Übung Priority Queue

Table of Contents

Version: 1.0

Ausgangssituation

> In computer science, a priority queue is an abstract data type which is like a regular queue or stack data structure, but where additionally each element has a "priority" associated with it. In a priority queue, an element with high priority is served before an element with low priority. In some implementations, if two elements have the same priority, they are served according to the order in which they were enqueued, while in other implementations, ordering of elements with the same priority is undefined.
>
> — wikipedia

> In der Informatik ist eine Vorrangwarteschlange (auch Prioritätenliste, Prioritätsschlange, Prioritätswarteschlange oder englisch priority queue genannt) eine spezielle abstrakte Datenstruktur, genauer eine erweiterte Form einer Warteschlange. Den Elementen, die in die Warteschlange gelegt werden, wird eine Priorität mitgegeben, die die Reihenfolge der Abarbeitung der Elemente bestimmt.
>
> — wikipedia

Aufgabe 1: Implementierung der Priority Queue

Implementieren Sie eine eigene Priority Queue mit folgenden Klassen:

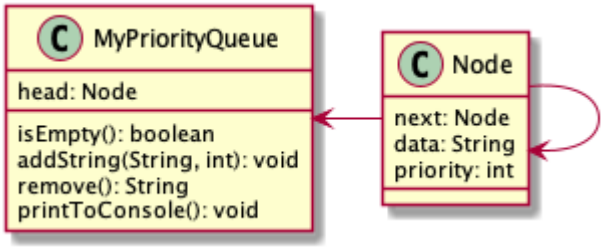


Figure 1. Class Diagram MyPriorityQueue

Es ist verboten, Arrays, Listen oder sonstige Collections zu verwenden. Die priority queue ist selbst zu implementieren.

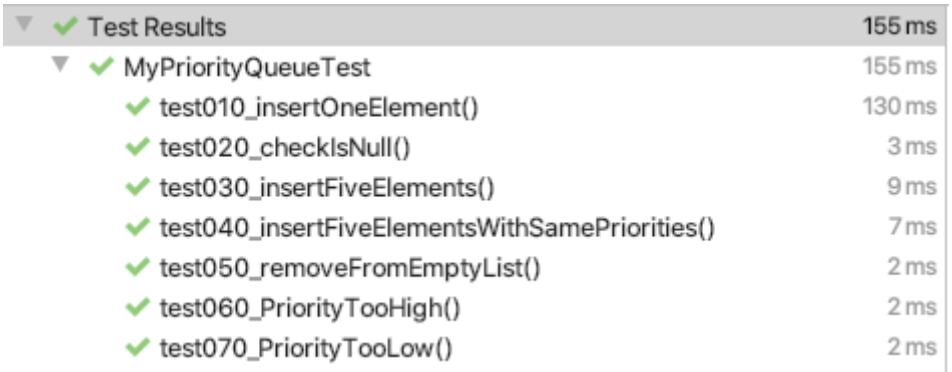
Table 1. Methoden

Methode	Beschreibung
isEmpty()	Rückgabe von `true`, wenn die priority queue leer ist, sonst `true`
add(String data, int priority)	Hinzufügen eines Elements mit einem String als Datum und der Priorität an der richtigen Stelle in der priority queue
remove()	Das Element mit der höchsten Priorität wird gelöscht und als Funktionswert zurückgegeben
printToConsole()	Der Inhalt der priority queue wird auf der Konsole ausgegeben

Die genaue Beschreibung der Methoden finden Sie in den Klassen.

Aufgabe 2: Erstellen der UnitTests

Erstellen Sie nun Unit-Tests, die die Funktion Ihrer priority queue systematisch testen



Ihre Unit-Tests sollen folgende Ausgabe ergeben:

```
<empty>
5 (chillen)
<empty>


<empty>


<empty>
80 (lernen)
80 (lernen)  5 (chillen)
80 (lernen)  15 (bingen)  5 (chillen)
90 (sporteln)  80 (lernen)  15 (bingen)  5 (chillen)
90 (sporteln)  80 (lernen)  15 (bingen)  5 (chillen)  2 (whatsappen)
80 (lernen)  15 (bingen)  5 (chillen)  2 (whatsappen)
15 (bingen)  5 (chillen)  2 (whatsappen)
5 (chillen)  2 (whatsappen)
2 (whatsappen)
<empty>


<empty>
80 (lernen)
80 (lernen)  5 (chillen)
80 (lernen)  5 (chillen)  5 (bingen)
90 (sporteln)  80 (lernen)  5 (chillen)  5 (bingen)
90 (sporteln)  80 (lernen)  5 (chillen)  5 (bingen)  2 (whatsappen)
80 (lernen)  5 (chillen)  5 (bingen)  2 (whatsappen)
5 (chillen)  5 (bingen)  2 (whatsappen)
5 (bingen)  2 (whatsappen)
2 (whatsappen)
<empty>


<empty>


<empty>
1 (priorität 1)
1 (priorität 1)  0 (priorität 120 -> 0)
0 (priorität 120 -> 0)
<empty>


<empty>
1 (priorität 1)
1 (priorität 1)  0 (priorität 10 -> 0)
0 (priorität 10 -> 0)
<empty>
```

Aufgabe 3: Zusätzliche Tests

Welche Zusätzlichen Tests fallen Ihnen noch ein - was sollte noch getestet werden. Tragen Sie Ihre Antworten in die Datei `Answers.adoc` ein.

Aufgabe 4: Datenbank

Persisitieren Sie den momentanen Imnhalt der priority queue in einer Datenbank.

Aufgabe 5: Testen Sie den Datenbankinhalt

Verwenden Sie hierzu AssertJ-DB.

1. Befüllen Sie die priority queue mit ein paar Einträgen
2. Persistieren Sie diese in der DB
3. Lesen Sie die Datenbank aus und kontrollieren Sie die Korrektheit der Einträge

Aufgabe 6: Arbeitsergebnisse dokumentieren

Tragen Sie in der Datei answers.adoc Ihre erstellten Aufgaben ein