객체지향프로그래밍 - 과제 #4

- 202184021 소프트웨어전공 박지민

1. 문제 정의

1. 두 종류의 프린터인 `InkJetPrinter`와 `LaserPrinter`가 존재한다.
2. 모든 프린터는 모델명(model), 제조사(manufacturer), 인쇄 매수(printedCount), 인쇄 종이 잔량(availableCount)을 가진다.
3. `InkJetPrinter`는 잉크 잔량(availableInk), `LaserPrinter`는 토너 잔량(availableToner)을 추가로 가진다.
4. `InkJetPrinter`는 rintInkJet 함수로 인쇄를 수행하며, `LaserPrinter`는 printLaser 함수로 인쇄를 수행한다.
5. 사용자는 어떤 프린터를 사용할지와 인쇄할 매수를 입력할 수 있다.
6. 인쇄 후 프린터의 남은 용지와 잉크 또는 토너의 잔량을 출력해야 한다.
7. 용지나 잉크/토너가 부족하면 인쇄할 수 없음을 사용자에게 알려야 한다.

2. 문제 해결 방법

2-1. 클래스 설계

- **Printer** 클래스**(**부모 클래스**)**
  - `std::string model` : 모델명
  - `std::string manufacturer` : 제조사
  - `int printedCount` : 인쇄 매수
  - `int availableCount` : 인쇄 종이 잔량
  - `Printer(...)` : 생성자
  - `bool print(int count)` : 인쇄를 수행하는 함수
    - availableCount, printedCount를 수정한다.
    - print 수행 후, 성공 시 true, 실패 시 false를 반환한다.
- **InkJetPrinter** 클래스**(**자식 클래스**)**
  - `int availableInk` : 잉크 잔량
  - `InkJetPrinter(...)` : 생성자, Printer(부모 클래스)의 멤버 변수는 Printer 생성자로 초기화한다.
    - 초기화 후 show() 함수를 호출하여 프린터 정보를 출력한다.
  - `void printInkJet(int count)` : 잉크젯 프린터의 인쇄를 수행하는 함수
    - print() 함수를 호출하여 인쇄를 수행한다.
    - page 당, 하나의 availableInk을 사용한다.
  - `void show()` : 프린터의 정보를 출력하는 함수

- **LaserPrinter** 클래스**(**자식 클래스**)**
 - `int availableToner` : 토너 잔량
 - `LaserPrinter(...)` : 생성자, Printer(부모 클래스)의 멤버 변수는 Printer 생성자로 초기화한다.

    - 초기화 후 show() 함수를 호출하여 프린터 정보를 출력한다.
 - `void printLaser(int count)` : 레이저 프린터의 인쇄를 수행하는 함수
    - print() 함수를 호출하여 인쇄를 수행한다.
    - page 당, 반개의 availableToner를 사용한다.
 - `void show()` : 프린터의 정보를 출력하는 함수

2-2. 동작 과정

1. main 함수에서 두 대의 프린터 객체인 `InkJetPrinter`와 `LaserPrinter`를 생성한다.
2. 사용자에게 인쇄할 프린터(1: 잉크젯, 2: 레이저)와 인쇄할 매수를 입력받는다.
3. 선택한 프린터의 인쇄 메소드를 호출하여 인쇄를 시도한다.
  - 잉크젯 프린터는 printInkJet(int pages) 메소드를 사용한다.
  - 레이저 프린터는 printLaser(int pages) 메소드를 사용한다.
4. 인쇄 후 각 프린터의 show() 메소드를 호출하여 현재 상태를 출력한다.
5. 용자에게 계속 인쇄할 것인지 묻고, 'y'를 입력하면 2번으로 돌아가고 'n'을 입력하면 프로그램을 종료한다.

3. 아이디어 평가

- 객체지향 프로그래밍의 적용
 - 부모 클래스인 Printer를 상속받아 자식 클래스인 InkJetPrinter와 LaserPrinter를 구현하였다.
 - 이를 통해 코드의 중복을 제거하고, 코드의 재사용성을 높였다.
- **OCP(Open-Closed Principle)**의 적용
 - 새로운 프린터 클래스를 추가할 때, Printer 클래스를 상속받아 새로운 클래스를 구현하면 된다.
 - 이를 통해, 기존의 Printer는 수정하지 않고, 확장 시에는 새로운 클래스를 추가하는 방식으로 OCP를 준수하였다.
- 코드의 유지보수성 향상
 - 중복 코드를 제거하여 코드의 가독성과 유지보수성을 높였다.
 - 클래스 구조를 체계적으로 설계하여 확장성과 재사용성을 향상시켰다.

4. 문제를 해결한 키 아이디어 또는 알고리즘 설명

- 상속
 - 상속을 통하여 클래스 간의 관계를 명확히 하였으며, 코드의 중복을 제거하였다.

- Printer 클래스를 통해 공통된 속성과 기능을 추상화하고, 자식 클래스에서 고유한 기능을 구현하였다.

- ex) printInkJet()과 printLaser()에서는 각각 availableInk는 page당 1, availableToner는 page당 0.5를 사용한다.

- 위의 문제를 상속을 통한 객체지향 프로그래밍을 통해 공통된 로직은 그대로 수행하고, 각각 클래스의 작은 부분을 수정하는 것만으로 쉽게 해결할 수 있었다.

- 캡슐화**: protected** 접근 지정자 사용

- 부모 클래스인 Printer의 멤버 변수를 protected로 선언하여 자식 클래스에서 접근할 수 있도록 하였다.
- 이로 인해, Printer 클래스를 상속받지 못하는 외부 클래스에서는 접근하지 못하지만, 자식 클래스에서는 접근 가능하여, 코드의 재사용성을 높였다.

- 코드의 중복 제거를 통한 효율성 개선

- 공통 기능을 부모 클래스인 Printer로 추출하고, 자식 클래스에서 필요한 부분만 구현하여 효율적인 코드를 작성하였다.
- 인쇄 후 상태를 출력하는 부분을 각 클래스의 show() 함수로 대체하여 코드의 중복을 없애고 구조를 단순화하였다.
- 이를 통해 유지보수 시 한 곳만 수정하면 전체 코드에 반영될 수 있도록 하였다.