Optional Assignment

Assem Abikhanova

Classes

- *Main* - main runnable class
- *Statistics* - class for printing final stats on the screen

```
public Statistics( ArrayList<Process> processes, double ave_waiting_time,
double ave_response_time, double ave_preemption) // constructor
public void print() // takes list of Process'es and prints them with average
values
```

- *InputProcessor* - responsible for reading data from input or arguments

```
public void readInput() //CASE: No input given
public void readFile(int procs_count, String filename) //CASE: User provides
number of processes (procs_count) and filename
public void generateInput(int num_of_procs) //CASE: User only gives number of
processes to run -> have to generate processes randomly
```

- *Process* - process object

```
protected long truntime; // total runtime
protected long thisruntime; // current running time
protected long lastpreempted; // last time when was preempted -> used for
calculating wait time
protected double vruntime; // vruntime
protected double slice; // slice time for which it runs before being inserted
back to the tree
private double weight; // weight of a process
private final int process_number;
private final int priority;
private final int burst_time; // in microseconds
private final int arrival_time; // in microseconds
private int wait_time; // in microseconds -> for stats
private int response_time; // in microseconds -> for stats
private int preempted_count; -> for stats
```

- *RBTNode* - Node of Red-Black tree with data as Process class and pointers to right, left children and parent node

```
//get all data with getters and setters
```

- *MyRedBlackTree* - Red-Black tree

```
public MyRedBlackTree(RBTNode root) // constructor
public RBTNode getRoot()  // returns root node
public RBTNode getMin() // returns min node in O(1)
public boolean insert(Process node) // return true on success; else false
public RBTNode delete(Process proc) // returns deleted node
```

```
public RBTNode delete(RBTNode node) // returns deleted node
public void print() // prints tree
```

Input format expected

User is expected to run file in two ways: 1. With the **number of programs to run** (all other fields will be generated randomly) 2. With the **number of programs to run** and the **filename** from which all information will be read 3. **No argument** (user will be asked to choose an option from ablove two cases)

```
File format expected to have:
#process | priority | burst_time_ms | arrival_time_ms
------------------------------------------------------------ file file_name.txt
1     0     325352     1
...
k     0     10036     1339
k+1 -10 12415     1346
...
n     10     141516     25145
------------------------------------------------------------
```

Formulas used in the program

```
sched_latency = 6.00
sched_min_granularity = 0.75
sched_wakeup_granularity = 1.00
period = max(sched_latency, active_num_of_procs * sched_min_granularity)
process_weight = 1024 * 1.25^(-priority)
vruntime += 1024 * process_current_running_time / process_weight
slice = (process_weight / total_weight) * period
```

NOTE:

Reference List

The following recources were used as a reference to Red-Black tree

- YouTube tutorial
- Live demonstration of RBT
- Node deletion process
- CFS formulas, i.e. period, vruntime, slice
- Weight formula was derived from values from this website