Week 5 - Research

(Researched from week 5 assignment notes)

1. What are the four pillars of Object-Oriented Programming? Explain each pillar.

   The four pillars of OOP consist of Inheritance, Encapsulation, Abstraction and Polymorphism

The idea of inheritance is that a class can adopt the traits and behaviors of another class, with the option for the first class to replace some of those traits and behaviors as needed.

The idea of encapsulation involves hiding complicated activities inside straightforward verbs.

The idea of keeping state and logic within is known as abstraction.

Polymorphism is the quality of existing in a variety of forms and having the capacity to carry out numerous tasks in a variety of ways.

2. What is the relationship between a Class and an Object?

A class specifies the properties of an object, such as the acceptable range of values and the default value. Object behavior is also described by a class. A member or "instance" of a class is an object. Every property of an object has a state in which those values are either explicitly declared by you or are determined by default settings.

3. What are the differences between checked and unchecked exceptions?

A checked exception is something that we want to explicitly look for. It is something that we do not have control over such as an input/output exception.

An unchecked exception is only checked when a programmer makes a mistake such as an out of bound exception

4. What are the differences between abstract classes and interfaces? When should you use one over the other?

The primary distinction between an abstract class and an interface is that the latter can have state through instance variables while the former cannot.

An interface can be used when you know what the class implementing the interface will do but want to leave the actual implementation 100% up to that class. When a class implements an interface, it must create a concrete implementation of each method defined in the interface.

5. What is unit testing and why is it important?

As applications become more complex, it becomes difficult to see the impact that change may have on the rest of the code. You may tweak something and unknowingly break three other areas of the code that were dependent on the code that was tweaked. To help mitigate this, we have an industry standard practice called Unit testing. Unit tests are code that you write that run your application code to make sure the output is as expected. It is an expensive practice but worth it. Most places require that you write unit test for all code. You write code that calls the method and says what you actually receive is equal to the expected output. If it is true, the test passes. If false, the test fails

6. What is your favorite thing you learned this week?

My favorite part of this week was understanding how object-oriented programming plays an integral part in making functional programming more succinct. In my previous lessons, I learned how to access properties or data on classes using dot notation. While this works, it is considered very poor practice in OOP. We need to encapsulate the data to only expose what needs to be used and to do this we use access modifiers.