

Figure 1: UML

3.1 StringConstructors

- This class is given to you. Don't change any of the provided functions as they will be used on FitchFork. You are welcome to add more functions if you want to use more data types in your testing.
- This class has three public functions which converts the output from toStringOneLine to a respective MaxHeap. Note that you are only provided with the Integer conversion functions but you can add more if you want to.

3.2 Node

- Don't change this class since it will be overwritten on FitchFork.
- This is a basic node class that will be used by the Leftist and Skew heaps.

3.3 MaxHeap

- Don't change this class since it will be overwritten on FitchFork.
- This is an abstract class that all three heap classes will inherit from.
- The following rules apply to all subclasses:
  - enqueue:
    - You may assume that duplicates values will not be added. Your code does not need to check for this you can just assume it will not happen.
  - peak:
    - If there are no values in the heap, then return `null`.
  - dequeue:
    - If there are no values in the heap, then return `null`.

3.4 DHeap

3.4.1 Members

- d
  - This is the degree of the heap.
  - This is the number of children that each node has.
- data
  - This is an array used to store the heap.
  - There will be no gaps in the array, and the array will always be perfectly sized to fit all data.

3.4.2 Functions

- Constructors
  - You are provided with a default constructor that intialises the heap to an empty heap.
  - You should implement the array constructor using the following rules:
    - Look up "Java variable arguments" if the input parameter type confuses you.
    - Ensure that the minimum degree of any heap is always 2.
    - Assign the passed-in array as the data array.
    - Apply the Floyd heapifying algorithm to the data array to ensure that the data is a valid heap.

- string functions
  - You are provided with two printing functions for debugging and testing purposes. These are toString and toStringOneLine.
  - Don't change these functions since they will be used on FitchFork.
- enqueue
  - Add the passed-in parameter to the heap.
- peak
  - Return the maximum value without removing it.
- dequeue
  - Return and remove the maximum value.

3.5 LeftistHeap, SkewHeap

3.5.1 Members

- root
  - This is the root of the heap. If the heap is empty then this will be null.

3.5.2 Functions

- string functions
  - You are provided with two printing functions for debugging and testing purposes. These are toString and toStringOneLine.
  - Don't change these functions since they will be used on FitchFork.
- enqueue
  - Add the passed-in parameter to the heap.
- peak
  - Return the maximum value without removing it.
- dequeue
  - Return and remove the maximum value.

4 Testing

As testing is a vital skill that all software developers need to know and be able to perform daily, approximately 10% of the assignment marks will be assigned to your testing skills. To do this, you will need to submit a testing main (inside the Main.java file) that will be used to test an Instructor-provided solution. You may add any helper functions to the Main.java file to aid your testing. In order to determine the coverage of your testing the jacoco tool. The following set of commands will be used to run jacoco:

```
javac *.java
rm -Rf cov
mkdir ./cov
java -javaagent:jacocoagent.jar=excludes=org.jacoco.*,destfile=./cov/output.exec
    -cp ./ Main
mv *.class ./cov
java -jar ./jacococli.jar report ./cov/output.exec --classfiles ./cov --html
    ./cov/report
```

This will generate output which we will use to determine your testing coverage. The following coverage ratio will be used:

$$\frac{\text{number of lines executed}}{\text{number of source code lines}}$$

and we will scale this ratio according to the size of the class.
The mark you will receive for the testing coverage task is determined using table 1:

Coverage ratio range	% of testing mark
0%-5%	0%
5%-20%	20%
20%-40%	40%
40%-60%	60%
60%-80%	80%
80%-100%	100%

Table 1: Mark assignment for testing

Note the top boundary for the Coverage ratio range is not inclusive except for 100%. Also, note that only the function stipulated in this specification will be considered to determine your mark. Remember that your main will be testing the instructor-provided code and as such it can only be assumed that the functions outlined in this specification are defined and implemented.

5 Upload checklist

The following files should be in the root of your archive

- DHeap.java
- LeftistHeap.java
- SkewHeap.java
- StringConstructors.java
- Main.java
- Any textfiles needed by your Main

6 Allowed libraries

- No imports allowed.

7 Submission

You need to submit your source files on the FitchFork website (https://ff.cs.up.ac.za/). All methods need to be implemented (or at least stubbed) before submission. Place the above-mentioned files in a zip named uXXXXXXXX.zip where XXXXXXXX is your student number. Your code must be able to be compiled with the Java 8 standard.

For this practical, you will have 5 upload opportunities and your best mark will be your final mark. Upload your archive to the appropriate slot on the FitchFork website well before the deadline. **No late submissions will be accepted!**