Best Practices for Assignment <2>: Polymorphism in Java

Summary of Learning:

- **Polymorphism Principle:** Polymorphism uses those methods to perform different tasks. This allows us to perform a single action in different ways.

- **Types of Polymorphism:**
  - **Compile-Time Polymorphism:** Achieved through method overloading, where methods in the same class have the same name but different parameters.
  - **Run-Time Polymorphism:** Achieved through method overriding, where derived classes provide their own implementation of methods from the base class. The specific method called is determined at runtime based on the object's actual type.

- **Method overloading:** Method overloading is a feature in Java that allows you to define multiple methods in the same class with the same name but different parameter lists.

- **Method overriding:** Method overriding is enables a subclass to provide its own implementation for a method that is already defined in its superclass

- **Code Reusability:** It promotes code reuse by allowing you to use the existing functionality of the superclass in the subclass. This reduces code duplication.

Alternative Ways to Complete the Assignment:

Best Practices:

Method Overloading:

- **Use Descriptive Method Names:** Choose meaningful method names that reflect the purpose of the overloaded methods. This enhances code readability and clarity.

- **Avoid Excessive Overloading:** While overloading is useful, avoid excessive method overloading. Too many overloaded methods with similar signatures can lead to confusion.

- **Avoid Changing Return Types:** Overloaded methods should have the same return type. Changing return types can lead to unexpected behavior.

Method Overriding:

- **Use @Override Annotation:** In languages like Java, use the @Override annotation when you intend to override a method. This helps catch errors at compile time.

- **Understand Inheritance Hierarchies:** Be aware of the inheritance hierarchy and how method overriding propagates through subclasses. Understand the behavior of the most specific method that is invoked.

- **Maintain Method Signatures:** Ensure that the overriding method has the same method signature (method name, return type, and parameters) as the overridden

method in the superclass. Any deviation can lead to errors and unexpected behavior.