SENG1110/SENG6110
Object Oriented Programming

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

FACULTY OF ENGINEERING AND BUILT ENVIRONMENT

www.newcastle.edu.au

Lecture 11

External files

Outline

- Text Files and Binary Files
- Creating a Text File
- Appending to a text File
- Reading from a Text File
- The Class `File`
- Programming Example: Reading a File Name from the Keyboard
- Using Path Names
- Methods of the Class `File`
- Defining a Method to Open a Stream
- Reading/writing objects

The Concept of a Stream

- Use of files
  - Store Java classes, programs
  - Store pictures, music, videos
  - Can also use files to store program I/O
- A *stream* is a flow of input or output data
  - Characters
  - Numbers
  - Bytes
- Streams are implemented as objects of special stream classes
  - Class `Scanner`
  - Object `System.out`

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Why Use Files for I/O

- Keyboard input, screen output deal with temporary data
  - When program ends, data is gone
- Data in a file remains after program ends
  - Can be used next time program runs
  - Can be used by another program

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Text Files and Binary Files

- All data in files stored as binary digits
  - Long series of zeros and ones
- Files treated as sequence of characters called *text files*
  - Java program source code
  - Can be viewed, edited with text editor
- All other files are called *binary files*
  - Movie, music files
  - Access requires specialized program

Creating a Text File

- File is empty initially
  - May now be written to with method `println`
- Data goes initially to memory buffer
  - When buffer full, goes to file
- Closing file empties buffer, disconnects from stream

Creating a Text File

- Class `PrintWriter` defines methods needed to create and write to a text file
  - Must import package `java.io`
- To open the file
  - Declare *stream variable* for referencing the stream
  - Invoke `PrintWriter` constructor, pass file name as argument
  - Requires `try` and `catch` blocks

Creating a Text File

- View CodeSamplesWeek11_Files
  `class TextFileOutput`

Sample screen output

```
Enter three lines of text:
A tall tree
in a short forest is like
a big fish in a small pond.
Those lines were written to out.txt
```

Resulting File

```
1 A tall tree
2 in a short forest is like
3 a big fish in a small pond.
```

You can use a text editor to read this file.

Creating a Text File

- A file has two names in the program
  - File name used by the operating system
  - The stream name variable
- Opening, writing to file overwrites pre-existing file in directory

Reading from a Text File

- Note CodeSamplesWeek11_Files
  `class TextFileInputDemo`
- Reads text from file, displays on screen
- Note
  - Statement which opens the file
  - Use of `Scanner` object
  - Boolean statement which reads the file and terminates reading loop

Appending to a Text File

- Opening a file new begins with an empty file
  - If already exists, will be overwritten
- Some situations require appending data to existing file
- Command could be

```
outputStream =
    new PrintWriter(
    new FileOutputstream(fileName, true));
```

- Method `println` would append data at end

Reading from a Text File

Sample screen output

```
The file out.txt
contains the following lines:

1 A tall tree
2 in a short forest is like
3 a big fish in a small pond.
```

Reading from a Text File

- Figure 10.3 Additional methods in class **`Scanner`**

Scannner_Object_Name`.hasNext()`
Returns true if more input data is available to be read by the method `next`.

Scannner_Object_Name`.hasNextDouble()`
Returns true if more input data is available to be read by the method `nextDouble`.

Scannner_Object_Name`.hasNextInt()`
Returns true if more input data is available to be read by the method `nextInt`.

Scannner_Object_Name`.hasNextLine()`
Returns true if more input data is available to be read by the method `nextLine`.

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Programming Example

- Reading a file name from the keyboard
- View CodeSamplesWeek11_Files
  `class TextFileInputDemo2`

Sample screen output

```
Enter file name: out.txt
The file out.txt
contains the following lines:

1 A tall tree
2 in a short forest is like
3 a big fish in a small pond.
```

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

The Class `File`

- Class provides a way to represent file names in a general way
  - A **`File`** object represents the name of a file
- The object
  `new File ("treasure.txt")`
  is not simply a string
  - It is an object that *knows* it is supposed to name a file

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Using Path Names

- Files opened in our examples assumed to be in same folder as where program run
- Possible to specify path names
  - Full path name
  - Relative path name
- Be aware of differences of pathname styles in different operating systems

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Methods of the Class File

- Recall that a `File` object is a system-independent abstraction of file's path name
- Class `File` has methods to access information about a path and the files in it
  - Whether the file exists
  - Whether it is specified as readable or not
  - Etc.

Defining a Method to Open a Stream

- Method will have a `String` parameter
  - The file name
- Method will return the stream object
- Will throw exceptions
  - If file not found
  - If some other I/O problem arises
- Should be invoked inside a `try` block and have appropriate `catch` block

Methods of the Class File

- Figure 10.4 Some methods in class `File`

`public boolean canRead()` Tests whether the program can read from the file.
`public boolean canWrite()` Tests whether the program can write to the file.
`public boolean delete()` Tries to delete the file. Returns true if it was able to delete the file.
`public boolean exists()` Tests whether an existing file has the name used as an argument to the constructor when the `File` object was created.
`public String getName()` Returns the name of the file. (Note that this name is not a path name, just a simple file name.)
`public String getPath()` Returns the path name of the file.
`public long length()` Returns the length of the file, in bytes.

Defining a Method to Open a Stream

- Example code

```
public static PrintWriter openOutputTextFile(String fileName)
                    throws FileNotFoundException, IOException
{
    PrintWriter toFile = new PrintWriter(fileName);
    return toFile;
}
```

- Example call

```
PrintWriter outputStream = null;
try
{
    outputStream = openOutputTextFile("data.txt");
}
< appropriate catch block(s) >
```

Other Input/Output situation

- Java also provides methods for reading and writing complete objects
- The process of writing and reading complete objects to and from files is called serialization
- All classes to be serialized must implement the Serializable interface
- This interface is defined in the java.io package

Serializing the Person and Couple Classes


```
import java.io.*;

public class Person implements Serializable {
      ...
      ...
}


import java.io.*;

public class Couple implements Serializable {
      ...
      ...
}
```


Writing Objects in Couple.java


```
...
Person she,he;
...
try
{
   FileOutputStream fos = new FileOutputStream("data");

   ObjectOutputStream oos = new ObjectOutputStream(fos);

   oos.writeObject(she);
   oos.writeObject(he);

fos.close();
}

catch(Exception e) {
   System.out.println("Error in output:" + e.toString());
}
```


Reading Objects in Couple.java


```
...
Person she,he;
...
try
{
   FileInputStream fis = new FileInputStream("data");

   ObjectInputStream ois = new ObjectInputStream(fis);

   she = (Person) ois.readObject(); // Note: cast
}

catch(Exception e)
{
    System.out.println("Error in output:" + e.toString());
}
```

Binary-File I/O with Class Objects - Example

- View CodeSamplesWeek11_Files
  `class Species, ClassObjectIODemo`

```
Records sent to file species.record.
Now let's reopen the file and echo the records.
The following were read
from the file species.record:
Name = Calif. Condor
Population = 27
Growth rate = 0.02%

Name = Black Rhino
Population = 100
Growth rate = 1.0%
End of program.
```

Sample screen output

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Your task

- Read
  - Lecture slides
  - Chapter 10

- Exercises
  - MyProgrammingLab

THE UNIVERSITY OF NEWCASTLE AUSTRALIA