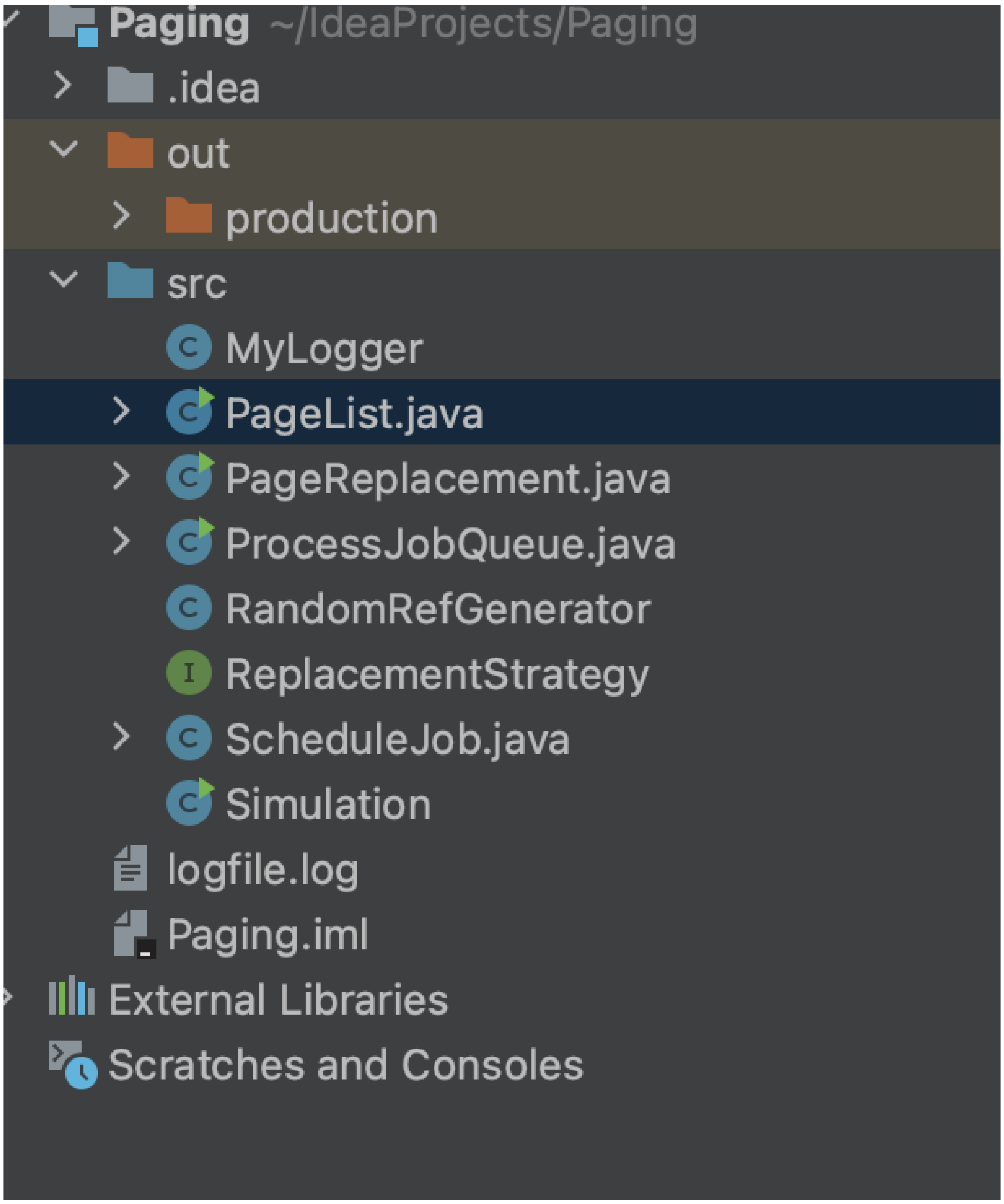
OS HW 3 Simulating Page Replacement Algorithms

Student Details

Divya Megharwade 07700009920 dmeharwade@scu.edu

File Structure



```
Paging ~/IdeaProjects/Paging
  .idea
  out
    production
  src
    MyLogger
    PageList.java
    PageReplacement.java
    ProcessJobQueue.java
    RandomRefGenerator
    ReplacementStrategy
    ScheduleJob.java
    Simulation
  logfile.log
  Paging.iml
External Libraries
Scratches and Consoles
```

Instructions on how to run the program

The attached zip file contains two directories
src - which consists of all the source files
out - that consists of all the executable files.

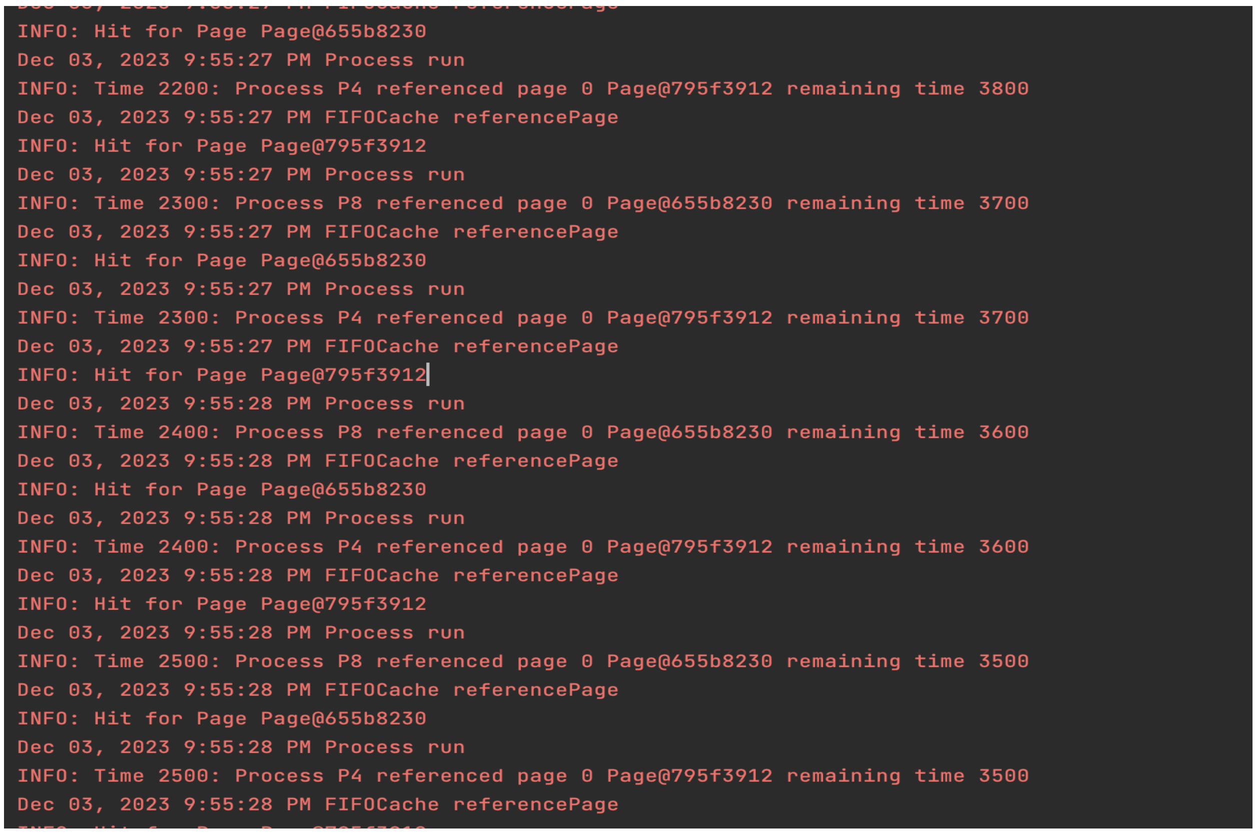
`To run cd out/production and execute`
`java Simulation`

Output

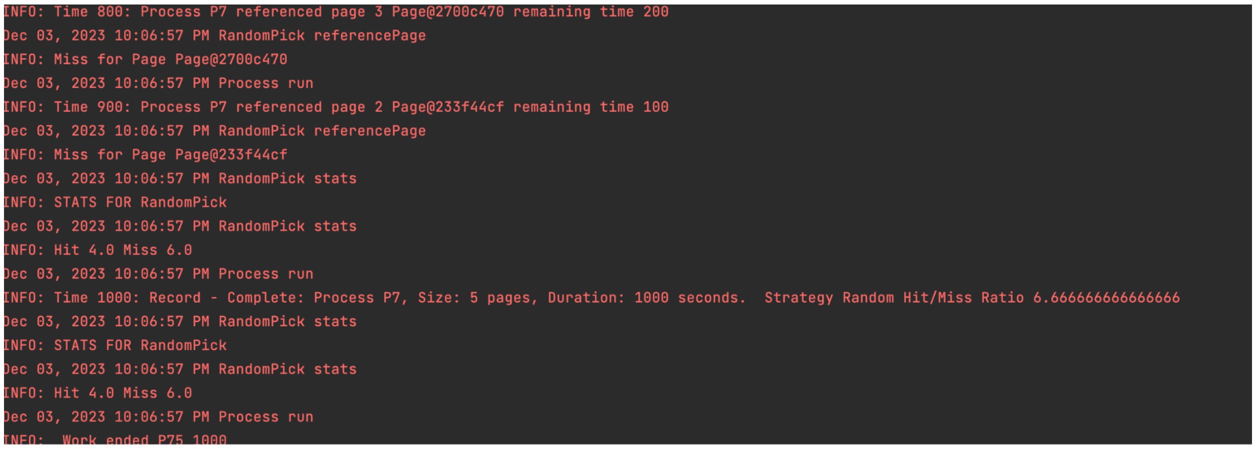
The Program takes around 25-30 minutes to complete.

Logs showing the freepages, allocation, task completion and hitRatio with pageReplacement strategy is printed periodically.

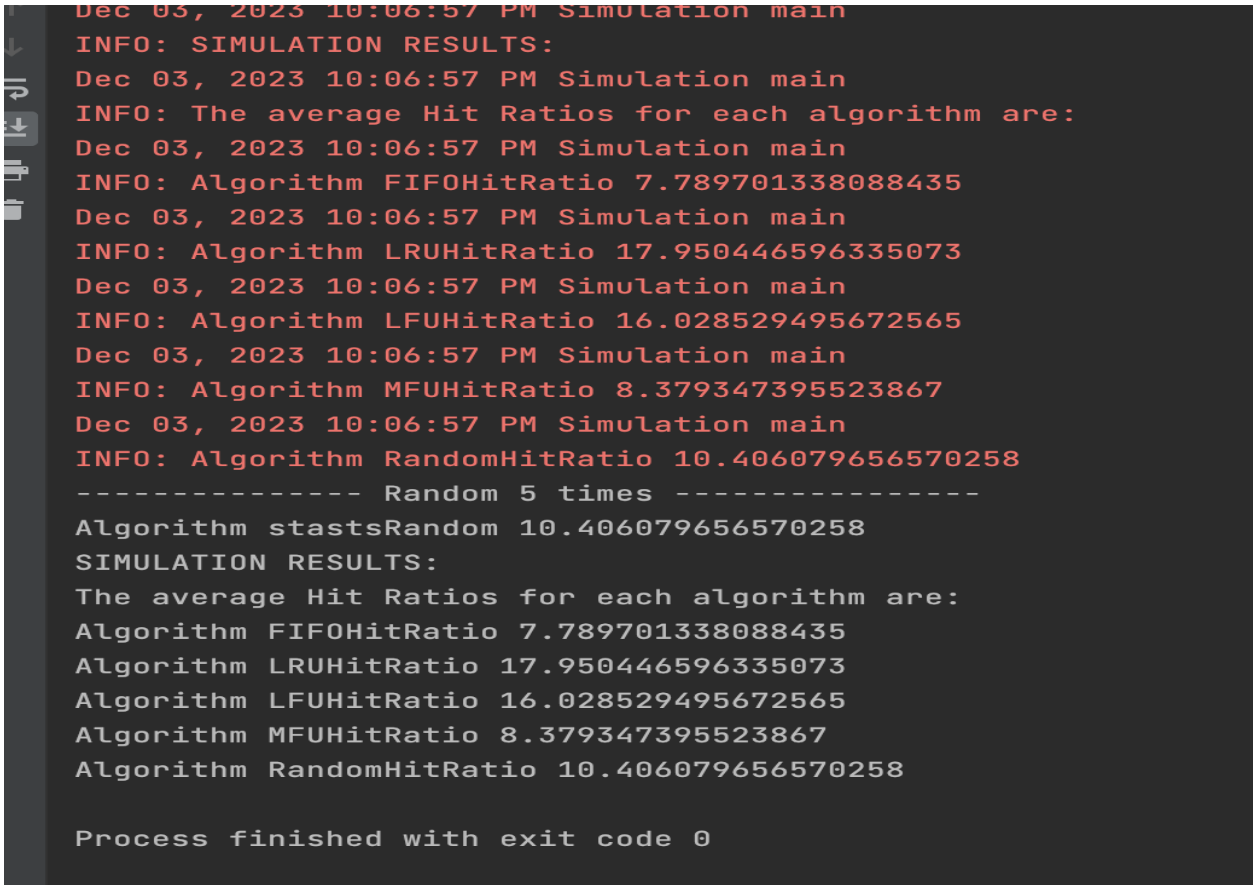
The final output shows cumulative stats for all the algorithms.



```
INFO: Hit for Page Page@655b8230
Dec 03, 2023 9:55:27 PM Process run
INFO: Time 2200: Process P4 referenced page 0 Page@795f3912 remaining time 3800
Dec 03, 2023 9:55:27 PM FIFOCache referencePage
INFO: Hit for Page Page@795f3912
Dec 03, 2023 9:55:27 PM Process run
INFO: Time 2300: Process P8 referenced page 0 Page@655b8230 remaining time 3700
Dec 03, 2023 9:55:27 PM FIFOCache referencePage
INFO: Hit for Page Page@655b8230
Dec 03, 2023 9:55:27 PM Process run
INFO: Time 2300: Process P4 referenced page 0 Page@795f3912 remaining time 3700
Dec 03, 2023 9:55:27 PM FIFOCache referencePage
INFO: Hit for Page Page@795f3912
Dec 03, 2023 9:55:28 PM Process run
INFO: Time 2400: Process P8 referenced page 0 Page@655b8230 remaining time 3600
Dec 03, 2023 9:55:28 PM FIFOCache referencePage
INFO: Hit for Page Page@655b8230
Dec 03, 2023 9:55:28 PM Process run
INFO: Time 2400: Process P4 referenced page 0 Page@795f3912 remaining time 3600
Dec 03, 2023 9:55:28 PM FIFOCache referencePage
INFO: Hit for Page Page@795f3912
Dec 03, 2023 9:55:28 PM Process run
INFO: Time 2500: Process P8 referenced page 0 Page@655b8230 remaining time 3500
Dec 03, 2023 9:55:28 PM FIFOCache referencePage
INFO: Hit for Page Page@655b8230
Dec 03, 2023 9:55:28 PM Process run
INFO: Time 2500: Process P4 referenced page 0 Page@795f3912 remaining time 3500
Dec 03, 2023 9:55:28 PM FIFOCache referencePage
```



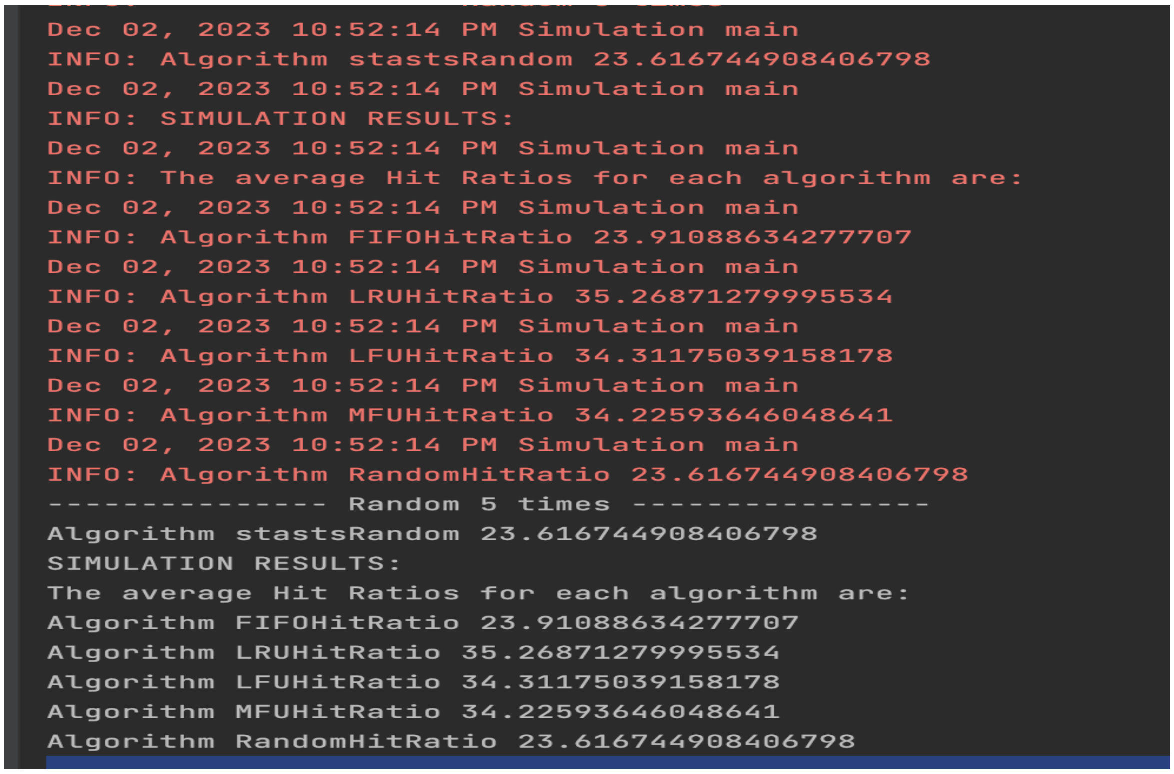
```
INFO: Time 800: Process P7 referenced page 3 Page@2700c470 remaining time 200
Dec 03, 2023 10:06:57 PM RandomPick referencePage
INFO: Miss for Page Page@2700c470
Dec 03, 2023 10:06:57 PM Process run
INFO: Time 900: Process P7 referenced page 2 Page@233f44cf remaining time 100
Dec 03, 2023 10:06:57 PM RandomPick referencePage
INFO: Miss for Page Page@233f44cf
Dec 03, 2023 10:06:57 PM RandomPick stats
INFO: STATS FOR RandomPick
Dec 03, 2023 10:06:57 PM RandomPick stats
INFO: Hit 4.0 Miss 6.0
Dec 03, 2023 10:06:57 PM Process run
INFO: Time 1000: Record - Complete: Process P7, Size: 5 pages, Duration: 1000 seconds.  Strategy Random Hit/Miss Ratio 6.666666666666666
Dec 03, 2023 10:06:57 PM RandomPick stats
INFO: STATS FOR RandomPick
Dec 03, 2023 10:06:57 PM RandomPick stats
INFO: Hit 4.0 Miss 6.0
Dec 03, 2023 10:06:57 PM Process run
INFO:  Work ended P75 1000
```



```
Dec 03, 2023 10:06:57 PM Simulation main
INFO: SIMULATION RESULTS:
Dec 03, 2023 10:06:57 PM Simulation main
INFO: The average Hit Ratios for each algorithm are:
Dec 03, 2023 10:06:57 PM Simulation main
INFO: Algorithm FIFOHitRatio 7.789701338088435
Dec 03, 2023 10:06:57 PM Simulation main
INFO: Algorithm LRUHitRatio 17.950446596335073
Dec 03, 2023 10:06:57 PM Simulation main
INFO: Algorithm LFUHitRatio 16.028529495672565
Dec 03, 2023 10:06:57 PM Simulation main
INFO: Algorithm MFUHitRatio 8.379347395523867
Dec 03, 2023 10:06:57 PM Simulation main
INFO: Algorithm RandomHitRatio 10.406079656570258
--------------- Random 5 times ----------------
Algorithm stastsRandom 10.406079656570258
SIMULATION RESULTS:
The average Hit Ratios for each algorithm are:
Algorithm FIFOHitRatio 7.789701338088435
Algorithm LRUHitRatio 17.950446596335073
Algorithm LFUHitRatio 16.028529495672565
Algorithm MFUHitRatio 8.379347395523867
Algorithm RandomHitRatio 10.406079656570258

Process finished with exit code 0
```

For frame size =6



```
Dec 02, 2023 10:52:14 PM Simulation main
INFO: Algorithm stastsRandom 23.616744908406798
Dec 02, 2023 10:52:14 PM Simulation main
INFO: SIMULATION RESULTS:
Dec 02, 2023 10:52:14 PM Simulation main
INFO: The average Hit Ratios for each algorithm are:
Dec 02, 2023 10:52:14 PM Simulation main
INFO: Algorithm FIFOHitRatio 23.91088634277707
Dec 02, 2023 10:52:14 PM Simulation main
INFO: Algorithm LRUHitRatio 35.26871279995534
Dec 02, 2023 10:52:14 PM Simulation main
INFO: Algorithm LFUHitRatio 34.31175039158178
Dec 02, 2023 10:52:14 PM Simulation main
INFO: Algorithm MFUHitRatio 34.22593646048641
Dec 02, 2023 10:52:14 PM Simulation main
INFO: Algorithm RandomHitRatio 23.616744908406798
--------------- Random 5 times ----------------
Algorithm stastsRandom 23.616744908406798
SIMULATION RESULTS:
The average Hit Ratios for each algorithm are:
Algorithm FIFOHitRatio 23.91088634277707
Algorithm LRUHitRatio 35.26871279995534
Algorithm LFUHitRatio 34.31175039158178
Algorithm MFUHitRatio 34.22593646048641
Algorithm RandomHitRatio 23.616744908406798
```

Code details

Simulation.java

// This is the entry file that needs to be invoked for the simulation.

/* The Simulation class creates a logger object, initializes the Process jobQueue

It Initializes the linked list for free pages, and a bunch of variables needed to begin.

It then loops through each of the simulation algorithms("FIFO", "LRU", "LFU", "MFU", "Random"),

in order and execute the Scheduling process 5 times for each algorithm.

For every execution of the algorithm it generates 150 new processes(threads).

It calculates the average hitRatio for each Page replacement algorithm and finally prints out the

results in order.

It also logs all the details to the logfile.log.

You can see concurrent processing running and referencing randomly generated page references every 100ms for each process till their service duration. The hit and miss metrics are computed for each reference in each page replacement algorithm and the averages are computed at the end of each iteration.

Finally the code prints out all the statistics for each PageReplacement Algorithm.

RandomReferenceGenerator.java
The RandomRefGenerator class generates the random reference string for
each process. The referenceGenerator function is invoked every 100 ms
by each process. It generates the random reference based on the below criteria:
1. generate a random number i which is <=10
2. If the i generated is <7 then the generate the reference page j such that the
delta is one of i-1, i, i+1
3. if i > 7 then generate the reference page j such that 0 <= j <= i-2 or j <= i+2 <= 10
This is to consider the locality of reference.
The function referenceGenerator returns random integer j

The sample reference string generated is below:
1 2 1 3 5 6 0 0 10 0 10 0 8 8 9 10 3 3 2 9 9 10 9 7 6 1 0 4 3 3 7 8 9 7 8 4 10 9 5 1 2 7 6 2 1 0 0 0 0 1 0 9 8 9 8 7 0 10 0 0 3 1 1 5 6 9 9 1 0 0 0 9 10 10 6 6 5 5 5 5 1 4 6 4 8 2 2 0 4 7 7 2 3 5 2 2 2 3 4 5

ScheduleJob.java
The scheduleProcesses function of ScheduleJobList class schedules the processes from the jobQueue for execution.
It checks the freepages for availability, if there are atleast 4 free pages it will
allocate the processes for execution and add them to an ArrayList scheduledProcessList
to track them for completion. Once the process has completed it will execute the join method,
extract the hitRate details for the process and then freeMemory for the respective process.
If the freepages are not available the scheduleProcesses fucntion will wait till one of the process finishes
its execution and picks up the next process from the job queue for processing.
scheduleProcesses runs till there are processes in the queue and till a time frame of 1 min has elapsed as
per the details mentioned in the assignment.
Even if one process has finished execution it will move on to schedule the next process.
Finally when all the processes are done it will free the memory for all.

ProcessJobQueue.java
This file contains the Process class & JobQueue Class.

The Process class extends thread class and implements the run function.
The process class has multiple fields like name, size, hitRatio, algo and page head.
It maintains a list of pages allocated to it by the schedule job process.
The constructor creates a new Process thread and when started executes the run method.
It invokes the referenceGenerator every 100 ms and references it till its service duration.

Based on the algo passed it creates the class for the algo and executes the page replacement algorithm.
It finally gets the stats from the page replacement algorithm and stores hitRatio in its field.
The JobQueue class creates a node for each process added to it by the scheduleJob process.
It maintains a linked list and when time for execution each node is picked from the head of the
linkedlist. it arranges the processes based on their arrival times.

PageList.java

This file contains the Page and FreePageList classes.
The Page is a single page node with size and a next pointer attributes.
The FreePageList creates the a linkedlist fo 100 pages for each round of execution.
It consists of the getNextFreePage and addToFreePageList functions.
getNextFreePage = will return the next free page form the freePagesList to the schedule process for allocation
addToFreePageList = will add back the free pages back to the freePages linked list.
PrintList - that prints the list
the getNextFreePage and addToFreePageList use a lock for thread synchronisation to ensure the free pages count stays
idempotent**.**

ReplacementStrategy,java

This is an interface that is used for by all the pagereplacement algorithms
due to the fact that they own a similar structure and methods making it simple
and easy to invoke.

PageReplacement.java

This file consists of all the PageReplacment Algorithms that the simulation requires.
namely FIFO, LRU, LFU, MFU and RandomPick.
Each of the pageReplacement Algorithms implement the ReplacementStrategy interace.
Each PageReplacement class captures the hit and miss ratio for each process and returns
the hitRatio stats.
Each class takes the capacity attribute for initilisation and use logger to log the details.
The classes are invoked using the strategy design pattern.

MyLogger.java

This is the logger class that creates a logfile.log to log all the messages.

It uses the filehandler class and the simpleFormatter for formatting the logs