4장. 클래스의 완성

담당교수 : 김미경

3장의 내용 정리

- 클래스에 대한 기본(7장까지 이어진다)
  - 무엇인가를 구현하기에는 아직도 무리!
- 클래스의 등장 배경
  - 현실 세계를 모델링
- 데이터 추상화 → 클래스화 → 객체화
- 접근 제어 : public , private, protected

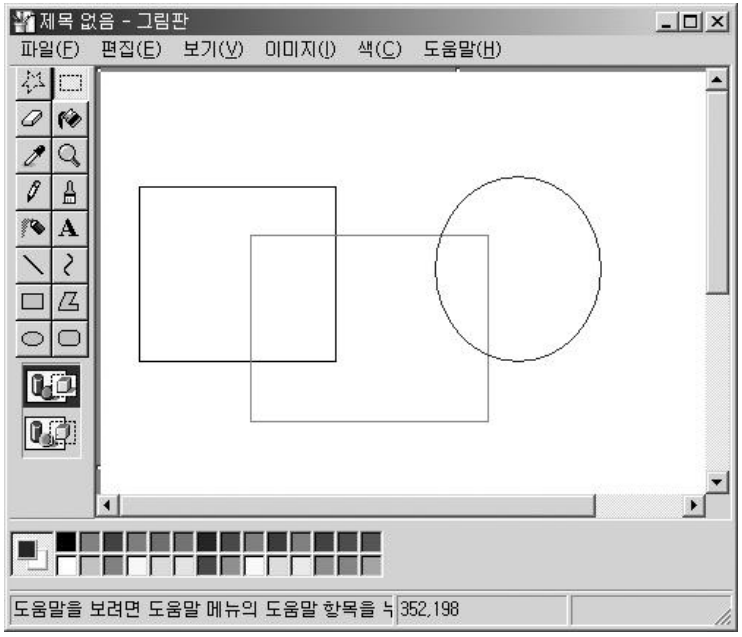
4장의 핵심 내용

- 클래스 디자인 기본 원칙
  - 캡슐화, 정보 은닉
  - 캡슐화와 정보 은닉의 유용성

- 클래스 객체의 생성과 소멸
  - 생성자, 소멸자
  - 생성자, 소멸자의 유용성

4-1 정보 은닉

- **정보 은닉의 필요성**
  - 프로그램의 안정적 구현과 관련
  - InfoHiding1.cpp, InfoHiding2.cpp, InfoHiding3.cpp




```
class Point
{
public:
        int x;
        int y;
};
```

4-1 정보 은닉

```cpp
/*   InfoHiding1.cpp*/
#include<iostream>
using std::cout;using std::endl;using std::cin;

class Point{
public:
    int x;   // x좌표의 범위 : 0 ~ 100
    int y;   // y좌표의 범위 : 0 ~ 100
};

int main(){
    int x, y;
    cout<<"좌표입력 : ";
    cin>>x>>y;

    Point p;
    p.x=x;
    p.y=y;

    cout<<"입력 된 데이터를 이용해서 그림을 그림"<<endl;
    return 0;
}
```

```cpp
/*   InfoHiding2.cpp*/
#include<iostream>
using std::cout;using std::endl;using std::cin;

class Point{
    int x;   // x좌표의 범위 : 0 ~ 100
    int y;   // y좌표의 점위 : 0 ~ 100
public:
    int GetX(){ return x; }
    int GetY(){ return y; }

    void SetX(int _x){ x=_x; }
    void SetY(int _y){ y=_y; }
};

int main(){
    int x, y;
    cout<<"좌표입력 : ";
    cin>>x>>y;

    Point p;
    p.SetX(x);
    p.SetY(y);

    cout<<"입력 된 데이터를 이용해서 그림을 그림"<<endl;
    return 0;
}
```

- 입력 오류 추가

```
15  void Point::SetX(int _x)
16  {
17      if(_x<0 || _x>100)
18      {
19          cout<<"X좌표 입력 오류, 확인 요망"<<endl;
20          return;
21      }
22      x=_x;
23  }
24  void Point::SetY(int _y)
25  {
26      if(_y<0 || _y>100)
27      {
28          cout<<"Y좌표 입력 오류, 확인 요망"<<endl;
29          return;
30      }
31      y=_y;
32  }
```

4-2 캡슐화(Encapsulation)

- **캡슐화(Encapsulation)의 기본 개념**
  - 관련 있는 데이터와 함수를 하나로 묶는 것,
  - Encapsulation1.cpp, Encapsulation2.cpp

```cpp
/*   Encapsulation1.cpp*/
#include<iostream>
using std::cout;using std::endl;using std::cin;
class Point{
    int x;    // x좌표의 범위 : 0 ~ 100
    int y;    // y좌표의 범위 : 0 ~ 100
public:
    int GetX(){ return x; }
    int GetY(){ return y; }

    void SetX(int _x);
    void SetY(int _y);
};

void Point::SetX(int _x){
    if(_x<0 || _x>100) {
        cout<<"X좌표 입력 오류, 확인 요망"<<endl;
        return;
    }
    x=_x;
}
void Point::SetY(int _y){
    if(_y<0 || _y>100)
    {
        cout<<"Y좌표 입력 오류, 확인 요망"<<endl;
        return;
    }
    y=_y;
}
```

```cpp

class PointShow
{
public:
    void ShowData(Point p)
    {
        cout<<"x좌표 : "<<p.GetX()<<endl;
        cout<<"y좌표 : "<<p.GetY()<<endl;
    }

};


int main()
{
    int x, y;
    cout<<"좌표입력 : ";
    cin>>x>>y;

    Point p;
    p.SetX(x);
    p.SetY(y);

    PointShow show;
    show.ShowData(p);

    return 0;
}
```

4-2 캡슐화(Encapsulation)

```
1  /*   Encapsulation2.cpp*/
2  #include<iostream>
3  using std::cout;using std::endl;using std::cin;
4  class Point{
5      int x;   // x좌표의 범위 : 0~100
6      int y;   // y좌표의 범위 : 0~100
7  public:
8      int GetX(){ return x; }
9      int GetY(){ return y; }
10     void SetX(int _x);
11     void SetY(int _y);
12     void ShowData();  //캡슐화를 위해 추가된 함수.
13 };
14
15 void Point::SetX(int _x){
16     if(_x<0 || _x>100) {
17         cout<<"X좌표 입력 오류, 확인 요망"<<endl;
18         return;
19     }
20     x=_x;
21 }
22 void Point::SetY(int _y){
23     if(_y<0 || _y>100)
24     {
25         cout<<"Y좌표 입력 오류, 확인 요망"<<endl;
26         return;
27     }
28     y=_y;
29 }
```

```
30 void Point::ShowData(){
31     cout<<"x좌표 : "<<x<<endl;
32     cout<<"y좌표 : "<<y<<endl;
33 }
34
35 int main()
36 {
37     int x, y;
38     cout<<"좌표입력 : ";
39     cin>>x>>y;
40
41     Point p;
42     p.SetX(x);
43     p.SetY(y);
44     p.ShowData();
45
46     return 0;
47 }
```

4-3 생성자와 소멸자

- 생성자의 필요성
  - 객체를 생성과 동시에 초기화 하기 위해서
  - 객체는 생성과 동시에 초기화되는 것이 좋은 구조!

- 생성자란?
  - 객체 생성 시 반드시 한번 호출되는 함수
  - 클래스와 같은 이름의 함수다.
  - 리턴형이 없으며 리턴 하지도 않는다.

4-3 생성자와 소멸자

```
struct Person{
    char name[10];
};
int main()
{
    Person p;
    cin>>p.name;
    return 0;
}
int main()
{
    char name[10];
    cin>>name;
    if(name이 적합하다면)
        Person p={name};
    return 0;
}
```

```
/*  Person1.cpp */

class Person
{
    char name[SIZE];
    char phone[SIZE];
    int age;
  public:
    void ShowData();
};

void Person::ShowData()
{
    cout<<"name: "<<name<<endl;
    cout<<"phone: "<<phone<<endl;
    cout<<"age: "<<age<<endl;
}

int main()
{
    Person p={"KIM", "013-113-1113", 22};
    p.ShowData();
    return 0;
}
```

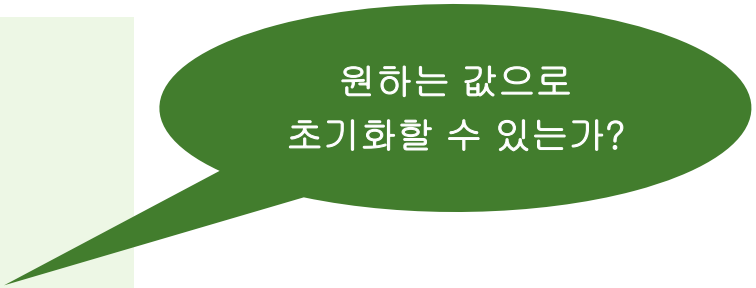
4-3 생성자와 소멸자

```
/*
   Person2.cpp
*/
class Person
{
      char name[SIZE];
      char phone[SIZE];
      int age;
  public:
      void ShowData(){ … 생략 … }
      void SetData(char* _name, char* _phone, int _age);
};
void Person::SetData(char* _name, char* _phone, int _age)
{
      strcpy(name, _name);
      strcpy(phone, _phone);
      age=_age;
}
int main()
{
      Person p;
      p.SetData("KIM", "013-333-5555", 22);
      return 0;
}
```

4-3 생성자와 소멸자

- **객체의 생성 과정(Constructor1.cpp)**
  - **첫째. 메모리 할당**
  - **둘째. 생성자의 호출**

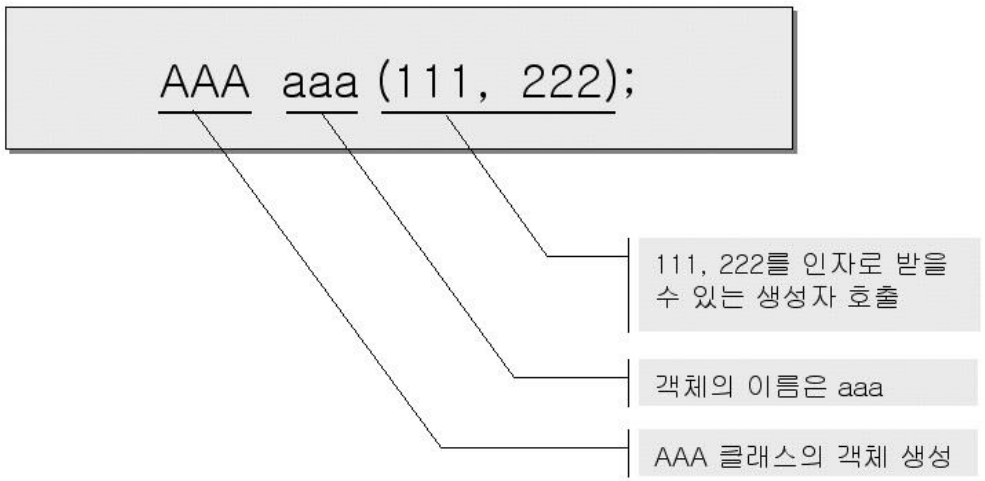
```
class AAA
{
    int i, j;
public:
    AAA()              //생성자
    {
        cout<<"생성자 호출"<<endl;
        i=10, j=20;
    }
    void ShowData()
    {
        cout<<i<<' '<<j<<endl;
    }
};
```



4-3 생성자와 소멸자

- **생성자를 통한 인자의 전달**
  - **이전의 객체 생성 방법은 구조체 변수의 선언과 동일한 방법!**
  - **Constructor2.cpp, Person3.cpp**

```
class AAA
{
    int i, j;
public:
    AAA(int _i, int _j)     //생성자
    {
        i=_i, j=_j;
    }
    void ShowData()
    {
        cout<<i<<' '<<j<<endl;
    }
};
```



4-3 생성자와 소멸자

```cpp
/*   Constructor2.cpp*/
#include<iostream>
using std::cout;
using std::endl;

const int SIZE=20;
class AAA
{
    int i, j;
public:
    AAA(int _i, int _j) //생 성 자 .
    {
        i=_i, j=_j;
    }
    void ShowData()
    {
        cout<<i<<' '<<j<<endl;
    }
};

int main()
{
    AAA aaa(111, 222);
    aaa.ShowData();

    return 0;
}
```

```cpp
/*   Person3.cpp*/
#include<iostream>
using std::cout;using std::endl;using std::cin;
const int SIZE=20;
class Person
{
    char name[SIZE];
    char phone[SIZE];
    int age;
public:
    Person(char* _name, char* _phone, int _age);
    void ShowData();
};

Person::Person(char* _name, char* _phone, int _age){
    strcpy(name, _name);
    strcpy(phone, _phone);
    age=_age;
}
void Person::ShowData(){
    cout<<"name: "<<name<<endl;
    cout<<"phone: "<<phone<<endl;
    cout<<"age: "<<age<<endl;
}
int main(){
    Person p("KIM", "013-333-5555", 22);
    p.ShowData();
    return 0;
}
```

4-3 생성자와 소멸자

- **public 생성자, private 생성자**
  - **public 생성자 : 어디서든 객체 생성 가능**
  - **private 생성자 : 클래스 내부에서만 가능**

```
#include <iostream>
using std::cout;
using std::cin;
using std::endl;

class A{
public:
     A(){
     cout<< "Test" <<endl;
     }
};
int main()
{
A a();      ➔A a;
return 0;
}
```

- **객체 생성의 두 가지 형태**

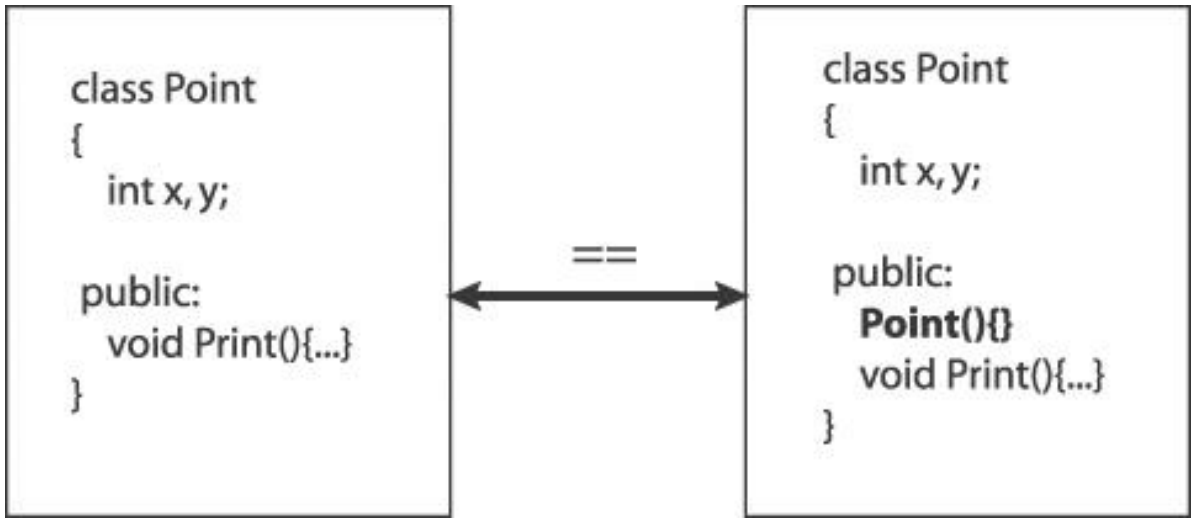
```
Person p = Person("KIM", "013-333-5555", 22);
Person p("KIM", "013-333-5555", 22);
```

- **오해하기 쉬운 객체 생성**

```
Person p("KIM", "013-333-5555", 22);        // 객체 생성
Person p( );                                // 객체 생성?
```

4-3 생성자와 소멸자

- 디폴트(default) 생성자
  - 생성자가 하나도 정의되어 있지 않은 경우
  - 자동으로 삽입이 되는 생성자
  - 디폴트 생성자가 하는 일? 아무것도 없다.
  - Default1.cpp

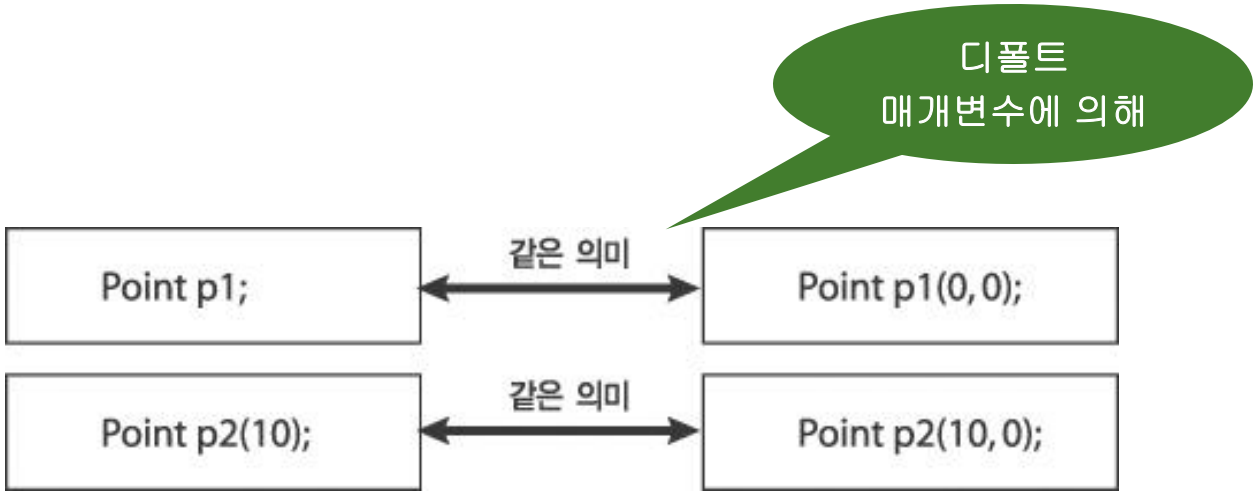


4-3 생성자와 소멸자

```
 1  /*    Default1.cpp*/
 2  #include<iostream>
 3  using std::cout;
 4  using std::endl;
 5
 6  class Point
 7  {
 8      int x, y;
 9  public:
10      Point(int _x, int _y)
11      {
12          x=_x, y=_y;
13      }
14      void ShowData()
15      {
16          cout<<x<<' '<<y<<endl;
17      }
18  };
19  int main()
20  {
21      Point p1(10, 20);    //10과  20을  인 자 로  받 는  생 성 자  호 출
22      p1.ShowData();
23
24      Point p2;    //void 생 성 자  호 출 .
25      p2.ShowData();
26      return 0;
27  }
```

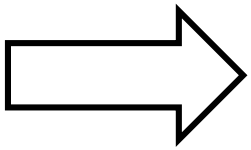
4-3 생성자와 소멸자

- **생성자의 특징**
  - 생성자도 함수다!
  - 따라서 함수 오버로딩이 가능하다
  - 디폴트 매개 변수의 설정도 가능하다.
  - Default2.cpp, Default3.cpp



4-3 생성자와 소멸자

```
1  /*   Default2.cpp*/
2  #include<iostream>
3  using std::cout;using std::endl;
4
5  class Point {
6      int x, y;
7  public:
8      Point() {
9          x=y=0;
10     }
11     Point(int _x, int _y)   {
12         x=_x, y=_y;
13     }
14     void ShowData() {
15         cout<<x<<' '<<y<<endl;
16     }
17 };
18
19 int main()
20 {
21     Point p1(10, 20);   //10과 20을 인자로 받는 생성자 호출
22     p1.ShowData();
23
24     Point p2;   //void 생성자 호출.
25     p2.ShowData();
26     return 0;
27 }
```



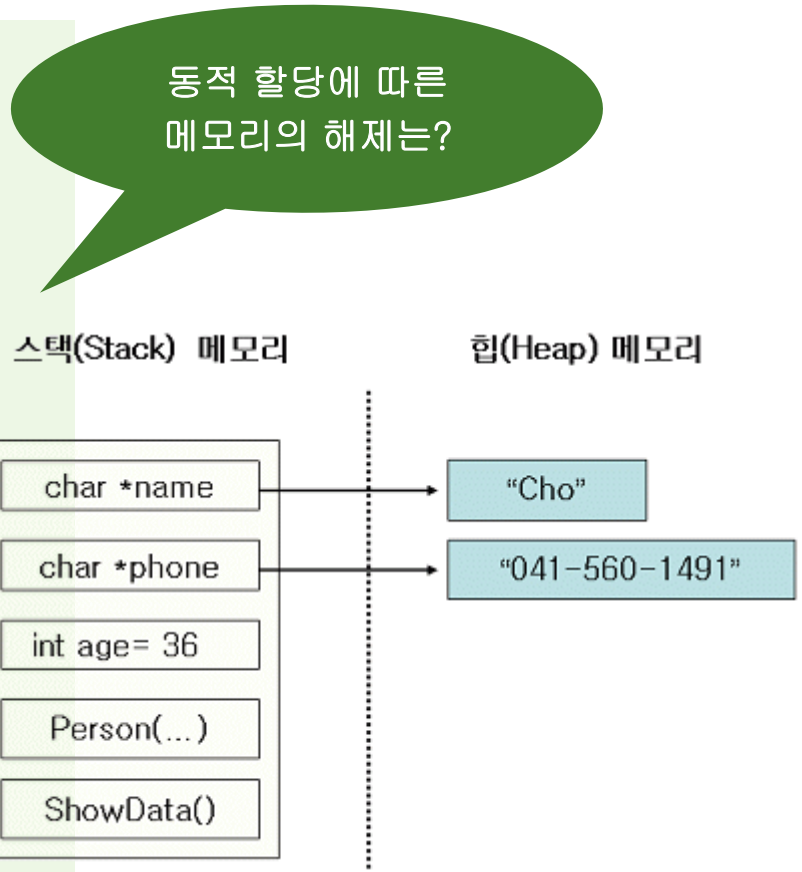
```
Point(int _x=0, int _y=0)
{
    x=_x, y=_y;
}
```

4-3 생성자와 소멸자

- 객체가 소멸되는 시점은?
  - 기본 자료형 변수, 구조체 변수가 소멸되는 시점과 동일하다.
- 함수 내에 선언된 객체
  - 함수 호출이 끝나면 소멸된다.
- 전역적으로 선언된 객체
  - 프로그램이 종료될 때 소멸된다.
  - 이렇게 객체 생성할 일 (거의) 없다!

▪ 생성자와 동적 할당(Person4.cpp)

```
class Person
{
    char *name;
    char *phone;
    int age;
public:
    Person(char* _name, char* _phone, int _age);
    void ShowData(){ … 생략 … }
};
Person::Person(char* _name, char* _phone, int _age)
{
    name=new char[strlen(_name)+1];
    strcpy(name, _name);

    phone=new char[strlen(_phone)+1];
    strcpy(phone, _phone);
    age=_age;
}
int main()
{   Person p("Cho", "041-560-1491", 36);
    p.ShowData();
    return 0;
}
```



- 동적 할당된 메모리 공간을 어디서도 해제해 주지 않고 있다.
  -> 메모리 누수 현상 발생
- 메모리를 자동으로 해제해 주는 함수 필요 -> 소멸자 함수

4-3 생성자와 소멸자

- **메모리를 해제하는 멤버 함수의 제공**
  - **Person5.cpp**

```
class Person
{
    char *name;
    char *phone;
    int age;
public:
    Person(char* _name, char* _phone, int _age);
    void DelMemory();
};

Person::Person(char* _name, char* _phone, int _age)
{
    … 생  략 …
}

void Person::DelMemory()
{
    delete []name;
    delete []phone;
}
```

만족하는가?

```
~Person();
```

```
Person::~Person()
{
    delete  [] name;
    delete  [] phone;
     cout << "소멸자 호출" << endl;
 }
```

4-3 생성자와 소멸자

- 생성자
  - 객체의 멤버 변수 초기화를 위해 객체 생성 시 자동 호출되는 함수

- 소멸자
  - 객체의 메모리 반환을 위해서 객체 소멸 시 자동 호출되는 함수
  - 클래스의 이름 앞에 ~가 붙은 형태
  - 리턴하지 않으며, 리턴 타입도 없다.
  - 전달인자 항상 void!
  - 따라서 오버로딩, 디폴트 매개 변수의 선언 불가능!

4-3 생성자와 소멸자

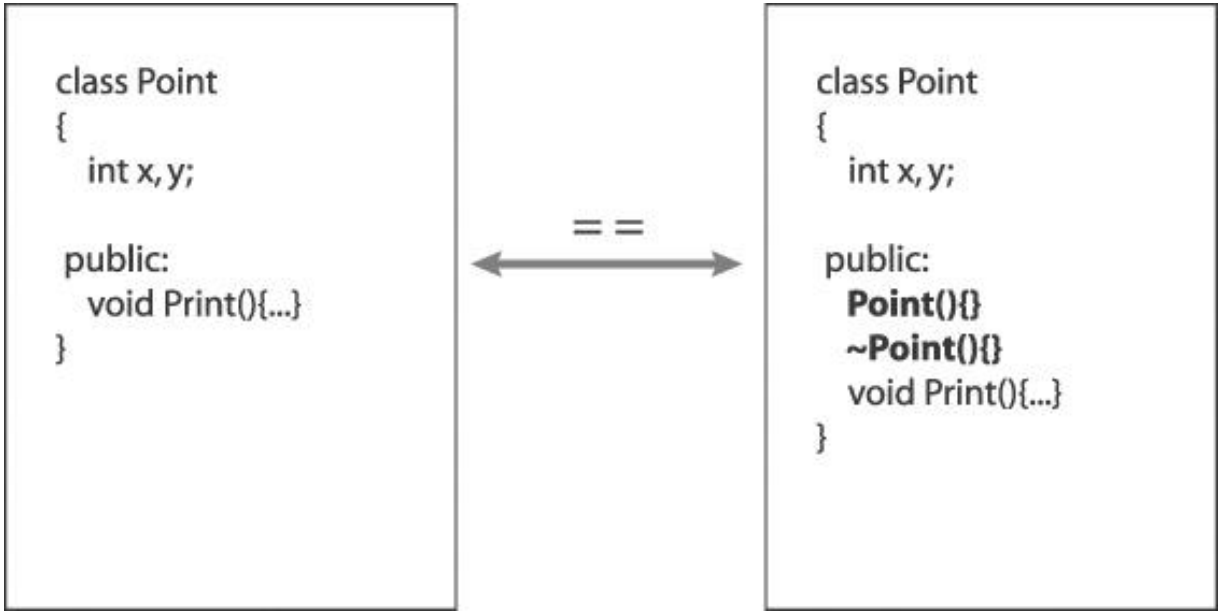
- 객체의 소멸 순서(Person5.cpp)
  - 첫째: 소멸자 호출
  - 둘째: 메모리 반환

```
class AAA
{
public:
    AAA() {
        cout<<"생성자 호출"<<endl;
    }
    ~AAA() {
        cout<<"소멸자 호출"<<endl;
    }
};
int main()
{
    AAA aaa1;          // “생성자 호출” 메시지 출력
    AAA aaa2;          // ”생성자 호출” 메시지 출력
    return 0;          // 리턴과 동시에 aaa1, aaa2 객체 소멸
}
```

```
1  /////   Person5.cpp   /////
2  #include<iostream>
3  #include <string.h>
4  using std::cout;using std::endl;
5
6  class Person{
7      char *name;
8      char *phone;
9      int age;
10 public:
11     Person(char* _name, char* _phone, int _age);
12     ~Person()
13     {
14         delete []name;
15         delete []phone;
16         cout << "소 멸 자  호 출 " << endl;
17     }
18     void DelMemory();
19     void ShowData();
20 };
21 Person::Person(char* _name, char* _phone, int _age){
22     name=new char[strlen(_name)+1];
23     strcpy(name, _name);
24
25     phone=new char[strlen(_phone)+1];
26     strcpy(phone, _phone);
27
28     age=_age;
29 }
35 int main()
36 {
37     Person p("KIM", "013-333-5555", 22);
38     p.ShowData();
39
40 //  p.DelMemory();
41
42     return 0;
43 }
```

4-3 생성자와 소멸자

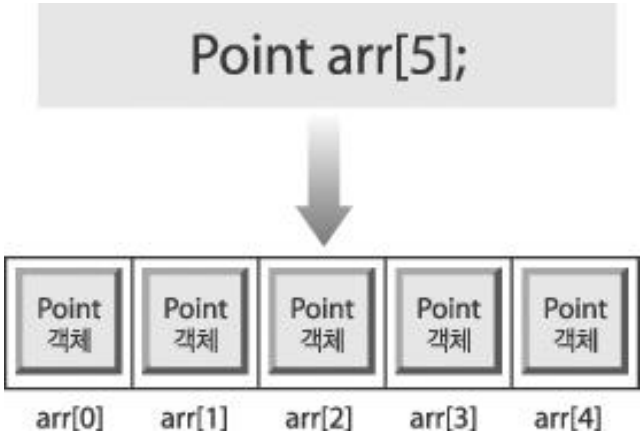
- 디폴트(Default) 소멸자
  - 객체의 소멸 순서를 만족시키기 위한 소멸자
  - 명시적으로 소멸자 제공되지 않을 경우 자동 삽입
  - 디폴트 생성자와 마찬가지로 하는 일 없다!



4-3 생성자와 소멸자

- 소멸자의 명시적 제공
  - 생성자에서 메모리 동적 할당을 하는 경우
  - Debugging 코드의 작성

- 객체 배열과 생성자
  - 객체 배열은 객체를 멤버로 지니는 배열이다.
  - 객체 배열은 기본적으로 void 생성자의 호출을 요구한다.
  - PointArr1.cpp



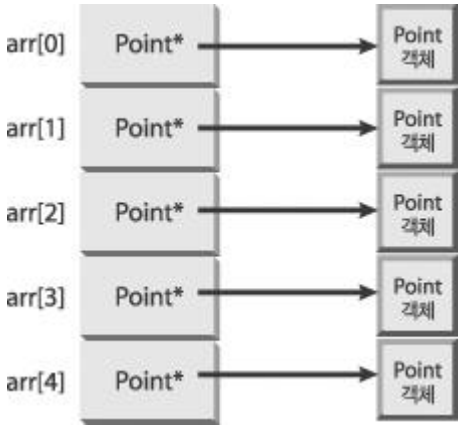
4-4 클래스와 배열

```cpp
/*    PointArr1.cpp*/

#include<iostream>
using std::cout;using std::endl;

class Point
{
    int x;
    int y;
public:
    Point(){
        cout<<"Point() call!"<<endl;
        x=y=0;
    }
    Point(int _x, int _y){
        x=_x;
        y=_y;
    }

    int GetX(){ return x; }
    int GetY(){ return y; }

    void SetX(int _x){ x=_x; }
    void SetY(int _y){ y=_y; }
};
```

```cpp
int main()
{
    Point arr[5];

    for(int i=0; i<5; i++)  {
        arr[i].SetX(i*2);
        arr[i].SetY(i*3);
    }

    for(int j=0; j<5; j++)
    {
        cout<<"x: "<<arr[j].GetX()<<' ';
        cout<<"y: "<<arr[j].GetY()<<endl;
    }

    return 0;
}
```

- **객체 포인터 배열**
  - 객체를 가리킬 수 있는 포인터를 멤버로 지니는 배열
  - 객체의 동적 생성 방법(PointArr2.cpp)

```
Point* arr[5];
```



4-4 클래스와 배열

```
29  int main()
30  {
31      Point* arr[5];
32
33      for(int i=0; i<5; i++)
34      {
35          arr[i]=new Point(i*2, i*3);  // new에 의한 객체 동적 생성.
36      }
37
38      for(int j=0; j<5; j++)
39      {
40          cout<<"x: "<<arr[j]->GetX()<<' ';
41          cout<<"y: "<<arr[j]->GetY()<<endl;
42      }
43
44      for(int k=0; k<5; k++)
45      {
46          delete arr[k];  // 힙에 저장된 객체 소멸.
47      }
48
49      return 0;
50  }
```

연습문제 4-4

- 직사각형을 나타내는 Rectangle 클래스와 원을 나타내는 Circle 클래스를 디자인 해 보자. 이 두 클래스는 넓이를 구하는 기능과 둘레를 구하는 기능(함수)을 가지고 있다. 다음에 제공되는 main 함수와 출력결과를 통해서 요구되는 Rectangle 클래스와 Circle 클래스를 디자인해 보자.

main 함수의 예

```
-------------------------------------------------------
int main(void)
{
  Rectangle rec(3, 4);  초기화로 Rectangle rec(가로(cm), 세로(cm))
  cout << "면적: " << rec.GetArea() << endl;
  cout << "둘레: " << rec.GetRound() << endl;


  Circle ring(5);  초기화로 Circle ring(원의 반지름 길이(cm))
  cout << "면적: " << ring.GetArea() << endl;
  cout << "둘레: " << ring.GetRound() << endl;


  return 0;
}
-------------------------------------------------------
```

연습문제 4-5

- 시(hour), 분(minute), 초(second) 정보를 지닐 수 있는 Time 클래스를 정의해 보자. 이 클래스는 멤버 변수가 지니고 있는 데이터를 적절히 출력하는 기능을 지녀야 한다. 출력 방식은 두 가지로 제공을 한다. 하나는 [시, 분, 초]의 형식을 띄며, 또 하나는 초 단위로 계산한 출력 결과를 보여준다. 제시되는 main 함수와 출력 결과를 참조하세요.

main 함수의 예

```
-------------------------------------------------------------
int main(void)
{
  Time  time1(10);   // 10시 0분 0초
  Time  time2(10, 20); // 10시 20분 0초
Time time3(10, 20, 30);  // 10시 20분 30초


time2.ShowTime();
time2.ShowTimeinSec();


return 0;
}
-------------------------------------------------------------
```

실행결과

```
-------------------------------------------------------------
[10시 20분 0초]
37200 초
-------------------------------------------------------------
```

연습문제 4-6

- 명함 정보를 지닐 수 있는 클래스를 정의해 보자. 클래스의 이름은 NameCard이고 이름, 전화번호, 주소, 직급 정보를 저장할 수 있어야 한다. 생성자 내에서 동적 할당하고, 소멸자에서 할당받은 메모리를 해제하는 형식으로 구현해 보자.

main 함수

```
-----------------------------------------------------------
int main(void)
{
   // NameCard Lee(이름, 전화번호, 주소, 직급);
  NameCard Lee("Lee Ji Sun", "333-333", "www.ezsun.net", "engineer");
  Lee.ShowData();
  return 0;
}
-----------------------------------------------------------
```

실행결과

```
-----------------------------------------------
이 름: Lee Ji Sun
전화번호: 333-3333
주 소: www.ezsun.net
직 급: engineer
```

실습 문제

실수부와 허수부로 되어 있는 복소수를 표현하고, 사칙연산이 가능한 Complex 클래스를 정의하고, 그 사용 예를 보여라.

Complex 클래스 정의 시 다음과 같은 특징을 고려하여 설계하세요.

- 생성자 함수에서는 실수부와 허수부를 초기화하라.

  예) Complex a(1,2); // a = 1+2i로 객체 생성과 더불어 초기화
- 복소수 12.4 + 15.6i를 Console 화면에 출력할 수 있는 기능을 포함하여야 하며, Complex 클래스의 public 함수인 Display() 함수를 이용하여, Console 화면에 복소수를 출력하는 기능을 포함하라
- 복소수 사칙연산 기능 포함

  예를 들어, 두 복소수 x = x1+y1i, y= x2+y2i 일 경우

  add(z=x+y): z = (x1+x2) + (y1+y2)i

  subtract(z=x-y): z = (x1-x2) +(y1-y2)i

  multiply(z=x*y): z = (x1*x2 - y1*y2)+(x1*y2+y1*x2)i

  divide(z=x/y): z = (x1*x2+y1+y2)/(x2^2 + y2^2) + (y1*x2-x1*y2)i/(x2^2+y2^2)
- 실수부와 허수부 값을 설정하고, 반환하는 함수 SetRe(), SetIm(), GetRe(), GetIm() 기능 포함

OOP2 만들기

1. 기존 작성된 OOP1의 cpp복사하여 OOP2프로젝트 만들기
2. struct를 class로 전환
    1. name을 char의 포인터로 변환하여 생성자와 소멸자 생성
    2. 이름, id, 잔액을 return 받는 멤버함수 추가
        Get….
    3. 잔액을 추가하는 멤버함수 : AddMoney
    4. 잔액을 찾는 멤버함수 : MinMoney
    5. 전제 자료를 확인하는 함수 : ShowAllData
3. Account pArray[100] 을 객체 배열의 포인터로 변경
4. MakeAccount 에 Account 객체 추가 하는 부분 변경==>객체의 동적할당, 생성자 이용
5. Deposit, Withdraw, Inquire 함수에서 Account클래스의 private으로 선언된 멤버 변수의 접근하기 위한 방법 변경.