객체지향 프로그래밍 4차 과제

문제

다음 그림과 같은 상속 구조를 갖는 클래스를 설계한다.
모든 프린터는 모델명(model). 제조사(manufacturer), 인쇄 매수 (printeCount), 인쇄 종이 잔량(availableCount)을 나타내는 정보와 print(int pages) 멤버 함수를 가지며, print()가 호출할 때마다 pages 매의 용지를 사용한다.
잉크젯 프린터는 잉크 잔량(availableInk) 정보와 printInkJet (int pages) 멤버 함수를 추가적으로 가지며, 레이저 프린터는 토너 잔량(availableToner) 정보와 역시 printLaser(int pages) 멤버 함수를 추가적으로 가진다. 각 클래스에 적절한 접근 지정으로 멤버 변수와 함수, 생성자, 소멸자를 작성하고, 다음과 같이 실행되도록 전체 프로그램을 완성하라. 잉크젯 프린터 객체와 레이저 프린터 객체를 각각 하나만 동적 생성하여 시작한다.

소스 수행 결과 화면 캡쳐

```
Officejet V40 , HP , 남은 종이 5장
남은 잉크 10
SCX-6X45 , 삼성전자 , 남은 종이 3장
남은 토너 20
프린터(1: 잉크젯, 2: 레이저)와 매수 입력>> 1 4
프린트하였습니다.
Officejet V40 , HP , 남은 종이 1장
남은 잉크 6
SCX-6X45 , 삼성전자 , 남은 종이 3장
남은 토너 20
계속 프린트 하시겠습니까(y/n)>> y
Officejet V40 , HP , 남은 종이 1장
남은 잉크 6
SCX-6X45 , 삼성전자 , 남은 종이 3장
남은 토너 20
프린터(1: 잉크젯, 2: 레이저)와 매수 입력>> 2 10
용지가 부족하여 프린트할 수 없습니다.
```

소스코드

```
#ifndef INKJETPRINTER_H
#define INKJETPRINTER_H

#include "Printer.h"

class InkJetPrinter : public Printer {
private:
    int availableInk;  // 잉크 잔량

public:
    InkJetPrinter(string model, string manufacturer, int avai
    void printInkJet(int pages);
    void printStatus() const override;
};

#endif // INKJETPRINTER_H
```

```
#include <iostream>
#include "InkJetPrinter.h"
using namespace std;

InkJetPrinter::InkJetPrinter(string model, string manufacture
    : Printer(model, manufacturer, availableCount), available

void InkJetPrinter::printInkJet(int pages) {
    if (pages > availableCount) {
        cout << "용지가 부족하여 프린트할 수 없습니다." << endl;
    } else if (pages > availableInk) {
        cout << "잉크가 부족하여 프린트할 수 없습니다." << endl;
    } else {
        availableCount -= pages;
        availableInk -= pages;
        cout << "프린트하였습니다." << endl;
    }
}

void InkJetPrinter::printStatus() const {
```

```
    Printer::printStatus();
    cout << "남은 잉크 " << availableInk << endl;
}
```

```
#ifndef LASERPRINTER_H
#define LASERPRINTER_H

#include "Printer.h"

class LaserPrinter : public Printer {
private:
    int availableToner;  // 토너 잔량

public:
    LaserPrinter(string model, string manufacturer, int avail
    void printLaser(int pages);
    void printStatus() const override;
};

#endif // LASERPRINTER_H
```

```
#include <iostream>
#include "LaserPrinter.h"
using namespace std;

LaserPrinter::LaserPrinter(string model, string manufacturer,
    : Printer(model, manufacturer, availableCount), available

void LaserPrinter::printLaser(int pages) {
    if (pages > availableCount) {
        cout << "용지가 부족하여 프린트할 수 없습니다." << endl;
    } else if (pages > availableToner) {
        cout << "토너가 부족하여 프린트할 수 없습니다." << endl;
    } else {
        availableCount -= pages;
        availableToner -= pages;
```

```
        cout << "프린트하였습니다." << endl;
    }
}

void LaserPrinter::printStatus() const {
    Printer::printStatus();
    cout << "남은 토너 " << availableToner << endl;
}
```

```
#ifndef PRINTER_H
#define PRINTER_H

#include <string>
using namespace std;

// 기본 Printer 클래스
class Printer {
protected:
    string model;
    string manufacturer;
    int printCount;   // 인쇄 매수
    int availableCount; // 인쇄 종이 잔량

public:
    Printer(string model, string manufacturer, int availableC
    virtual ~Printer();
    virtual void print(int pages);
    virtual void printStatus() const;
};

#endif // PRINTER_H
```

```
#include <iostream>
#include "Printer.h"
using namespace std;

Printer::Printer(string model, string manufacturer, int avail
```

```
    : model(model), manufacturer(manufacturer), availableCoun

Printer::~Printer() {}

void Printer::print(int pages) {
    if (pages > availableCount) {
        cout << "용지가 부족하여 프린트할 수 없습니다." << endl;
    } else {
        availableCount -= pages;
        printCount += pages;
        cout << "프린트하였습니다." << endl;
    }
}

void Printer::printStatus() const {
    cout << model << " , " << manufacturer << " , 남은 종이 " <
}
```

```
#include <iostream>
#include "InkJetPrinter.h"
#include "LaserPrinter.h"
using namespace std;

int main() {
    InkJetPrinter* inkJet = new InkJetPrinter("Officejet V40"
    LaserPrinter* laser = new LaserPrinter("SCX-6X45", "삼성전

    char continuePrint;
    int printerChoice, pages;

    do {
        // 프린터 상태 출력
        inkJet->printStatus();
        laser->printStatus();

        // 프린터 선택 및 매수 입력
        cout << "프린터(1: 잉크젯, 2: 레이저)와 매수 입력>> ";
```

```
        cin >> printerChoice >> pages;

        // 프린트 수행
        if (printerChoice == 1) {
            inkJet->printInkJet(pages); // 잉크젯 프린터에서 인쇄
        } else if (printerChoice == 2) {
            laser->printLaser(pages); // 레이저 프린터에서 인쇄
        }

        // 프린트 후 상태 바로 출력
        inkJet->printStatus();
        laser->printStatus();

        // 계속할지 여부 묻기
        cout << "계속 프린트 하시겠습니까(y/n)>> ";
        cin >> continuePrint;

    } while (continuePrint == 'y' || continuePrint == 'Y');

    // 동적 할당 메모리 해제
    delete inkJet;
    delete laser;

    return 0;
}
```

문제 정의

프린터 시스템을 구현하는 프로그램.

두 종류의 프린터, 즉 잉크젯 프린터와 레이저 프린터가 있고, 각각의 프린터는 모델명, 제조사, 남은 종이 수와 같은 공통 속성을 가지며, 프린트 시 남은 종이 수를 차감합니다.

잉크젯 프린터는 잉크 잔량을 관리하고, 레이저 프린터는 토너 잔량을 관리하여 각각의 자원이 부족할 경우 프린트를 진행하지 못하도록 합니다.

사용자는 특정 프린터와 프린트할 매수를 입력하여 인쇄를 요청할 수 있으며, 프로그램은 프린트 후 상태를 출력하고, 계속할지 여부를 묻습니다.

문제 해결 방법

- **클래스 설계**:
  - 공통 속성을 가지는 `Printer` 클래스를 기본 클래스로 정의합니다.
  - `Printer` 클래스를 상속하여 `InkJetPrinter`와 `LaserPrinter` 클래스를 구현합니다.
- **멤버 변수**:
  - `Printer` 클래스에는 `model`, `manufacturer`, `printCount`, `availableCount` 변수를 두어 모델명, 제조사, 남은 종이 수 등을 관리합니다.
  - `InkJetPrinter` 클래스에는 `availableInk` 변수를 추가하여 잉크 잔량을 관리합니다.
  - `LaserPrinter` 클래스에는 `availableToner` 변수를 추가하여 토너 잔량을 관리합니다.
- **멤버 함수**:
  - `Printer` 클래스의 `print()` 함수는 종이 잔량을 확인하고, 부족할 경우 경고 메시지를 출력합니다.
  - `InkJetPrinter` 클래스의 `printInkJet()` 함수는 종이와 잉크 잔량을 확인하여 둘 중 하나라도 부족할 경우 경고 메시지를 출력합니다.
  - `LaserPrinter` 클래스의 `printLaser()` 함수는 종이와 토너 잔량을 확인하여 둘 중 하나라도 부족할 경우 경고 메시지를 출력합니다.

알고리즘 설명

- **프린트 요청**:
  - 사용자가 프린트할 프린터(잉크젯 또는 레이저)와 매수를 입력합니다.
- **프린트 가능 여부 확인**:
  - `printInkJet()` 또는 `printLaser()` 함수를 통해 종이와 잉크/토너 잔량을 확인하고, 부족할 경우 "프린트할 수 없습니다" 메시지를 출력합니다.
  - 잉크와 토너가 충분하면 프린트를 진행하고, 종이와 잉크/토너 잔량을 차감합니다.
- **상태 출력**:
  - 프린트가 완료되거나 실패한 후, 잔여 종이 및 잉크/토너 상태를 출력합니다.
- **계속 여부 확인**:
  - 사용자가 `y` 또는 `n`을 입력하여 계속할지 여부를 결정합니다.

흐름 설명

- **프린터 객체 생성**:
  - `InkJetPrinter`와 `LaserPrinter` 객체를 동적으로 생성합니다.
- **프린트 요청 입력**:
  - 사용자에게 프린터(1: 잉크젯, 2: 레이저)와 프린트할 매수를 입력받습니다.
- **프린트 작업 수행**:
  - 선택한 프린터의 `printInkJet()` 또는 `printLaser()` 함수를 호출하여 프린트 작업을 시도합니다.
- **상태 출력**:
  - 프린트 후 즉시 두 프린터의 남은 종이, 잉크/토너 상태를 출력합니다.
- **계속 여부 확인**:
  - 사용자에게 계속할지 묻고, `y`를 입력하면 다시 프린트 요청을 받고, `n`을 입력하면 프로그램을 종료합니다.
- **메모리 해제**:
  - 프로그램 종료 시 동적으로 생성한 `InkJetPrinter`와 `LaserPrinter` 객체를 해제합니다.