This packet is intended to briefly introduce many concepts that will be very useful to you throughout the Hackathon. The topics are not in any particular order, but make sure you read through them all. You are allowed to use this packet to help you with any tasks throughout the Hackathon. More information about specific topics will either be provided on the tasks or can be found in the API.

**APIs**

Throughout the Hackathon, many methods and classes will not be given to you. It is important to be able to find them. An API (Application Program Interface) is a place where methods and properties are kept track of. Java maintains an API of all of its classes. This is the place where you should go whenever you want to see which classes might be useful to you, or which methods in each class you might want to use. It can be found at:

<https://docs.oracle.com/javase/7/docs/api/>

**Importing**

Java has designed many of its own classes. However, in order to use them you must import them. Import statements are put at the top of the program. “import” is a keyword, so don’t use it as an object name. Whenever you need to import a class, it will be mentioned in the task or you will be able to find it at the top of the class’ page on the API.

**Primitive types**

In many programming languages (including java), there are easier ways to store some kinds of data without using a class or a traditional object. They are called primitive types. There are five primitive types that will be useful for you today:

* int – an int stores a whole number
  + int i = 3;
* double – a double stores a decimal value
  + double d = 3.14;
* boolean – a boolean is either true or false
  + boolean b = true;
* String – a string is a sequence of characters
  + String s = “Hello”;
* char – a char is a single character
  + char c = ‘q’;

**Methods**

Methods are segments of code that can be called over and over again. They perform a specific function. Below are a few examples of methods:

public void methodOne()

This method header shows that methodOne is void, which means it doesn’t return anything. It simply completes the action inside the method body.

public String methodTwo()

This method header shows that the method returns a string. This means that it completes its action then sends a string back to where it was called.

public int methodThree(int arg)

This method returns an int, but it also takes an int *argument*. An argument is a value that can be passed into the method. The method can then use that value in its body.

public static void methodFour()

This is a static method. This just means it is a class method. The entire class performs this method instead of just one object. For example, println() is a static method because the entire System class performs it.

**Print vs Println**

Two common methods used in both debugging a program and producing output are System.print() and System.println(). The difference between them is that print prints out exactly what you put in the parentheses. However, println() also prints a new line after what you type. (Hint: This is very important in Task Four)

**Classes**

Java relies on the use of classes. Classes are basically blueprints for objects. They describe the properties and functions that every object of that class should have. Classes in java are declared in separate files and the filename must match the name of the class. An example of a class file would be Apple.java.

**Constructors**

Constructors are the section of a class that tell the computer about the default state of each object. When an object is created, the constructor gives it default values. Depending on how you write the constructor, the values could either be input when the object is created or just be a default specific to the class.

**Instantiation**

Intsantiation is a fancy way of saying create a new object. For primitive types, it is pretty easy, as shown above. However, it is a little more complicated for normal objects. The exact format depends on the constructor of each class. If the constructor takes arguments, include those in the parentheses. The general format is:

Object o = new Object(*arguments*);

**Eclipse**

Eclipse is an example of an IDE. An IDE, or just a programming environment, is a place that developers can write and run code. The environment usually has a way to compile the code into a form that is more easily read by the computer. It also has a place to run the program. Below is a description of the useful functions of Eclipse.