| | | | |
|---|---|---|---|
| Module: | CSC211 (2025) | Assessment: | Term 2 Practical 4 |
| Lecturer: | Mr. C. K. Baker | Total: | 50 marks |
| Duration: | Mon 2pm – Fri 2pm | | |

Instructions

For this practical, you will implement a priority queue using a heap data structure to manage tasks for the weekly sprint in your Software Development Team at TechNow, Bellville. In a dynamic development environment, tasks often differ in urgency and complexity—some require immediate attention, while others can be scheduled for later. A heap-based priority queue provides an efficient way to organise and retrieve tasks according to their priority, ensuring that the most critical work is addressed first.

There are 3 questions. Submit a compressed (.zip) file with all your code, your signed declaration of plagiarism and list of references. The submission file should be named XXYYZZZ. zip where XXYYZZZ corresponds to your student number.

N.B.

- work incrementally and systematically
- follow the marking rubric for mark allocation per question
- the data to test your work is provided in a text file called `tasks.txt`
- save and submit a working draft on iKamva once you have completed a question
- ensure all files are correctly named and submitted by Friday at 14:00

**Question 1 – setting up [15 marks]**

Study the class definitions below.

```
class Node {
    int task_id;
    String task_description;
    int priority;
    int durationInMinutes;
}

class PriorityQueue {
    int size;
    Node[] heap;
}
```

1.1 Create two files: Node.java and PriorityQueue.java

1.2 Copy the above class definitions into their respective files.

1.3 Write constructors for the Node class:

- default constructor
- loaded constructor

1.4 Write a constructor for the PriorityQueue class:

- sets size to 0
- initialises the array with a maximum size of 10 nodes

1.5 Use encapsulation to hide attributes of the Node class. Then, write accessor (getter) and mutator (setter) methods for the attributes of the Node class.

1.6 Use encapsulation to hide attributes of the PriorityQueue class. Then, write accessor (getter) and mutator (setter) methods for the attributes of the PriorityQueue class.

**Question 2 – building the heap [15 marks]**

2.1 Implement the method void clear()

- clears the heap and resets the size to 0

2.2 Implement the method Boolean isEmpty()

- returns true if the heap is empty, otherwise returns false.

2.3 Implement the method Boolean insert(Node x)

- adds a new node to the heap, maintaining the heap property.
- ensure that the heap is ordered by the key priority

2.4 Implement the method Node delete():

- removes the root node
- reduce the heap size
- use the helper method percolateDown(int hole) to reorder the heap

2.5 Implement the helper method void percolateDown(int hole)

**Question 3 – application [20 marks]**

Create a file named Main.java in which to write code for the following.

3.1 Write code to read and build the heap from data in the file <tasks.txt>

- each line in the file contains information about a task
  - task id
  - task description
  - priority
  - task duration (in minutes)
- transform the raw data into the appropriate attributes of the Node class
  - create a Node object and add it to the priority queue
  - repeat for each task

3.2 Write code to print out the tasks in the heap in order from highest to lowest priority

- the key is the attribute priority
- the lower the number, the greater the priority
- e.g. a task with priority 1 should be completed before a task with priority 2

**Data**

tasks.txt

```
// task_id, task_description, priority, duration (minutes)
1,Clean inbox,3,15
2,Finish report,1,120
3,Call supplier,2,30
4,Team meeting,4,60
5,Write documentation,5,90
6,Fix bug #302,1,45
7,Backup database,3,20
8,Code review,2,40
9,Update website,4,75
10,Plan sprint,2,50
```

**Example output**

| Order | Task ID | Task Description | Priority | Duration (min) |
|---|---|---|---|---|
| 1 | 2 | Finish report | 1 | 120 |
| 2 | 6 | Fix bug #302 | 1 | 45 |
| … | | | | |

**Total: 50 marks**

---