Reading/Writing Files, Webpages

CS2110, Recitation 9

Reading files/ webpages

I/O classes are in package java.io.
To import the classes so you can use them, use

import java.io.*;

Class File

An object of class File contains the path name to a file or directory. Class File has lots of methods, e.g.

- exists()
- canRead()
- canWrite()
- delete()
- createNewFile()
- length()
- ... (lots more) ...

File f = new File("res/map1.xml");

File path is relative to the package in which the class resides. Can also use an absolute path. To find out what absolute path's look like on your computer, use

f.getAbsolutePath();

Suppose f contains a File that describes a directory. Store in b a File[] that contains a File element for each file or directory in directory given by f

File[] b = f.listFiles();

File[] returns an array of file and directory names as Strings, instead of as File objects
f.mkdir() will create the directory if it does not exist.

Input Streams

Stream: a sequence of data values that is processed — either read or written — from beginning to end. We are dealing with input streams.

Read input stream for a file is by creating an instance of class FileReader:

FileReader fr = new FileReader(f);

fr.read() // get next char of file

Too low-level! Don’t want to do char by char.

Reading a line at a time

Class BufferedReader, given a FileReader object, provides a method for reading one line at a time.

FileReader fr = new FileReader(f);
BufferedReader br = new BufferedReader(fr);

Then:

String s = br.readLine(); // Store next line of file in s

When finished with reading a file, it is best to close it!

br.close();
Example: counting lines in a file
/** Return number of lines in f. Throw IOException if problems encountered when reading */
public static int getSize(File f) throws IOException {
  FileReader fr = new FileReader(f);
  BufferedReader br = new BufferedReader(fr);
  int n = 0; // number of lines read so far
  String line = br.readLine();
  while (line != null) {
    n = n+1;
    line = br.readLine();
  }
  br.close();
  return n;
}

FileReader(String)
When calling FileReader with a String argument s, s can be a name relative to the Eclipse project you are running.
When running a procedure main in Project a0, because folder SpeciesData is in a0, to read file A0.dat, we can use FileReader fr = new FileReader("SpeciesData/A0.dat");

Given method main an argument
public static void main(String[] args) {...}
In Eclipse, when you do menu item Run -> Run or Run -> Debug
Eclipse calls method main. Default is main(null);
To tell Eclipse what array of Strings to give as the argument,
Use menu item Run -> Run Configurations...
or Run -> Debug Configuration...

Class URL in package java.net
URL url = new URL("http://www. ... /links.html");
A URL (Universal Resource Locator) describes a resource on the web, like a web page, a jpg file, a gif file
The "protocol" can be:
http (HyperText Transfer Protocol)
https
ftp (File Transfer Protocol)

Window Run Configurations
This Arguments pane of Run Configurations window gives argument array of size 3:
args[0]: "SpeciesData/A0.dat"
args[1]: "2"
args[2]: "what for?"
Click Arguments pane
Quotes OK, but not needed because of space char
Quotes needed

Reading from an html web page
Given is URL url = new URL("http://www. ... /links.html");
To read lines from that webpage, do this:
1. Create an InputStreamReader:
   InputStreamReader isr = new InputStreamReader(url.openStream());
   Have to open the stream
2. Create a BufferedReader:
   BufferedReader br = new BufferedReader(isr);
3. Read lines, as before, using br.readLine()
**javax.swing.JFileChooser**

Want to ask the user to navigate to select a file to read?

```java
JFileChooser jd = new JFileChooser();
jd.setDialogTitle("Choose input file");
int returnVal = jd.showOpenDialog(null);
```

Starting always from the user’s directory can be a pain for the user. User can give an argument that is the path where the navigation should start.

```java
jd.showOpenDialog("/Volumes/Work15A/webpage/ccgb/");
```

returnVal is one of:
- JFileChooser.CANCEL_OPTION
- JFileChooser.APPROVE_OPTION
- JFileChooser.ERROR_OPTION

File f = jd.getSelectedFile();

**Writing files**

Writing a file is similar. First, get a BufferedWriter:

```java
FileWriter fw = new FileWriter("the file name",false);
BufferedWriter bw = new BufferedWriter(fw);
```

Then use

```java
bw.write("...");
```

to write a String to the file.

```java
bw.close(); // Don’t forget to close!
```

false: write a new file

true: append to an existing file

---