

Reading/Writing Files, Webpages

CS2110, Recitation 8

1

Reading files/ webpages

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

```
import java.io.*;
```

2

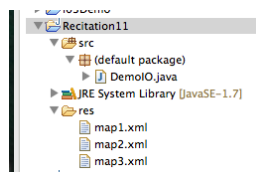
Class File

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

```
f.exists()    f.canRead()    f.canWrite()
f.delete()   f.createNewFile()
f.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();
```

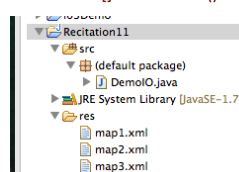
3

Class File

```
f.isDirectory()    f.listFiles()    f.list()    f.mkdir()
```

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()
```



`f.list()`: return an array of file and directory names as Strings, instead of as `File` objects

`f.mkdir()`: create the directory if it does not exist.

4

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);
```

`f` can be a `File` or a `String` that gives the file name

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

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

5

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
// (null if none)
```

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

```
br.close();
```

6

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;
}
    
```

Don't forget!

(write as while loop)

Always use this pattern to read a file!

```

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

Pattern to read a file

Always use this pattern to read a file!

```

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

```

line= br.readLine();
while (line != null) {
    Process line;
    line= br.readLine();
}
    
```

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());
            
```

Have to open the stream
2. Create a Buffered Reader:


```

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?

```

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

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

```

File f= jd.getSelectedFile();
    
```

```

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

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:

```

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

Then use

```

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

to write a String to the file.

```

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

false: write a new file
 true: append to an existing file