BASICS

http://www.flickr.com/photos/oskay/472097903/
Outline

- Computer Basics
- Programs and Languages
- Introduction to the Eclipse IDE
- Our First Program
  - Comments
- Algorithms
Hardware and Memory

• Most modern computers have similar components including
  • Input devices (keyboard, mouse, etc.)
  • Output devices (display screen, printer, etc.)
  • A processor
  • Two kinds of memory (main memory and auxiliary memory).
Main memory

- Working memory used to store
  - The current program
  - The data the program is using
  - The results of intermediate calculations
- Usually measured in megabytes or gigabytes (e.g. 8 gigabytes of RAM)
  - RAM is short for *random access memory*
  - A *byte* is a quantity of memory
Bits, Bytes, and Addresses

- A *bit* is a digit with a value of either 0 or 1.
- A *byte* consists of 8 bits.
- Each byte in main memory resides at a numbered location called its *address*. 
Main Memory

• Data of all kinds (numbers, letters, strings of characters, audio, video, even programs) are encoded and stored using 1s and 0s.

• When more than a single byte is needed, several adjacent bytes are used.
  • The address of the first byte is the address of the unit of bytes.

• When the computer is turned off, main memory is erased (volatile memory).
Auxiliary Memory

• Auxiliary memory uses devices such as a hard drive, DVD, USB drive, etc.
• Data (files) need to be “saved” to the auxiliary memory
• Data is still stored in bits and bytes
• When the computer is turned off, this data does not go away (persistent storage)
Programs

• A program is a set of instructions for a computer to follow.
• We use programs almost daily (email, word processors, video games, bank ATMs, etc.).
• When the computer follows the instructions it is running or executing the program.
View of Programming from 10,000 Feet

Your program

Alice ← Input string

Black box

Hi, Alice. How are are?

Output string
Languages

- **Machine language**
  - Low level, what the hardware understands
  - Tedious and error-prone to write
  - Specific to a particular type of computer

- **Natural language**
  - Imprecise and ambiguous
  - Hard to convert to instructions for the hardware

- **High level programming language**
  - Good balance between the two extremes
Becoming a Programmer: Step 1

Choose a language...

and hundreds more...
Our Choice: Java

• **Advantages**
  • Widely used, modern
  • Freely available, cross-platform
  • Features help novices learn to program

• **No perfect single language**
  • You'll learn many other languages
    • C/C++, assembly, Python, C#, JavaScript, PHP...
  • Programming skills translate easily between them

"There are only two kinds of languages: the ones people complain about and the ones nobody uses."

-Bjarne Stroustrup, father of C++
Your First Program

http://www.zazzle.com/baby_girls_first_java_program_hello_world_tshirt-235063563751392326 $23.95
How Java Works

**Source code:**
Plain text file created in some editor (notepad, vi, TextEdit, Eclipse, DrJava, ...)

```
public class HelloWorld
{
    public static void main(String [] args)
    {
        System.out.println("Hello world!");
    }
}
```

**"compiling"**
% `javac HelloWorld.java`

**Java bytecode:**
Intermediate language that any device running Java can understand (humans generally ignore this)

```
Compiled from "HelloWorld.java"
public class HelloWorld extends java.lang.Object{
    public HelloWorld();
    Code:
    0:      aload_0
    1:      invokespecial #1; //Method java/lang/Object."<init>":()V
    4:      return
    public static void main(java.lang.String[]);
    Code:
    0:      getstatic    #2; //Field java/lang/System.out:Ljava/io/PrintStream;
    3:      ldc           #3; //String Hello world!
    5:      invokespecial #4; //Method java/io/PrintStream.println:(Ljava/lang/String;)V
    8:      return
}
```
How Java Works

Java bytecode: Intermediate language that any device running Java can understand (humans generally ignore this)

"running"

% java HelloWorld
Eclipse

- **Eclipse IDE (Integrated Development Environment)**

```java
/* Name: Michele Van Dyne */
/* Description: Display the text "Hello world!" on the screen. */
/* Our first program */
*/

public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

Hello world!
Eclipse

- **Eclipse IDE (Integrated Development Environment)**
  - Recommended but not required
  - Free
  - Helpful features:
    - Syntax highlighting
    - Flagging likely mistakes
  - We will use mostly as a text editor
    - Ignoring 95% of its features
  - How to install?
    - See course web site, resources page
  - We'll still learn to work on the command line
Compiling and Running

• A Java program can involve any number of classes.
• The class to run will contain the words:

```java
public static void main(String[] args)
```

somewhere in the file

```java
public class CostCalc
{
    public static void main(String[] args)
    {
    
    }
}
```
Anatomy of a Java Program

Name of the class, must be in file named `CostCalc.java` (case sensitive!)

```java
public class CostCalc {
    public static void main(String[] args) {
    }
}
```

Extra things from the command line

The input that allows the program to produce variable output

All the action goes here (for now)

```bash
% java CostCalc bananas 12 0.21
To buy 12 bananas you will need $2.52
```
**Our First Program**

**args Array**

```java
public static void main(String [] args)

% java CostCalc bananas 12 0.21
To buy 12 bananas you will need $2.52
```

<table>
<thead>
<tr>
<th>identifier</th>
<th>meaning</th>
<th>value</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>args[0]</td>
<td>1st thing on command line after Java class name</td>
<td>&quot;bananas&quot;</td>
<td>String</td>
</tr>
<tr>
<td>args[1]</td>
<td>2nd thing on command line</td>
<td>&quot;12&quot;</td>
<td>String</td>
</tr>
<tr>
<td>args[2]</td>
<td>3rd thing on command line after Java class name</td>
<td>&quot;0.21&quot;</td>
<td>String</td>
</tr>
<tr>
<td>args.length</td>
<td># of things on command line</td>
<td>3</td>
<td>int</td>
</tr>
</tbody>
</table>
Command line args in Eclipse
Command Line args in Command Shell

```
C:\Users\Mich\Desktop\CSCI 135 Fall 2016\Workspace\00-intro>java ArgsExample apples 5 1.25
To buy 5 apples you will need $6.25
C:\Users\Mich\Desktop\CSCI 135 Fall 2016\Workspace\00-intro>
```
Some Terminology

<table>
<thead>
<tr>
<th>OFFICIAL</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>Arguments</strong> – the items inside the parentheses that provide data needed by methods</td>
<td></td>
</tr>
<tr>
<td>• <strong>Method</strong> – the code that is executed when called</td>
<td></td>
</tr>
<tr>
<td>• <strong>Variable</strong> – something that can store data</td>
<td></td>
</tr>
<tr>
<td>• <strong>Statement</strong> – an instruction to the computer; in Java it ends with a semicolon</td>
<td></td>
</tr>
<tr>
<td>• <strong>Syntax</strong> – the grammar rules for a programming language</td>
<td></td>
</tr>
<tr>
<td>• <strong>Flow of Control</strong> – the order in which instructions are executed</td>
<td></td>
</tr>
</tbody>
</table>
Algorithms

• By designing methods, programmers provide actions for objects to perform.
• An algorithm describes a means of performing an action.
• Once an algorithm is defined, expressing it in Java (or in another programming language) usually is easy.
Algorithms

- An algorithm is a set of instructions for solving a problem.
- An algorithm must be expressed completely and precisely.
- Algorithms usually are expressed in English or in pseudocode.
Example: Total Cost of All Items

- Write the number 0 on the whiteboard.
- For each item on the list
  - Add the cost of the item to the number on the whiteboard
  - Replace the number on the whiteboard with the result of this addition.
- Announce that the answer is the number written on the whiteboard.
Summary

- Computer Basics
- Programs and Languages
- Introduction to the Eclipse IDE
- Our First Program
  - Comments
- Algorithms