ESc101 | Lecture 39
Files (Contd.)

April 15, 2008

The Operating System views every file as a sequence of bytes. On the other hand, Java views files in two different ways:

Text file Sequence of characters

Binary file Sequence of bytes

For text files, there is a need to encode the characters. Various encoding schemes are used. Most prominent ones are ASCII, Unicode (UCS-2), UTF-8, UTF-16. These encodings define a mapping between characters and the corresponding byte sequences. Text file read involves reading enough bytes from the file and converting them to the character. Text file write is the complimentary process — the character is converted to the corresponding byte sequence and then written to the file. On the other hand, binary file read/write happens one byte at a time.

Following is the hierarchy of classes for file read/write:

- Binary Files
  - InputStream (for reading)
  - OutputStream (for writing)
- Text Files
  - Reader: this class has subclasses that are used for reading text files. Its subclasses are:
    - BufferedReader: uses a buffer to read the data. This essentially increases the I/O efficiency. Also, this was the preferred method before Scanner class was introduced.
    - CharArrayReader: Reads from a character array, which is essentially a byte sequence.
    - InputStreamReader: This class has a sub-class FileReader which is used to read from files. Note that this is different from InputStream class from the binary ones.
    - StringReader: This class is used to read data from strings.
  - Writer: this class has subclasses that are used for writing to text files. Its subclasses are:
    - BufferedWriter: this is the writer equivalent of BufferedReader.
    - CharArrayWriter: this class is the writer equivalent for CharArrayReader.
    - OutputStreamWriter: this has a subclass FileWriter which can be used to write to files.

* PrintWriter: This class is used to attach to any of the output streams — character arrays, strings, files and so on.
* StringWriter: this is the writer equivalent of StringReader.

The PrintWriter class has a constructor that takes object of Writer as argument. This implies that an object of PrintWriter class can be attached to any sub-type of Writer. For example, it can be attached to a FileWriter object to print to a file, or it can be attached to a StringWriter to write to a string. The System.out object is an object of PrintWriter that is attached to stdout, and hence prints to the console.

Consider the following code snippet:

```
int i = 123;
PrintWriter pw;
try {
    FileWriter myFile = new FileWriter(“test.out”);
    pw = new PrintWriter(myFile);
} catch (IOException e) {
    System.out.println(“Could not open file. Writing to stdout.”);
    pw = System.out;
}
pw.println(i); // or pw.print(i);
```

The above code tries to create a PrintWriter object pw and attach to the file named test.out. If the file creation fails due to some reason (e.g. no permission to create file), then Java throws IOException which is handled. The handler prints the error message and assigns System.out to pw. This works like a fall-back mechanism, wherein the output performed by pw.println() etc are “redirected” to the console in case the file cannot be opened.

There are four important methods in the Writer class:

close() to close the stream so that all buffers etc. are written to the stream and we do not lose the data if, say, the program terminates prematurely. After close() is called, no more output can be performed to the associated stream.

flush() to forcefully write all the buffers to the corresponding stream. This is sometimes required to clear some internal buffers. Note that this method does not close the stream.

write(char c) writes the character *c* to the stream, after converting it to the corresponding byte sequence

write(String s) writes the characters of string *s* after converting them to the corresponding byte sequences.

Following is a code snippet for CharArrayWriter:

```
char [] carray = new char[100];
PrintWriter pw = new PrintWriter(new CharArrayWriter(carray));
pw.print(25);
```

This code prints the number 25 to the character array *carray.*