

Project 5— Implementation of LRU/FIFO Paging Algorithms -**100 points**

(Due date: 12/19/2011/Monday Midnight at Moodle)

Your name:	Date:
------------	-------

=====How To Submit—Read Carefully, Please!=====

1. Create a directory “**project5_YourLastName**” (you must use this format for the directory name for this project; **Use Your Last Name**).
2. When having finished your project, copy **all the source files (*.java)** to these subdirectories, respectively—you should keep this folders clean: *only source code* files included.
3. A “**readme**” file is required for the project write-up that tells how to compile/run the programs and result screenshots ... *keep this readme simple!*
4. Compress directory “**project5_YourLastName**” and its contents into a **zip** or **rar** file with same name.
5. Submit the compressed file at Moodle.
6. **Penalty** for NOT following these submission instructions (10% ~100%).

In Chapter 9 (Virtual Memory), we introduced several algorithms on page replacement in virtual memory management, among which are FIFO (First-In, First-Out) and LRU (Least-Recently-Used). In this project, we will write a Java program that implements FIFO and LRU.

- Design and implement two *subclasses* of *ReplacementAlgorithm*—LRU and FIFO—that extend *ReplacementAlgorithm* class (available from this project zipped file and in the following).
 1. Each of these two classes will implement the *insert()* method, one class using the LRU page-replacement algorithm and other using the FIFO algorithm.
- There are two classes available to test your algorithm:
 1. *PageGenerator*—a class that generates page-reference strings with page numbers ranging from 0 to 4. The size of the reference string is passed to the *PageGenerator* constructor. Once a *PageGenerator* object is constructed, the *getReferenceString()* method returns the reference string as an array of integers.
 2. *Test*—used to test your FIFO and LRU implementations of the *ReplacementAlgorithm* abstract class. Testing is invoked with the following command:
 - **java Test <reference string size> <# of page frames> <#--to indicate which reference string will be used: 1—the sample one; 0—randomly generated one>**
- Apply the random page-reference string to each algorithm, and record the number of page faults incurred by each algorithm and page frame list (see the screenshots).
- Assume that **demand paging** is used.
- **Required for this programming project:**
 1. The two classes: **LRU and FIFO**, each of which must be **extended** from class *ReplacementAlgorithm!!!!*
 2. **You must use the *Test* class to test your algorithms.**
 3. **The output must include:**

- **The reference string you have generated/used for the testing of the algorithms.**
 - **The number of page faults.**
 - **The page frames.**
 - (see the sample screenshots).
4. Using the following reference string and **3** as the number of page frames to test the algorithms you have implemented:
 - **{7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1}.**
 - This string is included in the Test class.
 - You must use the command “**java Test 20 3 1**” to run the program and get the output.
 - Take a screenshot for the output.
 5. Using several randomly generated reference strings (with sizes from **25 to 50** and page frame numbers from **2 to 7**) to test the algorithms you have implemented:
 - You may use class *PageGenerator*.
 - You may use the command like “**java Test 30 3 0**” to run the program and get the output.
 - After several runs of your program, take one screenshot of a typical output.
 6. Your code should be compact and clean:
 - There are tons of examples for implementing FIFO and LRU—you may use them. But you need to *modify them to fit into* this project (i.e., the creation of subclasses of *ReplacementAlgorithm*).
 - Group discussion and cooperation is encouraged but your submission/code should be **100%** of your own work!—using different variables names, replacing with different selection/loop statements, changing the order of some statements...will **NOT** be tolerated and both (or more getting involved) will get 0 and be reported to the department and college.

Sample screenshots:

```
D:\Salem\2010Fall\CSC280\assignments\project5_sol>java Test 20 3 0
Page reference string:
4, 3, 3, 4, 0, 3, 3, 0, 4, 3, 2, 4, 3, 0, 1, 2, 1, 3, 4, 3.
```

```
*****LRU*****
Inserting 4: ==> 4, -1, -1,
Inserting 3: ==> 4, 3, -1,
Inserting 3: ==> 4, 3, -1,
Inserting 4: ==> 4, 3, -1,
Inserting 0: ==> 4, 3, 0,
Inserting 3: ==> 4, 3, 0,
Inserting 3: ==> 4, 3, 0,
Inserting 0: ==> 4, 3, 0,
Inserting 4: ==> 4, 3, 0,
Inserting 3: ==> 4, 3, 0,
Inserting 2: ==> 4, 3, 2,
Inserting 4: ==> 4, 3, 2,
Inserting 3: ==> 4, 3, 2,
Inserting 0: ==> 4, 3, 0,
Inserting 1: ==> 1, 3, 0,
Inserting 2: ==> 1, 2, 0,
Inserting 1: ==> 1, 2, 0,
Inserting 3: ==> 1, 2, 3,
Inserting 4: ==> 1, 4, 3,
Inserting 3: ==> 1, 4, 3,
LRU faults = 9
=====LRU Done=====
```

```
*****FIFO*****
Inserting 4: ==> 4, -1, -1,
Inserting 3: ==> 4, 3, -1,
Inserting 3: ==> 4, 3, -1,
Inserting 4: ==> 4, 3, -1,
Inserting 0: ==> 4, 3, 0,
Inserting 3: ==> 4, 3, 0,
Inserting 3: ==> 4, 3, 0,
Inserting 0: ==> 4, 3, 0,
Inserting 4: ==> 4, 3, 0,
Inserting 3: ==> 4, 3, 0,
Inserting 2: ==> 2, 3, 0,
Inserting 4: ==> 2, 4, 0,
Inserting 3: ==> 2, 4, 3,
Inserting 0: ==> 0, 4, 3,
Inserting 1: ==> 0, 1, 3,
Inserting 2: ==> 0, 1, 2,
Inserting 1: ==> 0, 1, 2,
Inserting 3: ==> 3, 1, 2,
Inserting 4: ==> 3, 4, 2,
Inserting 3: ==> 3, 4, 2,
FIFO faults = 11
=====FIFO Done=====
```

```
D:\Salem\2010Fall\CSC280\assignments\project5_sol>java Test 36 3 1
Page reference string:
7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1,
```

```
*****LRU*****
Inserting 7: ==> 7, -1, -1,
Inserting 0: ==> 7, 0, -1,
Inserting 1: ==> 7, 0, 1,
Inserting 2: ==> 2, 0, 1,
Inserting 0: ==> 2, 0, 1,
Inserting 3: ==> 2, 0, 3,
Inserting 0: ==> 2, 0, 3,
Inserting 4: ==> 4, 0, 3,
Inserting 2: ==> 4, 0, 2,
Inserting 3: ==> 4, 3, 2,
Inserting 0: ==> 0, 3, 2,
Inserting 3: ==> 0, 3, 2,
Inserting 2: ==> 0, 3, 2,
Inserting 1: ==> 1, 3, 2,
Inserting 2: ==> 1, 3, 2,
Inserting 0: ==> 1, 0, 2,
Inserting 1: ==> 1, 0, 2,
Inserting 7: ==> 1, 0, 7,
Inserting 0: ==> 1, 0, 7,
Inserting 1: ==> 1, 0, 7,
LRU faults = 12
=====LRU Done=====
```

```
*****FIFO*****
Inserting 7: ==> 7, -1, -1,
Inserting 0: ==> 7, 0, -1,
Inserting 1: ==> 7, 0, 1,
Inserting 2: ==> 2, 0, 1,
Inserting 0: ==> 2, 0, 1,
Inserting 3: ==> 2, 3, 1,
Inserting 0: ==> 2, 3, 0,
Inserting 4: ==> 4, 3, 0,
Inserting 2: ==> 4, 2, 0,
Inserting 3: ==> 4, 2, 3,
Inserting 0: ==> 0, 2, 3,
Inserting 3: ==> 0, 2, 3,
Inserting 2: ==> 0, 2, 3,
Inserting 1: ==> 0, 1, 3,
Inserting 2: ==> 0, 1, 2,
Inserting 0: ==> 0, 1, 2,
Inserting 1: ==> 0, 1, 2,
Inserting 7: ==> 7, 1, 2,
Inserting 0: ==> 7, 0, 2,
Inserting 1: ==> 7, 0, 1,
FIFO faults = 15
=====FIFO Done=====
```

PageGenerator class code:

```
public class PageGenerator
{
    private static final int DEFAULT_SIZE = 100;
    private static final int RANGE = 4;

    int[] referenceString;

    public PageGenerator() {
        this(DEFAULT_SIZE);
    }

    public PageGenerator(int count) {
        if (count < 0)
            throw new IllegalArgumentException();

        java.util.Random generator = new java.util.Random();
        referenceString = new int[count];

        for (int i = 0; i < count; i++){
            referenceString[i] = generator.nextInt(RANGE + 1);
        }
    }

    public int[] getReferenceString() {
        return referenceString;
    }
}
```

ReplacementAlgorithm class code:

```
public abstract class ReplacementAlgorithm
{
    // the number of page faults
    protected int pageFaultCount;

    // the number of physical page frame
    protected int pageFrameCount;

    /**
     * @param pageFrameCount - the number of physical page frames
     */
    public ReplacementAlgorithm(int pageFrameCount) {
        if (pageFrameCount < 0)
            throw new IllegalArgumentException();

        this.pageFrameCount = pageFrameCount;
        pageFaultCount = 0;
    }

    /**
     * @return - the number of page faults that occurred.
     */
    public int getPageFaultCount() {
        return pageFaultCount;
    }

    /**
     * @param int pageNumber - the page number to be inserted
     */
    public abstract void insert(int pageNumber);
}
```

Test class code:

```
public class Test
{
    public static void main(String[] args) {
        PageGenerator ref = new PageGenerator(new Integer(args[0]).intValue());

        int[] referenceString = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1};

        if ( (new Integer (args[2]).intValue() ) == 0 )
            referenceString = ref.getReferenceString();

        System.out.println("Page reference string:");
        for (int i = 0; i < referenceString.length; i++)
            System.out.printf("%d, ", referenceString[i]);
        System.out.println("\n");

        /** Use either the FIFO or LRU algorithms */
        ReplacementAlgorithm fifo = new FIFO(new Integer(args[1]).intValue());
        ReplacementAlgorithm lru = new LRU(new Integer(args[1]).intValue());

        // output a message when inserting a page
        System.out.println("*****LRU*****");
        for (int i = 0; i < referenceString.length; i++) {
            lru.insert(referenceString[i]);
        }

        // report the total number of page faults
        System.out.println("LRU faults = " + lru.getPageFaultCount());
        System.out.println("=====LRU Done=====");

        System.out.println();

        // output a message when inserting a page
        System.out.println("*****FIFO*****");
        for (int i = 0; i < referenceString.length; i++) {
            fifo.insert(referenceString[i]);
        }

        // report the total number of page faults
        System.out.println("FIFO faults = " + fifo.getPageFaultCount());
        System.out.println("=====FIFO Done=====");
    }
}
```