Announcements

- Zybooks Chapters 7 (Sections 16 - 20), Ch 8 (Sections 3 - 6), Ch 9 (All Sections)
- Program 2 Coming soon…
- Updated Schedule and Syllabus

Good Programming Practices

- Good Comments
- Appropriate Variable Names
- Appropriate Code Structure
- Frequent Compilation
- Program in Small Chunks
- Pseudo Code

Comments

Unnecessary comments:

Int arraySize; // This is the size of the array

String lastName; // This is a person’s last name

Good Comments

```
char letter1 = 'a';
char letter2; // The value of this variable will exclusively alternate between a, b, and c

// Loop until letter1 has reached the end of the alphabet
while (letter1 <= 'z') {
      letter2 = 'a';
       // Loop through all possible combinations between the current value of letter1 with
       letters: a, b, and c
       while (letter2 <= 'c') {
              System.out.println("" + letter1 + letter2 + " ");
              ++letter2;
       }
      ++letter1;
}
```

Bad Comments

- Incorrect Comments

```
Int phoneArrayLength;  // Array that holds phone objects
```

Variable Names

Good:

- Person[ ] personArray; // Notice the camelCase
- Int filterArrayLength;

Bad:

- Person[ ] PersonArray; // NOT camelCase
- Int filter_Array_Length; // Do not use underscores between words
- String foo; // What does this even mean???
- Int temp; // Avoid undescriptive names, what does temp mean? Tempurature or temporary?

Code Structure

- Do not ignore curly braces with single ‘if’ statements
- Properly Indent your code blocks
- Leave a line break (a line with nothing on it) between two unrelated sections of code to help improve readability

Compile Frequently

- Drastically improves development time (even if you’re in a rush)
- Catch runtime errors / bugs early on
- Keeps debugging runtime errors easy (Prevents cascading error effect)

Program in Small Chunks

DO NOT CREATE AN ENTIRE PROGRAM WITHOUT EVER HAVING COMPILED AND TESTED YOUR CODE!!!

Pseudo Code

- The practice of planning out a program BEFORE you begin coding
- Chronologically organize your program in small logical chunks
- You do not need to write correct programming syntax, but should make use of common control structures such as: if statements , for / while loops etc.

Example:

Create a square object that can calculate its own: Area, Parameter, and Center point.

How to use Java Documentation

- Check your JDK Version
- Google: Java JDK {Your JDK Version Number} {Class you want to know more about}
  - Example: Java JDK 11 String
  - The first link should be at something like: docs.oracle.com

Two Dimensional Arrays

An array of arrays

Example:

Int [ ][ ] x = new int[2][3];

This allocates a two element array where each element is an array of size 3

Two Dimensional Arrays (cont.)

Can also initialize like this:

```
Int[ ][ ] x = { {1, 2, 3}, {6, 5, 4} };
```

This is equivalent to:

```
Int[ ][ ] x = new int[ 2 ][ 3 ];
x[ 0 ][ 0 ] = 1;
x[ 0 ][ 1 ] = 2;
x[ 0 ][ 2 ] = 3;
x[ 1 ][ 0 ] = 6;
x[ 1 ][ 1 ] = 5;
x[ 1 ][ 2 ] = 4;
```

Classes vs Objects

Analogy:

A single blueprint can be used to create many buildings. Each building may be slightly different (perhaps in terms of color) but ultimately they are nearly the same.

A Class can be thought of as a blueprint for objects. An object is an instance of a Class. So while you might have one Class (blueprint) you can have many objects associated with that class.

Class

Example:

Class Foo {

}

Somewhere in main…

Foo bar = new Foo();
Foo bar2 = new Foo();
Foo bar3 = new Foo();

Notice there are now three instances (or objects) of the class Foo: bar, bar2, and bar3

Classes (cont.)

Classes can have multiple methods and variables associated with it

Those methods and variables exist with in the scope of the objects.

What is scope....?

Static

Static provides an exception to the previous slide, where the scope of the variable is now global and it will persist across objects.