- Administrative Stuff
  - Class
    - Lecture Times: MWF @ 1500-1700 in 32-141
    - Office Hours: TR   @ 1500-1700 in 66-144
  - Grading and Assignments
    - P/D/F format
    - grade = score > 60 ? P : D/F;
    - Number of anticipated assignments: 3-4
    - Exam: NONE
  - Piazza
    - Use for questions about the class and assignments
    - Use for clarification of assignments and minor notes
  - Google Feedback Form
    - Where you can tell us anonymously how we are doing
    - http://goo.gl/forms/QqEeNci6ba

- Eclipse **(SOURCE: MIT 6.005 FALL2014)**
  - Before installation
    - **JDK 8** (**Java Development Kit** for Windows, Linux, or Mac): Download the latest version. (You don't need NetBeans or Java EE. Also make sure you have the latest version of **Java Runtime Environment** JRE)

    - **Eclipse 4.4 (Eclipse Luna)**: Choose "Eclipse IDE for Java Developers," which will download a ZIP file. Unpack the ZIP file, go inside the resulting folder, and run Eclipse. Make sure Eclipse is configured to use Java 8 by doing the following:
      1. Click Window → Preferences → Java → Installed JREs and ensure that "Java SE 8" is the only one checked (on a Mac, the menu is Eclipse → Preferences → Java → Installed JREs).
      2. Also in Preferences, click Java → Compiler and set "Compiler compliance level" to 1.8. Click OK and Yes on any prompts.
    - Consult http://web.mit.edu/6.005/www/fa14/psets/ps0/ if you have questions.
    - Ignore everything about git since we won't use it in this class, but do read the Unit Testing section.

  - Shortcuts
    - Ctrl + Shift + S -> Save ALL (Do this often)
    - Ctrl + F11 -> Save and Launch(Run)
    - http://www.shortcutworld.com/en/win/Eclipse.html

- Java - Primitive Data **Types (PDTs)**
    - boolean
        - range: {true, false}
        - default: false
    - byte
        - 8 bit signed 2's complement integer
        - range: [-2^7 , 2^7)
        - default: 0
    - short
        - 16 bit signed 2's complement integer
        - range: [-2^15 , 2^15)
        - default: 0
    - char
        - 16 bit unicode character
        - range: { '\u0000','\u0001', ... ,'\uffff'}
        - default: '\u0000'
    - int
        - 32 bit signed 2's complement integer
        - range: [-2^31 , 2^31)
        - default: 0
    - long
        - 64 bit signed 2's complement integer
        - range: [-2^63 , 2^63)
        - default   0L or 0l
    - float
        - 32 bit floating point number
        - default: 0.0f or 0.0F
        - range: IT'S COMPLICATED
    - double
        - 64 bit floating point number
        - default: 0.0d or 0.0D
        - range: IT'S COMPLICATED
    - http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

- Java - Primitive Data **Structures**
  - Treated like a primitive data types but they are also like objects

  - Strings and arrays
    - like a PDTs they can be used without the keyword “new”
    - like an object they have methods that do operations

- Java - Naming Rules and Conventions
  - Rules (**MUST** be followed):
    - CANNOT start with a number
    - CANNOT use JAVA’S RESERVED WORDS
    - CANNOT start with special characters except ‘$’ (dollar sign) or ‘_’ (underscore)
    - CANNOT have whitespace
    - Case sensitive
    - Otherwise the name of a variable can be an unlimited sequence of UNICODE characters

  - Conventions (**SHOULD** be followed):
    - ALL variables in the program should be meaningful
    - ALL constants should be CAPS
    - ALL other variables should start with a lowercase and use camel case
    - itLooksLikeThisWhereTheFirstLetterOfAWordIsCapitalized

- Java - Keywords
  - Reserved words
  - http://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

- Java - Commenting
    - Line comment starts with double slashes
        - Anything that follows a double slash is a comment until the next line
    - Block comment starts with /* and ends with */
        - Anything inside the /* and */ are comments
    - JavaDocs are documentation of methods and instance variables
        - These are above the thing that they describe
        - They are like block comments but they begin with an extra asterisks
        - Anything inside the /** and */ are part of the JavaDocs

- Java - Casting
    - Turning a variable into another type
    - Sort of like making sure the units match
        - Automatic Casting - Done by Java
            - Usually upcasting
            - Done if there is no loss of “precision”

        - User forced casting
            - Usually downcasting but users CAN upcast as well
            - Downcasting CANNOT always be done

- Java - Scope
    - Braces
    - When something is created and when it is destroyed
        - Garbage Collector
    - Unlike C or C++ you do not allocate memory.
    - Java Manages all your memory for you

- Java - HelloWorld
    - Printing to console
        - use “System.out.print(“Hello World”);”
        - take for granted that System.out.print() prints stuff for now.
        - We’ll explore what “System”, “out” and “print” are later

    - Accept user input
        - Scanner class
        - There are other classes that allow user input such as bufferedReader

- Java - Control Statements
    - if
        - runs the if block when the condition is true

    - if / else
        - runs the if block when the condition is true
        - otherwise else block **ALWAYS** runs. Also called the trailing else.

    - if / else if
        - runs the the first if whose conditional returns true then breaks out of the if / else if structure.

    - if / else if / else
        - runs the first if whose conditional returns true then breaks out of the if / else if / else structure.
        - otherwise else block **ALWAYS** runs. Also a trailing else.

- Java - Loops
  - while
    - runs if the conditional is true
  - do while
    - runs once and keeps running if the conditional is true
  - for
    - for
    - for each
  - KEYWORDS: break and continue
    - break - kicks the program out of a control structure
    - continue - stops the current iteration and jumps to the next iteration.

- Java - Contracts: Arguments and Returns
  - Arguments and parameters
  - The return statement
  - Passing by reference
  - Satisfying conditions of input and output

- Java - Methods and modularity
  - Breaking up a larger problem into much smaller ones
  - Functions that does “stuff” in Java are called methods

- Java - Objects
  - What real life looks like
  - An abstract concept that can be made concrete
  - The “nouns” of Java

- Java - Class
  - The blueprints for objects
  - Contains all the particulars of what makes an object that object
  - Can be part of a hierarchy
  - Static Methods
  - Instance Methods

- Java - Inheritance (Think of a family)
    - The “nouns” are similar
    - Share common “physical characteristics”
    - Superclass
    - Subclass

- Java - Interfaces (Think of a work place)
    - The “verbs” are similar
    - Share kinds of actions they can do

- Java - Enumerated values

- Java - GUI

- Java