문제 정의

프린터의 종류에 따라 모델명, 제조사, 인쇄 매수, 인쇄 종이 잔량 등을 관리하고, 잉크젯 프린터와 레이저 프린터 각각 잉크 잔량과 토너 잔량을 따로 관리하는 프로그램을 작성해야 합니다. 프린터는 인쇄 요청 시, 페이지 수에 따라 용지와 잉크 또는 토너를 소비하며, 잔량이 부족할 경우 인쇄를 중단합니다.

문제 해결 방법

1. `Printer` 클래스는 기본 프린터 속성과 인쇄 기능을 가지며, 이를 상속받아 `InkJetPrinter`와 `LaserPrinter` 클래스를 각각 설계합니다.

2. 각 클래스에 잔량을 확인하는 함수(`isValidPrint`, `isValidInkJetPrint`, `isValidLaserPrint`)를 구현하여 잔량이 충분할 때만 인쇄를 수행하도록 합니다.

3. 기본 프린터 속성은 부모 클래스에서 관리하고, 잉크와 토너 잔량은 자식 클래스에서 개별적으로 관리하여 코드 중복을 줄입니다.

아이디어 평가

구현된 코드에서 각 클래스가 프린터의 잔량 확인과 인쇄 기능을 잘 수행하며, 잉크와 토너가 부족할 때와 용지가 부족할 때의 메시지도 정확히 출력됩니다. 사용자가 입력한 매수에 따라 인쇄 가능 여부가 검토됩니다.

문제를 해결한 키 아이디어 또는 알고리즘 설명

프린터의 기본 기능을 추상화한 `Printer` 클래스를 설계하고, 이를 상속받아 각각의 특성을 가진 `InkJetPrinter`와 `LaserPrinter` 클래스를 구현한 것이 핵심 아이디어입니다. 상속을 통해 코드 중복을 최소화하고, 각 프린터의 고유 기능을 개별적으로 처리함으로써 프로그램의 확장성을 높였습니다.

이제 각 코드에 주석을 통해 설명을 이어가겠습니다.

Pinter.h 코드

```
#ifndef PRINTER_H
#define PRINTER_H
#include <iostream>
#include <string>
using namespace std;
// Printer 클래스: 기본 프린터 속성 및 기능 정의
class Printer {
    string model;               // 프린터 모델명
    string manufacturer;      // 제조사
    int printedCount;          // 인쇄된 매수
    int availableCount;       // 사용 가능한 용지 수
protected:
    // 생성자: 모델명, 제조사, 용지 수를 설정하고 초기 인쇄 매수는 0으로 설정
    Printer(string mo = "", string me = "", int a = 0);
    // 인쇄 유효성 검사: 요청한 매수만큼 용지가 있는지 확인
    bool isValidPrint(int pages);
    // 인쇄 기능: 유효성 검사를 통과하면 인쇄 매수 증가 및 용지 감소
    void print(int pages);
    // 프린터 정보 출력: 모델명, 제조사, 남은 용지 수 출력
    void showPrinter();
};
// InkJetPrinter 클래스: Printer 클래스를 상속받아 잉크젯 프린터의 추가 속성 및 기능 정의
```

```
class InkJetPrinter : public Printer {
    int availableInk;    // 잉크 잔량
public:
    // 생성자: 기본 프린터 정보 외에 잉크 잔량을 추가로 설정
    InkJetPrinter(string mo = "", string me = "", int a = 0, int i = 0);
    // 잉크 잔량 유효성 검사: 요청한 매수만큼 잉크가 있는지 확인
    bool isValidInkJetPrint(int pages);
    // 잉크젯 프린터 인쇄 기능: 용지와 잉크가 충분할 때만 인쇄 수행
    void printInkJet(int pages);
    // 잉크젯 프린터 정보 출력: Printer 클래스의 정보 외에 잉크 잔량도 출력
    void showInkJetPrinter();
};
// LaserPrinter 클래스: Printer 클래스를 상속받아 레이저 프린터의 추가 속성 및 기능 정의
class LaserPrinter : public Printer {
    int availableToner; // 토너 잔량
public:
    // 생성자: 기본 프린터 정보 외에 토너 잔량을 추가로 설정
    LaserPrinter(string mo = "", string me = "", int a = 0, int t = 0);
    // 토너 잔량 유효성 검사: 요청한 매수만큼 토너가 있는지 확인
    bool isValidLaserPrint(int pages);
    // 레이저 프린터 인쇄 기능: 용지와 토너가 충분할 때만 인쇄 수행
    void printLaser(int pages);
    // 레이저 프린터 정보 출력: Printer 클래스의 정보 외에 토너 잔량도 출력
    void showLaserPrinter();
```

```
};
#endif // PRINTER_H
```

Printer.cpp 코드

```
#include "Printer.h"
// Printer 클래스 생성자 정의: 모델명, 제조사, 용지 수를 초기화하고 인쇄 매수를 0으로 설정
Printer::Printer(string mo, string me, int a) : model(mo), manufacturer(me), availableCount(a), printedCount(0) {}
// 인쇄 유효성 검사 함수: 남은 용지 수가 요청된 매수 이상인지 확인
bool Printer::isValidPrint(int pages) {
    return availableCount >= pages;
}
// 인쇄 함수: 유효성 검사 통과 시 인쇄 매수 증가 및 남은 용지 수 감소
void Printer::print(int pages) {
    if (isValidPrint(pages)) {
        printedCount += pages;      // 인쇄 매수 증가
        availableCount -= pages;    // 남은 용지 감소
    } else {
        cout << "용지가 부족하여 프린트할 수 없습니다." << endl;
    }
}
// 프린터 정보 출력 함수: 모델명, 제조사, 남은 용지 수 출력
void Printer::showPrinter() {
    cout << model << ", " << manufacturer << ", 남은 종이 " << availableCount << "장";
```

```
}
// InkJetPrinter 클래스 생성자 정의: 프린터 속성 외에 잉크 잔량 초기화
InkJetPrinter::InkJetPrinter(string mo, string me, int a, int i) : Printer(mo, me, a),
availableInk(i) {}
// 잉크 잔량 유효성 검사 함수: 남은 잉크가 요청된 매수 이상인지 확인
bool InkJetPrinter::isValidInkJetPrint(int pages) {
    return availableInk >= pages;
}
// 잉크젯 인쇄 함수: 용지와 잉크가 충분할 때만 인쇄 수행
void InkJetPrinter::printInkJet(int pages) {
    if (isValidPrint(pages)) {                    // 용지 유효성 검사
        if (isValidInkJetPrint(pages)) {          // 잉크 유효성 검사
            print(pages);                         // 인쇄 수행
            availableInk -= pages;                // 남은 잉크 감소
            cout << "프린트하였습니다" << endl;
        } else {
            cout << "잉크가 부족하여 프린트할 수 없습니다." << endl;
        }
    } else {
        cout << "용지가 부족하여 프린트할 수 없습니다." << endl;
    }
}
// 잉크젯 프린터 정보 출력 함수: 기본 정보와 잉크 잔량도 출력
void InkJetPrinter::showInkJetPrinter() {
    showPrinter();
```

```
    cout << ", 남은 잉크 " << availableInk << endl;
}
// LaserPrinter 클래스 생성자 정의: 프린터 속성 외에 토너 잔량 초기화
LaserPrinter::LaserPrinter(string mo, string me, int a, int t) : Printer(mo, me, a),
availableToner(t) {}
// 토너 잔량 유효성 검사 함수: 남은 토너가 요청된 매수 이상인지 확인
bool LaserPrinter::isValidLaserPrint(int pages) {
    return availableToner >= pages;
}
// 레이저 인쇄 함수: 용지와 토너가 충분할 때만 인쇄 수행
void LaserPrinter::printLaser(int pages) {
    if (isValidPrint(pages)) {                    // 용지 유효성 검사
        if (isValidLaserPrint(pages)) {          // 토너 유효성 검사
            print(pages);                          // 인쇄 수행
            availableToner -= pages;               // 남은 토너 감소
            cout << "프린트하였습니다" << endl;
        } else {
            cout << "토너가 부족하여 프린트할 수 없습니다." << endl;
        }
    } else {
        cout << "용지가 부족하여 프린트할 수 없습니다." << endl;
    }
}
// 레이저 프린터 정보 출력 함수: 기본 정보와 토너 잔량도 출력
void LaserPrinter::showLaserPrinter() {
```

```
    showPrinter();
    cout << ", 남은 토너 " << availableToner << endl;
}
```

Main.cpp 코드

```
#include "Printer.h"
int main() {
    // 잉크젯 및 레이저 프린터 객체 초기화
    InkJetPrinter ink("Officejet V40", "HP", 5, 10);
    LaserPrinter laser("SCX-6x45", "삼성전자", 3, 20);
    // 현재 작동 중인 프린터 정보 출력
    cout << "현재 작동중인 2대의 프린터는 아래와 같다" << endl;
    cout << "잉크젯 : ";
    ink.showInkJetPrinter();
    cout << "레이저 : ";
    laser.showLaserPrinter();
    int printer, paper; // 사용자 입력용 변수
    char ch;                // 계속 여부 확인용 변수
    while (true) {
        // 사용자로부터 프린터 종류와 인쇄 매수를 입력받음
        cout << endl << "프린터(1:잉크젯, 2:레이저)와 매수 입력>>";
        cin >> printer >> paper;
        // 선택한 프린터에 따라 인쇄 기능 호출
        switch (printer) {
```

```
            case 1:
                ink.printInkJet(paper);
                break;
            case 2:
                laser.printLaser(paper);
                break;
            default:
                break;
            }
            // 각 프린터의 상태를 출력
            ink.showInkJetPrinter();
            laser.showLaserPrinter();
            // 인쇄 작업을 계속할지 여부를 사용자에게 묻고, n을 입력하면 종료
            cout << "계속 프린트 하시겠습니까(y/n)>>";
            cin >> ch;
            if (ch == 'n') break;
        }
        return 0;
}
```