Assignment 2 – File Manipulation (8%)

Introduction

Assignment 2 contributes 6% towards your final course grade, and the assignment total is 10 marks. You should begin Assignment 2 in Module Five; it is due at the end of Module Six. Check your Course Schedule for the precise due date. Directions for submitting Assignment 2 to your Open Learning Faculty Member for grading can be found in the "Assignment and Project Instructions" document located on your Homepage. An assignment marking criteria follows at the end of this document.

Instructions

In this assignment, you will handle student records in a file named "`student_record`". Each record in the file consists of student number of integer type and student name of 32 characters, so the size of each record is 36 bytes. In this assignment, you need to write a Java program to store the following four records into the file:

72	James
56	Mark
87	John
30	Phillip
44	Andrew

Then, the program sorts the records in the file using any simple sorting algorithm such as bubble sorting, with the first field – student number, and prints all records in the file after sorting. When the program sorts the student records, the program should not read all records in memory at once. The program should move records in the file. For example, after the program reads the first two records, it may switch the records (because 72 > 56) and write them in the same position in the file.

> You need to submit Java programs and screen shots that show how your programs work.

Manipulating Files in Java

The following sections are prepared to help you understand how to manipulate files in Java application programs.

1. **File**
   - Basic idea to use files
     - Open – in Java, create an object for the given file
     - Read; write
     - Close
   - Sequential access
     - Just keep reading or writing
   - Random access
     - Read or write any place in a file
   - API reference: http://docs.oracle.com/javase/7/docs/api/java/io/File.html
2. **Sequential access**

   2.1 Writing

```
import java.io.*;

class FileWriteStreamTest {

    public static void main (String[] args) {
            FileWriteStreamTest f = new FileWriteStreamTest();
            f.writeMyFile();
    }

    void writeMyFile() {
            DataOutputStream dos = null;
            String record = null;
            int recCount = 0;

            try {
                    File f = new File("mydata.txt");
                    if (!f.exists())
                            f.createNewFile();

                    FileOutputStream fos = new FileOutputStream(f);
                    BufferedOutputStream bos = new BufferedOutputStream(fos);
```

```
                    dos = new DataOutputStream(bos);

                    dos.writeBytes("Test\n");
                    dos.writeBytes("Welcome\n");
                    dos.writeBytes("Operating System\n");
                    dos.writeBytes("File System\n");

            } catch (IOException e) {
                    System.out.println("Uh oh, got an IOException error!" +
e.getMessage());

            } finally {
                    // if the file opened okay, make sure we close it
                    if (dos != null) {
                            try { dos.close(); }
                            catch (IOException ioe) { }
                    }
            }
    }
}
```

2.2 Reading

```
import java.io.*;

class FileReadStreamTest {

    public static void main (String[] args) {
            FileReadStreamTest f = new FileReadStreamTest();
            f.readMyFile();
    }

    void readMyFile() {
            DataInputStream dis = null;
            String record = null;
            int recCount = 0;

            try {
                    File f = new File("mydata.txt");
                    if (!f.exists()) {
                            System.out.println(f.getName() + " does not exist");
                            return;
                    }
```

```
                    FileInputStream fis = new FileInputStream(f);
                    BufferedInputStream bis = new BufferedInputStream(fis);
                    dis = new DataInputStream(bis);

                    while ( (record=dis.readLine()) != null ) {
                            recCount++;
                            System.out.println(recCount + ": " + record);
                    }

            } catch (IOException e) {
                    System.out.println("Uh oh, got an IOException error!" +
e.getMessage());

            } finally {
                    // if the file opened okay, make sure we close it
                    if (dis != null) {
                            try { dis.close(); }
                            catch (IOException ioe) { }
                    }
            }
    }
}
```

3. Random access

```
import java.io.*;

class FileRandomAccessTest {

    public static void main (String[] args) {
            FileRandomAccessTest f = new FileRandomAccessTest();
            f.readWriteMyFile();
    }

    void readWriteMyFile() {
            RandomAccessFile raf = null;
            String s = null;

            try {
                    File f = new File("mydata.txt");
                    if (!f.exists())  // check if the file exists
                            f.createNewFile();  // create a new file
```

```
                    raf = new RandomAccessFile(f, "rw");  // open a file for random
access with "r", "rw"

                    if (raf.length() > 7) {  // the size of the file
                            raf.seek(7);  // move the file pointer
                            System.out.println(raf.readLine());  // read a line from
the file pointer
                            s = raf.readLine();
                            System.out.println(s);
                            raf.seek(raf.getFilePointer() - s.length());  // get the
file pointer
                            raf.writeBytes("Test RamdomAccessFile\n");  // write bytes
                    }

            } catch (IOException e) {
                    System.out.println("Uh oh, got an IOException error!" +
e.getMessage());

            } finally {
                    // if the file opened okay, make sure we close it
                    if (raf != null) {
                            try { raf.close(); }  // close the file
                            catch (IOException ioe) { }
                    }
            }
    }
}
```

Assignment Marking Criteria	Weighting
No syntax error: All requirements are fully implemented without syntax errors. Submitted screen shots will be reviewed with source code.	/5
Correct implementation: All requirements are correctly implemented and produce correct results Submitted screen shots will be reviewed with source code.	/5
Total	/10