Java Fundamentals Checklist

The following are several points of competency and skill that are intended to help identify areas of additional practice and study. Not all answers are found in the Codeup curriculum.

Knowledge

Java I

- [ ] I can explain how Java code runs: from being written to executing on an operation system.
- [ ] I can explain the difference between a primitive type and a reference type in Java.
- [ ] I can name several primitive types in Java.
- [ ] I can explain why multiple number primitive types are used.
- [ ] I can give an example of implicit and explicit casting.
- [ ] I can explain the difference between a runtime and compile-time error.
- [ ] I can identify the major editions of Java.
- [ ] I can define what a Java wrapper class is.

Java II

- [ ] I can articulate what OOP is.
- [ ] I can identify the four levels of visibility for a class/field/method and the identifiers used.
- [ ] I know what class of static methods is often used when working with arrays.
- [ ] I can explain the difference between static (class) fields/methods and instance fields/methods.
- [ ] I can explain the difference between method overloading and method overriding.
- [ ] I can identify the four pillars of object-oriented programming and give examples of each one.
- [ ] I can describe the differences between interfaces and abstract classes and their use cases.
- [ ] I can generally describe what the Java Collections Framework is and specifically talk about the differences between a list, set, and map.
- [ ] I can explain the difference between a checked and unchecked exception.
- [ ] I can explain what a Java annotation is, why they are used, and give examples.
- [ ] I can explain the difference between the throws and throw keywords with regard to exceptions.

Skills

Java I

- [ ] I can write a hello world Java program from scratch.
- [ ] I can write Java control statements (if, if/else, switch, ternary) and loops (while, do-while, for, for-each) from memory.
- [ ] I can write examples of the 8 Java primitive types.
- [ ] I can write examples of explicit casting and implicit casting.
- [ ] I can convert a numeric string to a number and a number to a numeric string.
- [ ] I can define a constant.
- [ ] I can write a program that takes in user input and returns an output to the user.
- [ ] I can compare two string values.
- [ ] I can write out a program comprised of multiple methods.
- [ ] I can get the length of string, trim it, get a portion of the string, create a new string replacing some portion of it.

Java II

- [ ] I can write a class that contains both static and instance fields and methods, constructors, setters and getters.
- [ ] I can write a program composed of multiple classes requiring instantiation (not just using classes as groups of static methods).
- [ ] I can write my own reference type and fill instances of it in an ArrayList and HashMap and use it as a return and input type of a method.
- [ ] I can write an abstract class and a subclass that extends from the abstract class.
- [ ] I can write multiple interfaces and implement their methods in another class.
- [ ] I can create arrays of various data types and sizes with and without using an array initializer.
- [ ] I can get the length of an array, get a specific element or range of elements, create a new array replacing some portion of it, and iterate over an array of elements.
- [ ] I can do the above with an ArrayList.
- [ ] I can comfortably use the helper methods of both Arrays and Collections classes.
- [ ] I can write a program that throws a runtime exception.
- [ ] I can write an example of method overloading and method overriding.
- [ ] I can write a program that reads and writes to a file.
- [ ] I can write methods and collections that work with a parent class or interface type but use instances of objects that extend from these parent classes or implement these interfaces (Polymorphism).