객체프로그래밍 정리노트 #3

202204103 ict융합공학부 이승훈
202204023 ict융합공학부 김보민

[3-1] Circle 클래스의 객체 생성 및 활용

승훈 풀이)
ㄴ Circle 클래스 생성 (변수/함수 선언)
ㄴ Circle getArea() 함수 구현 (반환할 값)
ㄴ main 함수 구현 (출력에 필요한 값)

```
//보민 , ->승훈
#include <iostream>
using namespace std;

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

double getArea() { //getArea()가 Circle 클래스 안에 있으니까 클래스 이름을 써줘야 해
	return 3.14*radius*radius;
}

int main() {
	int donut; // 도넛의 반지름을 새로 하나 정해줘야 해, 이때 c++에서는 클래스의 객체를
		선언할 때 클래스 이름을 같이 써줘
	donut.radius = 1;

	 -> donut 객체의 면적 알아내려면 어떻게 해야 하지?
	// donut 객체의 면적 알아내기 위해 새로운 변수(area)를
	설정하고 getArea() 함수를 통해 받아내면 돼

	cout << "donut 면적은 " << area << endl;
	Circle pizza;
	pizza.radius = 30;
	double area = pizza.getArea(); -> double area로 쓰면 왜 오류가 나지?
		// double area로 쓰면 안 되는 이유는 위에 이미 정의되어있기 때문이야
```

```
        cout << "pizza 면적은 " << area << endl;
}
```

(수정1 -> 최종)

```
#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;
        double area = donut.getArea();
        cout << "donut 면적은 " << area << endl;

        Circle pizza;
        pizza.radius = 30;
        area = pizza.getArea();
        cout << "pizza 면적은 " << area << endl;
}
```

[3-2] Rectangle 클래스 만들기

보민 풀이)
ㄴ Rectangle 클래스 선언, 내부에 필요한 변수, 함수 선언
ㄴ Rectangle 클래스 구현, 반환할 값 작성
ㄴ main 함수 구현

```
// 오류없이 성공
#include <iostream>
using namespace std;

class Rectangle {
public:
	int width;
	int height;
	double getArea();
};

double Rectangle::getArea() {
	return width * height;
}

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

[3-4] 생성자에서 다른 생성자 호출 연습 (위임 생성자 만들기)
위임 생성자를 만든 이유 달라진 점 서술

//(보민) 위임 생성자를 생성하지 않고 코드를 쓰게 되면

```
Circle::Circle() {
        radius = 1;
        cout << "반지름 " << radius << " 원 생성" << endl;
}
Circle::Circle(int r) {
        radius = r;
        cout << "반지름 " << radius << " 원 생성" << endl;
}
```

이런식이 될텐데 우리는 위임 생성자를 통해 코드 중복을 줄여줄 수 있어.

->(승훈) 그럼 내가 위임 생성자를 생성해서 코드를 수정해볼게

```
Circle::Circle() : Circle(1) { }

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

위임 생성자를 쓰기 위해선 타겟 생성자가 있어야 해.
이 코드에서 1줄은 위임 생성자로 볼 수 있고 그 아래 3줄은 타겟 생성자로 볼 수 있어.

// 저 코드를 다시 한 번 설명해보면 Circle()이 호출되면 Circle()은 자신의 코드를 실행하기 전에 Circle(int r)를 호출한 뒤, r에 1을 넘겨주어 반지름과 원 생성을 대신 하게 된다고 볼 수 있지.

-> 맞아. 이처럼 위임 생성자를 사용하려면 위임 생성자에서 타겟 생성자를 호출한 뒤에 자신의 코드를 추가하면 될 것 같아.

[p.29] 기본 생성자가 자동으로 생성되지 않는 경우

//(보민) Circle donut의 기본 생성자가 없어서 오류가 나는 것 같아

->(승훈) 그럼 기본 생성자인 Circle::Circle() {}을 만들어주면 되겠다.

// 맞아 그리고 Circle 클래스에도 생성자가 선언되지 않은 Circle();을 만들어줘야 해

-> 코드를 한 번 적어보자

```
class Circle {
public:
        int radius;
        double getArea();
        Circle(int r);
        Circle();
};
Circle::Circle(int r) {
        radius = r;
}
Circle::Circle() {}
int main() {
        Circle pizza(30);
        Circle donut;
}
```

이렇게 수정할 수 있을 것 같아.

// 맞아 잘 수정한 거 같아

[3-6] Rectangle 클래스 만들기

보민 풀이)
ㄴ Rectangle 클래스 선언, 내부에 필요한 변수, 함수, 생성자 선언
ㄴ Rectangle 클래스 구현 (인자의 개수에 따라 각각 다르게)
ㄴ main 함수 구현 (주어짐)

```
//보민, ->승훈
#include <iostream>
using namespace std;

class Rectangle {
public:
	int width;
	int height;
	Rectangle();
	Rectangle(int a, int b);
	Rectangle(int c);
	bool isSquare(); // boolean이 왜 오류가 나지?
			-> double 써야 하는 거 아니야?
			// 참 거짓을 판별해야 하니까 데이터 타입을 boolean을 쓴 거야.
				double 데이터 타입은 실수를 받을 때 사용해
				근데 오류가 나네. 검색해보자
			-> c++에서는 bool로 표현한다고 나오네
};

Rectangle::Rectangle() : Rectangle(1) { }

Rectangle::Rectangle(int c) {
	width = height = c;
}

Rectangle::Rectangle(int a, int b) {
	width = a; height = b;
}

bool Rectangle::isSquare() {
	if (width == height) return true;
	else return false;
}
int main() {
	Rectangle rect1;
	Rectangle rect2(3, 5);
```

```
    Rectangle rect3(3);
    if (rect1.isSquare()) cout << "rect1은 정사각형이다." << endl;
    if (rect2.isSquare()) cout << "rect2는 정사각형이다." << endl;
    if (rect3.isSquare()) cout << "rect3는 정사각형이다." << endl;
}
```

[3-8] 결과 예측

// (보민) 객체 생성은 선언 순서에 따라 진행되고 소멸은 생성의 역순으로 소멸되는 규칙에 따라 코드의 실행 순서를 예측해보면

1. globalDonut(1000); -> Circle::Circle() 생성 (1)
   globalPizza(2000); -> Circle::Circle() 생성 (2)

2. Circle mainDonut; -> Circle::Circle() 생성 (3)
   Circle mainPizza(30); -> Circle::Circle(int r) 생성 (4)

3. f();에서 Circle fDonut(100); -> Circle::Circle() 생성 (5)
   Circle fPizza(200); -> Circle::Circle() 생성 (6)

4. (6)~(1) 순서로 소멸할 것 같아.

-> (승훈) 내 생각도 그래. 객체 생성은 선언된 순서로 생성되고, 그 다음에 main 함수가 실행되고, 그 안의 f()가 실행될 것 같아. 소멸은 생성의 반대이기에

- 반지름 1000원 생성
- 반지름 2000원 생성
- 반지름 1원 생성
- 반지름 30원 생성
- 반지름 100원 생성
- 반지름 200원 생성
- 반지름 200원 소멸
- 반지름 100원 소멸
- 반지름 30원 소멸
- 반지름 1원 소멸
- 반지름 2000원 소멸
- 반지름 1000원 소멸
-

이렇게 될 거라고 생각해.