Reading & Writing Files

COMP163

"Programming today is a race between software engineers striving to build bigger and better idiot-proof programs, and the universe trying to build bigger and better idiots.

So far, the universe is winning."

Rick Cook

Second Exam

- The second exam in COMP163 will be on Wednesday, October 17
- The exam will cover topics since the first exam
  - IF statements
  - Loops
  - Reading and writing files

Lab Quiz

- There will be a quiz in lab on Thursday, October 18
- The quiz is just like a regular lab assignment, but you must do it by yourself
- You must be physically present in Graham 203 to take the lab quiz

Reading Files

- Java programs can read input from files as well as from the keyboard
- The Scanner method can be used to read files
- There are also many other classes and methods that can be used to read a file

Scanner with Files

- When you create an object of the Scanner class, you can specify a File object instead of System.in

```
java.io.File  hippo = new java.io.File("numbers.txt");
java.util.Scanner rhino = new java.util.Scanner( hippo);

        // read a number from the file
double sloth = rhino.nextDouble();
```

Locating the File

- The easiest way for your program to find the file you want to read is to put it in the same directory as your Java program
- You can also give a path name to Java

```
java.io.File  hippo = new java.io.File(
    "C:\\Users\\me\\Documents\\numbers.txt");
```

Escape Characters

- Remember that special characters in Java are represented by a backslash (\ ) and a letter
- The backslash and letter form one character

\t	tab
\n	new line (return)
\"	quote
\\	backslash

- Escape characters are only needed in the Java source code, not input

Reading with Scanner

- Reading from a file with a Scanner is just like reading from the keyboard
- All the usual methods are available
- Useful methods are the **`hasNext()`** methods which are true if there is more data in the file

Useful Scanner Methods

- **nextInt**() – read an int
- **nextDouble**() – read a double
- **nextLine**() – read the whole line as a String
- **next**() – read the next word as a String
- **hasNext**() – true if there is more *data*
- **hasNextInt**() – true if there is another int
- **hasNextDouble**() – true if there is another double
- **close**() – close the file when done

What will be displayed?

```
String platypus = "You\\me +\"world\"";
System.out.println( platypus );
```

A. You\\me +\"world\"
B. You\me +"world"
C. You\me "world"
D. You me world
E. error

Scanner File Exceptions

- When you create a Scanner object using a File object, it could throw an exception
- Your program **must** handle the possible exception
- The easiest way to do this is to throw the exception

```
public static void main( String[] unused ) throws
                    java.io.IOException {
```

What will be displayed?

```
Scanner keyboard = new Scanner(System.in);
while (keyboard.hasNextInt()) {
        int squirrel = keyboard.nextInt();
        System.out.print(squirrel);
}
```

when you type

```
2  7  dog
```

A. 2 7 dog
B. 2 7
C. 2
D. 7
E. error

```
/*  Example displaying big numbers in a file */
public class ReadFile {
  public static void main(String[] dog) throws  java.io.IOException {
    java.util.Scanner keyboard = new java.util.Scanner(System.in);
    System.out.print("Enter the filename >");
    String goat = keyboard.next();

    java.io.File  frog = new java.io.File( goat );
    java.util.Scanner  fish = new java.util.Scanner( frog );

    while ( fish.hasNextInt() ) {
          int cow= fish.nextInt();
          if ( cow > 100 ) {
               System.out.println( cow );
          }
    }
    fish.close();
  }}
```

Reading from the Web

- In Java you can read a file from a web server in almost the same way you read a file
- You need to create a **java.net.URL** object instead of a java.io.File object
- Instead of passing the File object to Scanner use **`openStream()`** method of a URL object
- You have to throw java.io.IOException

```
/*  Example reading a web file */
public class ReadWeb {
   public static void main(String[] args) throws
                                    java.io.IOException {
      java.util.Scanner keyboard = new
                    java.util.Scanner(System.in);
      System.out.print("Enter the URL >");
      String url = keyboard.next();

      java.net.URL webFile = new java.net.URL( url );
      java.util.Scanner  fileIn = new
            java.util.Scanner(webFile.openStream());
      String line;
      while ( fileIn.hasNext() ) {
          line = fileIn.nextLine();
          System.out.println( line );
      }
      fileIn.close();
   }
```

Write with other students

- Sometimes users will enter a URL without the “HTTP://” prefix *(which is required)*. Write a short Java segment that will add “HTTP://” if omitted.

Possible Solution

```
String prefix = url.substring(0,4);
if ( !prefix.equalsIgnoreCase("http") ) {
        url = "http://" + url;
}
```

System Class

- The System class contains several useful class fields and methods about your System or computer
- **in** – a java.io.InputStream object connected to "Standard input", usually your keyboard
- **out** – a java.io.PrintStream object connected to "Standard output", usually your screen
- **err** – a Printstream object where, by convention, error messages are to be displayed. Usually the same as out

System.out

- The **`out`** variable of the System class is a java.io.PrintStream object
- Objects of the PrintStream class have several methods including **`print`**, **`printf`** and **`println`**

The java.io.PrintWriter Class

- The java.io.PrintWriter class allows you to write data to a file using the **`print`** and **`println`** methods, as you have been using to display data on the screen
- Just as with the **`System.out`** object, the **`println`** method of the **`PrintWriter`** class will place a newline character after the written data
- The **`print`** method writes data without writing the newline character

Writing to a File

- To write to a file, create an object of the java.io.PrintWriter class

```
java.io.PrintWriter  rabbit = new
                              java.io.PrintWriter("stuff.txt");

rabbit.println("Hi");  // Written to stuff.txt
```

Warning: if the file already exists, it will be erased and replaced with a new file.

Terrible Mistake

```
public class MyProg {
                        …
java.io.PrintWriter  hare = new
                java.io.PrintWriter("MyProg.java");
                        …
hare.println("Hi");  // Overwrites your program
```

Using a PrintWriter

```
java.io.PrintWriter  aardvark = new
                        java.io.PrintWriter("Results.txt");
double dog = 42.0;
aardvark.println("The answer is " + dog);
aardvark.close();
```

Closing the File

- When you are done reading or writing a file, you should call the **`close()`** method on the Scanner or PrintWriter object
- Output files can *disappear* if they are not closed properly

Example Reading and Writing a File

```
import java.io.*;
public class RunningSum {
   public static void main(String[] args) throws
                              FileNotFoundException {
      double sum = 0.0, number;
      int count = 0;
      File inputFile = new File("numbers.txt");
      java.util.Scanner fileIn = new
                  java.util.Scanner(inputFile);
      PrintWriter fileOut = new PrintWriter("sum.txt");
      while (fileIn.hasNextDouble()) {
         number = fileIn.nextDouble();
         sum += number;
         count++;
         fileOut.println(number + " \t" + sum );
      }
      fileIn.close();
      fileOut.close();
      System.out.println(count + " numbers read");
   }
```

Write with your teams

- Write a simple program segment that will write the numbers 1 to 10 to the file myNum.txt

```
PrintWriter  trout = new PrintWriter("myNum.txt");
```

Possible Solution

```
PrintWriter  trout = new PrintWriter("myNum.txt");


for (int goat = 1; goat <= 10; goat++) {
        trout.println( goat );
}
```

File Exceptions

- When you create a Scanner or PrintWriter object for file I/O, it could throw java.io.IOException
- Your program must handle the possible exception
- The easiest way to do this is to throw the exception

```
public static void main( String[] unused ) throws
                                    java.io.IOException {
```

Bytes and Characters

- In Java, characters are stored using the Unicode format, which allows 16 bits (2 bytes) for each character
- Many files are created using the ASCII format which provides 8 bits (1 byte) per character
- If you are using the appropriate classes and methods to read a file, Java will automatically translate from ASCII to Unicode

Comments

Programs in COMP163 must have comments that

- Give your name at the top
- Provide a brief description of the program
- Describe the purpose of each variable

- Describe the purpose of each method and their parameters

Commenting Variables

- There must be a comment for each variable explaining the logical purpose of the variable
- The comment should tell the reader something that you do not know from reading the java

Good Comment

```
int sumYear = 0;   // sum of the input years
```

Miserable Comment

```
int sumYear = 0;   // declare sum as an int and set to 0
```

format Method

- The java.io.PrintWriter class has two identical methods, format and printf, to format output

  printf(String format, var1, var2, …)

- Writes the variables to the output as specified by the format string
- Very similar to printf in the C programming language

java.io.PrintWriter

- The java.io.PrintWriter class is commonly used to write data to a file or the screen
- **System.out** is an object of the class java.io.PrintWriter
- You can call the format and printf methods on System.out

```
System.out.printf("A piece of pie %5.3f\n", Math.PI);
```

Format Descriptors

- The format string may contain text with descriptors located in it
- The descriptors start with a percent sign, % followed optionally by a length and then a format type character

format	data type	result
'd'	int or long	The result is formatted as a decimal integer
'f'	double or float	The result is formatted as a floating point number
‘s’	String	The string

Output length

- You can specify a number between the % and the descriptor character to indicate the minimum number of characters to display
- For doubles, you can specify the maximum number of digits to the right of the decimal point

%*minlength*.*maxprecision***f**

Format Examples

```
double   e = 2.718281828459045;
System.out.format("answer is %5.3f", e);
```

will display **“answer is 2.718”**

```
System.out.format("answer is %8.4f", e);
```

will display **“answer is   2.7183”**

What will be displayed?

```
double dog = 9.87654321;
System.out.format("answer is %6.3f", dog);
```

A. `answer is  9.87654321`

B. `answer is  9.876`

C. `answer is  9.877`

D. `answer is 9.876`

E. `answer is 9.877`

Creating Columns

- You can specify a number between the % and the descriptor character to indicate the minimum number of characters to display
- This is useful to display strings or numbers in a column

%*minlength***s** *or* **%***minlength***d**

Formatting Integers

```
for (int num = 1; num < 10000; num*=17) {
    System.out.format("num %4d and %6d num squared \n",
                                    num, num*num);
}
```

generates the output

```
num    1 and       1 num squared
num   17 and     289 num squared
num  289 and   83521 num squared
num 4913 and 24137569 num squared
```

Newline Character

- Like the System.out.print method, format and printf do not automatically end with a new line
- You can put the new line character `'\n'` in the format string to specify where you want the output to go to a new line

What is displayed?

```
double x = 10.0;
double y = x * 2.0 / 3.0;
System.out.printf("y is %6.2f", y);
```

A. y is 10.00

B. y is   6.66

C. y is   6.67

D. y is   6.66667

E. y is   6.666666666667

Lab Quiz

- There will be a quiz in lab on Thursday, October 18
- The quiz is just like a regular lab assignment, but you must do it by yourself
- You must be physically present in Graham 203 to take the lab quiz

Second Exam

- The second exam in COMP163 will be on Wednesday, October 17
- The exam will cover topics since the first exam
  - IF statements
  - Loops
  - Reading and writing files