Simulator assignment

Mahmoud Kassab

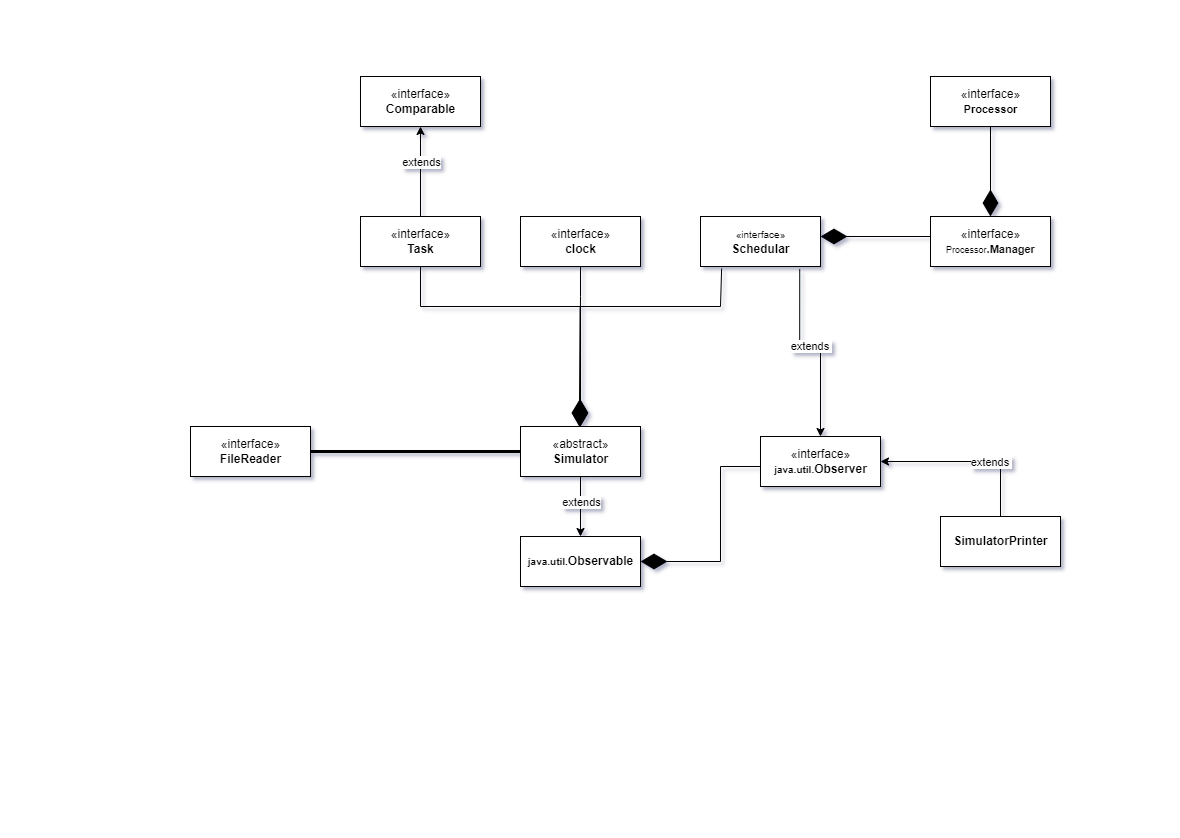
Introduction:

In this assignment we are asked to implement a basic CPU simulator that schedule tasks based on priority.

Design:

Since almost every operation occurs only when the clock is invoked, I believe the Observer Design pattern will be a good choice, especially for future expansion of the system (e.g., console printer ,store reports in files, etc.…).

Note: even though the java.util.Observable/Observer are deprecated[1], but they will do the job well.



UML Class Diagram

Data Structure & Algorithms:

I have used different types of data structures in for this project, taking in consideration the complexity of the each one of them to achieve the best performance possible.

1. tasks = Queue<Tasks>:

Represents all the tasks read from the file but hasn’t arrived yet, Since the I/O operations are expensive, reading all the tasks from the file at once and store them into a queue will be more efficient than reading from the file on each clock. For any future expansion if we wanted to dynamically add tasks in run time, it would be inserted in the queue.

1. Ready Tasks = PrioritiyQueue<Tasks>:

Represents all the tasks ready for execution, since the **Task** interface extends the **Comparable** interface, the priority queue will sort all the tasks based on their priority and execution time, as implemented in the subclass of the **Task** interface.

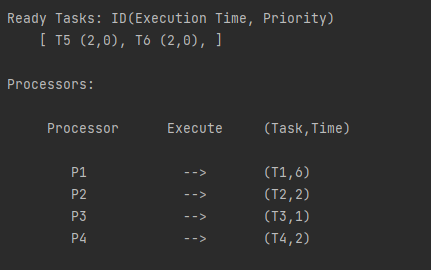
1. Ready Processors & Busy Processors = Queue<Processor>:

Separating the busy and ready processors will reduce the complexity when executing all the processors or assigning tasks to ready processors. There is no special reason to choose the queue to store them, just taking advantage of the FIFO.

1. Getter methods = Stream():

All data structures getters return streams to generalize the communication between the classes, and to take advantage of the streams pipeline operations, furthermore, streams do not modify the source object.

Tests:



The figure shows T5 and T6 are in the ready queue waiting for execution waiting for any processor to become available.

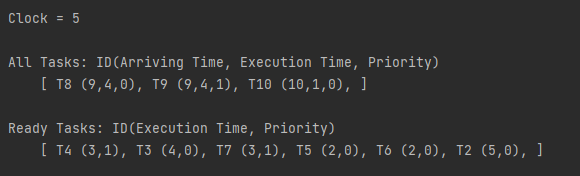
A picture containing text

Description automatically generated

The figure shows P1 and P2 available and ready to execute tasks waiting for any task to become available.

Since we are using a priority queue to store the ready tasks, only the first element in the queue will be sorted.

At the 5th clock T4 has to highest priority among the entire queue, but when T9 arrives at the 9th clock, the ready queue sets T9 as the highest priority, since its priority value = 1 and has the highest execution time among all ties.



5th Clock

Text

Description automatically generated

9th Clock

References:

1) [Java's Observer and Observable Are Deprecated in JDK 9](https://dzone.com/articles/javas-observer-and-observable-are-deprecated-in-jd)