



Java Input/Output (I/O)

Wellesley College CS230
Lecture 04
Thursday, February 8
Handout #11

Problem Set:

Assignment #1 due Tuesday, February 13

04 - 1



What is Input/Output (I/O)?

I/O is any means of receiving information from or transmitting information to user/file system/web.

Today we focus on textual information that can be entered/displayed in Linux shell.

Later in the semester we will cover Graphical User Interfaces (GUIs).

04 - 2



Some examples of File I/O in Linux

```
[fturbak@puma utils] cat ../text/cat-in-hat-4.txt
The sun did not shine.
It was too wet to play.
So we sat in the house
All that cold, cold, wet day.
```

```
[fturbak@puma utils] wc ../text/cat-in-hat-4.txt
4 23 100 ../text/cat-in-hat-4.txt
```

```
[fturbak@puma utils] wc ../text/cat-in-hat.txt
349 1620 7440 ../text/cat-in-hat.txt
```

```
[fturbak@puma utils] cp ../text/cat-in-hat.txt copycat
```

```
[fturbak@puma utils] wc copycat
349 1620 7440 copycat
```

04 - 3



Our Goal: Similar Operations in Java

```
import java.io.*; // Import classes from the Java I/O package
import java.net.*; // Import classes from the Java web package

public class FileOps {

    // We will write all of our code in this class, which can
    // be found in your ~/cs230/utils/FileOps.java file.

}
```

04 - 4



Reading File Contents Into a String

```
public static String fileToString (String infile) {
    try {
        BufferedReader reader = new BufferedReader(new FileReader(infile));
        // create a new file reader
        String result = ""; // variable for accumulating string from file;
        // next lecture, we'll see a StringBuffer is better for this
        String line = reader.readLine(); // read the first line of the file.
        while (line != null) { // line becomes null at end of file
            result = result + line + '\n';
            // readLine() omits the newline character, so add it back in
            line = reader.readLine(); // read the next line of the file
        }
        reader.close(); // close the file reader
        return result;
    } catch (IOException ex) {
        System.out.println(ex); // Handle file-not-found by displaying message
        return ""; // Return the empty string if file not found
    }
}
```

04 - 5



Displaying File Contents (like Linux cat)

```
/** A simple version of a method that displays file contents */
public static void displayFile (String infile) {
    System.out.println(fileToString(infile));
}
```

```
[fturbak@puma utils] java FileOps displayFile ../text/cat-in-hat-4.txt
The sun did not shine.
It was too wet to play.
So we sat in the house
All that cold, cold, wet day.
```

04 - 6



A More Efficient displayFile()

```
/** This version of displayFile() avoids reading entire file into a String
    object in Java memory. But it's way more complex! The simpler
    version is fine for many applications. */
public static void displayFile (String infile) {
    try {
        BufferedReader reader = new BufferedReader(new FileReader(infile));
        String line = reader.readLine(); // Read the first line of the file.
        while (line != null) { // Line becomes null at end of file
            System.out.println(line);
            line = reader.readLine(); // Read the next line of the file
        }
        reader.close();
    } catch (IOException ex) {
        System.out.println(ex);
    }
}
```

04 - 7



Creating a File from a String

```
/** Writes the contents string to the file named by outfile.
    Displays an errors message if outfile cannot be created. */
public static void stringToFile (String outfile, String contents) {
    try {
        BufferedWriter writer =
            new BufferedWriter(new FileWriter(outfile));
        // create a new file writer
        writer.write(contents); // write contents string to file
        writer.close(); // close file writer
    } catch (IOException ex) {
        System.out.println(ex); // Handle file-not-found
    }
}
```

04 - 8



Copying Files (like Linux cp)

```
/** A simple version of a method that copies infile to outfile */  
public static void copyFile (String infile, String outfile) {  
    stringToFile(outfile, fileToString(infile));  
}
```

```
[fturbak@puma utils] java FileOps copyFile ../text/cat-in-hat-  
4.txt copycat-4
```

```
[fturbak@puma utils] java FileOps displayFile copycat-4  
The sun did not shine.  
It was too wet to play.  
So we sat in the house  
All that cold, cold, wet day.
```

We could avoid the large intermediate string of copyFile() by expanding fileToString() and writing one line at a time.

04 - 9



wordCount() (like Linux wc)

```
/** A simple version of a Linx-like word count (wc) method */  
public static void wordCount (String infile) {  
    String contents = fileToString(infile);  
    int chars = contents.length(); // number of chars in file  
    int lines = contents.split("\n").length; // number of lines in file  
    int words = contents.split("\\s+").length;  
    // contents.split("\\s+") splits contents around every  
    // substring of one or more whitespace chars.  
    System.out.println(lines + "\t" + words + "\t" + chars + "\t" + infile);  
}
```

```
[fturbak@puma utils] java FileOps wordCount ../text/cat-in-hat.txt  
349 1620 7440 ../text/cat-in-hat.txt
```

```
[fturbak@puma utils] wc ../text/cat-in-hat.txt  
349 1620 7440 ../text/cat-in-hat.txt
```

04 - 10



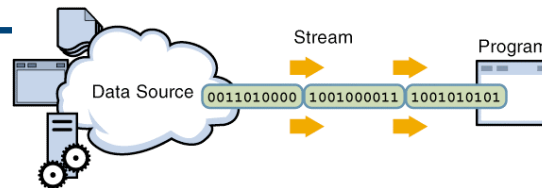
Testing wordCount()

```
/** A simple version of a Linx-like word count (wc) method */
public static void wordCount (String infile) {
    String contents = fileToString(infile);
    int chars = contents.length(); // number of chars in file
    int lines = contents.split("\n").length; // number of lines in file
    int words = contents.split("\\s+").length;
    // contents.split("\\s+") splits contents around every
    // substring of one or more whitespace chars.
    System.out.println(lines + "\t" + words + "\t" + chars + "\t" + infile);
}
```

04 - 11



Input Streams



- IO comes in InputStreams
 - The keyboard: `System.in`
 - A file: `FileInputStream()`
 - The web: `url.openStream()`
- What you read is unformatted
 - Format it into chars with an **InputStreamReader**
 - For a file you use **FileReader** (String filename)
- What you read has unpredictable length
 - Restrict it into lines with a **BufferedReader**

04 - 12



Reading lines from an Input Stream

```
public static String readLineFromInputStreamReader
    (InputStreamReader isReader) {
    try {
        BufferedReader reader = new BufferedReader(isReader);
        return reader.readLine();
    } catch (IOException ex) {
        System.out.println(ex);
        return "";
    }
}
```

04 - 13



Reading lines from other Sources

```
public static String readLineFromFile (String infile) throws IOException {
    InputStreamReader fr = new FileReader(infile);
    return readLineFromInputStreamReader(fr);
}

public static String readLineFromURL (String urlName) throws IOException {
    InputStreamReader ir = new InputStreamReader(new URL(urlName).openStream());
    return readLineFromInputStreamReader(ir);
}

public static String readLineFromKeyboard (String prompt) throws IOException {
    System.out.println(prompt);
    InputStreamReader ir = new InputStreamReader(System.in);
    return readLineFromInputStreamReader(ir);
}
```

04 - 14