Input and Output

Oct 11

Scanner, FIleReader, try..catch

MultiDimensional Arrays

- Arrays can have 2 dimensions (or more)!!
  - int[][] chessBoard = new int[8][8];
  - chessBoard[0][0] = 1;

• Whiteboards

- Write a program that defines a 10x10 array
  - Into each spot in array put the value (row+1) * (column+2)
- Then print the contents of the array with all of row on one line
  - `2 4 6 8 10 12 14 16 18 20`
  - `4 6 8 10 12 14 16 18 20 22`
  - `etc`

I/O Overview

- Most programs transform some sort of input to some sort of output
- So far we've seen runtime args as input, and printing to console as output
- When we use System.out.println, this sends data to the "standard output", which by default displays text in the Terminal
- We can also read data from "standard in" which by default reads text from the Terminal -- ie the keyboard
  - Other input sources may include files, databases, etc.
  - Other output targets may include files, graphics, sound, etc.
- To simplify reading/writing, we use libraries (like System.out.println) that other people have created

Reading from Standard.in

Using Scanner

- Standard.in defaults to the keyboard
  - Easiest use of Standard.in is within the Scanner class
  - Start a Scanner and tell it to read Standard.in
    - new Scanner(Standard.in)
  - Scanner has lots of really useful functions
    - next
    - nextInt
    - nextDouble
    - etc

```
import java.util.Scanner;

public class StdIn113 {

    public static void main(String[] args) {
        int[] forward = new int[5];
        Scanner scnr = new Scanner(System.in);
        for (int i = 0; i < 5; i++) {
            System.out.println("Enter an integer");
            forward[i] = scnr.nextInt();
        }
    }
}
```

More on Using Scanner

- Previous program reads exactly 5 items
- Suppose task is to get as much as use enters then compute average
- Need a way
  - for user to say "I am done"
  - for program to use recognize

- CTRL-d
- Scanner hasNext() function
  - This requires that the user hit CTRL-d
    - How do they know?? What else can we do??

```
import java.util.Scanner;

public class SIAverage {
    public static void main(String[] args) {
        int sum = 0;
        int count = 0;
        Scanner scnr = new Scanner(System.in);
        while (scnr.hasNext()) {
            sum += scnr.nextInt();
            count++;
        }
        System.out.println("Average " + (sum / count));
    }
}
```

Input from Files

Scanner again

- Scanner can also read files!
  new Scanner(new File("file.txt"));
- Rather than Standard.in we have new File("file.txt")
  - This tells Scanner to read a file
- Otherwise the program is much the same
- BUT this does not compile!!!!

```
import java.util.Scanner;
import java.io.File;
public class File113 {

    public static void main(String[] args) {
        int[] forward = new int[5];
        Scanner scnr = new Scanner(new File("File113.java"));
        for (int i = 0; i < 5; i++) {
           forward[i] = scnr.nextInt();
        }
    }
}
```

Only Change

try..catch

a Java gotcha

```
javac File113.java
File113.java:7: error: unreported exception FileNotFoundException;
must be caught or declared to be thrown
        Scanner scnr = new Scanner(new File("File113.java"));
                       ^
1 error
```

- For some types of problems java requires that the programmer provide code for handling the issue.
- The syntax for this is
- What kind of problems might occur?
- What is "Handling the problem"?

```
try {
// Code that could have problems
} catch (Exception e) {
// Code for "handling" the problem
}
```

Handling .. first try

```
import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;
public class File113tc {

    public static void main(String[] args) {
        String[] forward = new String[5];
        try {
            Scanner scnr = new Scanner(new File("File113.java"));
        }
        catch (FileNotFoundException fnf) {
            System.out.println("Problem: " + fnf);
        }
        for (int i = 0; i < 5; i++) {
            forward[i] = scnr.nextInt();
        }
    }
}
```

Handling .. again

```
import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;
public class File113tc2 {

    public static void main(String[] args) {
        int[] forward = new int[5];
        Scanner scnr = null;
        try {
            scnr = new Scanner(new File("File113.java"));
        }
        catch (FileNotFoundException fnf) {
            System.out.println("Problem: " + fnf);
            return;
        }
        for (int i = 0; i < 5; i++) {
            forward[i] = scnr.nextInt();
        }
    }
}
```

Finale

- Wrap code in try catch
  - watch out for scope
- Anything you open should be closed
  - reading from something that you closed will cause an error
- When reading, always make sure that there is something to read.

```
import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;
public class File113tc3 {

    public static void main(String[] args) {
        try {
            String[] forward = new String[5];
            Scanner scnr = new Scanner(new File("test.txt"));
            for (int i = 0; i < 5; i++) {
                if (scnr.hasNext()) {
                    forward[i] = scnr.next();
                }
            }
            scnr.close();
        }
        catch (FileNotFoundException fnf) {
            System.out.println("Problem: " + fnf);
            return;
        }
    }
}
```

Reading the Keyboard

- Change the line:
  `Scanner scnr = new Scanner(new File("test.txt"));`
  to
  `Scanner scnr = new Scanner(System.in);`

- That is all.

Java FileReader

a different way to read

- Scanner is great for many tasks; sometimes you need more control
  - scanner breaks things up by "words"
    - What is you need letters?
      - could use scanner to get a string then get letters out of the string
- BUT
  - Scanner is really slow
  - it can get confused on very large files
- So use a different class -- FileReader

FileReader

Constructor Summary

Constructors

Constructor and Description
FileReader(**File** file) Creates a new FileReader, given the File to read from.
FileReader(**FileDescriptor** fd) Creates a new FileReader, given the FileDescriptor to read from.
FileReader(**String** fileName) Creates a new FileReader, given the name of the file to read from.

Method Summary

Methods inherited from class java.io.InputStreamReader

close, getEncoding, read, read, ready

Methods inherited from class java.io.Reader

mark, markSupported, read, read, reset, skip

read

```
public int read()
         throws IOException
```

Reads a single character.

Overrides:

read in class Reader

Returns:

The character read, or -1 if the end of the stream has been reached

Throws:

IOException - If an I/O error occurs

FileReader

```
import java.io.FileReader;

public class FileReader113 {
    public static void main(String[] args) {
        try {
            FileReader fileR = new FileReader("test.txt");
            while (fileR.ready()) {
                int read = fileR.read();
                System.out.println(read);
            }
        } catch (Exception ee) {
            System.out.println("Problem " + ee);
        }
    }
}
```

```
1 2
3 4

49
32
50
10
51
32
52
10
```

ASCII for '1'

ASCII for SPC

ASCII for '\n'

Even More ways to get input

- Other things in the java.io package
  - BufferedReader, etc
- java.nio package
  - can be faster than io but only if you are reading a lot
    - I never user it

Output

```
import java.io.PrintWriter;

public class Write113 {
    public static void main(String[] args) {
        try {
            PrintWriter pw = new PrintWriter("outfile.txt");
            pw.println("Hello");
            pw.close();
        } catch (Exception ee) {
            System.out.println("Problem writing " + ee);
        }
    }
}
```

System.out !!?

close writers just like you close readers. Yes, you can close System.out. Don't.