Java 101

Object Oriented Programming

A Brief Recap

- Classes are collections of variables and functions

- To use a class, you must create an object from it
  - Person myObject = new Person();

- When an object is created, that class's constructor method is run automatically
  - Constructor methods have the same name as the class

More Recap

- Each object you create is a copy of the class but can then updated to be different than other objects of the same class
- You access attributes (variables) and methods (functions) of an object by using myObject.attribute and myObject.method()
- Use private, protected, and public to restrict access

Using ‘this’

Within class methods, use ‘this’ to refer to the object that is using the method

For example, within a class ‘Person’:

```
private numberOfEyes = 2;

public function loseAnEye() {
    this.numberOfEyes = this.numberOfEyes - 1;
}
```

Why Use ‘this’?

Mainly used to keep variables straight within methods

For example, within a class ‘Person’:

```
private numberOfEyes = 2;

public function defineEyes(numberOfEyes) {
    this.numberOfEyes = numberOfEyes;
}
```

Inheritance

Class inheritance is a way to build upon previously defined classes, also called class extension

The base class is called the parent class

The class that inherits the parent class class is called the subclass (could also say this class extends the parent class)

Inheritance Example

```
public class Person {
    // Define some things here
}

public class Customer extends Person {
    // Define more stuff here
}
```

Uses of Inheritance

Inheritance can be used to either add things to the parent class or overwrite existing attributes or methods

If you want to overwrite something, just give it the same name in the subclass that is in the parent and any object created from the subclass will contain the update

Abstract Classes

Abstract classes are base classes that cannot be instantiated (cannot have objects created from them) with the intention that they will be extended

Used when you want to create a variety of classes but they all have something similar

Example: Mammal abstract class -> Animal subclasses

Interfaces

Interfaces are similar to abstract classes, they are basically an outline of a class that must be implemented

Every method in an interface MUST be implemented in subclasses (not true for abstract classes)

Rather than being extended, interfaces are implemented

Interface Example

```
interface Reptile {
    void setNumberOfScales(number);
}
public class Dinosaur implements Reptile {
    void setNumberOfScales(number) {
        this.numberOfScales = number;
    }
}
```

Encapsulation

Encapsulation is an object oriented design principle that stresses the idea of making as much of a class private as possible

This minimizes what outside classes and programs can do with your objects

Can implement this with getter and setter methods to get and set the attributes of an object

Getters and Setters

For every attribute within a class, there must be a method to change or return the value of that attribute

```
public class BankCustomer {
    private int SSN = 123456789;
    public int getSSN() { return this.SSN; }
    public void setSSN(newSSN) { this.SSN = newSSN; }
}
```

Polymorphism

Polymorphism is an object oriented design principle that stresses the idea of creating methods that overwrite their parent methods but still produce the same outcome

This preserves the expected outcome for all subclasses of a shared parent class in order for them to all still act the same way

Example of Polymorphism

Suppose we have a parent class Animal with a method printNumberOfAppendages(), a subclass of Animal called Mammal could have arms and legs that would make up appendages while another subclass of Animal called Fish could have fins that make up its appendages

In this example, no matter the subclass, we still want to print the number of appendages with this method so we need to overwrite the method in each subclass but they’ll still print the same thing

Git

Git is the “stupid content tracker”, stupid as in, it won’t do anything intelligently like merge your code on its own if there are conflicts

Git is used as a revision control system used to track changes in code and allow resetting code to previous states and merging files together

Useful for both large and small projects

Github

Github is a website that stores repositories of code

People create accounts on it, send code to it (using Git or SVN), and submit bug reports on it

There are similar websites like Bitbucket, Beanstalk, Gitlab, etc.

Git Workflow

- Initialize a Git repository on your local computer
- Make some files, write some code, and so on
- When you get to a good milestone, add your files to the Git staging area
- When you're done adding files, commit what's in the staging area
- When you want to send your code to a remote repository (Github), push your code to it