Scheduling algorithm OS project

**Scheduling algorithms:**

* *Shortest remaining time next:*

A pre-emptive that checks the remaining running time of all queued processes with each arrival of a new process. The schedule then runs the process with the shortest remaining time. This algorithm prefers short run time process and may starve longer processes.

* *Fair-share algorithm:*

A pre-emptive algorithm that assigns each user a priority and thus a cpu time share. The processes of a certain user share the assigned cpu time equally using the round robin strategy. This may starve processes of a user that is assigned a very low priority.

* *Lottery bucket algorithm:*

A pre-emptive algorithm that assigns a certain number of tickets to each process based on the priority of the user owning that ticket. Following each time quantum, a random ticket is selected and the process that owns that ticket is run for the next quantum.

**Main Classes:**

* *Components.Process:*

Represents a user process with a name, arrival time, running time and an owner username.

* *Components.ticket:*

Represents a ticket to be given to a process in the lottery bucket algorithm.

* *Components.User:*

Represents a user with an assigned priority and a name.

* *ShedulingAlgorithms.* *SchedulingAlgorithm:*

An abstract class to act as a parent of all scheduling algorithm classes. Defines two instance variables, a list of users and a list of processes.

* *ShedulingAlgorithms.* *FairShare:*

Represents the fair-share algorithm defines method simulate that returns an array of processes arranged in time slots according to the order of running.

The method starts by calculating the total time slots needed based on the total runtime, and then assigns a process to each time slot based on who has the turn to run. The turn is determined by the priority of the user and the number of process it s/he owns.

* *ShedulingAlgorithms.* *Lottery:*

Represents the lottery bucket algorithm, and defines method simulate. The method starts by calculating the time slots needed based on the total runtime of all processes.

Each process is then assigned a several tickets based on the priority of the owner of the process. A ticket is selected using its index and a random number generator (Java.util.Random) and the process owning that ticket is assigned to the current time slot. Returns an array of processes arranged according to the current time slots.

* *ShedulingAlgorithms.* *ShortestRemainingTimeNext:*

Represents the shortest remaining time next algorithm. Defines a method simulate that starts by calculating the total number of time slots needed based on the total runtime of all processes. This method considers the arriving time of all processes and for each time slot picks the process with the minimum running time of all arrived processes. This process is assigned to the current time. Returns an array of processes arranged according to the current time slots.

* *Simulation.* *Simulation:*

Controls the simulation of the project. Starts by taking input from the user about the number of users in the simulation, the priority of each user and the number of processes of each user. The user then selects the scheduling algorithm to be used in the simulation and the resulting schedule is output as a table.

**How to run:**

Start by running the Simulation.Simulation main method, Enter the number corresponding to the algorithm you want to simulate, Enter the number of users, then enter each user as a pair of name and priority sperated by a space, then enter the number of processes, then enter the process details as in the project description. The program will simulate the algorithm and then prints each process of which user is run in which timeslot.

**References:**

Process Scheduling Lecture 5