Object Oriented Programming
Introduction

Mr. D Bylsma
Chris
Adrian

1 Hmmmmmm

Bad programmers write code for machines to run. Good programmers write code for humans to read. In reality, one of the biggest challenge programmers face is writing elegant code; code that is readable, maintainable, and easily extensible. Previously in this course, you have (most likely) been learning language syntax; this is how I draw a rectangle, here's how I do math, here's how I detect keyboard events, and I put them together to make a simple animation. This is good; you're practicing your problem solving skills, which is another fundamental part of "computer science." However, in this next section of IT 12, we will begin focusing on **software design**, or "writing good code" (and at the same time problem solving). This will likely be the hardest section yet, as we're looking for different qualities in your work. However, when the learning goals finally click, I expect you will really appreciate the time you spent learning this properly :)

2 Object Oriented Programming? I thought this was Java!

We all know English, but that doesn't make us good writers. Likewise, we can learn many computer languages, but that doesn't make us good programmers. Some of you literature-types may also already know that a good writer who knows multiple languages is usually a good writer in all of them. This is the first step moving away from what's called "cowboy coding" and towards **software design**. This time, you are not learning a new language to fool around with; you are learning **Object Oriented Programming**. We will learn OOP in Java; learning Java is a byproduct of learning OOP design.

3 What is OOP?

Object Oriented Programming is a **programming paradigm**: sort of like a pattern/convention/rule of thumb people follow to help write good code. It's proven to be fairly successful at doing its job. Because it's quite readable and maintainable, most programmers like it (say, over functional programming) and it's often advertised as a feature for different languages ("PHP5 is now OOP!"). In OOP, you (the programmer) are concerned with **modeling the *data*** in an application. This may sound obvious, but think back to the assignments you did in the past; you probably had a few big files with variables and functions littered all over the place.

Ironically, in the first part of this section of the course you will be doing the same cowboy coding as before. First, you will learn a new language (Java) but with a stronger focus on algorithms and "better" problem solving (better methods of solving a problem vs. whatever works). Next, you can learn the new syntax and procedures for OOP. **This is not easy. You must be willing to learn this new thing**. We'll give more concrete details about OOP later, although the pseudo-definition above is key to OOP.

4 What is Java

Java is very much an object-oriented language. It was developed/designed/released in the early 1990s, and in fact — meant to be simple (it stripped out many features of C++, but despite losing on many powerful features, Java was a fun environment to code up something in. Today, it's sorta becoming bloated with new, "great" features, but that's a discussion for another setting). Nevertheless, I say it's the first "real" language most of you will be programming in (except for Javascript, which is a powerful, powerful language when done right). So it seems like there's more to it, but you just have to devote time, effort, and learn it.

I'll use some technical terms, because it helps anchor the important points.

Java is statically typed. This means all your types (booleans, ints, Strings, and later your custom classes) are checked at compile time. This makes coding a little more tedious; you have to specify these types everywhere and rules for what you can do are very rigid, but because another program is checking your work, it helps get rid of errors.

Java is strongly typed. This means there are strict rules about what you can and cannot do with types. With the exception of a few cases (e.g. adding a String and a Number in Java converts the Number to a String if the Number is the second operand), Java doesn't allow types to be intermixed. This is related to (but not tied to) the term above; if you try to do String myString = 10 + "10"; in Java, it will complain when you compile it (In a dynamic, strong language, you may not need to compile it, or it won't complain when you compile it, but it will when you run it).
CORA: Compile once, run anywhere. Java is a programming language. There is a specification about the "grammar", or rules/syntax for this language. Using this grammar, people have built compiles for Java. A Java compiler usually, but doesn't have to, compiles your Java code to bytecode (.java to .class files). Java bytecode is a small set of platform independent basic computer instructions. When someone tries to run a Java program, they need a Java Virtual Machine. There is a JVM for each platform, which reads the simple bytecode each time the program executes, and takes care of any environment inconsistencies. That's how you can compile Java once, and run it anywhere (assuming it has a JVM).
Getting back to Java as an object oriented: the power of Java comes mostly from it's strict adherence to OOP. Each file is a **class** (we'll talk about what that is later), so you're forced to use the OOP paradigm. That's why we chose to use Java for this section, although PHP, Python, even ActionScript could do.

5 The course to come

In the first half of this unit (approximately assignments 1-5), you will learn Java syntax. Ironically, you will be doing most of the same cowboy coding as before, although we will focus more on algorithms and stricter problem solving (vs. "whatever works"). In the second half of the unit, we will look at OOP.
A large part of the course will be taught through video tutorials. They are not fun, and it's easy to zone out. But do watch them if you want to learn.

5.1 Marking

While marks should rarely be a concern while you're learning, I want this section's values to be clear from the start. We will be judging:

- Whether your program does what it's supposed to do
- How your program does it (whether it does it well) (software design: writing maintainable code)
- Whether you followed programming conventions (software design: writing readable code)
- Decomposition of tasks

6 Decomposition

Decomposition is breaking down a large task into many small tasks. For example, let's say in your VB project you wanted to read an XML file of a user's bookmarks for your custom web browser. You may break the tasks down into some structure like (I am using Java syntax to define these methods, but you should have an intuitive understanding of what they mean):

- List¡Bookmarks¿ readAndReturnBookmarks(String filename)
  - String openFile(String filename)
  - List¡String¿ parseBookmarks(String fileContents)
  - Bookmark parseSingleBookmark(String savedString)
    - String getBookmarkNameFromString(String savedString)
    - String getBookmarkURLFromString(String savedString)
    - Bookmark createBookmark(String name, String URL)

It is hard at first deciding where to break a program down, but you must try. We have chosen this as one of the core learning goals, as it is a key, key concept to not only OOP but good program design in general. Plus, it is a fairly simple concept **if you are open minded and commit to learning**.
A few tips:

- Break the problem/task/sequence down to pseudocode steps
- Use good names for each function (that will help you understand the problem, and also is important for writing good code)

- Identify the responsibilities of each function

As an example for the last two points, say you have a function calculateNextPrime(int start). In your assignment, you're supposed to print the next prime number. But should the printing go in this function, or should you get the next prime number (with this function), then print it, in the parent method? "Calculate next prime" is a fairly good name; "calculate" implies the function is doing some computation, and when the **responsibility** of a function is some computation, you should return the result.
Note: The createBookmark(String, String) function above is usually replaced by a constructor - something we'll learn more about when we really dive into OOP
Note: When a function is part of a object (again, something we'll talk about when we're directly in OOP), we call it a ***method***

7 Problem Solving

In this unit, we focus explicitly on problem solving. Since you are at the end of your IT 12 year, we expect you to be able to take care of yourself. **Primarily, this means not being afraid to try new things on your own, and trying to find the answer yourself**. Google is your friend; Google everything. Before asking a question, always try at least 3 Google queries (the more you use it, the better you'll get at finding things - in this case, specifically finding programming question-answers). I am mostly a self-taught programmer - I learned 95% of my computer science knowledge from the internet. When I took Computer Science 310 at UBC, I knew more than most students (highest mark on the midterm) — all stuff learned from Google. **Even more important than OOP is your problem solving attitude. It is a desire to learn, not merely be given the answer**
Recommended reading: http://whathaveyoutried.com

7.1 Pseudocode

Pseudocode is a way of describing how to do something using a hybrid between your native language (English) and a computer language. For example, to calculate a prime number we may have:

```
initialize a counter (integer)
while we not found a prime number
        increment the counter
        if the counter is a prime number
                return the counter
```

There are a couple things to note:

- This code looks somewhat like your real code
- It's not perfect, but
- It makes what you're supposed to do obvious

You should always write pseudocode when you don't know how to do something. Yoda says: "Do or do not. There is no try." I take that as: if you don't know, **just do something** (i.e. write pseudocode). Don't just sit there like a frightened deer on the road. Pseudocode is how you solve a problem; that's 90% of the problem. Once you have pseudocode, you only need to translate it into a computer language, and that Google and others can help you with
If you want to learn more, read the Wikipedia article on pseudocode

8 Coding Conventions

You may say: why do I have to comment my code? Why do I have to indent my code a certain way? It all compiles to the same stuff anyway. Mainly, we want you to get into the habit of looking for and following conventions. In programming, conventions are almost always good; they make your code easier to read, understand, and maintain.
"I want to be the only person able to understand my code. That means I'm important to the company!" Being crucial to your job is the worst way to not be promoted.
See this thread (first comment): http://workplace.stackexchange.com/questions/9235/

8.1 Variable, Function, and Class Naming

Related to coding conventions are good names. Names should be descriptive but not too long. Variables and functions are **lowerCamelCase**, and Classes are **UpperCamelCase**. This makes them easy to identify.