Explain how I use polymorphism and overriding

22101063 임성빈

In the provided code, Polymorphism and Method Overriding, two core concepts of object-oriented programming, are extensively utilized. Using these principles enhances the flexibility, scalability, and maintainability of the program.

**1. Polymorphism**

Polymorphism allows objects with the same interface to operate in multiple forms. In Java, this is typically implemented through inheritance or interfaces. Here, polymorphism is illustrated using inheritance.

Example:

The Student and Employee classes inherit from the Person class and implement its abstract method writeOutput(). This means that any object of type Person can invoke writeOutput(), but the actual behavior is determined by the concrete subclass.

In the Main class, the writeOutput() method can be called like this:스크린샷, 텍스트, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명

This code calls the writeOutput() method for all objects in the Person list (Under, Grad, Faculty, Staff). Depending on the actual type of the object, the appropriate overridden writeOutput() method is executed. This is a great example of polymorphism, increasing code flexibility and reusability.

**2. Overriding (Method Overriding)**

Method overriding occurs when a subclass redefines a method inherited from a superclass. This allows the subclass to modify the inherited method to fit its specific needs.

Example:

The abstract method writeOutput() defined in the Person class is implemented differently in the Student and Employee classes. Furthermore, Student and Employee are further specialized into Under, Grad, and Faculty, Staff, respectively, with each subclass overriding this method again.텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

(오른쪽에 email 있음)

Each subclass redefines writeOutput() to display different information based on the type of object. This is a classic use case of method overriding, utilizing the same interface (writeOutput() method) to produce varying outcomes based on the object type.

**Conclusion**

By leveraging polymorphism and method overriding, specific behaviors of each class are defined, a single reference variable can manage objects of various types, and appropriate methods are invoked at runtime. This significantly enhances code flexibility and makes maintenance and expansion easier.