Reading/Writing Files, Webpages

CS2110, Recitation 9
I/O classes are in package `java.io`. To import the classes so you can use them, use

```java
import java.io.*;
```
An object of class **File** contains the path name to a file or directory. Class **File** has lots of methods, e.g.

```java
f.exists()   f.canRead()   f.canWrite()
f.delete()   f.createNewFile()
f.length()   ...
```

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

```java
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()
**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:

```java
FileReader fr = new FileReader(f);
```

```java
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 IO Exception 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;
}

Example: counting lines in a file

/**
  * Return number of lines in f.
  * Throw IO Exception 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;
}

Always use this pattern to read a file!

  line = first line;
while (line != null) {
  Process line;
  line = next line;
}
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

```java
FileReader fr = new FileReader("SpeciesData/A0.dat");
```

Folder `SpeciesData`, in project `a0`, has several files in it.
Given method main an argument

```java
public static void main(String[] args) { ... }
```

Parameter: String array

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...

(see next slide)
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 needed because of space char

Quotes OK, but not needed
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)
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());

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);

File f = jd.getSelectedFile();

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

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

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.
Writing files

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

```java
FileWrite 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