Welcome to CSSE 220

- We are excited that you are here:
  - Start your computer and get ready for our first class session
  - If you haven’t followed the instructions in the email I sent last night, please do that now
  - Pick up a quiz from the back table and answer the first two questions

Course Introduction, Starting with Java

CSSE 220—Object-Oriented Software Development

Rose-Hulman Institute of Technology

Agenda

- Instructor intro
- A few administrative details
- Verify Eclipse and Subclipse configuration
- Java *vs*. Python
- Examine and modify simple Java programs

Daily Quizzes

- I expect you to answer every question.
- Stop me if I don't cover a question!

A Tour of the On-line Course Materials

- Moodle
- Syllabus
- Schedule

Evening lab assistants, F-217

- 7-9 PM Sunday-Thursday

Programming is not a spectator sport

- And neither is this course
- Ask, evaluate, respond, comment!
- Interrupt me! Even with statements like, *"I have no idea what you were just talking about."*
- I do not intend for classroom discussions to go over your head. Don't let them!

Ok, let’s write our first Java program!

- Hello world

Checkout today's project (HW1)

- New Eclipse workspace, Java perspective (there is probably already a csse220 workspace on your computer)
- Go to SVN Repository view, at bottom of the workbench
  - If it is not there, **Window → Show View → Other → SVN → SVN Repositories**
- Right-click in SVN view, then choose **New Repository Location**
  - **http://svn.csse.rose-hulman.edu/repos/csse220-201510-*your_username***
- Right-click HW1 project and choose Checkout
  - **Accept default options**

Get help immediately if you're stuck!

HelloPrinter.java

- To run a Java program:
  - Right-click the .java file in Package Explorer view
  - Choose **Run As → Java Application**
- Change the program to say hello to a person next to you
- Introduce an error in the program
  - See if you can come up with a different error than the person next to you
- Fix the error that the person next to you introduced

A First Java Program

In Java, all variable and function definitions are inside *class* definitions

main is where we start

```
public class HelloPrinter {

    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }

}
```

System.out is Java's standard output stream. This is the variable called **out** in the **System** class.

System.out is an *object* from the **PrintStream** class. **PrintStream** has a *method* called **println( ).**

Q8

Introduction to Java

Things Java Has in Common with Python

- Classes and objects
- Lists (but no special language syntax for them like Python)
- Standard ways of doing graphics and GUIs
- A huge library of classes/functions that make many tasks easier
- Nice integration with the Eclipse IDE

Why Java?

- Widely used in industry for large projects
  - From cell phones
    - including smart phones—Android platform
  - To global medical records
- Highlights essential topic of the class – Object Orientation
- Similar to other popular languages C#, Objective-C
- Less complex than C++
- Most popular language according to the TIOBE Programming Community Index [November 2013]

http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Guess what language is #2

Q9

Interlude: JavaScript and Java

Java is to Javascript as Ham is to Hamster

From Wikipedia (edited, bullets added to enhance PowerPoint readability):

- The change of name to JavaScript roughly coincided with Netscape adding support for Java technology in its web browser.
- The name caused confusion, giving the impression that JavaScript was a spin-off of Java.
- The choice has been characterized by many as a marketing ploy by Netscape to give JavaScript the cachet of what was then the hot new web-programming language.
- It has also been claimed that the language's name is the result of a co-marketing deal between Netscape and Sun, in exchange for Netscape bundling Sun's Java runtime with its then-dominant browser.

Basic Java Functions and Conditionals

- Let’s go through the ConditionalExamples.java file

Javadoc comments

```java
/**
 * Has a static method for computing n!
 * (n factorial) and a main method that
 * computes n! for n up to Factorial.MAX.
 *
 * @author Mike Hewner & Delvin Defoe
 */
public class Factorial {
    /**
     * Biggest factorial to compute.
     */
    public static final int MAX = 17;

    /**
     * Computes n! for the given n.
     *
     * @param n
     * @return n! for the given n.
     */
    public static int factorial (int n) {
        ...
    }

    ...

}
```

We left out something important on the previous slide – comments!

Java provides Javadoc comments (they begin with /**) for both:

- Internal documentation for when someone reads the code itself
- External documentation for when someone re-uses the code

Comment your own code now, as indicated by this example. Don't forget the @author tag in HelloPrinter.

Writing Javadocs

- Written in special comments: /** … */
- Can come before:
  - Class declarations
  - Field declarations
  - Constructor declarations
  - Method declarations
- Eclipse is your friend!
  - It will generate Javadoc comments automatically
  - It will notice when you start typing a Javadoc comment

In all your code:

- Write appropriate comments:
  - Javadoc comments for public fields and methods.
  - Explanations of anything else that is not obvious.
- Give self-documenting variable and method names:
  - Use name completion in Eclipse, Ctrl-Space, to keep typing cost low and readability high
- Use Ctrl-Shift-F in Eclipse to format your code.
- Take care of all auto-generated TODO’s.
  - **Then delete the TODO comment**.
- Correct ALL compiler warnings. Quick Fix is your friend!

Q10 - 11

HW1 & READING DUE
BEFORE NEXT SESSION

IT'S ON THE SCHEDULE PAGE.

AS ALWAYS, POST ON PIAZZA (OR EMAIL ME) IF YOU HAVE ANY QUESTIONS