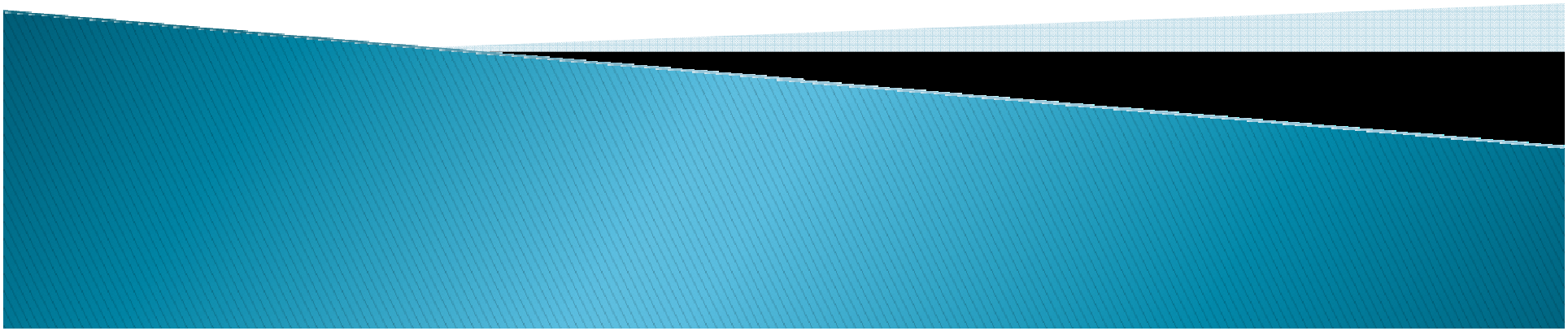
CSSE 220 Day 2

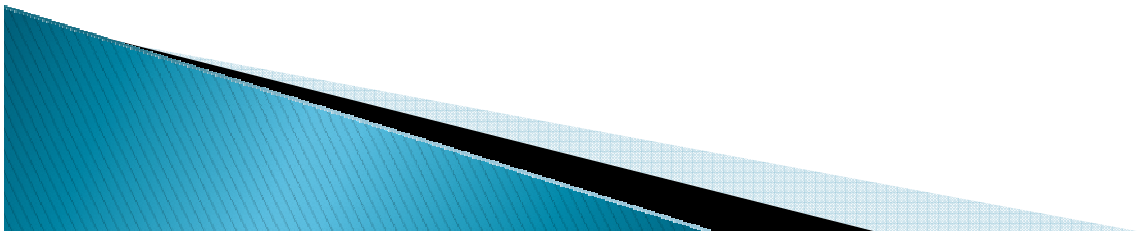
Course Intro
Instructor Intro
Java Intro, Continued



Your questions about …

- The syllabus
- Java
- etc.

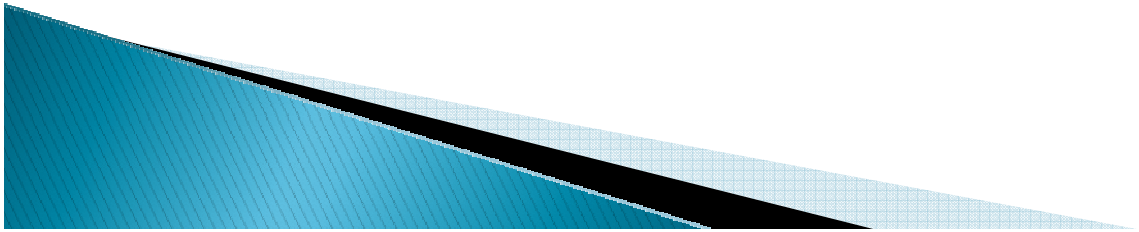
- To submit your homework, do Team > Share
  - Your repository name is csse220-200830-username
  - Use your old SVN password.



Laptops in the Classroom

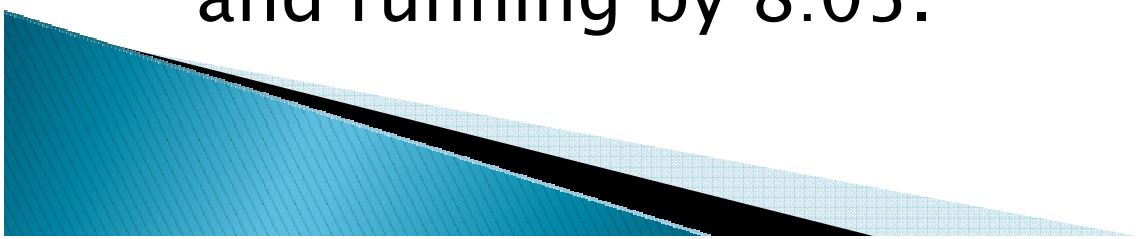
Note to assistants: If you notice someone using a laptop for non-course stuff during class discussions, please give that person a gentle reminder.

- You will generally need to use your laptops during at least a portion of every class period. Please be sure to bring your laptop, a power brick, and a network cable to class.
- You, me, and YouTube
  - Turn off IM and email software and only use other software for things directly related to class
  - If you choose to use non-class-related software or websites during class, you must sit in the next-to-last row.



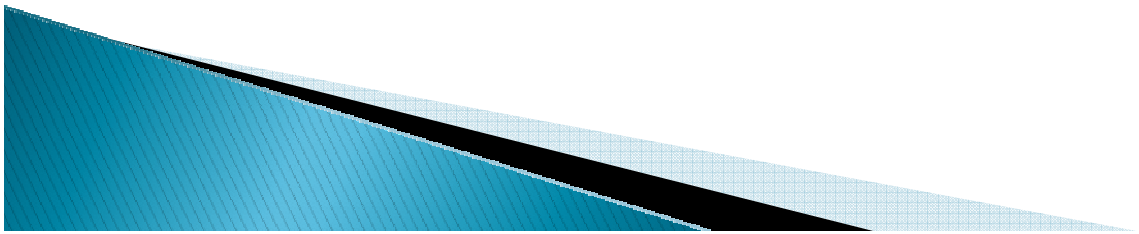
More announcements

- Cell Phones
  - please set ringers to silent or quiet.
    - Minimize class disruptions.
    - But sometimes there are emergencies.
- Personal needs
  - If you need to leave class for a drink of water, a trip to the bathroom, or anything like that, you need not ask me. Just try to minimize disruptions.
- Please be here and have your computer up and running by 8:05.



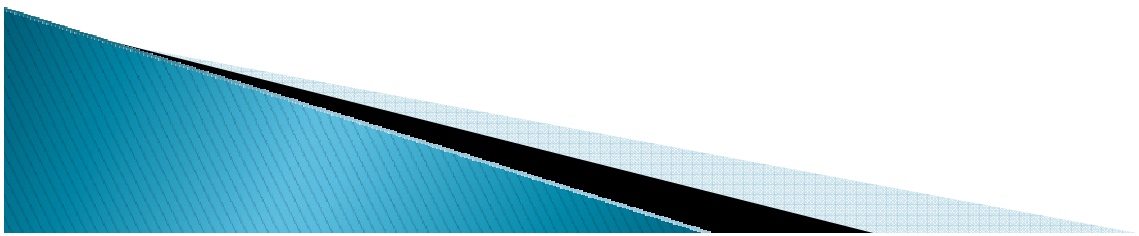
Bonus points for reporting bugs

- In the textbook
- In any of my materials.
- Use the Bug Report Forum on ANGEL
- More details in the Syllabus.



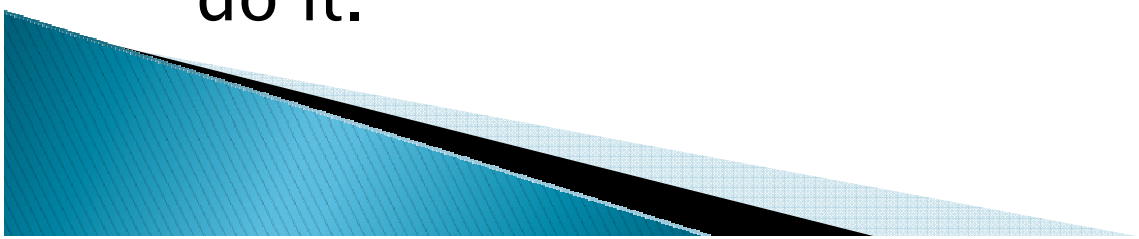
Plagiarism

- Plagiarism has sometimes been a problem in courses at this level. I won't look hard for it, but if I do happen to find it, watch out!
- Of course, copying from the work of previous terms' students is just as bad as copying from this term's students.
- If you use someone else's ideas, attribute them.
- If you use someone else's code, don't submit it!



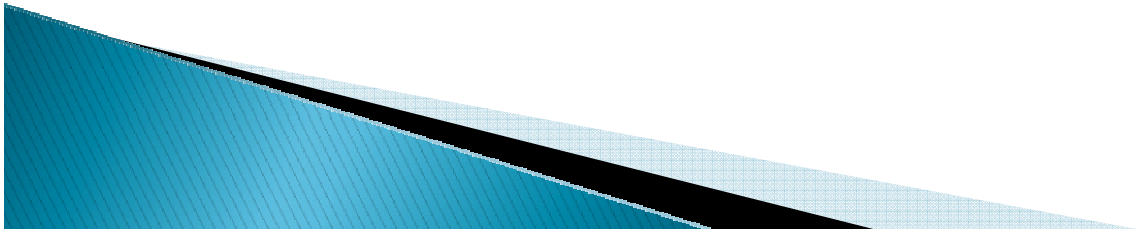
Penalty for cheating

- See the syllabus
- Get help in ways that increase understanding
- Don't get or give help that bypasses learning.
- My usual penalty for plagiarism or cheating.
  - If the assignment or exam is worth N points, your score will be -N (not zero)
  - Why?
- In cases of electronic copying (and perhaps other cases), the penalty may apply to both giver and receiver.
- If you are not sure whether a certain kind of collaboration is appropriate, ask me before you do it.



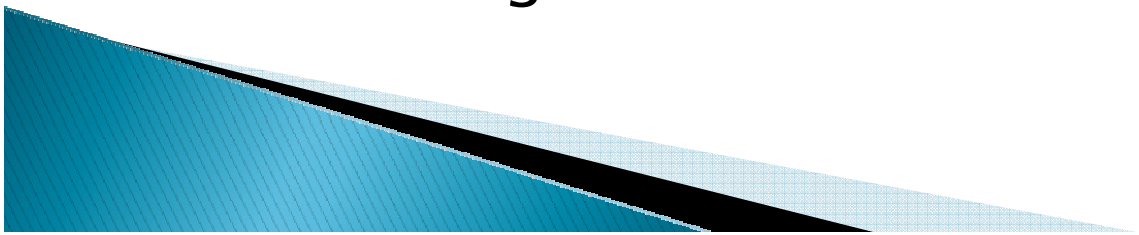
Some major emphases of 220

- Reinforce and extend OO ideas from 120
  - Major emphasis on inheritance
  - GUI programming using Java Swing
- Data Structures
  - Introduce Algorithm efficiency analysis
  - Abstract Data Types
  - Specifying and using standard Data Structures
  - Implementing simple data structures (lists)
- Recursion
- Simple Sorting and searching
- A few additional Software Engineering concepts



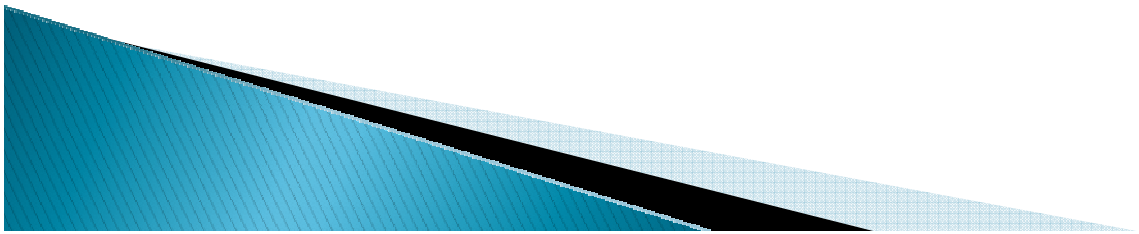
What will I spend my time doing?

- Larger programming problems, mostly outside of class.
  - Exploring the JDK documentation to find the classes and methods that you need.
  - Debugging!
  - Reviewing other students' code.
- Reading (a lot to read at the beginning; less later).
  - Thinking about exercises in the textbooks.
  - Some written exercises, mostly from the textbook.
  - Small programming assignments in class (some to be continued for homework).
- Discussing the material with other students.



Even in lecture…

- This course is about participating, doing.
  - When we are having a class discussion, you may not always be the one to answer aloud, but try to THINK the answer before someone else verbalizes!
- Consider Mary and Bob:
  - Mary is active and engaged
  - Bob just sits there “absorbing”
  - Outcomes?



Ask user for value (new way)

Import the **Scanner** class from the **java.util** package.

System.in is Java's standard input stream. Note that this means the variable called **in** in the **System** class.

Other **Scanner** methods include nextDouble(), nextLine(), nextBoolean, hasNextInt(). hasNextline().

```
import java.math.BigInteger;
import java.util.Scanner;

public class Factorial_6_Scanner {
   public static final int MAX = 25;

   public static BigInteger factorial(int n) {
     BigInteger prod = BigInteger.ONE;
     for (int i=1; i<=n; i++)
       prod = prod.multiply(new BigInteger(i +""));
     return prod;
   }

  public static void main(String[] args) {
     Scanner scanner = new Scanner(System.in);
     System.out.print("Enter a nonnegative integer: ");
     int n = scanner.nextInt();
     System.out.println(n + "! = " + factorial(n) );
  }
}
```

If we do not do the import, we can write

```
java.util.Scanner sc = new java.util.Scanner(System.in);
```

So **import** is a simple convenient shortcut

Ask user for value (old way)

Using the new Scanner class is easier than this approach. But you will often see the old approach in other people's code (including Mark Weiss' code).

Think of this as the "magic incantation" for getting set up to read from standard input.

readline( ) returns the next line of input as a String.

Since **readline( )** could generate an IO Exception, the try/catch is required.

```
import java.math.BigInteger;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.BufferedReader;

  // omitted definition of the factorial method

  public static void main(String[] args) {
     BufferedReader in =
       new BufferedReader(
        new InputStreamReader(System.in));
     String line = "";
     System.out.print("Enter a positive integer: ");
     try {
        line = in.readLine();
     } catch (IOException e) {
        System.out.println("Could not read input");
     }
     int n = Integer.parseInt(line);
     System.out.println(n + "! = " + factorial(n) );
  }
}
```

parseInt( ) takes a string that represents an integer and returns the corresponding int value. It is somewhat similar to Python's **int( )** function.

What if a user types something wrong?

If any exception gets thrown by the code in the **try** clause, the **catch** clauses are tested in order to find the first one that matches the actual exception type.

If none match, the exception is thrown back to whatever method called this one.

If it is never caught, the program crashes.

```
import java.math.BigInteger;

public class Factorial_9_InputErrors {

   public static BigInteger factorial(int n) {
      if (n < 0)
         throw new IllegalArgumentException();
      BigInteger prod = BigInteger.ONE;
      for (int i = 1; i <= n; i++)
         prod = prod.multiply(new BigInteger(i + ""));
      return prod;
   }

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter a nonnegative integer: ");
      try {
        int n = scanner.nextInt();
        System.out.println(n + "! = " + factorial(n));
      } catch (InputMismatchException e) {
         System.out.println("Must be an integer");
      } catch (IllegalArgumentException e) {
         System.out.println("Cannot be negative");
      }
   }
}
```

Java's five Exception keywords

- Discuss with the person next to you (for two or three minutes):
  - First, tell that person something she/he probably does not know about you.
- What does each of the following mean?
- What can you say about how it is used?

try	Run some code that might throw an exception
catch	if this type of exception is thrown, run the following code … (i.e. handle the exception).
finally	Run this code at the end, whether there is an exception or not.
throw	I discovered something I don't know how to handle. Does anyone who called me know what to do?
throws	This method might throw this kind of checked exception. If it does, I'm not handling it! Documentation for the compiler and users!

Factorial recursive

Recursive factorial definition:

$n! = 1 \quad \text{if } n = 0$

$n! = (n-1)!\, n \quad \text{if } n > 0$

```
import java.math.BigInteger;

public class Factorial_10_Recursive {
   public static final int MAX = 30;

   /* Return the factorial of n */
   public static BigInteger factorial(int n) {
      if (n < 0)
         throw new IllegalArgumentException();
      if (n == 0)
         return BigInteger.ONE;
      return new BigInteger(n+ "").multiply(factorial(n-1));
    }

   public static void main(String[] args) {
      for (int i=0; i <= MAX; i++)
         System.out.println(i + "! = " + factorial(i) );
   }
}
```

Recursive basically means:
The method calls itself.

Speed up Factorial Calculation with Caching

Store previously-computed values in an array called vals

```
import java.math.BigInteger;

public class Factorial_11_Caching {
   public static final int MAX = 2000;
   static int count = 0;   // How many values have we cached so far?
   static BigInteger[] vals = new BigInteger[MAX+1];  // the cache
   static { vals[0] = BigInteger.ONE; }       // Static initializer

   /* Return the factorial of n */
   public static BigInteger factorial(int n) {
      if (n < 0 || n > MAX)
         throw new IllegalArgumentException();
      if (n <= count)  // If we have already computed it …
         return vals[n];
      BigInteger val =
         new BigInteger(n+ "").multiply(factorial(n-1));
      vals[n] = val;  // Cache the computed value before returning it
      count = n;
      return val;
   }
                  // Code for main()omitted. Same as in previous example.
}
```

Interlude – From worldmag.com

WORLD MAGAZINE | Archives *1996 to the Present*

Quick Takes
ODDBALL OCCURRENCES

Bovine force

If Linda and Charles Everson Jr. had been driving just a bit faster, they may not have celebrated another anniversary. While celebrating their first anniversary, the Michigan couple was driving on Highway 150 alongside a cliff near Manson, Wash., when something fell from above and crushed the hood of their minivan. Instead of falling rocks, it was a falling cow. The 600-pound cow, which had fallen from 200 feet up, crushed the front of their minivan, but the couple escaped unscathed. The 1-year-old bovine wasn't so fortunate.

wenatcheeworld.com
THE FIERCELY INDEPENDENT VOICE OF NORTH CENTRAL WASHINGTON

The cow heard around the world

Media milking falling-heifer story to death

The couple were driving back to their Manson hotel on Highway 150 Sunday after attending a church service in Chelan when the 600-pound heifer named Michelle dropped from above and landed on the hood of their Buick Terraza and bounced off onto the road. Everson said he was stunned and kept on driving, repeating to himself "I don't believe it. I don't believe it."

File Input/Output

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

public class FileIOTest {
/* Copy an input file to an output file, changing all letters to uppercase.
   This approach can be used for input processing in almost any program. */
   public static void main(String[] args) {
      String inputFileName =  "sampleFile.txt";
      String outputFileName = "upperCasedFile.txt";
      try {
         Scanner sc = new Scanner(new File(inputFileName));
         PrintWriter out = new PrintWriter(new File(outputFileName));
         while (sc.hasNextLine()){  // process one line
            String line = sc.nextLine();
            line = line.toUpperCase();
            for (int i= 0; i< line.length(); i++)
            // normally we might do something with each character in the line.
               out.print(line.charAt(i));
            out.println();
         }
         out.close();
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

More File Input/Output

```
import java.util.Scanner;
import java.io.*;

public class TryFileInputOutput {

  public static void main(String[] args) {
    String inFileName=null ,outFileName = "outFile.txt";
    Scanner fileScanner;
    PrintWriter out;
```

Essentially the same as before

Keep looping until user enters the name of an input file that we can actually open.

```
    try {
      Scanner sc = new Scanner(System.in);
      while (true) // until we get a valid file.
        try {
          System.out.print("Enter input file name: ");
          inFileName = sc.nextLine();
          fileScanner = new Scanner(new File(inFileName));
          break; // we have a valid file, so exit the loop.
        } catch(FileNotFoundException e) {
          System.out.println("Did not find file " + inFileName + ". Try again!");
      }
```

```
    out = new PrintWriter(new File (outFileName));
    while (fileScanner.hasNextLine()){  // process one line
      String line = fileScanner.nextLine();
      line = line.toUpperCase();
      for (int i=0; i<line.length(); i++)
        out.print(line.charAt(i));  // process each char on the line
      out.println();
    }
    out.close();
    fileScanner.close();
    System.out.println("Done!");
  } catch (IOException e) {
    e.printStackTrace();
  }
```

Essentially the same as before

Primitive types

Primitive Type	What It Stores	Range
`byte`	8-bit integer	–128 to 127
`short`	16-bit integer	–32,768 to 32,767
`int`	32-bit integer	–2,147,483,648 to 2,147,483,647
`long`	64-bit integer	-2^{63} to $2^{63} - 1$
`float`	32-bit floating-point	6 significant digits (10^{-46}, 10^{38})
`double`	64-bit floating-point	15 significant digits (10^{-324}, 10^{308})
`char`	Unicode character	
`boolean`	Boolean variable	`false` and `true`

figure 1.2

The eight primitive types in Java

Java *switch* statement

figure 1.5

Layout of a switch statement

```
1  switch( someCharacter )
2  {
3    case '(':
4    case '[':
5    case '{':
6      // Code to process opening symbols
7      break;
8
9    case ')':
10   case ']':
11   case '}':
12     // Code to process closing symbols
13     break;
14
15   case '\n':
16     // Code to handle newline character
17     break;
18
19   default:
20     // Code to handle other cases
21     break;
22 }
```

A program that uses switch

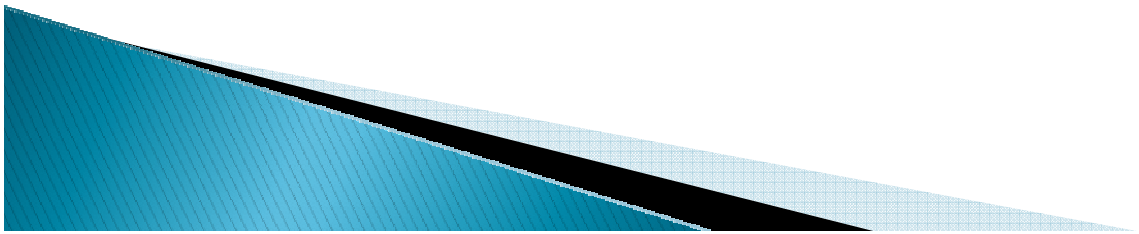
```
// Adapted from Java Examples in a NutShell 3rd Ed, by David Flanagan.
// The children's game FizzBuzz.

public class FizzBuzz {
  public static void main(String[] args) {
    for(int i = 1; i <= 100; i++) {          // count from 1 to 100
      switch(i % 35) {                       // What's the remainder mod 35?
        case 0:                              // For multiples of 35... 1 2 3 4
          System.out.print("fizzbuzz ");     // print "fizzbuzz".
          break;                             // Don't forget this statement!
        case 5: case 10: case 15:            // If the remainder is any of these
        case 20: case 25: case 30:           // then the number is a multiple of 5
          System.out.print("fizz ");         // so print "fizz".
          break;
        case 7: case 14: case 21: case 28: // For any multiple of 7...
          System.out.print("buzz ");         // print "buzz".
          break;
        default:                             // For any other number...
          System.out.print(i + " ");         // print the number.
          break;
      }
      if (i%10 == 0) System.out.println();
    }
  }
}
```

```
1 2 3 4 fizz 6 buzz 8 9 fizz
11 12 13 buzz fizz 16 17 18 19 fizz
buzz 22 23 24 fizz 26 27 buzz 29 fizz
31 32 33 34 fizzbuzz 36 37 38 39 fizz
41 buzz 43 44 fizz 46 47 48 buzz fizz
51 52 53 54 fizz buzz 57 58 59 fizz
61 62 buzz 64 fizz 66 67 68 69 fizzbuzz
71 72 73 74 fizz 76 buzz 78 79 fizz
81 82 83 buzz fizz 86 87 88 89 fizz
buzz 92 93 94 fizz 96 97 buzz 99 fizz
```

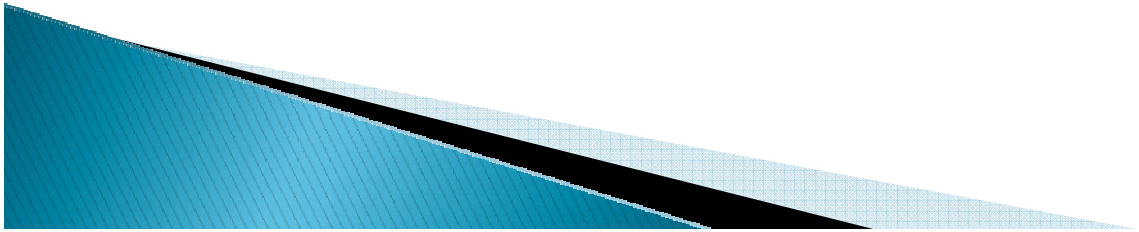
Unit Testing

- What do you think it is?
  - Testing parts of your code in isolation before putting them all together
- Why is it a good thing?
- How do I write good test cases?
  - Test all types of inputs, including boundary cases.
  - Practice!
- How easy is it to do it in Eclipse?
  - Fairly so, with JUnit
- I will **often** give you unit tests to help you write correct code.
  - Here are some to test last night's homework…
  - Check out HW1Test from:
    http://svn.cs.rose-hulman.edu/repos/csse220-200830-username



In all your code:

- Write appropriate comments:
  - Javadoc comments for public fields and methods.
  - Explanations of anything else that is not obvious.
- Give explanatory variable and method names:
  - Use name completion in Eclipse, Alt-/ to keep typing cost low and readability high
- Use local variables and static methods (instead of fields and non-static methods) where appropriate.
  - “where appropriate” includes any place where you can’t explicitly justify doing otherwise.
- Use Ctrl-Shift-F in Eclipse to format your code.



Start HW 2

- Go there now