National University of Computer & Emerging Sciences (NUCES) Islamabad,

Department of Computer Science

DATA STRUCTURES – FALL 2021

LAB 07

Learning Outcomes

In this lab you are expected to learn the following:

- Queue Data Structure in JAVA

Queue

This lab requires you to implement your own Queue Data Structure using Arrays, its operations and using those operations to perform different tasks.

Queue is an abstract data type or a linear data structure, in which the first element is inserted from one end called the **REAR**, and the removal of the existing element takes place from the other end called as **FRONT** thus making it **FIFO (First In First Out).**

FRONT **REAR**

2	**3**	**4**	**5**	**6**

Note: All the implementation must be based on Generics

TASK 1

Implement Class Queue, its data members, getters and setters.

TASK 2

Implement operations listed below in Class Queue.

Queue ()
A non-parameterized constructor that creates an empty queue. Where should the front and rear of an empty queue point to?

enqueue ()
Inserts the element at the rear of the queue.

dequeue ()
Removes the element from the front of the queue.

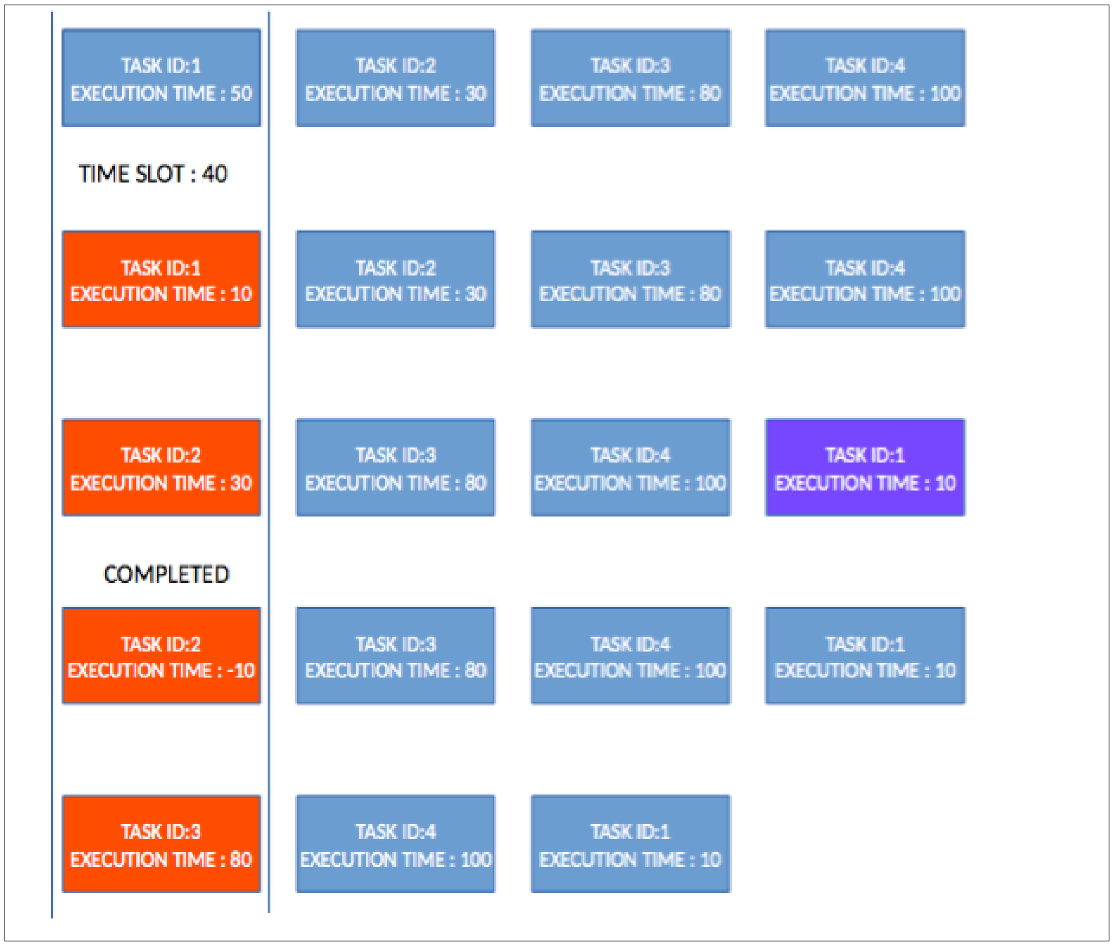
peek()
Returns the value of the element at the front of the queue.

isEmpty ()
Returns True if the queue is empty else returns False.

isFull ()
Returns True if the queue is full else returns False.

TASK 3

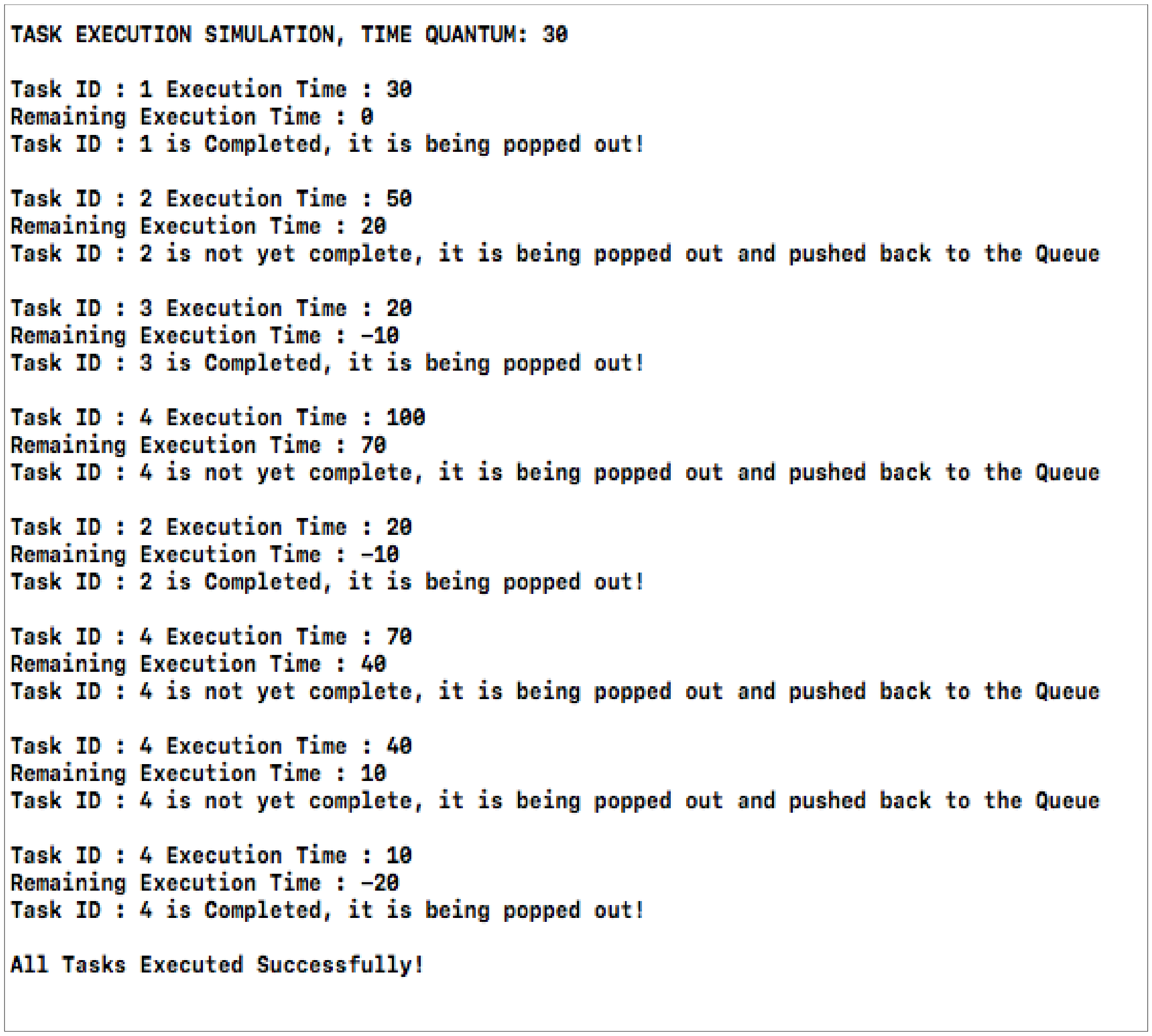
Round robin is a scheduling algorithm that an operating system uses to time share computational resources of a processor between tasks. Each task is given a specific time slot **(Quantum)** to execute on a processor (CPU Time), once this time slot expires and the task is not yet completed, it is preempted **(dequeue)** and added to the back of the queue **(enqueue)** with its **Remaining Execution Time**. Then the next task (**front)** in the queue is selected and this process continues until all tasks have finished execution. Your task is to simulate this process using a Queue.



First create a Class **Task** having **taskID** and **execution time** as data members also create its constructor, getters and setters. The queue will contain objects of the Task class as shown in figure above.

Now Implement a function **roundRobin()** in Class **Main** that takes as an input a Queue of Tasks and quantum. Then it simulates the process of task execution.

Sample of output is displayed below:



```
TASK EXECUTION SIMULATION, TIME QUANTUM: 30

Task ID : 1 Execution Time : 30
Remaining Execution Time : 0
Task ID : 1 is Completed, it is being popped out!

Task ID : 2 Execution Time : 50
Remaining Execution Time : 20
Task ID : 2 is not yet complete, it is being popped out and pushed back to the Queue

Task ID : 3 Execution Time : 20
Remaining Execution Time : -10
Task ID : 3 is Completed, it is being popped out!

Task ID : 4 Execution Time : 100
Remaining Execution Time : 70
Task ID : 4 is not yet complete, it is being popped out and pushed back to the Queue

Task ID : 2 Execution Time : 20
Remaining Execution Time : -10
Task ID : 2 is Completed, it is being popped out!

Task ID : 4 Execution Time : 70
Remaining Execution Time : 40
Task ID : 4 is not yet complete, it is being popped out and pushed back to the Queue

Task ID : 4 Execution Time : 40
Remaining Execution Time : 10
Task ID : 4 is not yet complete, it is being popped out and pushed back to the Queue

Task ID : 4 Execution Time : 10
Remaining Execution Time : -20
Task ID : 4 is Completed, it is being popped out!

All Tasks Executed Successfully!
```