File I/O

CSC 116 – Section 002
March 16, 2005

Input and Output

- File Input: action of reading data from a file [Wu]
- File Output: action of saving, or writing, data to a file [Wu]
- File Access: refers to reading or writing from a file

Creating a File Object

- You can create a new File object by specifying the path of the file that you wish to access
  - Absolute path
  - Relative path
- If the file does not exist, then it is created
- Example:

  File f = new
  File("/afs/unity.ncsu.edu/users/s/sesmith5/CSC116/code/Hello.java");

File Methods

- canRead()
  - Returns true if the application can read the file
- canWrite()
  - Returns true if the application can write to the file
- getPath()
  - Returns a String containing the pathname
- isFile()
  - Returns true if the abstract pathname is a normal file
- isDirectory()
  - Returns true if the abstract pathname is a directory
- close()
  - Closes the file

Data Storage

- char – represents a single character
  - Written as symbols enclosed in single quotes
  - Special binary codes are used to represent a single character (ASCII)
- Text data storage
  - 12345 is stored as ‘1’ ‘2’ ‘3’ ‘4’ ‘5’
- Binary data storage
  - 12345 is stored as 00 00 30 39
- Data storage classes are in the *java.io* package

Reading Text Data

- Use FileReader class
- FileReader(String pathname)
  - Constructor of a FileReader object
- FileReader(File file)
  - Constructor of a FileReader object
- read()
  - Returns an int containing the binary value of a single character
  - Throws IOException

Reading Text Data Example

```
FileReader r = new FileReader("input.txt");
int next = r.read();
char c;
if(next != -1)
   c = (char)next;
r.close():
```


Writing Text Data

- Use FileWriter class
- FileWriter(String pathname)
- FileWriter(String pathname, boolean append)
- FileWriter(File file)
- FileWriter(File file, boolean append)
- write(String str)
  - Writes the String or character to the file
  - Throws IOException

Writing Text Data Example

```
FileWriter w = new FileWriter("output.txt");
char c = 'a';
w.write(c);
w.write("This is a String");
w.close();
```


Reading Binary Data

- Use FileInputStream class
- Can be used to read executables and all file types
- FileInputStream(String pathname)
  - Constructor of a FileReader object
- FileInputSteam(File file)
  - Constructor of a FileReader object
- read()
  - Returns an int containing a byte of data
  - Throws IOException

Reading Binary Data Example

```
FileInputStream inStream = new
   FileInputStream("input.txt");
int next = inStream.read();
byte b;
if(n != -1)
   b = (byte) next;
inStream.close();
```


Writing Binary Data

- Use FileOutputStream class
- FileOutputStream(String pathname)
- FileOutputStream (String pathname, boolean append)
- FileOutputStream (File file)
- FileOutputStream (File file, boolean append)
- write(int b)
  - Writes the byte to the file
  - Throws IOException

Writing Binary Data Example

```
FileOutputStream outStream = new
  FileOutputStream("output.txt");
byte b = 56;
outStream.write(b);
outStream.close();
```


Close Files

- Always make sure to close all the files that you've opened!
  - Use the close() method
- Example:

```
FileReader r = new FileReader("input.txt");
r.close();
```

Reading Text Data Lines

- Use FileReader and Buffered Reader classes
- BufferedReader: “Read text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines.” [Java API]
- BufferedReader.readLine()
  - Returns a String from the file
  - Throws an IOException

Reading Text Data Lines Example

```
FileReader r = new FileReader(“input.txt”)
BufferedReader in = new BufferedReader(r);
String line = in.readLine();
reader.close();
```

Writing Text Data Lines

- Use FileWriter and PrintWriter classes
- PrintWriter: “Print formatted representations of objects to a text-output stream.” [Java API]
- PrintWriter does not have any methods that throw IOExceptions
- PrintWriter.print(String line)
  - Prints a line of text
- PrintWriter.println(String line)
  - Prints a line of text with a new line at the end
- PrintWriter does not flush the buffer until a println is called

Writing Text Data Lines Example

```
FileWriter w = new FileWriter(“output.txt”);
PrintWriter out = PrintWriter(w);
out.println(“Hello World!”);
w.close();
```

References

- Jason Schwarz's Lecture 16 slides: http://courses.ncsu.edu/csc116/
- Wu – Chapter 9 and 12
- Java API – File, FileReader, FileWriter, FileInputStream, and FileOutputStream