Scheduling Algorithms

Question

This project involves implementing several different process scheduling algorithms. The scheduler will be assigned a predefined set of tasks and will schedule the tasks based on the selected scheduling algorithm. Each task is assigned a priority and CPU burst. The following scheduling algorithms will be implemented:

1. First-come, first-served (FCFS), which schedules tasks in the order in which they request the CPU.
2. Shortest-job-first (SJF), which schedules tasks in order of the length of the tasks' next CPU burst.
3. Priority scheduling, which schedules tasks based on priority.
4. Round-robin (RR) scheduling, where each task is run for a time quantum (or for the remainder of its CPU burst).
5. Priority with round-robin, which schedules tasks in order of priority and uses round-robin scheduling for tasks with equal priority.

Answer

FCFS

对于FCFS来说，只需要对Driver中定义的queue每次取出队列第一个数据并输出即可。拓展Algorithm类，重载schedule和pickNextTask，需要注意的是，这里的queue的数据类型是List，get之后需要手动删除队首。

```
@Override
public void schedule(){
    System.out.println("FCFS");
    while(!queue.isEmpty()){
        Task tmp = pickNextTask();
        System.out.println(tmp.getName()+"\t"+tmp.getPriority()+"\t"+tmp.getBurst());
        queue.remove(0);
    }
}
@Override
public Task pickNextTask(){
    return this.queue.get(0);
}
```

运行结果如下：

```
C:\Users\lenovo\IdeaProjects\schedule\src>java Driver FCFS schedule.txt
FCFS
T1      4       20
T2      3       25
T3      3       25
T4      5       15
T5      5       20
T6      1       10
T7      3       30
T8      10      25
```

SJF

这里用java的PriorityQueue库，这里定义了一个新的类ComparableSmall，这个类拓展了优先级队列Comparable类，私有变量包括name、tid、priority、burst，在SJF中优先级队列按照burst从小到大排列。重载Compatable类的compareTo函数，如下：

```
    @Override
    public int compareTo(ComparableSmall cp) {

        int cpTime = cp.getTime();
        String cpName = cp.getName();

        if (this.getTime() < cpTime) {
            return -1;
        }

        if (this.getTime() > cpTime) {
            return 1;
        }

        if (this.getTime() == cpTime) {
            return this.getName().compareTo(cpName);
        }

        // Should not reach here
        return 0;
    }
```

然后将queue中的Task放入到优先级队列中，最后输出即可，重载的schedule函数如下：

```
@Override
public void schedule(){
    Queue<ComparableSmall> sjfQueue = new PriorityQueue<>();
    while(!queue.isEmpty()){
        Task tmp = pickNextTask();
        sjfQueue.add(new ComparableSmall(tmp.getPriority(),
tmp.getName(),tmp.getBurst()));
        queue.remove(0);
```

```
        }
        System.out.println("Shortest-job-first");
        while (sjfQueue.peek() != null) {
            ComparableSmall tmp = sjfQueue.peek();
            System.out.println(tmp.getName()+"\t"+tmp.getId()+"\t"+tmp.getTime());
            sjfQueue.remove();
        }
    }
```

运行结果如下：

```
C:\Users\lenovo\IdeaProjects\schedule\src>java Driver SJF schedule.txt
Shortest-job-first
T6      1       10
T4      5       15
T1      4       20
T5      5       20
T2      3       25
T3      3       25
T8      10      25
T7      3       30
```

Priority

priority和SJF类似，区别在于SJF的优先级队列是按照burst从小到大排序的，而priority是按照priority从大到小排序的，只需要在SJF的基础上，重载Comparable的compareTo的函数即可，代码如下：

```
@Override
public int compareTo(ComparableBig cp) {

    int cpPriority = cp.getPriority();
    String cpName = cp.getName();

    if (this.getPriority() > cpPriority) {
        return -1;
    }

    if (this.getPriority() < cpPriority) {
        return 1;
    }

    if (this.getPriority() == cpPriority) {
        return this.getName().compareTo(cpName);
    }

    // Should not reach here
    return 0;
}
```

代码运行结果：

```
C:\Users\lenovo\IdeaProjects\schedule\src>java Driver PRI schedule.txt
Priority Scheduling
T8      10      25
T4      5       15
T5      5       20
T1      4       20
T2      3       25
T3      3       25
T7      3       30
T6      1       10
```

RR

对于RR来说，在一个Task没有运行完之前，它会一直在队列中，这样实际上RR就是一个先进先出的队列，如果burst小于时间切片的大小，则不再放入队列中，如果大于，则burst应减去splice，再加入到队尾。 代码如下：

```
while(!RRQueue.isEmpty()){
    Task tmp = RRQueue.poll();
    System.out.println("Will run Name: " + tmp.getName());
    System.out.println("Tid: " + tmp.getTid());
    System.out.println("Priority: " + tmp.getPriority());
    System.out.println("Burst: "+tmp.getBurst() + "\n");

    if(tmp.getBurst()>10){
        tmp.setBurst(tmp.getBurst()-splice);
        RRQueue.offer(tmp);
    }
    else{
        System.out.println("Task "+ tmp.getName() +" finished.\n");
    }
}
```

运行结果如下：

> C:\Users\lenovo\IdeaProjects\schedule\src>java Driver RR schedule.txt RR Scheduling
>
> Will run Name: T1 Tid: 0 Priority: 4 Burst: 20
>
> Will run Name: T2 Tid: 1 Priority: 3 Burst: 25
>
> Will run Name: T3 Tid: 2 Priority: 3 Burst: 25
>
> Will run Name: T4 Tid: 3 Priority: 5 Burst: 15
>
> Will run Name: T5 Tid: 4 Priority: 5 Burst: 20
>
> Will run Name: T6 Tid: 5 Priority: 1 Burst: 10
>
> Task T6 finished.
>
> Will run Name: T7 Tid: 6 Priority: 3 Burst: 30
>
> Will run Name: T8 Tid: 7 Priority: 10 Burst: 25

> Will run Name: T1 Tid: 0 Priority: 4 Burst: 10

> Task T1 finished.

> Will run Name: T2 Tid: 1 Priority: 3 Burst: 15

> Will run Name: T3 Tid: 2 Priority: 3 Burst: 15

> Will run Name: T4 Tid: 3 Priority: 5 Burst: 5

> Task T4 finished.

> Will run Name: T5 Tid: 4 Priority: 5 Burst: 10

> Task T5 finished.

> Will run Name: T7 Tid: 6 Priority: 3 Burst: 20

> Will run Name: T8 Tid: 7 Priority: 10 Burst: 15

> Will run Name: T2 Tid: 1 Priority: 3 Burst: 5

> Task T2 finished.

> Will run Name: T3 Tid: 2 Priority: 3 Burst: 5

> Task T3 finished.

> Will run Name: T7 Tid: 6 Priority: 3 Burst: 10

> Task T7 finished.

> Will run Name: T8 Tid: 7 Priority: 10 Burst: 5

> Task T8 finished.

Priority RR

对于priority RR来讲，首先先用priority中的优先级队列进行排序，然后对于优先级队列里的priority相同的值进行RR操作。具体的判断条件是，当队首Task的priority和当前正在操作的Task相同时，将当前Task和队首Task放入队列RRQueue中，直到队首Task的priority不等于当前的Task或者队列为空时，对RRQueue进行与RR相同的操作即可。代码如下：

```
while (pQueue.peek() != null) {
    ComparableBig tmp = pQueue.poll();
    if(pQueue.peek() != null && pQueue.peek().getId() == tmp.getId()){
        Queue<Task> RRQueue = new LinkedList<>();
        Integer splice = 10;
        RRQueue.offer(new Task(tmp.getName(),tmp.getId(),tmp.getTime()));
        while (pQueue.peek() != null && pQueue.peek().getId() == tmp.getId()){
            tmp = pQueue.poll();
            RRQueue.offer(new Task(tmp.getName(),tmp.getId(), tmp.getTime()));
        }
        while(!RRQueue.isEmpty()){
            Task Tmp = RRQueue.poll();
            System.out.println("Will run Name: " + Tmp.getName());
            System.out.println("Priority: " + Tmp.getPriority());
            System.out.println("Burst: "+Tmp.getBurst() + "\n");
```

```
            if(Tmp.getBurst()>10){
                Tmp.setBurst(Tmp.getBurst()-splice);
                RRQueue.offer(Tmp);
            }
            else{
                System.out.println("Task "+ Tmp.getName() +" finished.\n");
            }
        }
    }
    else {
        System.out.println("Will run Name: " + tmp.getName());
        System.out.println("Priority: " + tmp.getId());
        System.out.println("Burst: "+tmp.getTime() + "\n");
        System.out.println("Task " + tmp.getName() + " finished\n");
    }

}
```

运行结果如下：

> C:\Users\lenovo\IdeaProjects\schedule\src>java Driver PRI-RR schedule.txt Priority with RR Scheduling
>
> Will run Name: T8 Priority: 10 Burst: 25
>
> Task T8 finished
>
> Will run Name: T4 Priority: 5 Burst: 15
>
> Will run Name: T5 Priority: 5 Burst: 20
>
> Will run Name: T4 Priority: 5 Burst: 5
>
> Task T4 finished.
>
> Will run Name: T5 Priority: 5 Burst: 10
>
> Task T5 finished.
>
> Will run Name: T1 Priority: 4 Burst: 20
>
> Task T1 finished
>
> Will run Name: T2 Priority: 3 Burst: 25
>
> Will run Name: T3 Priority: 3 Burst: 25
>
> Will run Name: T7 Priority: 3 Burst: 30
>
> Will run Name: T2 Priority: 3 Burst: 15
>
> Will run Name: T3 Priority: 3 Burst: 15
>
> Will run Name: T7 Priority: 3 Burst: 20
>
> Will run Name: T2 Priority: 3 Burst: 5
>
> Task T2 finished.

> Will run Name: T3 Priority: 3 Burst: 5

> Task T3 finished.

> Will run Name: T7 Priority: 3 Burst: 10

> Task T7 finished.

> Will run Name: T6 Priority: 1 Burst: 10

> Task T6 finished