Alesha Ray
Due 11/29/2020

Page-Replacement Algorithm

(Report)

Overview

This implementation of the Page-Replacement Algorithm program was written in an Object Oriented format. Classes were created for the three algorithms to be implemented for calculating page faults: FIFO (First-In First-Out), OPT (Optimal), and LRU (Least Recently Used). There is also an abstract Algorithms class from which the three previous algorithms extend, and a Main class which drives the program.

The program takes in a specific frame count from the user. The frame count and the page-reference string, which is a randomly generated list of 100 values ranging from 0-49, is passed to all three page-replacement algorithms. Each algorithm's run method is called which will then count the number of page faults based on the reference string.

Explanation of Results

Since the program can be run by script or manually, the results can be seen in two different ways. If the program is run manually, the user has the option of whether or not to choose *verbose* output. If the *-v* flag is passed in when the program is run, the name of the page-replacement algorithm, the run (index) indicator, and the number of page faults is displayed for each algorithm.

If the program is run using the ***runs*** script, it will loop 30 times, each of which the loop count will be passed in as the specified frame count. The program will then generate a data log file which contains the run count, algorithm name, page-replacement string of 100 values, and page fault for each algorithm on a separate file labeled by the specific run. The data logs are then moved into the data folder and can be used for analysis purposes.

Alesha Ray
Due 11/29/2020

Run Instructions

The program is provided with a ***runs*** script that makes running the application simple. The *runs* script compiles the program and runs it with each of 30 different frame counts, starting from 1. The *runs* script then prints out a log file for each of the different runs based on the frame count, and then copies the log file to the data folder. The log files are then used for data analysis purposes such as a plot graph that shows the output of the 30 different runs for each algorithm (90 runs total).

Running the program with script:

```
./runs.sh
```

After navigating into the src/main/java folder of the project directory, where the runs.sh script is located, enter the above line. This will create the class files for each class, run the program, and save log files into the data folder.

The console will automatically print out the status of the log files being generated, written to, and closed. These files are moved into the data folder and are labeled based on the loop index which is also the number of frames for that fun. These files can be used to create a plot graph similar to the one shown below.

Running the program manually:

```
javac com/aleshamray/pra/Main.java
java com.aleshamray.pra.Main ## -v (optional)
```

After navigating into the src/main/java folder of the project directory, run the above commands. The first command compiles the program. The second runs the program, passing in the number of frames (indicated by ## above), and a -v flag which will show the algorithm name and page fault count for each algorithm. The -v flag is optional as a single log file with the output of all runs will be created, similar to the individual data files from the scripted run.

Cleaning up the created files

```
./clean.sh
```

This will remove all class object files and the log file(s) created as a result of the program.

Alesha Ray
Due 11/29/2020

Graph Plot

(This plot is also available in larger pdf format with assignment submission)

Fame Count	FIFO Page Faults	OPT Page Faults	LRU Page Faults
1	100	100	100
2	93	83	93
3	96	83	96
4	99	83	99
5	85	69	85
6	86	67	86
7	89	64	89
8	84	64	84
9	86	58	85
10	80	53	78
11	81	56	83
12	74	53	72
13	78	52	79
14	72	55	74
15	72	53	72
16	62	50	66
17	65	45	72
18	75	47	71
19	63	50	64
20	69	47	65
21	65	44	65
22	59	43	58
23	56	43	58
24	63	45	63
25	63	43	58
26	52	44	54
27	56	42	55
28	58	44	58
29	55	45	49
30	50	44	52