Project 5—Implementation of LRU/FIFO Paging Algorithms
(**Due date: 12/11/2008/Thursday**)

Your name:	Date:

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 classes—LRU and FIFO—that extend *ReplacementAlgorithm* class (avail from this project zipped file and in the following).
    1. Each of these 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 t hat generates page-reference strings with page numbers ranging from 0 to 9. 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 intergers.
    2. **Test**—used to test your FIFO and LRU implementations of the ReplacementAlgorithm abstract class. Test is invoked as:
        - **java Test <reference string size> <# of page frames>**
- Apply the random page-reference string to each algorithm, and record the number of page faults incurred by each algorithm.
- Implement the replacement algorithms so that the number of page frames can vary from 1 to 7.
- Assume that **demand paging** is used.
- **Required for this programming project:**
    1. **The two classes: LRU and FIFO**
    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.**

1. (**30 points**) 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};** (the string size is **20).**
- Take a screenshot for the output.

2. (**70 points**) 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 the PageGenerator class.
- Take screenshot for the outputs.

Important:

- In addition to above compressed source java files, a **readme** file (PDF or Doc or DOCX format) is required for your submission. Check the following on how to submit your project.
    - Result screenshots are required in your Readme file.

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

1. Create a directory "**project5_YourLastName**" (you must use this format for the directory name for this project; **Use Your Last Name. For example, if your** last name is Smith, you should create directory with the name of "project3_Smith"
2. Create "**project51src**" … "**project52src**" subdirectories under "project5_YourLastName" directory.
3. Under these subdirectories, you can put ONLY java files (source) files. This should be clean and comprehensive—that is, I will javac *.java and I can test your code.
4. If you have used some IDE, you can compress the package files in other subdirectories than the above six ones and tell me how to run in the **readme** file.
5. A "**readme**" file is required for the project write-up that tells how to compile in which IDE (not required if not having used any IDE but a simple command line), result screenshots , … keep this readme simple!
    a. This "readme" must reside in the "**project5_YourLastName**" dir in the format of .txt, .pdf, or .doc/docx.
6. Compress the "**project5_YourLastName**" dir and its contents into a zipped/rar-ed file with same name.
7. Submit the compressed file to the instructor by email.
8. Double check your work before submission. Significant penalty (10—100 points)will be applied if your submission does not follow the above instruction!

PageGenerator class code:

```
public class PageGenerator
{
  private static final int DEFAULT_SIZE = 100;
  private static final int RANGE = 9;

 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 = ref.getReferenceString();

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

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

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