Week 5

Kevin McKenzie

4 Pillars of OOP

Inheritance - Inheritance refers to the ability for a class to take properties and methods from another class. Causing less instantiations of code written. Using the extends key-word for pairing a child class or (sub) to a parent or (super) class.

Abstraction - Abstraction refers to the non-instantiation of a class/method. An abstract class can have both implemented methods and non implemented methods within them showing what is needed within a class or an interface. Helping to organize and give a direction of what functions/properties are needed within.

Encapsulation - Encapsulation refers to the bundling of variables and methods within a class to hide and prevent access from outside sources. Encapsulation uses 4 access modifiers - private , public, protected, No Modifier, in order to do so. Use setters and getters to modify private properties.

Polymorphism - Polymorphism is simply the difference in classes that inherit one another from the extends keyword. When inheriting from a class the basic functions/properties is defaulted as the base, then the differentiating information that makes up each subclass is then polymorphic.

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

The difference between a class and an object in OOP is similar to the way land and property is in the real world. When you create a class in OOP that class becomes the canvas of your creation. Then the object becomes the actual thing you're putting on/in that canvas or property. That object can be anything.

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

A checked exception is outside of our code, checked exceptions are in our control and they show a red error when something is wrong. A try-catch block is used to manage checked exceptions. The try catch block is used to replace an exception in your program.

An unchecked exception is an error we have no control over and that dont allow our code to run when it occurs.

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

- An abstract class can consist of both method signatures with and without an implimenting code body. Abstract classes help to create regular classes by giving method signatures for the implementing class to iterate however fit. Organizing what each class should have.
- An Interface can only have non implemented methods and variables. The interface automatically assumes all methods are abstract resulting in only non implemented methods to run.

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

Unit testing is the first part in a 4 part process it takes to successfully test your program for public use. Unit testing helps to find errors and bugs in the code. Testing individual pieces of the program for efficient functionality before going public.

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

I believe learning about OOP was pretty interesting. Considering how for the first month of the course what we learned was mainly CS jargon and understanding the language of java. Now that I know a little more about the industry by grasping what object oriented programming means I feel a lot more confident in my journey with Promineo Tech. With each day the more I learn the more confident I get in working in the field.

https://youtu.be/zbVAU7lK25Q

https://youtu.be/j0lBrYSlYaU

https://youtu.be/3WSIjS1Qgug

https://youtu.be/jFQfulEd8sU