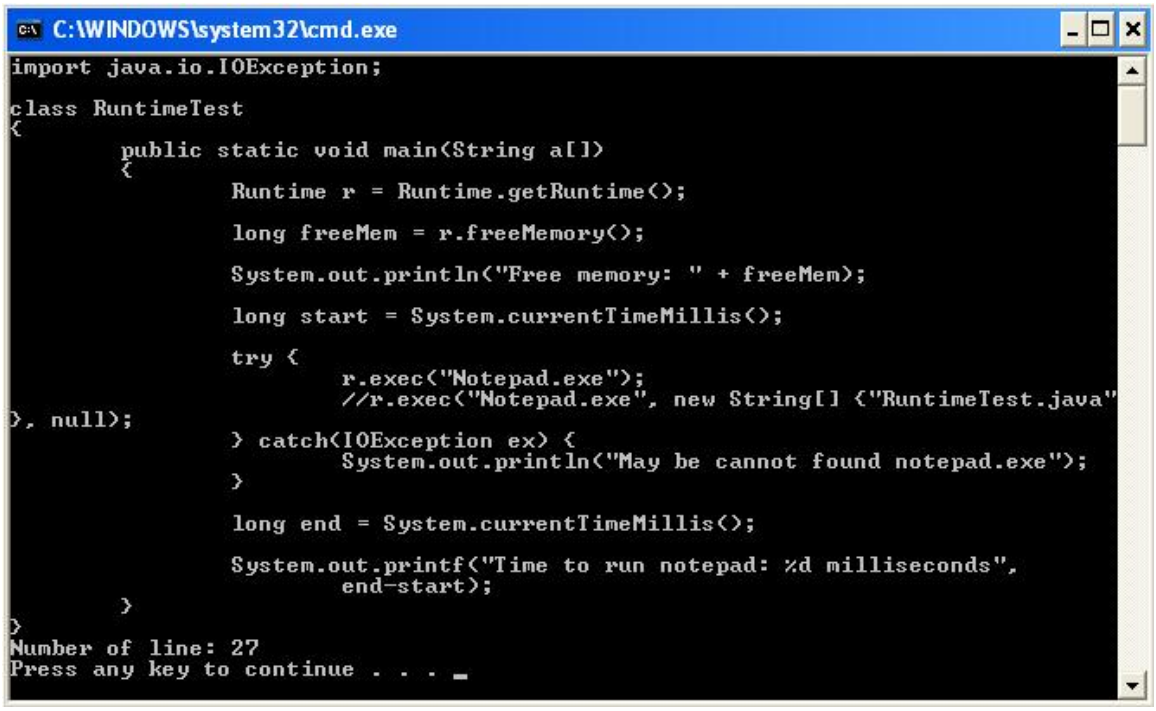
1. This class demonstrates the usage of a buffer stream, read character lines of a file

```
import java.io.*;

class BufferedReaderDemo
{
   public static void main(String a[])
   {
      int count = 0;
      try {
         InputStreamReader iReader = new InputStreamReader(
            new FileInputStream("RuntimeTest.java"));
         BufferedReader bReader = new BufferedReader(iReader);

         String line;
         count = 0;
         while ((line = bReader.readLine()) != null) {
            System.out.println(line);
            count++;
         }

         iReader.close();
         bReader.close();
      } catch(IOException ex) {
         ex.printStackTrace();
      }

      System.out.println("Number of line: " + count);
   }
}
```

The result:



2. Write an application that uses BufferedReader and BufferedWriter classes

```
import java.io.*;

public class BufferedReaderWriter {

    public static void main(String args[]) {

        String a0, a1, a2;

        if (args.length != 3){
            a0 = "words.txt";
            a1 = "wordsout.txt";
            a2 = "3";
        } else{
            a0 = args[0];
            a1 = args[1];
            a2 = args[2];
        }

        SimpleEncryption se = new SimpleEncryption();
        se.encrypt(a0, a1, Integer.parseInt(a2));

        /* print content of encrypted file */
        System.out.println("Displaying   encrypted   words   from
words.txt...");
        se.viewFileContent(a1);
```

```
    }
}
class SimpleEncryption {

   void encrypt(String source, String dest, int shiftSize) {

      BufferedReader reader;
      BufferedWriter writer;

      try {
         reader = new BufferedReader(new FileReader(source));
         writer = new BufferedWriter(new FileWriter(dest));
         String line = reader.readLine();
         int data;
         while (line != null) {
            for (int i = 0; i < line.length(); i++) {
               data =(int) line.charAt(i) + shiftSize;
               writer.write(data);
            }
            writer.write((int)'\n');
            line = reader.readLine();
         }
         reader.close();
         writer.close();
      } catch (IOException ie) {
         System.out.println("Failed    encrypting    the    file
content.");
      }

   }

   void viewFileContent(String filename) {

      BufferedReader reader;
      try {
         reader = new BufferedReader(new FileReader(filename));
         String line = reader.readLine();
         while (line != null) {
            System.out.println(line);
            line = reader.readLine();
         }
         reader.close();
      } catch (IOException ie) {
         System.out.println("Failed to open file for reading.");
      }
   }

}
```

You must create two input.txt file and output.txt file to run application

The output of the program:

3. In this exercise, you are going to create a file to store an object.

```
import java.io.*;

class Student implements Serializable
{
   String rollno;
   String name;
   transient int age;
}

public class ObjectStream
{
   public static void writeObject() {
      Student obj = new Student();
      obj.rollno = "A009";
      obj.name = "james";
      obj.age = 16;

      try {
         ObjectOutputStream oos = new ObjectOutputStream(
            new FileOutputStream("student.dat"));
         oos.writeObject(obj);
         oos.close();
      } catch(FileNotFoundException fe) {
         System.out.println("Cannot find the file");
      } catch(IOException ioe) {
         System.out.println("Error in save object");
         ioe.printStackTrace();
      }
   }


   public static void readObject() {
         Student obj = null;
         try {
            ObjectInputStream ois = new ObjectInputStream(
               new FileInputStream("student.dat"));
            obj = (Student)ois.readObject();
            ois.close();
```

```
        } catch(FileNotFoundException fe) {
           System.out.println("Cannot find the file");
        } catch(ClassNotFoundException ce) {
           System.out.println("Class not found");
        } catch(IOException ioe) {
           System.out.println("Error in loading object");
           ioe.printStackTrace();
        }

        System.out.println("Rollno: " + obj.rollno + ", name: "
           + obj.name + ", age: " + obj.age);
   }
   public static void main(String args[])
   {
      writeObject();

      System.out.println("The stored student:");
      readObject();
   }
}
```

The result:

Do It Yourself

3.1. Do workshop of the module 4

3.2. Create a class Student includes: name, age, mark and necessary methods. Using FileWriter, FileReader and BufferedReader to write a program that has functional menu:

Menu

---

1. Save to File
2. Read File

```
3. Exit

Your choice: _
```

+ Save to File: input information of a student and write that information into a file named sv.txt – each student information is stored in a line. Use tab to separate each field of the object. Student objects are stored in an array.

+ Read File: read and display information of students

3.3. Write a Java program that takes a list of filenames on the command line and prints out the number of lines in each file. The program should create one thread for each file and use these threads to count the lines in all the files at the same time. Use java.io.LineNumberReader to help you count lines. You'll probably want to define a LineCounter class that extends Thread or implements Runnable. Now write a variant of your program that uses your LineCounter class to read the files sequentially, rather than at the same time. Compare the performance of the multithreaded and single-threaded programs, using System.currentTimeMills( ) to determine elapsed time. Compare the performance of the two programs for two, five, and ten files.

3.4. Redo the exercise 3.2 by using the classes ObjectInputStream and ObjectOutputStream

References

+ Java tutorials

+ Javadoc

+ Java2s.com

+ Javapassion.com

+ Java almanac
http://www.exampledepot.com