# Week 5 Research

## Collin Wahlund

**OOP:**

The four pillars of Object-Oriented Programming are Abstraction, Encapsulation, Inheritance, and Polymorphism.

Abstraction is the idea of hiding the details of how the class works from the end user standpoint. Instead of calling the object and performing a high number of operations on it, the program can call a functionality of the object and rely that the results will be as expected. Interfaces and abstract classes are ways to do this as they will define what needs to be done without direct operations, allowing the details to be handled by classes that implement them.

Encapsulation is bundling the needed data and code into a single operational unit. The object will contain both the data required and the operations that can take place. A good part of this is making anything not necessary to be exposed be private, using methods to perform the needed operations and data access. This also gives us control about how the data goes in, allowing for things like calculations and error checking the input, without giving direct access to the data except as we allow.

Inheritance is a class inheriting attributes from a parent class. This allows you to extend a more general class into a more specific one. For example, an employee parent class could be extended into hourly and salaried with the same basic information an employee does, but specialized data and methods that are relevant only to the type of employee we are talking about.

Polymorphism is the ability to have a class do different things depending on what is needed. Examples of this would be class constructors or methods that take a different number of parameters. Depending on what kind and number of parameters given we could perform items differently depending on the need.

---

The relationship between a class and an object is that the class is like the blueprint for something to be constructed, while the object is the final product. That is to say an object is an instantiation of its class, an item which could be of many, that utilize the same class.

Resources: https://www.geeksforgeeks.org/four-main-object-oriented-programming-concepts-of-java/

https://stackoverflow.com/questions/10839131/implements-vs-extends-when-to-use-whats-the-difference

**Exceptions:**

An exception is an unexpected issue with a program that causes it to malfunction. This kind of error if not caught will cause an issue and end the operations.

The difference between checked and unchecked exceptions is that checked exceptions will be caught when you compile your program with the compiler, while unchecked exceptions will not.

Checked exceptions are usually issues the programmer could notice and make sure to watch for in logic. An array out of index issue is an example of one that should be caught by the programming logic.

An unchecked exception is one that happens at run time. Operations may have items that could go wrong, so if there is something inherently that could go wrong it is important to catch the exception if it were to occur so as not to end the program unexpectedly. An example of this would be in an operation that adds data to a file. What if the file does not exist? How will the program handle this event? By setting up code to handle unchecked exceptions we can make our programs more resilient.

Resources: https://www.geeksforgeeks.org/exceptions-in-java/

https://docs.oracle.com/javase/tutorial/essential/exceptions/definition.html