

소스 구현 설명

202304178 구서연

1. 문제 정의 :

상속에 대하여 정확하여 코드의 중복을 줄이고, 그에 따라 멤버 변수와 멤버 함수에 적절한 접근 지정자를 매칭시킬 수 있다. 동적생성을 하고 포인터로 활용한 후 반납할 수 있다.

2. 문제 해결 방법 :

- 상속관계를 파악

기본 클래스(base class) - `Printer`

파생 클래스(derived class) - `InkJetPrinter`, `LaserPrinter`

프린터 공통 속성끼리 묶어서 모든 `Printer`의 특징은 클래스 `Printer`로 구현한다, 개별 프린터의 세부적인 특징은 각자 구현한다. 클래스 `Printer`의 경우, 멤버 변수들은 `private`으로 하고, 멤버 함수들은 자식 클래스도 접근이 가능하도록 `protected`로 만들었다.

```
5  class Printer {
6      string model;
7      string manufacturer;
8      int printedCount;
9      int availableCount;
10     protected:
11         Printer(string mo, string me, int a);
12         ~Printer();
13         bool isValidPrint(int pages);
14         void print(int pages);
15         void showPrinter();
16     };
17
```

- 상속과 확장

잉크젯 프린터와 레이저 프린터가 기본 프린터와 다르게 갖고있는 세부적인 특징은, 잉크젯 프린터는 잉크 잔량(`availableInk`) 멤버 변수와 `printInkJet(int Pages)` 멤버 함수를 추가적으로 가진다. 레이저 프린터는 토너 잔량(`availableToner`) 멤버변수와 `printLaser(int Pages)` 멤버 함수를 추가적으로 가진다. 추가로 각각 생성자와 소멸자를 선언해 주고, 프린트가 가능한 것인지를 판단하는 `bool` 함수 `isValidPrint(int pages)`, `isValidInkJetPrint (int pages)`, `isValidLaserPrint (int pages)`를 만들어준다.

```
18 class InkJetPrinter : public Printer {
19     int availableInk;
20     public:
21         InkJetPrinter(string mo, string me, int a, int i);
22         ~InkJetPrinter();
23         bool isValidInkJetPrint(int pages);
24         void printInkJet(int pages);
25         void showInkJetPrinter();
26     };
27
```

```

28 class LaserPrinter : public Printer {
29     int availableToner;
30 public:
31     LaserPrinter(string mo, string me, int a, int t);
32     ~LaserPrinter();
33     bool isValidLaserPrint(int pages);
34     void printLaser(int pages);
35     void showLaserPrinter();
36 };

```

- 생성자 & 소멸자

```

Printer::Printer(string mo = "", string me = "", int a = 0) {
    this->model = mo;
    this->manufacturer = me;
    this->availableCount = a;
    this->printedCount = 0;
}
Printer::~Printer(){}

```

Printer 클래스의 생성자에서는 프린터의 기본 속성(모델명, 제조사, 남은 용지 수, 인쇄된 페이지 수)을 초기화한다. this를 사용하여 이 객체의 멤버 변수값을 매개변수로 전달된 값으로 초기화를 한다.

```

LaserPrinter::LaserPrinter(string mo = "", string me = "", int a = 0, int t = 0) : Printer(mo, me, a) {
    this->availableToner = t;
}
LaserPrinter::~~LaserPrinter() {};

```

이 생성자는 LaserPrinter 객체가 생성될 때 프린터의 모델명, 제조사, 남은 용지 수와 토너 잔량을 초기화한다. 부모 클래스 Printer의 생성자를 호출하여 공통 속성을 정하고 난 후, LaserPrinter 클래스 고유의 availableToner 속성을 t로 초기화한다.

```

InkJetPrinter::InkJetPrinter(string mo = "", string me = "", int a = 0, int i = 0) : Printer(mo, me, a) {
    this->availableInk = i;
}
InkJetPrinter::~~InkJetPrinter() {};

```

InkJetPrinter 클래스는 Printer 클래스를 상속받으며, 추가적으로 잉크젯 프린터에 필요한 availableInk 속성을 i로 초기화한다. 이때, 디폴트값은 0이다.

- 동적 생성 & 반납

InkJetPrinter와 LaserPrinter 객체를 동적으로 생성하고, 활용한 이후 객체를 삭제한다.

```

InkJetPrinter* ink = new InkJetPrinter("Officejet V40", "HP", 5, 10);
LaserPrinter * laser = new LaserPrinter("SCX-6x45", "삼성전자", 3, 20);

```

InkJetPrinter 객체 ink와 LaserPrinter 객체 laser를 동적으로 생성한다. ink와 laser는 InkJetPrinter와 LaserPrinter 객체 자체가 아니라, InkJetPrinter와 LaserPrinter 타입의 포인터이다.

ink와 laser는 객체가 아닌 객체의 주소를 담고 있는 포인터이므로, 멤버 함수에 접근할 때

화살표(->) 연산자를 사용하여 포인터가 가리키는 객체의 멤버에 접근한다.

```
ink->showInkJetPrinter();  
laser->showLaserPrinter();
```

```
delete ink;  
delete laser;
```

delete를 사용하여 공간을 반납함으로써 메모리 누수를 방지하고 프로그램의 메모리 효율을 유지할 수 있다.

- 반복문

```
while (true) {  
  
    cout << endl << "프린터(1:잉크젯, 2:레이저)와 매수 입력>>";  
    cin >> printer >> paper;  
    switch (printer) {  
        case 1: ink->printInkJet(paper); break;  
        case 2: laser->printLaser(paper); break;  
        default: break;  
    }  
    ink->showInkJetPrinter();  
    laser->showLaserPrinter();  
  
    cout << "계속 프린트 하시겠습니까(y/n)>>";  
    cin >> ch;  
    if (ch == 'n') break;  
    while (ch != 'y')  
    {  
        cout << "계속 프린트 하시겠습니까(y/n)>>";  
        cin >> ch;  
        if (ch == 'n') break;  
    }  
}
```

반복문을 통해 사용자로부터 명령어를 입력받고 n이 입력될 때까지 계속 반복한다. switch 문을 통해 잉크젯 프린터와 레이저 프린터 중 실행할 것을 택한다.

3. 아이디어 평가 :

아이디어 1 :

상속을 활용해 코드의 중복을 줄이는 것이 핵심

-> Printer 클래스에서 공통적인 기능을 정의한 후, InkJetPrinter와 LaserPrinter에서 각각의 세부적인 기능을 구현

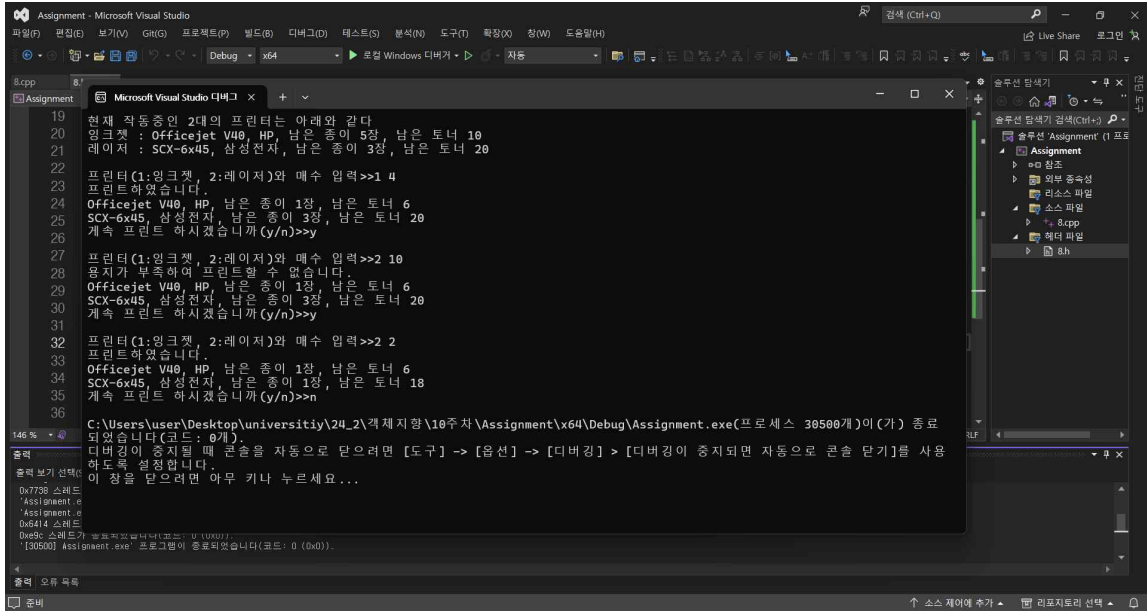
아이디어 2 :

자식 클래스의 생성자에서 부모 클래스의 생성자 호출

-> 부모 클래스 Printer의 생성자를 호출하여 공통 속성을 정하고 난 후, 자기 클래스 고유의 속성을 매개변수로 주어진 값으로 초기화한다. 안 주어졌다면 디폴트값으로 초기화한다.

총정리 : 프린터의 공통 기능을 부모 클래스에서 제공함으로써 코드 중복을 최소화했으며, 개별 프린터 특성은 각각의 클래스에서 정의하여 유연성을 높였습니다.

- 실행 화면



```
19 현재 작동 중인 2대의 프린터는 아래와 같다
20 잉크젯 : Officejet V40, HP, 남은 종이 5장, 남은 토너 10
21 레이저 : SCX-6x45, 삼성전자, 남은 종이 3장, 남은 토너 20
22 프린터 (1:잉크젯, 2:레이저)와 매수 입력>>1 4
23 프린트 하였습니다.
24 Officejet V40, HP, 남은 종이 1장, 남은 토너 6
25 SCX-6x45, 삼성전자, 남은 종이 3장, 남은 토너 20
26 계속 프린트 하시겠습니까 (y/n)>>y
27 프린터 (1:잉크젯, 2:레이저)와 매수 입력>>2 10
28 용지가 부족하여 프린트할 수 없습니다.
29 Officejet V40, HP, 남은 종이 1장, 남은 토너 6
30 SCX-6x45, 삼성전자, 남은 종이 1장, 남은 토너 18
31 계속 프린트 하시겠습니까 (y/n)>>y
32 프린터 (1:잉크젯, 2:레이저)와 매수 입력>>2 2
33 프린트 하였습니다.
34 Officejet V40, HP, 남은 종이 1장, 남은 토너 6
35 SCX-6x45, 삼성전자, 남은 종이 1장, 남은 토너 18
36 계속 프린트 하시겠습니까 (y/n)>>n
C:\Users\user\Desktop\university\24_2\객체지향\10주차\Assignment\x64\Debug\Assignment.exe(프로세스 30500개)이 (가) 종료
되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
D:\7758 소헤드
'Assignment.r
D:\6414 소헤드
D:\950 소헤드(가) 실행된 프로그램이 종료되었습니다(코드: 0 (0x0)).
[30500] Assignment.exe' 프로그램이 종료되었습니다(코드: 0 (0x0)).
```