Department of Computer Science

CS21120: Data Structures and Algorithm Analysis
Assignment 2 - **Scheduling**

1 Background

A scheduler is a central part of a multitasking computer operating system. Your task in this assignment is to implement a scheduling algorithm to work within a simulation framework, and to run some experiments, documenting your results in a professional manner as a scientific report.

2 The tasks in detail

For this assignment you are required to do three things:

1. Understand some existing code.
2. Write new classes to fit into the existing structure.
3. Write a report that discusses the results that you obtain from running the new code in the existing framework.

2.1 Understand some existing code

Obtain the code from the web server at the location http://pcbo.dcs.aber.ac.uk/blog/teaching/cs21120 or from BlackBoard.

The code consists a GUI which runs experiments together with a class that loads scheduler routines based on the contents of a properties file, and some code that implements a FirstComeFirstServed scheduling algorithm.

The Application can be run by using the command line

```
java uk.ac.aber.rcs.cs211.schedulersim.SimulatorApplication
```

This class produces results as shown from a demo run below:

```
Simulator run Fri Jan 12 12:43:46 GMT 2007
uk.ac.aber.rcs.cs211.schedulersim.scheduler.FirstComeFirstServed
DataFile name:  /Users/rcs/Documents/workspace/211SchedulerSim/bin/Test.jobs
CPU2    13
CPU1    37
CPU3    56
MIXED1  63
IO1     75
Mean elapsed duration   48
Total CPU time  75
```

You are required to integrate your code with the existing code, so spend some time looking at the source code and running it to familiarise yourself with how it operates. Pay particular attention to the `uk.ac.aber.rcs.cs211.schedulersim.Scheduler` interface, as you will have to implement this interface in your classes. There is an example implementation in `uk.ac.aber.rcs.cs211.schedulersim.scheduler.FirstComeFirstServed` that you should look at, as you may be able to adapt this rather than write from scratch.

Make sure you understand the `Schedulers.properties` file, and the data files. The data file structure is described at the beginning of each file.

2.2 Write new classes to fit into the existing structure.

You must use your understanding of the code to write at least one new scheduling routine to add to the simulation system. I suggest that you use the existing class implementing *First Come First Served* as a starting point.

I would suggest that the easiest scheduler that you could provide an implementation for would be a *Round Robin* scheduler. This scheduler gives each process on the ready queue a slice of time on the CPU and then returns the executed job to the back of the queue to wait its turn again. This should require very small changes to the *First Come First Served* algorithm.

This is the **ABSOLUTE MINIMUM** of development work that you should do. In order to make your report more convincing, and more interesting, you should consider implementing at least one more scheduling algorithm.

You might want to consider implementing a priority queue, either one that proritises on overall job length, or prioritises on amount of I/O done so far in the job, or some ratio of run time and I/O. If you're feeling confident with your coding, you might want to try implementing more than one of these algorithms. Wikipedia offers some simple descriptions of a number of scheduling algorithms including *Shortest Job Next*, and *Shortest Remaining Time*

If you use a non-deterministic scheduler algorithm, like *Lottery Scheduling*, then you should realise that you will not get the same results each time you run it, and so will have to make multiple runs in order to get results for the report.

In order to make your algorithms appear in the menu, you should add them into the *Schedulers.properties* file.

You should **NOT** modify the provided classes in any way whatsoever. If you do, you will be marked down.

2.3 Discuss the results that you obtain from running the new code in the existing framework.

There are a number of data files that have been provided which represent a set of running programs. You should test each of these data files together with each of the algorithms, and keep a record of the results of your experiments.

You may want to try creating your own data files to run other experiments, and the file format is described in comments at the beginning of each file.

You should examine both the individual run times for jobs, as well as mean and total elapsed times.

2.4 The Experiments and Report

The report should be in the style of a scientific report, presenting methodology, results and conclusions. It should have an abstract on the first page, and an executive summary at the end. The style of the report must use the ACM SIG Proceedings Template to format your report, which can be downloaded from their website http://www.acm.org/sigs/pubs/proceed/template.html

The report should be a **maximum of 6 pages**, including diagrams, but not including source code. The source code should not be part of the report, but must be included in the submission. You should only submit source code that is not distributed to you.

In your report you must address the following issues:

- How quickly the first job finishes
- The mean time to finish a job
- The overall processing time for all jobs

You should compare and contrast different algorithms with several data files, and comment on whether the results are what you expected.

3 The Submission

- You must submit a report detailing the techniques used and the results of your analysis in a formal style. You will note that from the marking scheme (below) that the majority of the marks are for the report. You will have to produce the code in order to write the report, but you are expected to produce a quality scientific report for the majority of the marks.
- You must submit a printed copy of all *your* Java source code for all new and modified classes, which should be well structured and commented. You do not have to submit printed copies of the source code that I have provided on the web site.
- You must also submit an electronic copy of your report and code to blackboard. There will be 2 submission entries on blackboard, one that will take a submission of the PDF version of your report (for SafeAssign purposes) and the other that will take the submission of the ZIP file containing all your code.

  NOTE: Your files must be named abc9_cs211_a2.pdf and abc9_cs211_a2.zip where abc9 is replaced with your username. The zip file must be structured such that you have four directories in the top level of it:

  - *report* containing the source and PDF of the report;
  - *code* containing the source code;
  - *design* containing class diagrams and brief descriptions of classes;
  - *testing* containing testing evidence.

The assignment must be handed in to the letterbox in reception as well as submitted electronically to blackboard - the standard Computer Science collection system. You must attach a signed declaration of originality.

This assignment is exempt from anonymous marking, and as such should not be made anonymous. You should use @author tags in the JavaDoc comments in your source code and other places where appropriate.

This assignment is due on Friday 3rd May 2013 between 10am and 4pm. Late submissions will not be accepted and will be treated in line with University policy. Printing queues are not an excuse for a late hand-in, please ensure that you have the assignment ready to hand in at least 24 hours before the deadline.

4 The Marking Scheme

This assignment is worth 25% of the marks for the course CS21120, therefore you are expected to spend somewhere around 30 hours working on it.

Implementation of Algorithms 30% (A maximum of 10% per algorithm implemented)

Report on experiments 70%

Richard Shipman March 14, 2013