“When a programming language is created that allows programmers to program in simple English, it will be discovered that programmers cannot speak English.”

- Anonymous

Based on slides by Marty Stepp and Stuart Reges from http://www.buildingjavaprograms.com/
What We Will Do Today

- What are computer languages?
- Java editors
  - text editor and command line
  - BlueJ
- First programming concepts
  - output with println statements
  - syntax and errors
- structured algorithms with static methods
- identifiers, keywords, and comments
Computers and Computer Languages

- Computers are everywhere
  - how many computers do you own?
- Computers are useful because they run various programs
  - program is simply a set of instructions to complete some task
  - how many different programs do you use in a day?
Definitions

- **program**: A set of instructions that are to be carried out by a computer.

- **program execution**: The act of carrying out the instructions contained in a program.
  - this is done by feeding the instructions to the CPU

- **programming language**: A systematic set of rules used to describe computations, generally in a format that is readable and editable by humans.
  - in this class will are using Java
High Level Languages

- Computers are fast
  - Intel® Core™ i7-3770 Processor released April 2012
  - made up of about 1.4 Billion transistors (a switch that is on or off)
  - performs tens of billions of operations per second

- Computers are dumb
  - They can only carry out a very limited set of instructions
    - on the order of 100 or so depending on the computer's processor
    - machine language instructions, aka instruction set architecture (ISA)
    - Add, Branch, Jump, Get Data, Get Instruction, Store
Machine Code

- John von Neumann - co-author of paper in 1946 with Arthur W. Burks and Hermann H. Goldstine,
  - "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument"

- One of the key points
  - program commands and data stored as sequences of bits in the computer's memory

- A program:
  
  111000111000000000
  0101011011100000
  0110100001000000
  0000100000001000
  0001011011000100
  0001001001100001
  0110100001000000
  0000111000000011
Say What?

- Programming with Strings of bits (1s or 0s) is not the easiest thing in the world.

- Assembly language
  - mnemonics for machine language instructions

```
.ORIG x3001
LD R1, x3100
AND R3, R3 #0
LD R4, R1
BRn x3008
ADD R3, R3, R4
ADD R1, R1, #1
LD R4, R1
BRnzp x3003
```
High Level Languages

- Assembly language, still not so easy, and lots of commands to accomplish things

- High Level Computer Languages provide the ability to accomplish a lot with fewer commands than machine or assembly language in a way that is hopefully easier to understand

```c
int sum = 0;
int count = 0;
while( list[count] != -1 ) {
    sum += list[count];
    count = count + 1;
}
```
Java

There are hundreds of high level computer languages. Java, C++, C, Basic, Fortran, Cobol, Lisp, Perl, Prolog, Eiffel, Python.

The capabilities of the languages vary widely, but they all need a way to do:
- declarative statements
- conditional statements
- iterative or repetitive statements

A compiler is a program that converts commands in high level languages to machine language instructions.
public class Hello
{
    public static void main(String[] args)
    {
        System.out.println("Hello World!");
    }
}

This would be in a text file named Hello.java
DEMO of writing and running a program via notepad and
the command line
Running a program

1. Write it.
   - **code** or **source code**: The set of instructions in a program.

2. Compile it.
   - **compile**: Translate a program from one language to another.
   - **byte code**: The Java compiler converts your code into a format named *byte code* that runs on many computer types.

3. Run (execute) it.
   - **output**: The messages printed to the user by a program.
Bigger Java program!

```java
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
        System.out.println();
        System.out.println("This program produces");
        System.out.println("four lines of output");
    }
}
```

- Its output:
  Hello, world!
  This program produces
  four lines of output

- **console**: Text box into which the program's output is printed.
Every executable Java program consists of a class, that contains a method named main, that contains the statements (commands) to be executed.
System.out.println

- A statement that prints a line of output on the console.
  - pronounced "print-linn"

- Two ways to use System.out.println:
  - System.out.println("<text>");
    Prints the given message as output.
  - System.out.println();
    Prints a blank line of output.
Syntax

- **syntax**: The set of legal structures and commands that can be used in a particular language.
  - Every basic Java statement ends with a semicolon ;
  - The contents of a class or method occur between { and }

- **syntax error (compiler error)**: A problem in the structure of a program that causes the compiler to fail.
  - Missing semicolon
  - Too many or too few { } braces, braces not matching
  - Class and file names do not match
  - ...
Syntax error example

```java
1  public class Hello {
2       p__ublic static void main(String[] args) {
3           System.owt.println("Hello, world!")_
4       }
5  }
```

▸ Compiler output:

```
Hello.java:2: <identifier> expected
       poublic static void main(String[] args) {
              ^
Hello.java:3: ;' expected
         ^
2 errors
```

– The compiler shows the line number where it found the error.

– The error messages sometimes can be tough to understand:
  • Why can’t the computer just say “You misspelled ‘public’”?
An Important Realization

- Computers are stupid.

- Computers can’t read minds.

- Computers don’t make mistakes.

- If the computer is not doing what we want, it’s because WE made a mistake.
More on syntax errors

- Java is case-sensitive
  - Hello and hello are not the same

```
1 Public class Hello {
2   public static void main(String[] args) {
3     System.out.println("Hello, world!");
4   }
5 }
```

Hello.java:1: class, interface, or enum expected
Public class Hello {
    ^
1 error
Names

- You must give your program a name.

```java
public class SubstitutionCipherDecoder {

    // Naming convention: capitalize each word (e.g. MyClassName)

    // Your program's file must match exactly (SubstitutionCipherDecoder.java)
    //  - includes capitalization (remember, Java is "case-sensitive")
```
Identifiers

- **identifier**: A name given to an item in your program.
  - must start with a letter, underscore, or $\$
  - subsequent characters can be any of those or a number

- **legal**:
  - _myName
  - TheCure
  - ANSWER_IS_42
  - $bling$

- **illegal**:
  - me+u
  - 49ers
  - side-swipe
  - Ph.D's
Keywords

- **keyword**: An identifier that you cannot use because it already has a reserved (special) meaning in Java.

<table>
<thead>
<tr>
<th>abstract</th>
<th>default</th>
<th>if</th>
<th>implements</th>
<th>private</th>
<th>this</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>do</td>
<td>import</td>
<td>instanceof</td>
<td>protected</td>
<td>throw</td>
</tr>
<tr>
<td>break</td>
<td>double</td>
<td>int</td>
<td>interface</td>
<td>return</td>
<td>throws</td>
</tr>
<tr>
<td>byte</td>
<td>else</td>
<td>long</td>
<td>static</td>
<td>short</td>
<td>transient</td>
</tr>
<tr>
<td>case</td>
<td>extends</td>
<td>native</td>
<td>strictfp</td>
<td>void</td>
<td>try</td>
</tr>
<tr>
<td>catch</td>
<td>final</td>
<td>new</td>
<td>super</td>
<td>volatile</td>
<td>while</td>
</tr>
<tr>
<td>char</td>
<td>finally</td>
<td>package</td>
<td>switch</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td>float</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>const</td>
<td>for</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>continue</td>
<td>goto</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Because Java is case-sensitive, you could technically use `Class` or `cLaSs` as identifiers, but this is very confusing and thus **strongly discouraged**.
clicker question

Which of the following is not a syntactically correct Java identifier for the name of a class?

A. static
B. Void
C. FirstProgram
D. _My_program
E. More than one of A - D is not a syntactically correct Java identifier.
Strings

- **string**: A sequence of text characters.
  - Starts and ends with a " (quotation mark character).
    - The quotes do not appear in the output.
  - Examples:
    - "hello"
    - "This is a string. It's very long!"

- Restrictions:
  - May not span multiple lines.
    - "This is not a legal String."
  - May not contain a " character.
    - "This is not a "legal" String either."

- This begs the question…
Escape sequences

- **escape sequence**: A special sequence of characters used to represent certain special characters in a string.

  - `\t` tab character
  - `\n` new line character
  - `\"` quotation mark character
  - `\\` backslash character

- Example:
  ```java
  System.out.println("\\hello\\nhow\\tare "\"you\"?\\\"");
  ```

- Output:
  ```java
  \hello
  how are "you"?
  ```
clicker Question

How many visible characters does the following println statement produce when run?

```java
System.out.println("\t\nn\\\\t\"\tt");
```

A. 0
B. 1
C. 2
D. 3
E. 4
What sequence of println statements will generate the following output?

This program prints the first lines of the song "slots".

"She lives in a trailer"
"On the outskirts 'a Reno"
"She plays quarter slots in the locals casino."
What sequence of println statements will generate the following output?

A "quoted" String is 'much' better if you learn the rules of "escape sequences."

Also, "" represents an empty String. Don't forget to use " instead of " ! "' is not the same as " 
What is the output of the following `println` statements?

```java
System.out.println("\ta\tb\tc");
System.out.println("\\\\");
System.out.println("'");
System.out.println("\\\\\\\\");
System.out.println("C:\nin\the downward spiral");
```
Answer to Practice Program 3

Output of each println statement:

```
  a b c
  \\`
  `""``
C: in he downward spiral
```
Write a `println` statement to produce this output:

```
/ \ // \ \ /// /// \ \``
Answer to Practice Program 4

println statement to produce the line of output:

System.out.println("/ \ \ // \ \ \ \ \ // \ \ \ \ \ \ \ \ \ \ \ \ \ \") ;