소스 파일 구현 설명

202304148 조하린

다음과 같은 상속 구조를 갖는 클래스를 설계한다.
프린터 ㅡ상속--> 잉크젯 프린터, 레이저 프린터
모든 프린터는 모델명(model), 제조사(manufacturer), 인쇄 매수 (printedCount), 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 추가로 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.

파일에 대한 설명

PrinterInheritance 프로젝트

(1) printer.h - 클래스 선언, 함수 선언
    멤버 변수: model, manufacturer, printedCount, availableCount,
        availableInk, availableToner
    멤버 함수: 생성자 Print(), print(int pages), showPrinter(),
        생성자 InkJetPrint(), printInkJet(int pages), showInkJetPrinter(),
        생성자 LaserPrint(), printLaser(int pages), showLaserPrinter()
(2) printer.cpp - 클래스의 멤버 함수 구현
    멤버 함수: 생성자 Print(), print(int pages), showPrinter(),
        생성자 InkJetPrint(), printInkJet(int pages), showInkJetPrinter(),
        생성자 LaserPrint(), printLaser(int pages), showLaserPrinter()
(3) main.cpp
    프로그램 시작 및 실행
    객체 생성: 잉크젯 프린터 객체 ink, 레이저 프린터 객체 laser
    함수 호출 및 결과 출력

문제 정의

프린터기 2대가 있다. 각 프린터는 모델명, 제조사, 인쇄 매수, 인쇄 종이 잔량, 잉크 잔량, 토너 잔량에 대한 정보를 가지고 있다. 키보드로부터 사용할 프린터 종류와 인쇄할 페이지 수를 입력받는다. 프로그램은 프린트가 가능한지를 판단하고, 잔량이 부족하면 에러 메시지를 출력하고, 프린트가 가능하다면 성공 메시지 출력과 함께 잔량을 업데이트한다. 그리고 프린트를 이어서 할 것인지 여부를 입력받아 프로그램을 반복하거나 종료한다.

1. 클래스와 구성요소

(1) 클래스

- 기본 클래스 Printer
- 파생 클래스 InkJetPrinter, LaserPrinter

(2) 구성요소

- 변수
  - string model: 모델명
  - string manufacturer: 제조사
  - int printedCount: 인쇄 매수
  - int availableCount: 인쇄 종이 잔량
  - int availableInk: 잉크 잔량
  - int availableLaser: 레이저 잔량
  - int printer: 선택할 프린터
  - int pages: 인쇄할 페이지 수
- 함수
  - Printer 생성자
  - void print(int pages) 함수
  - void showPrinter() 함수
  - InkJetPrinter 생성자
  - void printInkJet(int pages) 함수
  - void showInkJetPrinter() 함수
  - LaserPrinter 생성자
  - void printLaser(int pages) 함수
  - void showLaserPrinter() 함수

2. 입력 : 키보드로부터 사용할 프린터 종류와 인쇄할 페이지 수를 입력받는다.

3. 반복문과 조건문
   1) while 반복문을 통해 전체 프로그램을 계속해서 진행시킨다.
   2) if-else 조건문: 잉크젯 프린터, 레이저 프린터 중 사용할 프린터 선택.
   3) if-else 조건문: 용지, 잉크, 토너를 잔량과 비교하여 인쇄 가능 여부를 확인.
   4) if-else 조건문: 인쇄를 계속 진행할 것인지 여부 선택.

4. 상속 - 접근 지정
   - 기본 클래스 Printer
   - 파생 클래스 InkJetPrinter, LaserPrinter
   - 접근 지정자 protected, public

5. 객체 동적 할당
   - 잉크젯 프린터 객체와 레이저 프린터 객체를 각각 하나만 동적 생성하여 시작

6. 생성자, 소멸자
   - 생성자: 각 클래스의 멤버 변수 초기화
   - 소멸자: 동적 할당된 객체 메모리 해제

문제 해결 방법 (아이디어)

1. 클래스 - 변수, 함수

(1) 기본 클래스 Printer
- 멤버 변수
  - string model: 모델명
  - string manufacturer: 제조사
  - int printedCount: 인쇄 매수
  - int availableCount: 인쇄 종이 잔량
- 멤버 함수
  - Printer 생성자: 프린터의 멤버 변수 초기화
  - void print(int pages) 함수
    인쇄 종이 잔량과 인쇄 매수를 비교해 출력 가능 여부를 판단하고,
    출력이 가능하다면 출력 후 printedCount, availableCount 업데이트
  - void showPrinter() 함수: 프린터 정보 및 상태 출력

(2) 파생 클래스 InkJetPrinter
- 멤버 변수
  - int avilableInk: 잉크 잔량
- 멤버 함수
  - InkJetPrinter 생성자: 잉크젯 프린터의 멤버 변수 초기화
  - void printInkJet(int pages) 함수
    잉크젯 프린터의 남아있는 용지 수와 인쇄 매수를 비교해 출력이 가능한 페이지 수를 확인하고, 남아있는 잉크 양과 인쇄 매수당 필요한 잉크 양을 비교하여 최종적으로 잉크젯 프린터의 출력 가능 여부를 판단하고, 잉크가 부족하면 출력이 가능하다면 출력 후 availableInk 업데이트
  - void showInkJetPrinter() 함수: 잉크젯 프린터 상태를 출력

(3) 파생 클래스 LaserPrinter
- 멤버 변수
  - int availableLaser: 레이저 잔량
- 멤버 함수
  - LaserPrinter 생성자: 레이저 프린터의 멤버 변수 초기화
  - void printLaser(int pages) 함수
    남아있는 토너 양과 인쇄 매수당 필요한 토너 양을 비교하여 출력 가능 여부 판단하고, 출력이 가능하다면 출력 후 availableLaser 업데이트
  - void showLaserPrinter() 함수: 레이저 프린터 상태를 출력

2. 상속 - 접근 지정자
   - 기본 클래스 Printer로부터 상속받은 InkJetPrinter, LaserPrinter 클래스
   - 상속을 통해 printer 클래스의 멤버들을 파생 클래스에서 재사용하게 하는데, 파생 클래스가 기본 클래스를 protected로 상속받아 기본 클래스의 public, protected 멤버들이 모두 protected 접근 지정으로 변경되어 사용한다.

3. 키보드 입력
   - 입력 스트림 객체 cin을 통해 사용할 프린터와 인쇄 매수를 입력받아 printer와 pages 변수에 저장한다.

4. 반복문과 조건문
   1) while 반복문을 통해 전체 프로그램을 계속 실행한다.
      사용자가 종료를 선택할 때까지 계속 실행된다.
   2) if-else 조건문을 통해 선택한 프린터가 잉크젯 프린터인지 레이저 프린터인지 확인하여 각각에 맞는 것을 실행한다.
   3) if-else 조건문을 통해 용지, 잉크, 토너를 잔량과 비교해 출력 가능 여부 확인.
   4) if-else 조건문을 통해 프린트를 계속 진행할 것인지 여부를 선택받고,
      프로그램을 반복하거나 종료한다.

5. continute문과 break문
   - continue문: 반복문에서 현재 반복 건너뛰기
   - break문: 반복문제서 현재 반복문을 종료

6. this 포인터
   - 현재 객체의 변수와 매개변수 구분
   - this->model, this->manufacturer, this->printedCount, this->availableCount, this->availableInk, this->availableToner

7. 객체 동적 할당
   - new 연산자를 사용하여 InkJetPrinter와 LaserPrinter 객체를 동적으로 생성

8. 소멸자
   - new 연산자를 사용하여 InkJetPrinter와 LaserPrinter 객체를 동적으로 생성하였으므로 delete 연산자를 사용해 메모리에서 해제되도록 한다.

9. pages * 0.5 - 부동소수점 연산

문제 해결 키 (아이디어)

공통으로 가지는 정보는 기본 클래스인 Printer에 정의하고, 상속을 통해 기본 클래스의 멤버들을 파생 클래스에 상속해준다. 파생 클래스 InkJetPrinter, LaserPrinter는 각 클래스에서 추가로 가지는 정보만 정의한다.

(1) 기본 클래스 Printer
    공통 변수: model, manufacturer, printedCount, availableCount
    공통 함수: Print() 생성자, print(int pages), showPrinter()

(2) 파생 클래스 InkJetPrinter
    추가 변수: availableInk
    추가 함수: InkJetPrint() 생성자, printInkJet(int pages), showInkJetPrinter()

(3) 파생 클래스 LaserPrinter
    추가 변수: availableLaser
    추가 함수: LaserPrint() 생성자, printLaser(int pages), showLaserPrinter()

프로그램 순서

1. 프로그램 시작
2. 프린터 2대의 정보(모델명, 제조자, 인쇄 종이 잔량, 잉크/토너 잔량) 제공
3. 사용할 프린터기와 인쇄 매수 입력
4. 출력 가능 여부 판단 - 용지, 잉크, 토너 잔량
5. 출력 및 잔량 업데이트
6. 추가 인쇄 여부 선택
   yes - 3번으로 돌아감
   no - 7번으로 진행
7. 프로그램 종료

아이디어 평가

1. 상속
   - 기본 클래스 Printer
     모든 프린터가 공통으로 가지는 정보 모델명(model), 제조사(manufacturer), 인쇄 매수 (printedCount), 인쇄 종이 잔량(availableCount)과 멤버 함수인 print(int pages), showPrinter()를 Printer 클래스에 정의했다.
   - 파생 클래스 InkJetPrinter
     잉크젯 프린터는 잉크를 사용하여 인쇄하는 프린터이다. 공통으로 가지는 기본 정보들은 Printer 클래스에서 멤버들을 상속받아 사용하고, 추가로 잉크 잔량 (availableInk) 변수와 printInkJet(int pages), showInkJetPrinter() 멤버 함수를 InkJetPrinter 클래스에 정의했다.
   - 파생 클래스 LaserPrinter
     레이저 프린터는 토너를 사용하여 인쇄하는 프린터이다. 공통으로 가지는 기본 정보들은 Printer 클래스에서 멤버들을 상속받아 사용하고, 추가로 토너 잔량 (availableToner) 변수와 printLaser(int pages), showLaserPrinter() 멤버 함수를 LaserPrinter 클래스에 정의했다.
   - 공통 정보를 기본 클래스인 Printer에 정의하고, 파생 클래스 InkJetPrinter, LaserPrinter가 이를 상속받는다. 파생 클래스 InkJetPrinter, LaserPrinter는 각 클래스에서 추가로 가지는 정보만 정의한다. 따라서 코드의 중복을 줄일 수 있고, 재사용성에 유리하다.

2. 접근 지정자
   - 클래스 간의 상속을 통해 기본 클래스의 기능을 파생 클래스에서 재사용한다.
   - 기본 클래스의 멤버들이 상속을 통해 파생 클래스의 멤버로 확장될 때, 기본 클래스 멤버의 접근 지정은 상속 조건에 따라 달라진다.
   - protected 상속
     기본 클래스를 protected로 상속받으면, 기본 클래스의 protected, public 멤버들은 모두 protected 접근 지정으로 변경되어 파생 클래스에 상속 확장됨.

3. this 포인터
   this를 사용하지 않다면 클래스의 멤버 변수와 메서드의 매개 변수 이름이 동일해서 두 개의 변수를 만들어야 하는데, this를 사용함으로써 코드의 가독성을 높이고 변수들의 혼동을 줄일 수 있었다.

4. 객체 동적 생성
   new 연산자를 사용하여 객체를 동적 생성 함으로써 메모리를 할당할 수 있다.

5. 소멸자와 delete
   동적으로 할당된 객체는 프로그램 종료 시 반드시 메모리 해제가 필요하기 때문에 소멸자를 명시적으로 정의해야 한다. 하지만 main 함수에서 객체만 동적 할당이 되었기 때문에 delete 사용만으로도 소멸자가 자동으로 생성된다. 소멸자는 delete 호출 시 자동으로 실행되어, 메모리를 해제하므로 delete만으로도 충분하다.

6. 소수점 계산
   - pages / 2 - 정수형 연산
     pages가 정수이기 때문에 정수 나누기로 처리가 되기 때문에 소수점 이하 값이 버려져 pages가 홀수일 경우 제대로 계산이 되지 않는다.
   - pages * 0.5 - 부동소수점 연산
     따라서 곱셈으로 처리해야 정확한 결과값을 얻을 수 있다.

7. 파일 분리
   파일 분리를 통해 각 파일의 역할에 맞게 나뉘어 코드의 가독성을 향상할 수 있었다. 또한 코드를 재사용할 수 있고, 각 클래스를 독립적으로 관리할 수 있게 되었으며 디버깅에 용이해졌다.