



BIRZEIT UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER ENGINEERING

OPERATING SYSTEMS
ENCS3390

OS Project Report

Prepared by:

Tariq Odeh	1190699
Qays Safa	1190880
Mahmoud Samara	1191602

Instructor: Dr. Ayman Hroub & Ahmed Affaneh

Date: 10th January 2022

Abstract

In this project, we will learn about memory management, with a focus on virtual memory. It entails developing a simulator for testing page replacement algorithms. To do this we're going to make a paging simulator. It will read a set of data files containing page traces for individual jobs and then replicate the paging requirements of those programs. In our project we will make two main paging replacement algorithms that are First in First Out (FIFO) and Least Recently used (LRU). Moreover, we will deal with a new type of input files that is configuration file that will include information for the process and main information for the memory. In addition, as a bonus we have to make scheduling function using round robin technique with fixed time quantum. Finally, we will explain all the code we written and attach screen shots for the run of the program.

Table of Contents

1. Theory.....	1
1.1. Scheduling with Round Robin algorithm	1
1.2. Page replacement algorithm (FIFO)	2
1.3. Page replacement algorithm (LRU)	3
2. Design and implementation	4
2.1. Input File	4
2.2. Code & Output	5
2.3. User interface	10
3. Conclusion.....	12
4. References	13

List Of Figures

Figure 1: Round Robin Scheduling.....	1
Figure 2: FIFO algorithm	2
Figure 3: LRU algorithm	3
Figure 4: Input File	4
Figure 5: Main menu	5
Figure 6: FIFO/LRU menu	6
Figure 7: How to read file	6
Figure 8: Round Robin output	7
Figure 9: FIFO output.....	8
Figure 10: LRU output.....	9
Figure 11: interface main page.....	10
Figure 12: interface main menu	10
Figure 13: interface FIFO	10
Figure 14: interface LRU	11
Figure 15: interface RR	11

1. Theory

1.1. Scheduling with Round Robin algorithm

The round-robin (RR) scheduling algorithm is designed especially for timesharing systems and Interactive systems. It is similar to FCFS scheduling, but preemption is added to enable the system to switch between processes. A small unit of time, called a time quantum or time slice, is defined. Round Robin Scheduling is very much practical and there is no starvation (no convoy effect) because every process gets CPU for a certain amount of time unit/quantum. When time quantum increases context switching decreases, and response time increases. But whenever time quantum decreases context switching increases and response time decreases.[1]

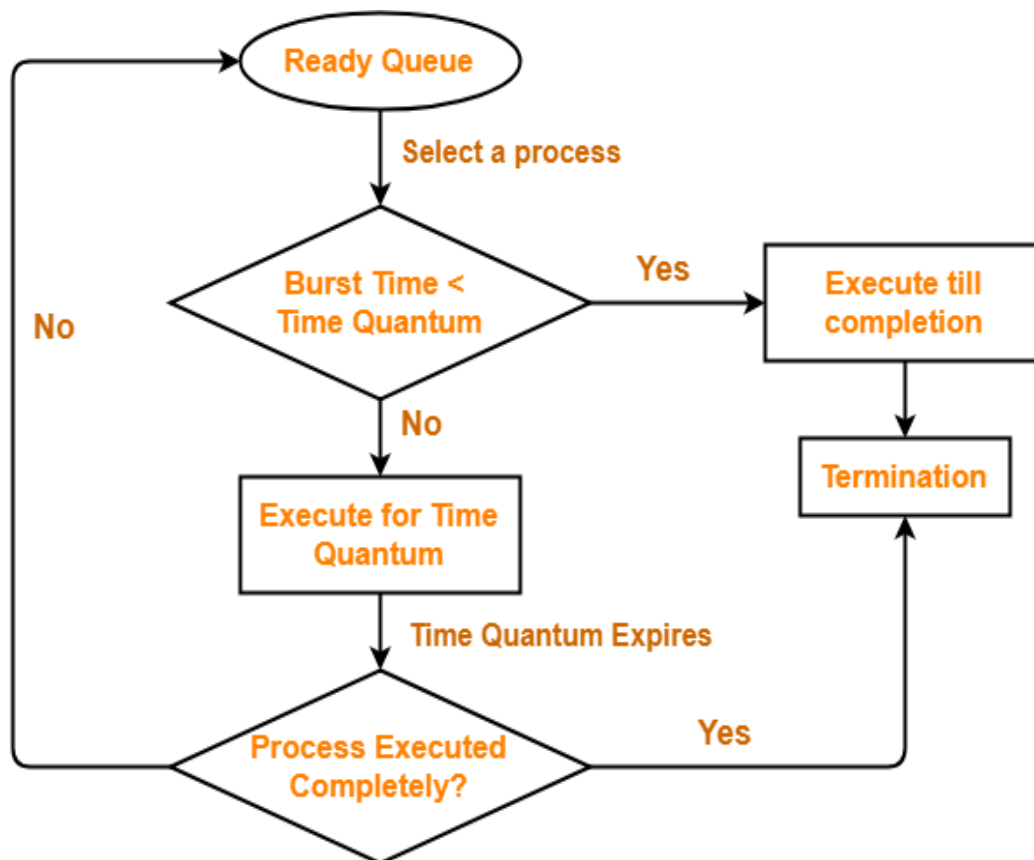


Figure 1: Round Robin Scheduling

1.2. Page replacement algorithm (FIFO)

The FIFO (first-in, first-out) method takes a straightforward approach to this problem. We use a queue in main memory to keep track of all the pages. We'll add it to the queue and proceed as soon as a page arrives. As a result, the oldest page will always be at the front of the queue. We now remove the first page in the queue, which is also the oldest, when a new page arrives and there is no space in the memory. This procedure is repeated until the operating system gets page flow. [2]

Page replacement algorithms like FIFO are used when there is a new page request, and there is not enough space in the main memory to allocate the new page. Hence, a page replacement algorithm decides which page it should replace so that it can allocate the memory for the new page. The steps of a page replacement can be summarized in a flowchart:

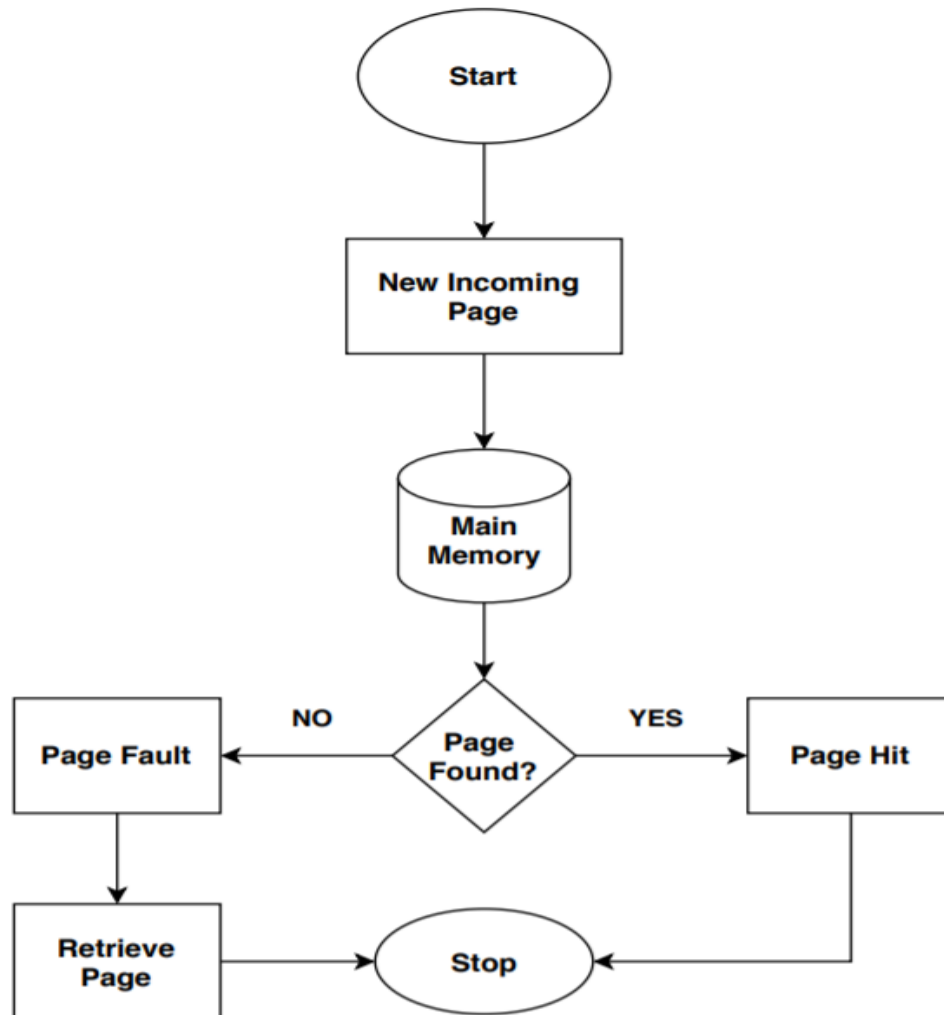


Figure 2: FIFO algorithm

1.3. Page replacement algorithm (LRU)

Least Recently Used (LRU) algorithm is a Greedy algorithm where the page to be replaced is least recently used. It is one of the algorithms that were made to approximate if not better the efficiency of the optimal page replacement algorithm at the point when a page must be replaced with LRU page replacement picks the page that has not been utilized for the longest time frame. We can think about this technique as the optimal page-replacement algorithm glancing in reverse in time, as opposed to advance.[3]

The following figure shows the behavior of the program in paging using the LRU page replacement policy:

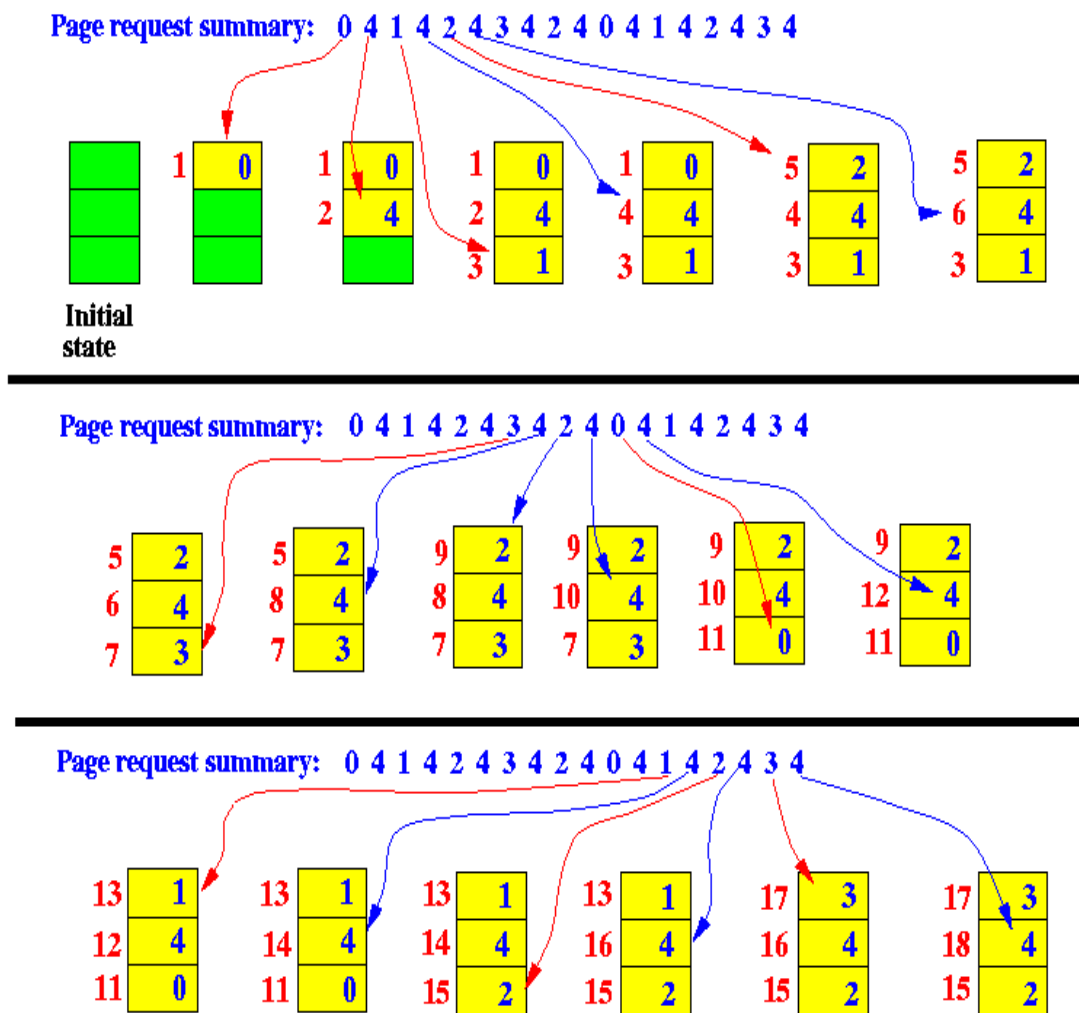


Figure 3: LRU algorithm

2. Design and implementation

2.1. Input File

First of all, we will talk about the input file and the main classes for our project. As we can see in the following figure it shows all classes for our project and the input file. In the input file we put the information as what the formula is: the first line shows the number of processes, second line shows size of physical memory in frames, third line shows minimum frames per process, from forth line until the end of input file it is the information for each process like (PID, start time, Duration time, Size, etc....) and the number of lines here equal the number we entered in the first line. We can see that our input file can be edited with any number or character but the name is constant that is (config.txt).

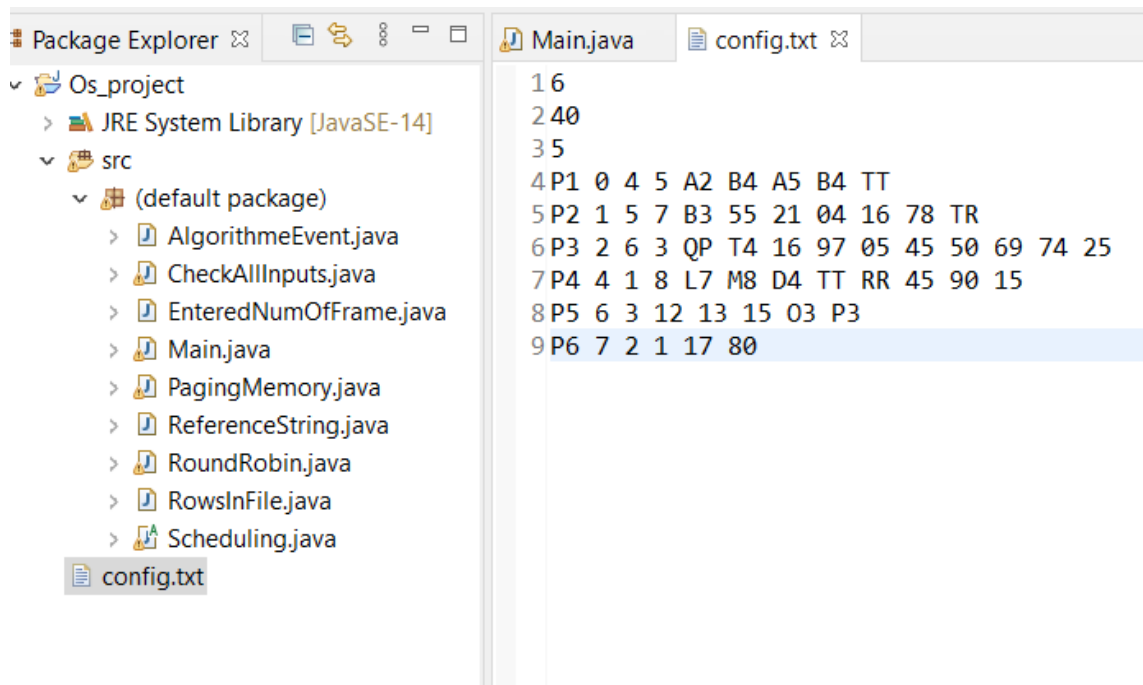


Figure 4: Input File

2.2. Code & Output

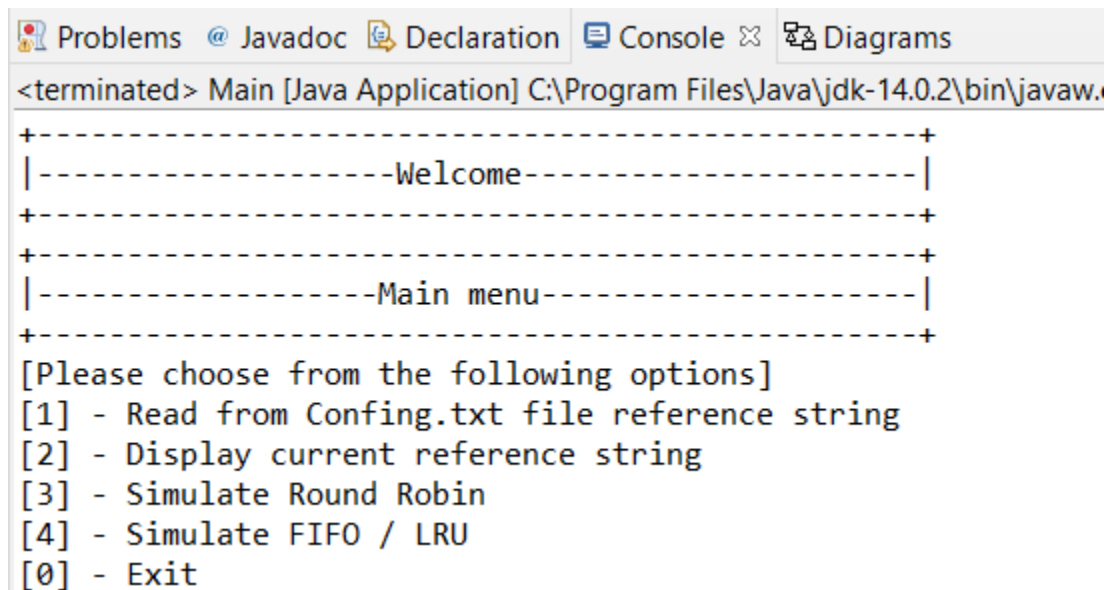
Code: In our project we have three main algorithms (RR, FIFO, LRU) so we made java class for round robin and java class for both FIFO and LRU its' name is (Paging memory). In the main class we read the input file and called all classes depend on cases and what the user will enter number from the menu that is in the main class. In addition, when the user will choose FIFO/LRU it will enter another menu that is special for these two algorithms. Also, as we know we must delete the last 12 bits from memory trace to get the page number for each process and as we know the memory trace numbers are in hexadecimal and each number or character is 4 bits then we must delete the last 3 strings from the memory trace to get the page number.

```
System.out.println("-----Welcome-----");
System.out.println("-----");

Scanner sc = new Scanner(System.in) ; //read physical frame numbers
ReferenceString Ref_string = null ; //to save data for each process for use in algorithm
PagingMemory inf_p ;
int read_1 ;

while (true) { //Main loop & Main menu
    System.out.println("-----");
    System.out.println("-----Main menu-----");
    System.out.println("-----");
    System.out.println("[Please choose from the following options]");
    System.out.println("[1] - Read from Confing.txt file reference string");
    System.out.println("[2] - Display current reference string");
    System.out.println("[3] - Simulate Round Robin");
    System.out.println("[4] - Simulate FIFO / LRU");
    System.out.println("[0] - Exit");
    System.out.println();

    read_1 = Integer.parseInt(sc.next()) ; //read input from user
    sc.nextLine() ;
}
```

The screenshot shows the IDE's console window with the following output:

```
<terminated> Main [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe
+-----+
|-----Welcome-----|
+-----+
+-----+
|-----Main menu-----|
+-----+
[Please choose from the following options]
[1] - Read from Confing.txt file reference string
[2] - Display current reference string
[3] - Simulate Round Robin
[4] - Simulate FIFO / LRU
[0] - Exit
```

Figure 5: Main menu

```

case 4:          //FIFO / LRU scheduling algorithm
System.out.println("Plese enter the number of Physical Page Frames:") ;
int Physical_frames = sc.nextInt() ;
System.out.println("Number of page frames set to [" + Physical_frames + "]" ) ;
int read_2 ;

Back_Menu:      //use it to exit from switch and while loop
while (true) {
    System.out.println();
    System.out.println("+-----+");
    System.out.println("|-----FIFO / LRU-----|");
    System.out.println("+-----+");
    System.out.println("[Please choose from the following options]");
    System.out.println("[1] - Read reference string") ;
    System.out.println("[2] - Generate reference string") ;
    System.out.println("[3] - Display current reference string") ;
    System.out.println("[4] - Simulate FIFO") ;
    System.out.println("[5] - Simulate LRU") ;
    System.out.println("[0]- Go back") ;
}

```

The screenshot shows the IDE's console window with the following output:

```

+-----+
|-----FIFO / LRU-----|
+-----+
[Please choose from the following options]
[1] - Read reference string
[2] - Generate reference string
[3] - Display current reference string
[4] - Simulate FIFO
[5] - Simulate LRU
[0]- Go back

```

Figure 6: FIFO/LRU menu

```

24
25     int N = 0 ;          //number of processes
26     int M = 0 ;          //size of physical memory in frames
27     int S = 0 ;          //minimum frames per process
28     int r = 3 ;          //to read N, M and S
29

```

The screenshot shows the IDE's console window with the following output:

```

<terminated> Main [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (Jan 23, 2022, 6:54:1
+-----+
|-----Welcome-----|
+-----+
+-----+
|-----Main menu-----|
+-----+
[Please choose from the following options]
[1] - Read from Config.txt file reference string
[2] - Display current reference string
[3] - Simulate Round Robin
[4] - Simulate FIFO / LRU
[0] - Exit

1
# [Config.txt has been successfully read]

```

Figure 7: How to read file

Output: For our output after reading the input file the user can choose the algorithm he wants:

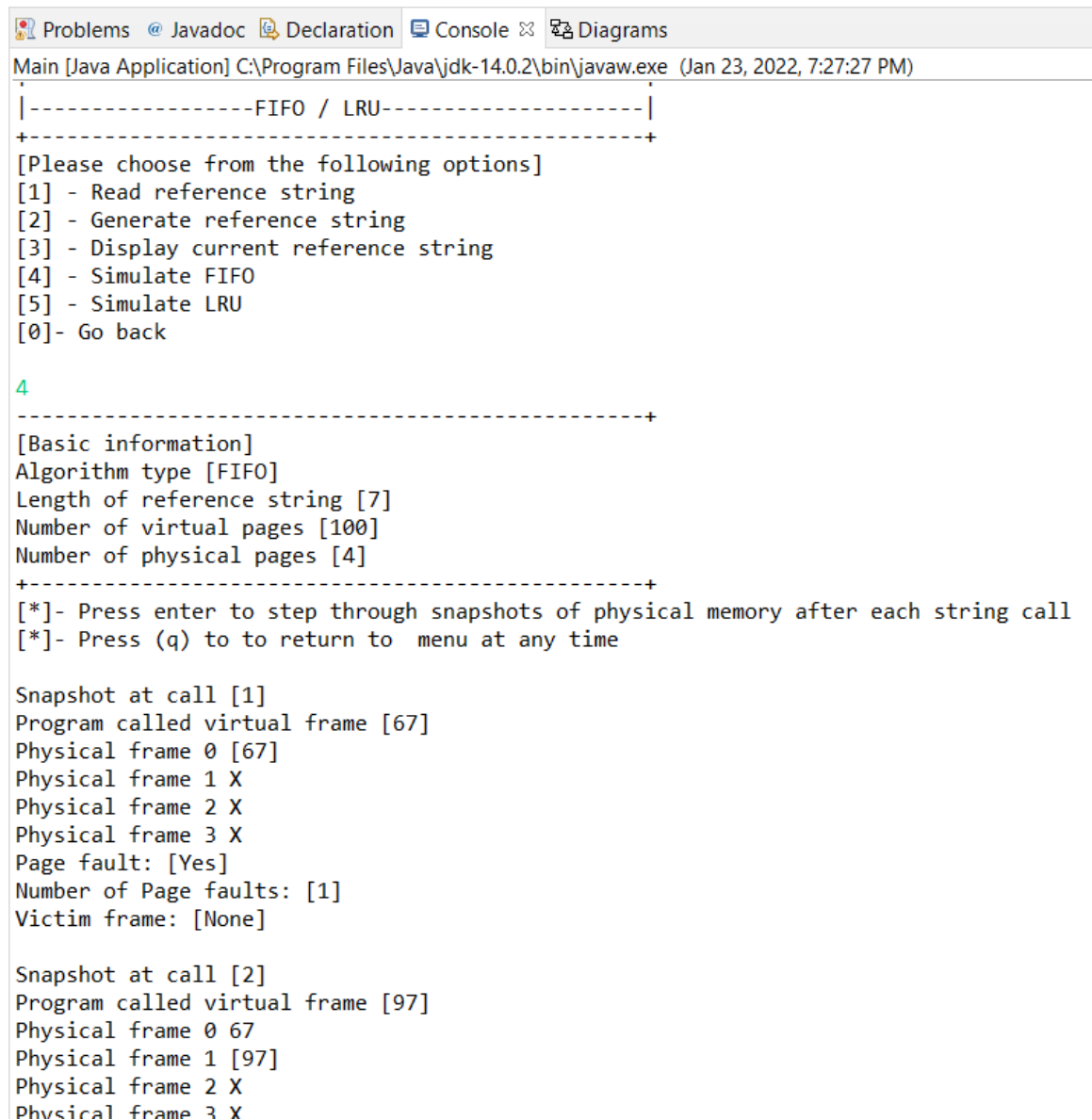
1. Round Robin:

If the user chooses number 3 (RR) then he must enter number for time quantum. After that, all information's will be printed in a schedule include (Process ID, Arrival time, burst time, waiting time and Turnaround time). Finally, the average time for both waiting time and Turnaround time will be printed.

Figure 8: Round Robin output

2. FIFO:

For FIFO algorithm first user will choose number (4) from the main menu then he will go to another menu to make choices for user if he wants to make his own string or he wants the string to be general and of course the number of frames will be entered. Finally, the algorithm will start working by including all numbers to the physical frame and in each step will be shown which number will enter now and which one is killed, at the end the total number of page faults will be printed.



```
Problems @ Javadoc Declaration Console Diagrams
Main [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (Jan 23, 2022, 7:27:27 PM)
|-----FIFO / LRU-----|
+-----+
[Please choose from the following options]
[1] - Read reference string
[2] - Generate reference string
[3] - Display current reference string
[4] - Simulate FIFO
[5] - Simulate LRU
[0]- Go back

4
-----+
[Basic information]
Algorithm type [FIFO]
Length of reference string [7]
Number of virtual pages [100]
Number of physical pages [4]
+-----+
[*]- Press enter to step through snapshots of physical memory after each string call
[*]- Press (q) to to return to menu at any time

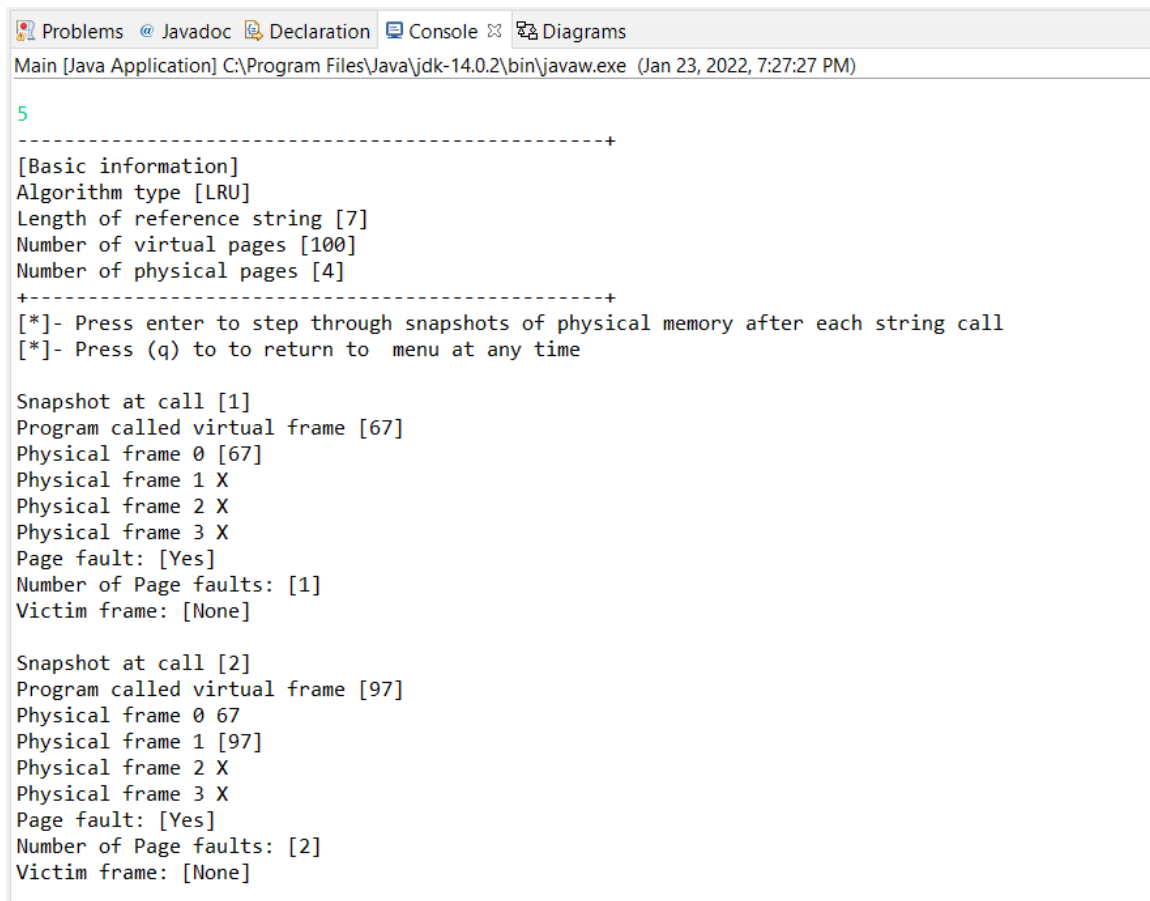
Snapshot at call [1]
Program called virtual frame [67]
Physical frame 0 [67]
Physical frame 1 X
Physical frame 2 X
Physical frame 3 X
Page fault: [Yes]
Number of Page faults: [1]
Victim frame: [None]

Snapshot at call [2]
Program called virtual frame [97]
Physical frame 0 67
Physical frame 1 [97]
Physical frame 2 X
Physical frame 3 X
```

Figure 9: FIFO output

3. LRU:

For LRU algorithm first user will choose number (4) from the main menu then he will go to another menu to make choices for user if he wants to make his own string or he wants the string to be general and of course the number of frames will be entered. Finally, the algorithm will start working by including all numbers to the physical frame and in each step will be shown which number will enter now and which one is killed, at the end the total number of page faults will be printed.



```
Problems @ Javadoc Declaration Console Diagrams
Main [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (Jan 23, 2022, 7:27:27 PM)

5
-----+
[Basic information]
Algorithm type [LRU]
Length of reference string [7]
Number of virtual pages [100]
Number of physical pages [4]
-----+
[*]- Press enter to step through snapshots of physical memory after each string call
[*]- Press (q) to return to menu at any time

Snapshot at call [1]
Program called virtual frame [67]
Physical frame 0 [67]
Physical frame 1 X
Physical frame 2 X
Physical frame 3 X
Page fault: [Yes]
Number of Page faults: [1]
Victim frame: [None]

Snapshot at call [2]
Program called virtual frame [97]
Physical frame 0 67
Physical frame 1 [97]
Physical frame 2 X
Physical frame 3 X
Page fault: [Yes]
Number of Page faults: [2]
Victim frame: [None]
```

Figure 10: LRU output

2.3. User interface



Figure 11: interface main page



Figure 12: interface main menu

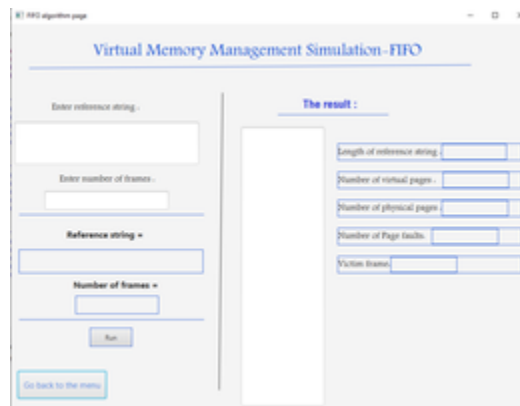


Figure 13: interface FIFO

Figure 14: interface LRU

Figure 15: interface RR

From figures above we show the user interface that we made as what the output of our code is. But note that we didn't connect the user interface with the code because we got many errors in eclipse and time constraints.

3. Conclusion

From this project, one can see and test different page replacement algorithms, compare between them, and so know what is better for a specific purpose, since some algorithms may give less page faults than some other algorithms and other algorithms may be better for other cases. Simulating Algorithms is good, since we can see their efficiency before applying them in real, and it was very nice to have a project to train us to do something like this.

4. References

- [1] Agarwal, S. (2020, May 24). *LRU page replacement algorithm*. Yuvayana. Retrieved January 8, 2022, from <https://er.yuvayana.org/lru-page-replacement-algorithm-in-operating-system/>
- [2] Datta, S. (2020, October 19). *How does fifo page replacement work?* Baeldung on Computer Science. Retrieved January 8, 2022, from <https://www.baeldung.com/cs/fifo-page-replacement>
- [3] Java read files. (2020, April 4). Retrieved January 6, 2022, from https://www.w3schools.com/java/java_files_read.asp
- [4] *Page replacement algorithms in operating systems*. GeeksforGeeks. (2021, August 4). Retrieved January 7, 2022, from <https://www.geeksforgeeks.org/page-replacement-algorithms-in-operating-systems/>
- [5] *Round robin process and examples*. MyCareerwise. (2019, May 1). Retrieved January 8, 2022, from <https://mycareerwise.com/content/round-robin-process-and-examples/content/exam/gate/computer-science>