Overview

This lecture provides an intro to the Java File API
The Classes & Methods you need to read/write files

Everything you need is part of java.io package
So at the top of your code you should add:

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

(Or import the individual classes if you prefer)

The File Class

Everything is based around the "File" class

This is used to represent any file in the filesystem

Providing access to information about that file

As well methods to manipulate it

Directories

Directories/folders are a special type of file in Java

This makes sense in some ways...

When you list the contents of a directory:

```
File[] documents = documentFolder.listFiles();
```

You will get back a bunch of files
(Some of which will be subdirectories)

To check if a particular file is a directory, use:

```
someFile.isDirectory();
```

Exceptions

Pretty much anything we attempt to do to a file can cause exceptions to be generated

As a result, everything needs to be inside a "try"
And we need to catch various exceptions, including:

- IOException
- FileNotFoundException

(As appropriate to the current situation)

Filenames

When accessing a file, we identify it by filename

A String composed of the name & path of the file

For example:

```
String filename = "cv.txt";
```

Paths can be relative:

```
String filename = "../documents/cv.txt";
```

Or absolute:

```
String filename = "/user/bob/documents/cv.txt";
```

Platform Independent Separator

The problem is that the following:

```
String name = "email/cv.txt";
```

Would only work on OSX or Linux (not Windows)

Instead we should really use:

```
String name = "email" + File.separator + "cv.txt";
```

This will work no matter which platform we run it on

(File.separator is replaced with the correct character)

Checking if a file exists

Before we attempt to do anything to a file
We should make sure that file actually exists !

This is done by creating a new instance of File class
Passing it the name of the file we are interested in:

```
String name = "email" + File.separator + "cv.txt";
File fileToOpen = new File(name);
```

We can then check to see if this file actually exists:

```
if(fileToOpen.exists()) {
   // Do something to the file !
}
```

LESSON LEARNT

A File Object doesn't represent an ACTUAL file

It only represents a POTENTIAL file !

Creating new Files

If the file doesn't exist, then we should create it !
Done by asking the File instance to create the file:

```
fileToOpen.createNewFile();
```

This returns a boolean (the success of the action)
It might be worth checking this !
(You might not have write permission to the folder)

Creating new Directories

If we want to create a directory instead of a file

Then we need to use the "mkdir" method:

```
String name = "email";
File emailFolder = new File(name);
emailFolder.mkdir();
```

Or "mkdirs" to create multiple levels of folder:

```
String name = "docs" + File.separator + "email";
File emailFolder = new File(name);
emailFolder.mkdirs();
```

Reading and Writing

Java provides various Helper Classes to aid with IO

Offering different alternatives to access file content

The one that we will focus on makes use of:

- FileReader: Class to read data from a File
- FileWriter: Class to write data to a File

FileWriter

FileWriter allows us to write chars or Strings to a file:

```
String name = "email" + File.separator + "cv.txt";
File fileToOpen = new File(name);
FileWriter writer = new FileWriter(fileToOpen);
writer.write("Hello\n");
writer.write('a');
writer.flush();
writer.close();
```

FileReader

FileReader allows us to read chars from a file:

```
String name = "email" + File.separator + "cv.txt";
File fileToOpen = new File(name);
FileReader reader = new FileReader(fileToOpen);
char[] buffer = new char[10];
reader.read(buffer, 0, buffer.length);
reader.close();
```

More Advanced Readers and Writers

Readers and Writers are quiet low-level

And are very much reminiscent of C code

Instead, we can use some more high-level classes:

- BufferedReader
- BufferedWriter

Lets look at BufferedReader in more detail...

BufferedReader

BufferedReader can read in a whole line as a String:

```
String name = "email" + File.separator + "cv.txt";
File fileToOpen = new File(name);
FileReader reader = new FileReader(fileToOpen);
BufferedReader buffReader = new BufferedReader(reader);
String firstLine = buffReader.readLine();
buffReader.close();
```

Other useful methods of the File Class

- delete
- renameTo
- setReadable
- setWritable
- list
- getName
- getPath
- getParent
- length