Announcements

- Midterm 10/2 (this Wednesday)
  - Only need yourselves and a pen / pencil
  - Will not need a calculator… so don’t bring one or use one in the exam...
- Still working on Program 3… Will push back the due date again to ensure you have at least a week to work on the program

Questions for the Midterm

Streams

- An **OutputStream** is a class that supports writing information to some sort of output.
    - It provides several methods for writing a sequence of bytes to a destination.
- **System.out** is a predefined output stream that is associated with a system's standard output, this is usually set to be a computer's screen.
- The "**out**" variable in the **System** class is a reference to a **PrintStream** object
- The **PrintStream** class inherits from the OutputStream class.
- The **PrintStream** class adds functionality such as: **print** and **println** that you commonly use

Streams (cont.)

- An **InputStream** class supports reading from an input, such as a keyboard.
  - It provides several methods that allow a programmer to extract bytes from a particular source
- **System.in** is a predefined input stream that is associated with a system's standard input (a keyboard).
- System.in is an input **byte stream**, every 8 bits is returned as an int. This inconvenience is solved by using a **Scanner** object.
- The **Scanner** class wraps the input stream, then reads in a sequence of bytes and converts the correct number of bytes into a desired data type.
  - In order to use the Scanner class, you must import it like so: import java.util.Scanner

Streams (cont.)

- When using an InputStream we need a throws clause to the definition of main because both input and output streams may throw an IOException
    - public static void main (String[] args) throws IOException {

Streams (cont.)

The Scanner class has many convenient methods that are at your disposal. The sections of methods that you particularly care about, are the ones that begin with the word: “next”

Let’s take a look at Java’s documentation...

Streams (cont.)

How to read input from a file:

- The **FileInputStream** class (which is derived from InputStream), allows a programmer to read bytes from a file.
    - In order to use the FileInputStream class you must import it like so: import java.io.FileInputStream;
- You must also import the IOException class since you will need it for when FileInputStream throws file I/O exceptions
    - import java.io.IOException;
- An example of throwing an IOException might be:
    - When you try to read from a file that doesn't exist

Streams (cont.)

- The constructor to a FileInputStream object is a String with the desired file name
  - FileInputStream fStream = new FileInputStream(“midterm1answers.txt”);
- The FileInputStream class only supports a basic byte stream, so using a Scanner object with it is useful.
- Using the Scanner object you can call one of the many “next” methods and access the content in the file.
- Once you are done accessing the file then you should call the “close” method on the FileInputStream class.
  - fStream.close();

Streams (cont.)

- If you want to access all of the content in a file rather than a small portion, then use a while loop in conjunction with the Scanner class's "hasNext" method:

```
FileInputStream fStream = new FileInputStream("midterm2answers.txt");
Scanner scanner = new Scanner(fStream);
while ( scanner.hasNext() ) {
        System.out.println( scanner.next() );
}
```

Streams (cont.)

How to write output to a file:

- A **FileOutputStream** allows a programmer to write bytes to a file.
  - import java.io.FileOutputStream;
- FileOutputStream inherits from the OutputStream class, it only supports writing basic bytes as output, which is why we use a: **PrintWriter** object.
  - PrintWriter will function similar to how Scanner did for FileInputStream, acting as a convenient wrapper
- The PrintWriter class allows a programmer to output various data types to the underlying character stream in a manner similar to System.out.
  - PrinterWriter pWriter = new PrintWriter(fOutStream);

Streams (cont.)

- Sometimes you will notice your PrintWriter doesn't seem to be writing to the output like it should.
- This is likely because a **buffer** is being filled with the information you are trying to write out but has not yet triggered the actual release of that information to the desired destination.
    - You can think of a buffer as essentially an array holding the information you are trying to write out.
- The solution is to trigger the release of the information in the buffer via the "**flush**" method.

Let's look at an example...