Homework#04

문제 정의

같은 정보를 가지는 페인터(InkJetPrinter, LaserPrinter)간의 정보 충돌을 가상 상속을 통해 다중 상속의 모호성을 해결하는 코드를 작성한다.

단, 남은 종이나 남은 잉크가 없을 경우 동작이 불가능 하다는 문구를 띄우게한다.

문제 해결 방안

1. 일단 공통적으로 가지는 변수인 모델, 제조사, 인쇄 매수나 종이 잔량의 변수를 입력 받는 프린터 클래스를 작성한다.
   (이 때, 우리는 프린터 간의 정보 충돌을 막기 위해 다중 상속이 아닌 가상 상속으로 선언한다.)

```cpp
#include <iostream>
#include <string>

//공통으로 가지는 변수들을 프린터로 정의
class Printer {
protected:
    std::string model;
    std::string manufacturer;
    int printedCount;
    int availableCount;

public:
    Printer(const std::string& model, const std::string& manufacturer, int availableCount)
        : model(model), manufacturer(manufacturer), printedCount(0), availableCount(availableCount) {}

    virtual ~Printer() {}

    virtual void print(int pages) = 0; // 순수 가상 함수
    virtual void show() const = 0;
};
```

2. Printer 클래스에서 공통으로 가지는 변수가 아닌 잉크 잔량(availableInk)와 토너 잔량(availableToner)에 대한 각각의 프린터에 대한 정보를 Printer 클래스로 상속 받아 기본 틀을 받은 후, 기능들을 정의한다.

```
//공통으로 가지지 않는 변수(잉크 개수)
class InkJetPrinter : public Printer {
    int availableInk;

public:
    InkJetPrinter(const std::string& model, const std::string& manufacturer, int availableCount, int availableInk)
        : Printer(model, manufacturer, availableCount), availableInk(availableInk) {}

    ~InkJetPrinter() override {}

    void print(int pages) override { //main 함수에서 pages의 개수를 입력 받고
        if (availableCount >= pages && availableInk >= pages) {
            printedCount += pages;
            availableCount -= pages;
            availableInk -= pages; //잉크와 남은 종이 개수만큼을 pages에 할당한 변수 크기만큼 제거
            std::cout << "인쇄했습니다.₩n";
        }
        else {
            std::cout << "용지가 부족하거나 잉크가 부족합니다.₩n";
        }
    }

    void show() const override {
        std::cout << model << ", " << manufacturer << ", 남은 종이 " << availableCount << "장, 남은 잉크 " << availableInk << "₩n";
    }
};
```

(TonerPrinter도 잉크 잔량만을 공유하지 않을 뿐, 하는 기능은 똑같기 때문에 토너 잔량으로만 바꾸고 클래스를 정의한다.)

```
//공통적으로 가지지 않는 변수(토너 개수)
class LaserPrinter : public Printer {
    int availableToner;

public:
    LaserPrinter(const std::string& model, const std::string& manufacturer, int availableCount, int availableToner)
        : Printer(model, manufacturer, availableCount), availableToner(availableToner) {}

    ~LaserPrinter() override {}

    void print(int pages) override { //InkJetPrinter와 동일한 방법으로 코딩 및 제거
        if (availableCount >= pages && availableToner >= pages) {
            printedCount += pages;
            availableCount -= pages;
            availableToner -= pages;
            std::cout << "인쇄했습니다.₩n";
        }
        else {
            std::cout << "용지가 부족하거나 토너가 부족합니다.₩n";
        }
    }

    void show() const override {
        std::cout << model << ", " << manufacturer << ", 남은 종이 " << availableCount << "장, 남은 토너 " << availableToner << "₩n";
    }
};
```

3. 어떤 프린터를 사용할 지 또, 그 프린터를 사용하면 얼마나 인쇄할 것인지에 대한 변수들을 입력 받고 그 변수들에 대응하는 결과 값들을 도출 시킨다.

```
int main() {
    InkJetPrinter inkJet("Officejet V40", "HP", 5, 10);
    LaserPrinter laser("SCX-6x45", "Samsung", 3, 20);

    while (true) { //사용을 끝낼 때 까지 무한 반복
        std::cout << "현재 작동중인 2 대의 프린터는 아래와 같다₩n";
        inkJet.show();
        laser.show();

        std::cout << "프린터(1:잉크젯, 2:레이저)와 매수 입력>> "; //어떤 프린터를 쓸지 구분하는 코드 + 얼마 복사할지의 코드
        int choice, pages;
        std::cin >> choice >> pages;

        if (choice == 1) {
            inkJet.print(pages);
        }
        else if (choice == 2) {
            laser.print(pages);
        }
        else { //만약 1,2중 프린터를 고르지 않거나 인쇄 매수가 초과될 때
            std::cout << "잘못된 선택입니다.₩n";
            continue;
        }

        inkJet.show();
        laser.show();

        std::cout << "계속 프린트 하시겠습니까(y/n)>> "; //종료나 재시작을 알리는 코드
        char cont;
        std::cin >> cont;
        if (cont == 'n' || cont == 'N') { //만약 'n'이나 'N'을 입력 받으면
            break; //코드가 꺼지고 소멸자를 실행한다.
        }
    }

    return 0;
}
```

문제 해결을 위해 사용된 아이디어 및 평가

이 문제는 코드의 상속과 다형성을 이용해 두 프린터를 생성하는 문제였다. 프린터를 다 구현했다 할지라도 결과를 도출 시킬 때, 겹치는 변수가 있어 어떤 변수를 나타내는건지 모호할 지도 모르는 코드를 virtual 상속을 통해 해결함으로써 코드가 더 간편해지는

결과를 낳았던 것 같다. 그리고 main() 함수에선 N을 눌러야만 코드를 종료시킬 수 있기에 한정적인 for 반복문보단 while 반복문을 사용해 영구적으로 사용가능 하다는 점에서 효율적이었다.

마지막으로 각 프린터에 대응하는 값을 if문으로 비교하고 그에 대한 결과값을 각 프린터에 정의해 main함수에선 show()나 print(pages)를 통해 짧게 도출시킬 수 있어서 좋았다.