Report to assignment number 6

Maryam Soleimani

CPU:

In the startSimulation method we have the arrlist tasks with the type of Task then while tasks is not empty we keep defining a variable called shortest and assign the first index of tasks arrlist to it.

Our goal is to find the shortest task and make a thread for that. So what we do here is that we get the first index of the tasks arrlist and keep compare it with other indices and whenever we found an index with shorter processing time, we assign that one to shortest. When we keep doing this in a loop at the end of the loop we end up having the task with the shortest processing time. After getting out of the loop we remove that shortest one from the tasks and then make a thread for that then we run that thread(start) and after it Is stopped we add the id of that task to the executed tasks array list. This process continues until the tasks arraylist becomes empty and we have the executedTasks array list which is sorted from the shortest processing time to the longest.

FindMultiples:

Here first I define a public static arraylist called dividends to store the numbers that are divided by 3,5 or 7.

Then I defined a synchronized method to add items to dividends. It is synchronized in order to prevent that some threads at the same time effect dividends. Then we have the class thread and attributes : devisor (3,5,7) and n and then the thread method. In the run method it checks the numbers one to n and the ones that are divided will be added to the dividends arrlist.

In the getSum method we define three individual threads for 3,5 and 7 and start them and then join them.

So far some of the dividends we have stored in the dividends arrlist are repeated so in order to eliminate them and have only one of each of them we can use the following piece of code:

```
Set<Integer> set = new HashSet<>(dividends);
    dividends.clear();
    dividends.addAll(set);
```

Then by looping through dividends we can get the sum of all elements:

```
for (int i : dividends){
      sum+=i;
    }
```