Xi XIA
2016213482
2016215117

Exercises

9.21 Consider the following page reference string:

7, 2, 3, 1, 2, 5, 3, 4, 6, 7, 7, 1, 0, 5, 4, 6, 2, 3, 0 , 1.

Assuming demand paging with three frames, how many page faults would occur for the following replacement algorithms?

- LRU replacement
- FIFO replacement
- Optimal replacement

- LRU replacement: 18 page faults

page	7	2	3	1	2	5	3	4	6	7	7	1	0	5	4	6	2	3	0	1
	7	7	7	1		1	3	3	3	7		7	7	5	5	5	2	2	2	1
frame		2	2	2		2	2	4	4	4		1	1	1	4	4	4	3	3	3
			3	3		5	5	5	6	6		6	0	0	0	6	6	6	0	0

- FIFO replacement: 17 page faults

page	7	2	3	1	2	5	3	4	6	7	7	1	0	5	4	6	2	3	0	1
	7	7	7	1		1		1	6	6		6	0	0	0	6	6	6	0	0
frame		2	2	2		5		5	5	7		7	7	5	5	5	2	2	2	1
			3	3		3		4	4	4		1	1	1	4	4	4	3	3	3

- Optimal replacement: 13 page faults

page	7	2	3	1	2	5	3	4	6	7	7	1	0	5	4	6	2	3	0	1
	7	7	7	1		1		1	1	1			1		1	1	1	1		
frame		2	2	2		5		5	5	5			5		4	6	2	3		
			3	3		3		4	6	7			0		0	0	0	0		

9.30 A page-replacement algorithm should minimize the number of page faults. We can achieve this minimization by distributing heavily used pages evenly over all of memory, rather than having them compete for a small number of page frames. We can associate with each page frame a counter of the number of pages associated with that frame. Then, to replace a page, we can search for the page frame with the smallest counter.

a. Define a page-replacement algorithm using this basic idea. Specifically address these problems:

i. What is the initial value of the counters?

ii. When are counters increased?

iii. When are counters decreased?

iv. How is the page to be replaced selected?

Define a page-replacement algorithm addressing the problems of:

i. Initial value of the counters—0.

ii. Counters are increased—whenever a new page is associated with that frame.

iii. Counters are decreased—whenever one of the pages associated with that frame is no longer required.

iv. How the page to be replaced is selected—find a frame with the smallest counter. Use FIFO for breaking ties.

b. How many page faults occur for your algorithm for the following reference string with four page frames?

1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2.

14 page faults

c. What is the minimum number of page faults for an optimal page replacement strategy for the reference string in part b with four page frames?

11 page faults

Programming Project

Page replacement policy

1. **FIFO (First IN, First OUT)**
   - FIFO implements a queue.
   - A FIFO replacement algorithm links with each page the time when that page was added into the memory
   - The oldest page is chosen when a page is going to be replaced. We can create a FIFO queue to hold all the pages present in the memory disk. At the head of the queue we replace the page. We insert page at the tail of the queue when a page is added into the memory disk.

   Implementation:
   - 1 array, pageFrameList[pageFrameCount]
   - When there is page fault, it replaces the page in the frame after the previously replaced frame

```
/**
 * @param pageNumber the number of the page to be
 *                   inserted into the page frame list.
 */
void insert(int pageNumber) {
    //TODO: insert code here
    //fault flag
    boolean flag;
    flag = search(pageNumber);
    if (!flag) {
        pageFrameList[firstIndex] = pageNumber;
        firstIndex++;
        if (firstIndex == pageFrameList.length) {
            firstIndex = 0;// back to the head of frame
        }
        System.out.print("frame: ");
        for (int j = 0; j < pageFrameList.length; j++) {
            if (pageFrameList[j] != -1) {
                System.out.print(pageFrameList[j] + "    ");
            } else
                System.out.print("     ");
        }
        System.out.print(" --> page fault!");
```

```java

        System.out.println();
        pageFaultCount++;
    }
    //page hit
    else {
        System.out.print("frame: ");
        /* display the frame buffer array */
        for (int j = 0; j < pageFrameList.length; j++) {
            if (pageFrameList[j] != -1) {
                System.out.print(pageFrameList[j] + "    ");
            } else
                System.out.print("     ");
        }
        System.out.print(" --> page hit!");
        System.out.println();
    }
}
```

2. LRU (Least Recently Used)

- On a page fault, the frame that was least recently used is replaced.

Implementation:

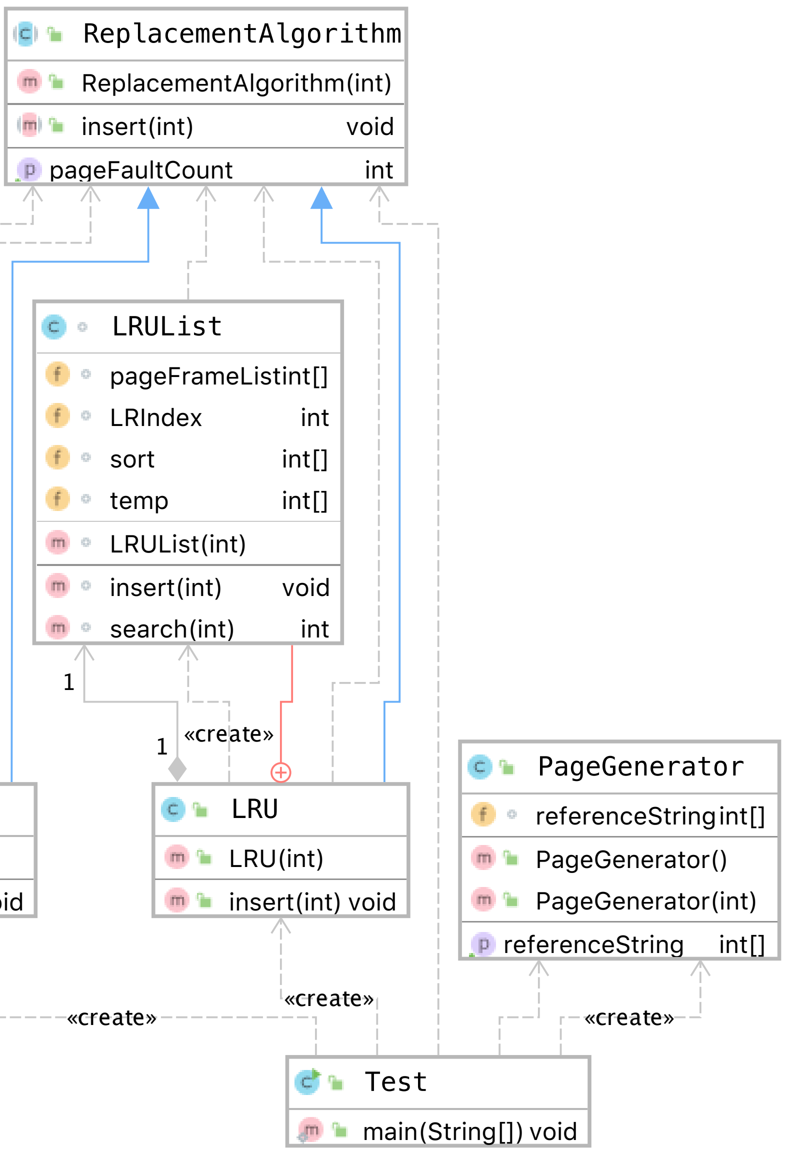
- 1 array, pageFrameList[pageFrameCount]
- Two additional arrays, sort[pageFrameCount] & temp[pageFrameCount], where sort[] stores the sorted list of pages from most recently used to least recently used and temp[] is the temporary array used to update the list
- When page fault occurs, it finds the index of the LRU from pageFrameList[] based on the last element of sort[] and replaces that page
- Each time a page is referenced, update sort[]

```java
/**
 * @param pageNumber the number of the page to be
 *                   inserted into the page frame list.
 */
void insert(int pageNumber) {
    //TODO: insert code here
    //fault flag
    //-1 is page fault
    int flag;
    flag = search(pageNumber);


    //page fault, get the least used index
    for (int j = 0; j < pageFrameCount && flag == -1; j++) {
        if (pageFrameList[j] == sort[pageFrameCount - 1]) {
            LRIndex = j;
            break;
        }
    }

    //page fault, replace the least recently used page
    if (flag == -1) {
        pageFrameList[LRIndex] = pageNumber;
        System.out.print("frame: ");
        /* display frame buffer array */
        for (int j = 0; j < pageFrameList.length; j++) {
            if (pageFrameList[j] != -1) {
```

Xi XIA
2016213482
2016215117

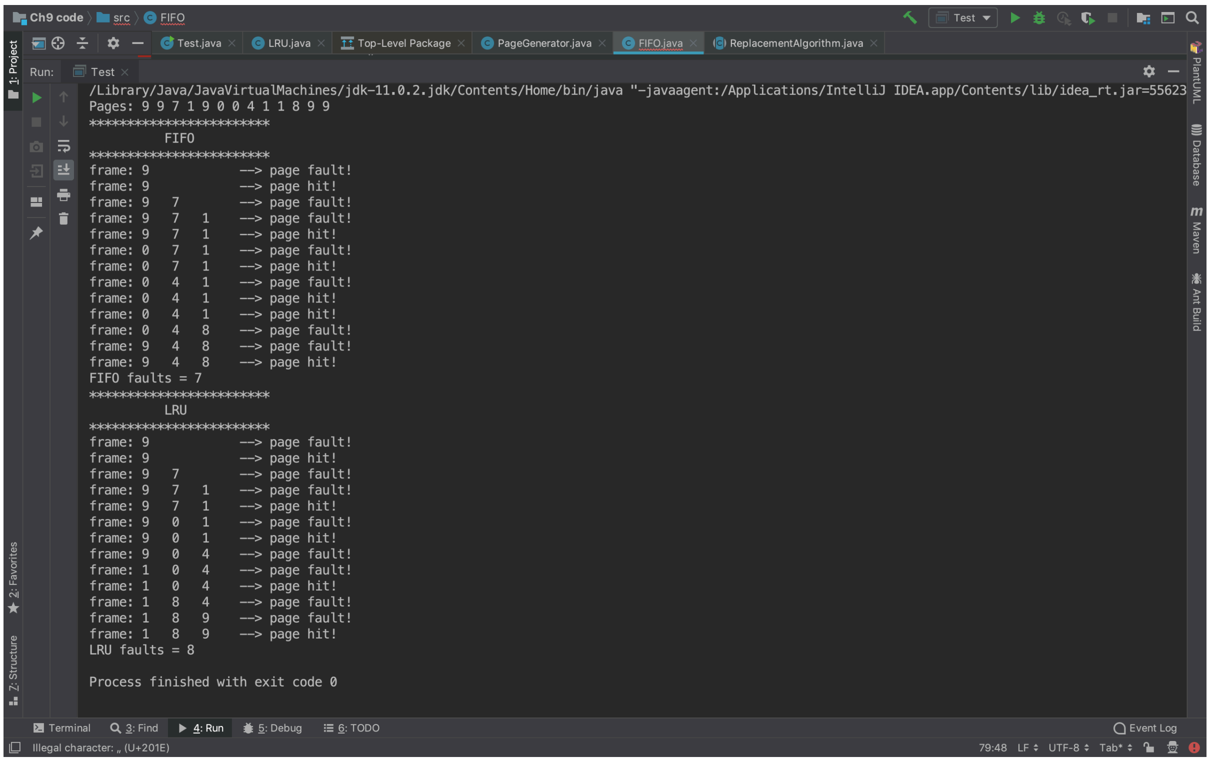
```
                System.out.print(pageFrameList[j] + "    ");
            } else
                System.out.print("     ");
        }
        System.out.println(" --> page fault!");
        pageFaultCount++;
    }
    //page hit
    else {
        /* display frame buffer array */
        System.out.print("frame: ");
        for (int j = 0; j < pageFrameList.length; j++) {
            if (pageFrameList[j] != -1) {
                System.out.print(pageFrameList[j] + "    ");
            } else
                System.out.print("     ");
        }
        System.out.println(" --> page hit!");
    }
```

UML



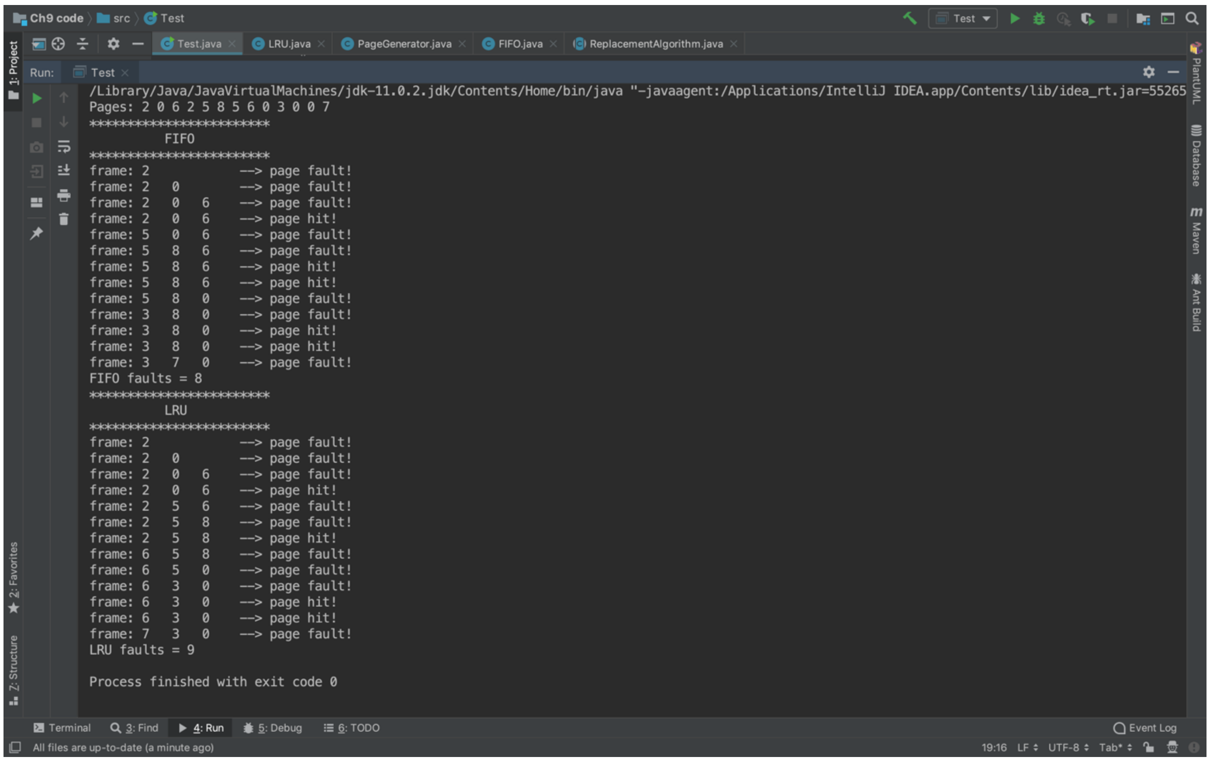
Xi XIA
2016213482
2016215117

Results



```
/Library/Java/JavaVirtualMachines/jdk-11.0.2.jdk/Contents/Home/bin/java "-javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=55623
Pages: 9 9 7 1 9 0 0 4 1 1 8 9 9
************************
          FIFO
************************
frame: 9            --> page fault!
frame: 9            --> page hit!
frame: 9   7        --> page fault!
frame: 9   7   1    --> page fault!
frame: 9   7   1    --> page hit!
frame: 0   7   1    --> page fault!
frame: 0   7   1    --> page hit!
frame: 0   4   1    --> page fault!
frame: 0   4   1    --> page hit!
frame: 0   4   1    --> page hit!
frame: 0   4   8    --> page fault!
frame: 9   4   8    --> page fault!
frame: 9   4   8    --> page hit!
FIFO faults = 7
************************
          LRU
************************
frame: 9            --> page fault!
frame: 9            --> page hit!
frame: 9   7        --> page fault!
frame: 9   7   1    --> page fault!
frame: 9   7   1    --> page hit!
frame: 9   0   1    --> page fault!
frame: 9   0   1    --> page hit!
frame: 9   0   4    --> page fault!
frame: 1   0   4    --> page fault!
frame: 1   0   4    --> page hit!
frame: 1   8   4    --> page fault!
frame: 1   8   9    --> page fault!
frame: 1   8   9    --> page hit!
LRU faults = 8

Process finished with exit code 0
```

Illegal character: „ (U+201E)



```
/Library/Java/JavaVirtualMachines/jdk-11.0.2.jdk/Contents/Home/bin/java "-javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=55265
Pages: 2 0 6 2 5 8 5 6 0 3 0 0 7
************************
          FIFO
************************
frame: 2            --> page fault!
frame: 2   0        --> page fault!
frame: 2   0   6    --> page fault!
frame: 2   0   6    --> page hit!
frame: 5   0   6    --> page fault!
frame: 5   8   6    --> page fault!
frame: 5   8   6    --> page hit!
frame: 5   8   6    --> page hit!
frame: 5   8   0    --> page fault!
frame: 3   8   0    --> page fault!
frame: 3   8   0    --> page hit!
frame: 3   8   0    --> page hit!
frame: 3   7   0    --> page fault!
FIFO faults = 8
************************
          LRU
************************
frame: 2            --> page fault!
frame: 2   0        --> page fault!
frame: 2   0   6    --> page fault!
frame: 2   0   6    --> page hit!
frame: 2   5   6    --> page fault!
frame: 2   5   8    --> page fault!
frame: 2   5   8    --> page hit!
frame: 6   5   8    --> page fault!
frame: 6   5   0    --> page fault!
frame: 6   3   0    --> page fault!
frame: 6   3   0    --> page hit!
frame: 6   3   0    --> page hit!
frame: 7   3   0    --> page fault!
LRU faults = 9

Process finished with exit code 0
```

All files are up-to-date (a minute ago)