13-신은총, 신은근

정리노트 #3 (9/25)

① 코드를 보고, 질문 생각하기

```cpp
#include <iostream>
using namespace std;

class Circle {
public:
    int radius;
    double getArea();
};

double Circle::getArea() {
    return 3.14*radius*radius;
}

int main() {
    Circle donut;
    donut.radius = 1; // donut 객체의 반지름을 1로 설정
    double area = donut.getArea(); // donut 객체의 면적 알아내기
    cout << "donut 면적은 " << area << endl;

    Circle pizza;
    pizza.radius = 30; // pizza 객체의 반지름을 30으로 설정
    area = pizza.getArea(); // pizza 객체의 면적 알아내기
    cout << "pizza 면적은 " << area << endl;
}
```

Circle 선언부

Circle 구현부

```
donut 면적은 3.14
pizza 면적은 2826
```

* 대화

은호: 이 코드는 제일 내가 iostream 헤더를 포함하고, 표준 네임스페이스를 사용 한다.
은총: C#에서는 선언부, 구현부로 나누어져 있어 각각 맞는 변수를 선언해야 할 것 같다.
은호: 그럼결국 main 함수에서 선언부, 구현부의 메서드를 호출하네
은총: `::`은 무슨 의미 일까?
은호: Circle의 범위를 정하는 것 같은데... 근데 C#에서는 다 선언부 구현부를 다 나눠놨을까?
은총: 나중에 클래스를 따로 쓰기위해서 아닐까?
은호: 결국, 이 코드는 `Circle` 클래스 객체 `donut`을 생성 하고, 반지름 1로 설정 한 후 getArea 메서드를 호출하여 donut 객체의 넓이를 계산하고 출력하네..
은총: 맞네.. 그럼 비슷 하게 사각형 넓이도 구할 수있겠다.

* 의문점 해결.

① :: 의 의미 → 범위 지정 연산자
② 선언부, 구현부로 나눈이유 : 클래스를 다른 파일에서 활용하기 위해서이다.

---

② public 선언 개념.

```cpp
#include <iostream>
using namespace std;

class Rectangle { // Rectangle 클래스 선언부
public:
    int width;
    int height;
    int getArea(); // 면적을 계산하여 리턴하는 함수
};

int Rectangle::getArea() { // Rectangle 클래스 구현부
    return width*height;
}

int main() {
    Rectangle rect;
    rect.width = 3;
    rect.height = 5;
    cout << "사각형의 면적은 " << rect.getArea() << endl;
}
```

* 대화

은총: 이 코드를 보니까 width, height, 멤버함수 `getArea()` 모두 public으로 선언되어 있어
은호: 그렇네, 왜 public으로 선언되어 있을까?
은총: 아마도, 자바처럼 public 키워드는 클래스의 멤버를 외부에서 접근할 수 있도록 하는 것 같다.
은호: 아 그러면.. main 함수에서 멤버 변수에 접근 할당 할 수있고, rect.getArea()와 같이 멤버함수를 호출하여 원하는 정보를 얻을 수 있네
은총: 그러면 public 말고 다른 접근 지정자도 쓸 수 있겠네.
은호: public, private, protected 와 같은 접근 지정자를 써서 접근을 제어하는 방법도 알아봐야겠다.

* 결론

public으로 선언하면 외부코드가 해당 멤버에 직접 엑세스하고 조작할 수 있다.
단, 데이터 무결성 희생 할 수 있으므로 private 등 다른 접근 지정자를 써야 한다.

③ 왜? 위임생성을 하는가?

```
#include <iostream>
using namespace std;

class Circle {
public:
    int radius;
    Circle(); // 위임 생성자
    Circle(int r); // 타겟 생성자
    double getArea();
};

Circle::Circle() : Circle(1) { } // 위임 생성자

Circle::Circle(int r) { // 타겟 생성자
    radius = r;
    cout << "반지름 " << radius << " 원 생성" << endl;
}

double Circle::getArea() {
    return 3.14*radius*radius;
}
```

호출

호출. r에 1 전달

```
int main() {
    Circle donut; // 매개 변수 없는 생성자 호출
    double area = donut.getArea();
    cout << "donut 면적은 " << area << endl;

    Circle pizza(30); // 매개 변수 있는 생성자 호출
    area = pizza.getArea();
    cout << "pizza 면적은 " << area << endl;
}
```

```
반지름 1 원 생성
donut 면적은 3.14
반지름 30 원 생성
pizza 면적은 2826
```

* 대화

은둔: Circle() 생성자는 Circle(int r) 생성자를 위임하고 있어

은총: 이런 위임 생성자를 왜 사용하는 걸까?

은둔: 아마 중복된 코드를 방지하기 위해서 쓰지 않을까?

은총: 아, 그럼, 코드를 재사용하고 중복을 피할 수 있겠네

은둔: 위임 생성자를 쓰면 같은 클래스 내에서 다른 생성자를 호출해 초기화 작업을 공유할 수 있겠네

은총: 아 그러면 코드 중복을 줄일 수 있겠네

* 결론

위임생성자를 사용하는 이유는 코드중복을 방지하는데 유용한 기능이다.
클래스 내 초기화 코드 공유 한다.