**General Java Notes**

**VARIABLES:**

* Local variables: not visible outside of the method
* Class/Instance variables: declared inside of class but outside any method
* Class: static. Single shared value for all instances, even if none exist
* Instance: non-static. Separate value for each instance; aka member/field variables
* Reference variables: data type is an object type; refers to an object

-Must be declared as one of the following object types: class, interface, or array type

**KEYWORD FINAL:**

Final Primitive Variables:

Final Reference Variables:

Final Methods:

Final Classes:

* Cannot be extended or inherited; can’t be a superclass but CAN be a subclass
* All methods within the class are implicitly final as well

**Object** means a real word entity such as pen, chair, table etc.

**Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

* Object: entity that has a state and behavior
* Class
* Inheritance: one object acquires all the properties and behaviors of a parent object
  + Provides code reusability
* Polymorphism: one task is performed by different ways
  + Java – method overloading/overriding is used to achieve this
* Abstraction: showing only the necessary details/info to the user
* Encapsulation:
  + Class is the best example

OO Programming: attributes and behavior are contained within a single object

* Data is referred to as attributes
* Functions are referred to as methods
* By combining data and methods in the same entity, access to data is controlled

\*It’s normally better to build small objects with specific tasks rather than large objects with many tasks

**Wrappers - OOP**

Wrappers are used to convert any data type to an object. Each primitive type has a corresponding Wrapper type/ Class that is used in the conversion of primitive types to objects. The wrapper classes do this by “wrapping” primitive values inside the object so that these values can then be used in mechanisms only allowed for objects. These classes also have methods to unwrap the object and return the primitive data type. This is important because it allows us to do things with primitive types like include them in an Array List, give them a null value, or treat them polymorphically like all other objects. ( Discussion 03 )

Wrapper Classes:

Primitive: Wrapper:

* boolean -> Boolean
* char -> Character
* byte -> Byte
* short -> Short
* int -> Integer
* long -> Long
* float -> Float
* double -> Double

How to use them