File I/O in Java

By: Austin Fahsl, Shane Bruggerman, Sailesh Shamsunder

Summary:

Input from and output to files in java is done mainly through streams. Streams are a sequence of data that one can use and manipulate to write to and read from files. The class FileWriter extends OutputStreamWriter, which is a type of stream used to write to text files, and the class FileReader extends InputStreamReader, which is a type of stream used to read from text files. These types of readers and writers directly tell the operating system how to change the file. Several different classes exist for reading and writing files, including a Scanner for input and PrintWriter for output. The Scanner 'scans' for formatted input and parses what it finds as primitives or Strings. PrintWriter prints formatted output to a file and has access to most of the print methods built into the System such as print(String) and println(String). There is also a File object in java, which is constructed given the parameter of its file path as a string.

How do we create File I/O streams?

Specifically, the classes being used will be BufferedWriter and BufferedReader. These are special streams, which do not talk to the operating system directly, but instead go through a 'buffer'. This makes them more efficient and easier to use, mostly because of their newLine() and readLine() methods. To create a new BufferedWriter and BufferedReader, the following code is used:

```
File f = new File("C:\\Users\\fahslaj\\Documents\\testFile.txt");

// The FileWriter(File,Boolean) constructor creates a new writer given for the specific file.
// The Boolean determines whether the writer should add on to the end of the existing data
// (true) or erase everything from the file and start anew (false);

BufferedWriter wr = new BufferedWriter(new FileWriter(f,true));
                                // and
BufferedReader re = new BufferedReader(new FileReader(f));
```

As you can see, the BufferedWriter and BufferedReader take a FileWriter and FileReader parameter. After one is done with the BufferedWriter or BufferedReader, they should be closed with the close() method.

Note: BufferedWriter does not write to the file until the close() method is called, so be sure to close the writer before reading the same file if changes are to be saved.

How do we write and read?

Writing using a BufferedWriter is done with the write(String) method. This method adds the text onto the current end of the stream. The newLine() method is to be called to start a new line. Ex:

```
wr.write("Hello!");        wr.newLine();        wr.write("This is the next line!");
```

Reading using a BufferedReader is done with the read() or readLine() method. The read() method returns the next character in the stream. The readLine() method will return the next line in the stream, which is also the next line in the file, as one long String.

Exceptions:

Many of the methods used in input and output to files throw exceptions. A FileNotFoundException is thrown when trying to access a file that does not exist n the computer's external drives. An IOException is a general exception that occurs when many other input or output errors appear. A specific example of the IOException is the EOFException, which is thrown when a file reader reaches the end of the file stream (End of File Exception).

Sources:

- Oracle Java 7 API database
- Oracle Tutorials: Buffered Streams