객체지향프로그래밍 (202004100 장주훈, 202084023 이병길)

**4주차**

클래스와 객체

세상 모든 것은 객체이고 객체는 캡슐화 가능하다. (객체 내 데이터에 대한 보안, 보호, 외부접근 제한) + 외부와의 인터페이스(정보 교환 및 통신)를 위해 객체의 일부분 공개

|  |
| --- |
|  |

클래스는 설계도, 객체는 각 멤버(멤버 별 특성이 다름)

**클래스 선언부**와 클래스 구현부로 구성(분리하는 이유는 클래스를 다른 파일에서 활용하기 위함)

* class 키워드를 이용
* 접근 권한 지정 ◼ private, public, protected 중의 하나

**클래스 구현부**

* 클래스에 정의된 모든 멤버 함수 구현

|  |
| --- |
| #include <iostream>  using namespace std;  class Circle {  public:  int radius;  double getArea();  };  double Circle::getArea(){  return radius \* radius \* 3.14;  }  int main(){  Circle donut;  donut.radius = 1;  double area = donut.getArea();  cout << "donut의 면적은" << area << endl;  Circle pizza;  pizza.radius = 30;  area = pizza.getArea();  cout << "pizza의 면적은" << area << endl;  } |
| 어렵거나 힘들었던 점  클래스 선언 후, 메인 함수에서 바로 클래스를 불러와 객체 선언하는 것이 헷갈렸고, 멤버 함수의 호출 방식이 헷갈렸다.  절차지향적에서는 미리 다 선언을 하고 가져다 쓰는 형식인데 객체지향적에서는 함수안에서 바로 객체를 선언하고 진행하는 부분에서 헷갈리는 부분이 있었다.  또한, 객체에서 멤버 값을 넣어줄 때 객체이름과 멤버 사이의 ‘.’ 연산자 사용이 낯설었다.  **+ Public: 접근지정자 사용에 대해서는 아직 잘 모르겠다.**  **>> 이유: cpp는 원래 pravite을 디폴트 값이라 public의 접근지정자를 사용하지 않으면 오류가 발생하게 된다.** |

|  |
| --- |
| #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;  } |
| 헷갈렸던 점.  선언부에서 원래는 함수선언과 동시에 함수실행을 같이하는데 선언만하고 후에 구현부에서 실행하는 과정이 문법적으로 헷갈렸다. |

생성자: 객체가 생성되는 시점에서 자동으로 호출되는 멤버 함수

* 멤버함수와 쓰는 방식이 다름 >> 리턴 타입 없음. void 타입도 안됨
* 반드시 클래스 이름과 동일
* 객체 생성 시 오직 한 번만 호출(임의로 호출x)
* 매개변수가 다르면 다른 생성자임.
* 생성자가 선언되어 있지 않으면 기본 생성자 자동으로 생성(컴파일러에 의해 자동 생성)

목적 ◼ 객체가 생성될 때 객체가 필요한 초기화를 위해

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

Q3. 생성자가 하나라도 선언된 클래스의 경우(아래 코드에서 circle donut이 맞는가?)

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

의견 1: 컴파일러에서 생성자가 하나라도 선언되었다면 기본 생성자는 생성되지 않을 것 같다.

의견 2: 매개변수 받는 생성자를 선언했으므로 컴파일러는 기본생성자를 만들지 않을 것이다.

실습 3-6

|  |
| --- |
| #include <iostream>  using namespace std;  class Rectangle {  public:  int width;  int height;  bool isSquare();  Rectangle();  Rectangle(int a);  Rectangle(int a, int b);  };  Rectangle::Rectangle() : Rectangle(1){}  Rectangle::Rectangle(int a){  width = a;  height = a;  }  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;  } |
| 헷갈렸던 점.  if문에 사용되는 isSqaure()함수에서 true 반환하는 것을 생각해야 되기 때문에 bool Rectangle::isSqaure()함수를 만들어줘야 한다. |

소멸자: 객체가 소멸되는 시점에서 자동으로 호출되는 함수(더 이상 사용되지 않는 좀비로 남음)

* 자바에서는 알아서 소멸되지만, C++에서는 직접 소멸자를 사용해서 메모리 해제를 시켜야 함.
* 소멸자 함수의 이름은 클래스 이름 앞에 ~를 붙인다.
* 소멸자는 리턴 타입이 없고, 어떤 값도 리턴하면 안됨
* 중복 불가능
* 소멸자가 선언되어 있지 않으면 기본 소멸자가 자동 생성

new를 이용하여 동적으로 생성된 객체의 경우

* new를 실행하는 순간 객체 생성
* delete 연산자를 실행할 때 객체 소멸

Q5. 생성순서와 소멸순서를 적어보기.

|  |
| --- |
| #include <iostream>  using namespace std;  class Circle {  public:  int radius;  Circle();  Circle(int r);  ~Circle();  double getArea();  };  Circle::Circle() {  radius = 1;  cout << "반지름 " << radius << " 원 생성" << endl;  }  Circle::Circle(int r) {  radius = r;  cout << "반지름 " << radius << " 원 생성" << endl;  }  Circle::~Circle() {  cout << "반지름 " << radius << " 원 소멸" << endl;  }  double Circle::getArea() {  return 3.14 \* radius \* radius;  }  Circle globalDonut(1000);  Circle globalPizza(2000);  void f() {  Circle fDonut(100);  Circle fPizza(200);  }  int main() {  Circle mainDonut;  Circle mainPizza(30);  f();  } |
| 코드만 보고 순서 결과예측  생성: 전역>main() > f()  소멸: f() > main() > 전역 |

**접근 지정자**

◼ private: 동일한 클래스의 멤버 함수에만 제한함(C++ 디폴트 값)

◼ public: 모든 다른 클래스에 허용

◼ protected: 클래스 자신과 상속받은 자식 클래스에만 허용(상속)

**텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명**

**인라인 함수 사용 이유(**함수 호출시 컴퓨터 구조상 오버헤드가 발생할 수 밖에 없는데 오버헤드를 해결하기 위해 인라인 함수 사용을 한다.)

Ex) 텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

* 매크로와 유사
* 함수 호출에 따른 오버헤드 존재하지 않음 >> 프로그램의 실행 속도 개선

But, 인라인 함수 코드의 삽입으로 컴파일된 전체 코드 크기 증가

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**C++ 구조체**(C 언어와의 호환성 때문 + 구조체의 디폴트 접근 지정 – public)

구조체 객체 생성 ⁄ struct 키워드 생략

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**이때 문제점!! 중요!!**

#include”a.h”를 한 b.h, c.h가 있으면

main에서 #include”a.h” #include”b.h” 이러면 오류 발생!

따라서, #ifndef A\_H #define A\_H #endif 이런 식으로 하면 충돌을 피할 수 있음.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명