Lab 1: Introduction to Java

General Programming Guidelines

Our **goal** is to get started using Java and learn how to create a class from scratch. Before we get started on the mini-assignment, let's go over some general Java and programming guidelines:

- Packages
- Methods and Parameters
- Constructors

Packages

Java programs are built upon classes. As the programs you write grow in size, though, and you find yourself with more and more classes, you'll want to organize these classes somehow to prevent overlap and unintended consequences. Thankfully, Java organizes classes into ***packages*** to keep code organized and manageable. A package is a group of related classes. You can think of a package as a stand alone group of classes that performs a certain task. For example, if your program was a video game, its packages could include the graphics package, the physics package, the Artificial Intelligence package, etc. Packages can also be nested (the physics package could contain a math package). *All Java classes should belong to a package*[1], and you specify that package by typing:

```
package <some package>;
```

at the very *beginning of the file* (before you even declare the class), and *without the angle brackets*. All this being said, the programs in CS15 aren't big enough to need multiple packages, **so the name of your packages will simply be the names of the programs.**

Methods and Parameters

A class can't do anything without you defining some capabilities for it. If you had a **`Robot`** class, you would want it to do something cool like walk, dance, cook, maybe even do your CS15 assignments for you. Java models these capabilities using methods. Methods have a specific syntax; refer to lectures if you don't remember.

[1] The only real exception is when you are writing a very small Java program that consists of only one class.

Sometimes a method needs outside information in order to perform its task. A way to pass information to a method is through parameters. For example, a **`Robot`**'s **`cook(...)`** method needs to know what to cook. You can tell it what to cook using parameters. If you didn't use parameters, you would end up writing a cook method for every possible dish your Robot can cook! Just think: **`cookChicken(), cookPork(), cookSteak(), cookTomatoes().`** Or you can write a method that takes in a **`Food`** that you want your Robot to cook as a parameter: **`cook(Food food)`**. When a method takes a parameter, it can perform operations on that parameter, or use the information contained in that parameter to perform other operations.

Constructors

Let's not forget constructors; they are special methods that are automatically called when an object gets instantiated (i.e. every time you call **`new <someConstructor>();`**). Every class needs one[2], and they are usually used to instantiate instance variables and set default values. Going back to our Robot example, when Java goes to build a new Robot, it will look in the constructor to see what default values should be given to its color, name, height, weight, etc. The programmer can then set these values when defining the constructor. Constructors have to be named the same as their class, so a Robot class will have a **`Robot()`** constructor. Look into the lecture slides for specific syntax.

Just as in any method, parameters are very useful for constructors. Most prevalently, they are used to set up associations (knowledge of other objects) when an object is instantiated. Remember the **`DogGroomer`** and **`PetShop`** example from lecture? We can use a parameter in the **`Robot`** class to let the Robot know which student it belongs to and what color the student ordered its Robot to be.

Pair Programming

Several labs in CS15 will be done with a partner assigned by your section TAs. Please read the CS15 Pair Programming Missive before completing this lab.

[2] If you don't write a constructor, Java will automatically create one for you, but it won't do anything.