Power C++

제10장 생성자와 소멸자

이번 장에서 학습할 내용

- 생성자
- 소멸자
- 초기화 리스트
- 복사 생성자
- 디폴트 멤버 함수

생성자

- 생성자(contructor): 객체가 생성될 때에 필드에게 초기값을 제공하고 필요한 초기화 절차를 실행하는 멤버 함수

객체의 일생

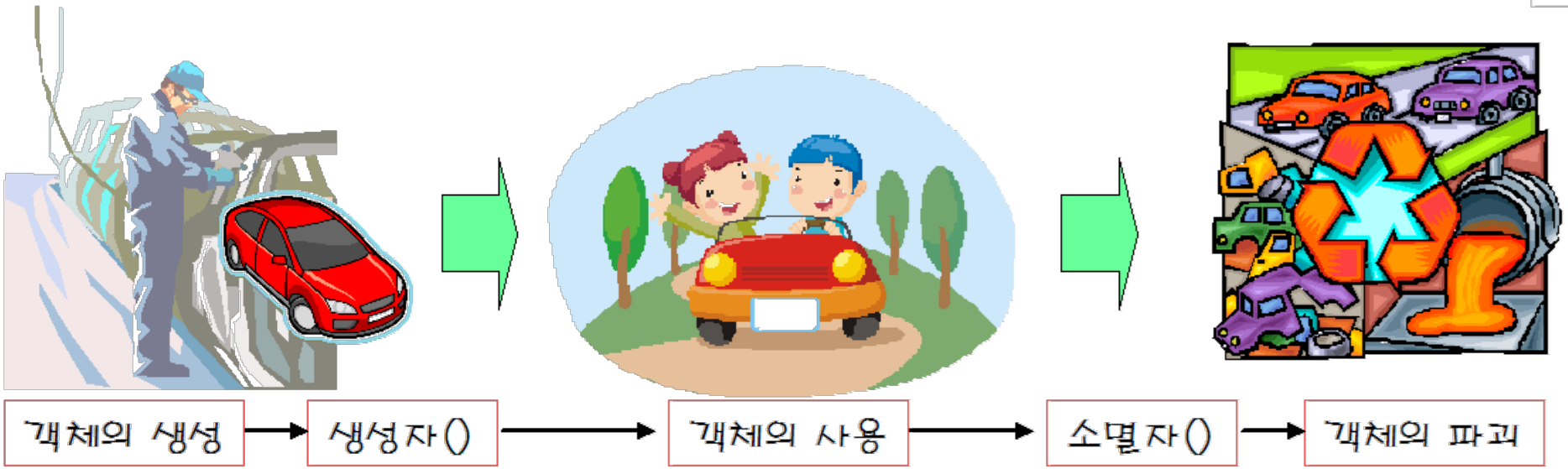


그림 10.2 객체의 일생

생성자의 특징

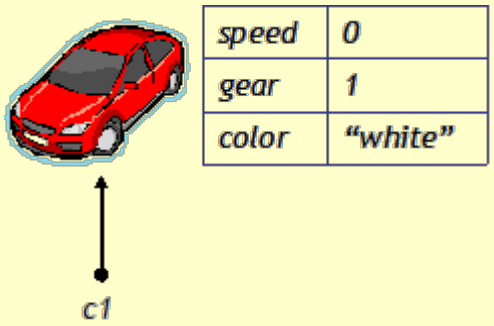
- 클래스 이름과 동일하다
- 반환값이 없다.
- 반드시 public 이어야 한다.
- 중복 정의할 수 있다.

```
class Car
{
        ...
public:
        Car()
        {
                ...
        }
};
```

생성자

디폴트 생성자

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

class Car {
private:
	int speed; // 속도
	int gear; // 기어
	string color; // 색상
public:
	Car()
	{
		cout << "디폴트 생성자 호출" << endl;
		speed = 0;
		gear = 1;
		color = "white";
	}
};
int main()
{
	Car c1; // 디폴트 생성자 호출
	return 0;
}
```



생성자의 외부 정의

```
Car::Car()
{
        cout << "디폴트 생성자 호출" << endl;
        speed = 0;
        gear = 1;
        color = "white";
}
```

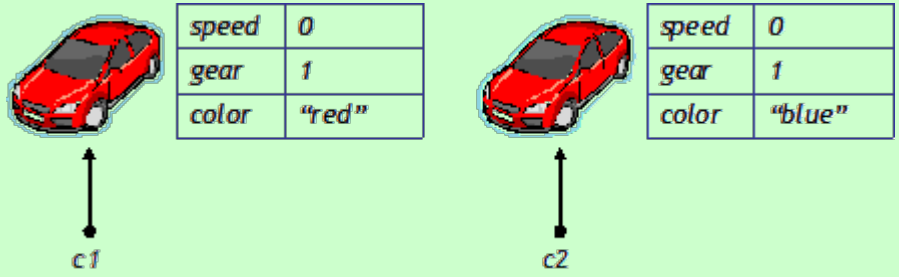
매개 변수를 가지는 생성자

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

class Car {
private:
    int speed;                  // 속도
    int gear;                   // 기어
    string color;               // 색상
public:
    Car(int s, int g, string c)   : speed(s), gear(g), color(c)로 사용할 것
    {
        speed = s;
        gear = g;
        color = c;
    }
    void printInfo();
};
```

매개 변수를 가지는 생성자

```
void Car::printInfo()
{
    cout << "속도: " << speed << endl;
    cout << "기어: " << gear << endl;
    cout << "색상: " << color << endl;
}

int main()
{
    Car c1(0, 1, "red");                    // 생성자 호출
    Car c2 = Car(0, 1, "blue");             // 이런 식으로도 생성자 호출이 가능하다.
    c1.print();
    c2.print();
    return 0;
}
```

```
속도: 0
기어: 1
색상: red
속도: 0
기어: 1
색상: blue
```



생성자의 중복 정의

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

class Car {
private:
    int speed;                  // 속도
    int gear;                   // 기어
    string color;       // 색상
public:
    Car()
    {
        cout << "디폴트 생성자 호출" << endl;
        speed = 0;
        gear = 1;
        color = "white";
    }
```

매개 변수를 가지는 생성자

```
Car(int s, int g, string c)
    {
        cout << "매개 변수가 있는 생성자 호출" << endl;
        speed = s;
        gear = g;
        color = c;
    }
};
int main()
{
    Car c1;                      // 디폴트 생성자 호출
    Car c2(100, 0, "blue"); // 생성자 호출
    return 0;
}
```

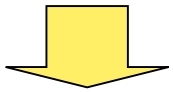
```
매개 변수가 있는 생성자 호출
디폴트 생성자 호출
계속하려면 아무 키나 누르십시오 . . .
```

생성자 호출의 다양한 방법

```
int main()
{
        Car c1;   // ①디폴트 생성자 호출
        Car c2(); // ②이것은 생성자 호출이 아니라 c2()라는 함수의 원형 선언
        Car c3(100, 3, "white");     // ③생성자 호출
        Car c4 = Car(0, 1, "blue"); // ④이것은 먼저 임시 객체를 만들고 이것을
                                    c4에 복사
        return 0;
}
```

생성자를 하나도 정의하지 않으면?

```
class Car {
      int speed;          // 속도
      int gear;           // 기어
      string color;       // 색상
};
```



컴파일러가 비어있는 디폴트 생성자를 자동으로 추가한다.

```
class Car {
      int speed;          // 속도
      int gear;           // 기어
      string color;       // 색상
public:
      Car() { }
}
```

디폴트 매개 변수

```
Car(int s=0, int g=1, string c="red")
{
        speed = s;
        gear = g;
        color = c;
}
```

디폴트 생성자를 정의한 것과 마찬가지 효과를 낸다.

생성자에서 다른 생성자 호출하기

```
...
class Car {
    int speed; // 속도
    int gear; // 기어
    string color; // 색상
public:
    // 첫 번째 생성자
    Car(int s, int g, string c) {
        speed = s;
        gear = g;
        color = c;
    }
    // 색상만 주어진 생성자
    Car(string c) {
        Car(0, 0, c);                    // 첫 번째 생성자를 호출한다.
    }
};
int main()
{
    Car c1("white");
    return 0;
}
```

중간 점검 문제

1. 만약 클래스 이름이 MyClass라면 생성자의 이름은 무엇이어야 하는가? MyClass
2. 생성자의 반환형은 무엇인가? 없음
3. 생성자는 중복 정의가 가능한가? ㅇ
4. 클래스 안에 생성자를 하나도 정의하지 않으면 어떻게 되는가?

   커파일러가 자동으로 디폴트 생성자 생성

소멸자

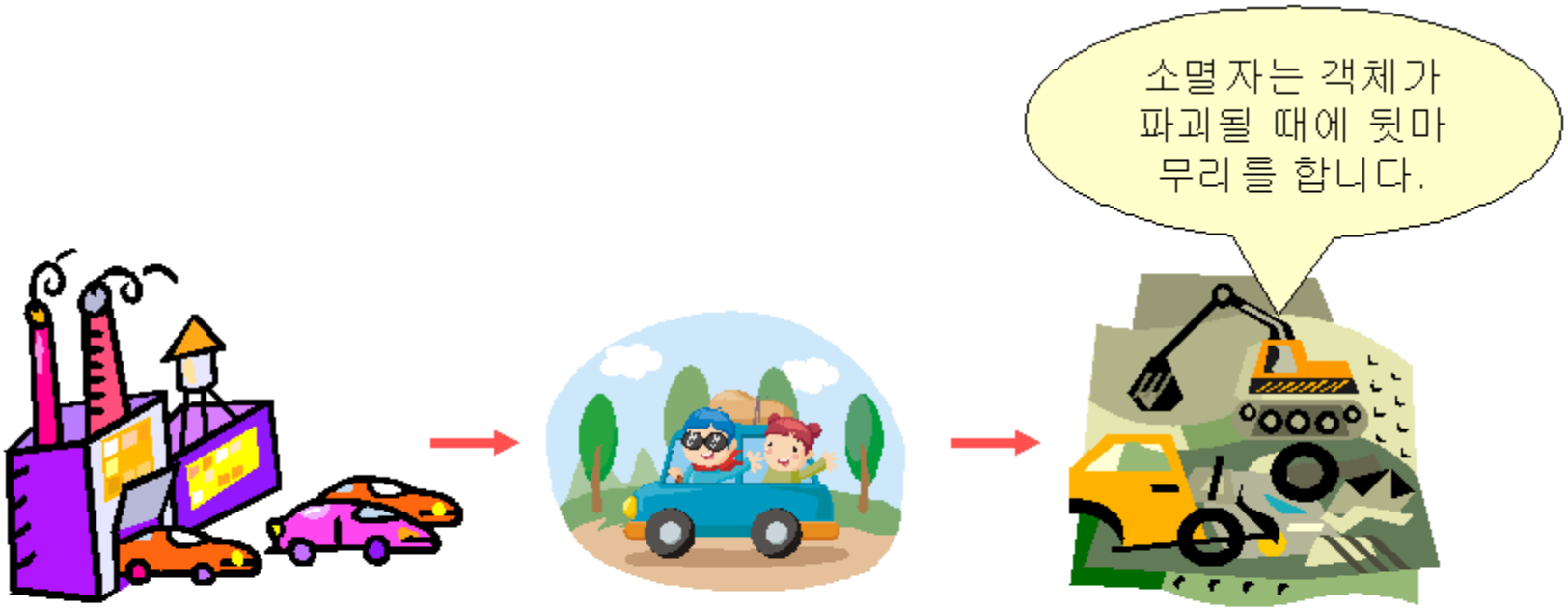


그림 10.4 소멸자의 개념

소멸자의 특징

- 소멸자는 클래스 이름에 ~가 붙는다.
- 값을 반환하지 않는다.
- public 멤버 함수로 선언된다.
- 소멸자는 매개 변수를 받지 않는다.
- 중복 정의도 불가능하다.

입력을 받지 않음

```
class Car
{
        ...
public:
        ~Car()
        {
                ...
        }
};
```

소멸자

소멸자

```
class Car {
private:
	int speed;			// 속도
	int gear;			// 주행 거리
	string color;		// 색상
public:
	Car()
	{
		cout << "생성자 호출" << endl;
		speed = 0;
		gear = 1;
		color = "white";
	}
	~Car()
	{
		cout << "소멸자 호출" << endl;
	}
};
int main()
{
	Car c1; // 디폴트 생성자 호출
	return 0;
}
```

{ }가 끝나서 c1이 죽을 때 자동으로 소멸자 함수 실행. ->여기선 소멸자 안만들어도 상관 X

```
매개 변수가 있는 생성자 호출
디폴트 생성자 호출
계속하려면 아무 키나 누르십시오 . . .
```

유용한 소멸자

```
...
class Car {
      int speed; // 속도
      int gear; // 기어
      char *color; // 색상
public:
      // 첫 번째 생성자
      Car(int s, int g, char *c) {
            speed = s;
            gear = g;
            color = new char[strlen(c)+1];
            strcpy(color, c);
      }
      ~Car() {
            delete [] color;
      }
};

int main()
{
      Car c(0, 1, "yellow");
      return 0;
}
```

stack에 있음

동적 메모리 반납

color가 가리키는 것으로 오면 OS로부터 받아와 동적 할당을 했기 때문에 소멸자 사용

디폴트 소멸자

- 만약 프로그래머가 소멸자를 정의하지 않았다면 어떻게 되는가?
- 디폴트 소멸자가 자동으로 삽입되어서 호출된다

```
class Time {
        int hour, minute, second;
public:
        print() { … }
}
```

~Time { } 을 넣어준다.

중간 점검 문제

1. 만약 클래스 이름이 MyClass라면 소멸자의 이름은 무엇이어야 하는가?
2. 소멸자의 반환형은 무엇인가?
3. 소멸자는 중복 정의가 가능한가?

초기화 리스트

- 멤버 변수를 간단히 초기화할 수 있는 형식

```
Car(int s, int g, string c) : speed(s), gear(g), color(c) {
        ...// 만약 더 하고 싶은 초기화가 있다면 여기에
}
```

상수 멤버의 초기화

- 멤버가 상수인 경우에는 어떻게 초기화하여야 하는가?

컴파일러마다 다르기 때문에 이런 게 있다고만 알고 있고 중요한건 Car(): MAX_SPEED(300)은 선언과 동시에 초기화하는 것이라는 것.

```
class Car
{
        const int MAX_SPEED = 300;          // 컴파일 오류!
        int speed;          // 속도
        ...
}
```

아직 생성이 안됐음!

```
class Car
{
        const int MAX_SPEED;
        int speed;          // 속도
public:
        Car()
        {
                MAX_SPEED = 300;          // 컴파일 오류!

        }
}
```

상수를 변경할 수 없음!

상수 멤버의 초기화

```
class Car
{
        const int MAX_SPEED;
        int speed;          // 속도
public:
        Car() : MAX_SPEED(300)
        {
        }
};
```

레퍼런스 멤버의 초기화

```
#include <iostream>
#include <string>
using namespace std;
class Car
{
	string& alias;
	int speed;		// 속도
public:
	Car(string s) : alias(s)
	{
		cout << alias << endl;
	}
};

int main()
{
	Car c1("꿈의 자동차");
	return 0;
}
```

```
꿈의 자동차
계속하려면 아무 키나 누르십시오 . . .
```

객체 멤버의 경우

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

class Point
{
    int x, y;
public:
    Point(int a, int b) : x(a), y(b)
    {
    }
};

class Rectangle
{
    Point p1, p2;
public:
    Rectangle(int x1, int y1, int x2, int y2) : p1(x1, y2), p2(x2, y2)
    {
    }
};
```

생성자 호출

중간 점검 문제

1. 초기화 리스트를 반드시 사용하여서 초기화해야되는 멤버의 타입은?
2. 클래스 MyClass의 상수 limit를 초기화 리스트를 사용하여서 초기화 하여 보라.

복사 생성자

- 한 객체의 내용을 다른 객체로 복사하여서 생성

```
class Student
{
public:
    Student(const Student& other)
        : ____________________
    { }
};

iny main()
{
        Student a;
        Student b = a;  ( copy constructure )

         b = a;  ( operator constructor )
}
```

객체

복사생성자

객체

복사 생성자의 특징

- 자동으로 디폴트 복사 생성자가 생성된다.
- 자신과 같은 타입의 객체를 매개 변수로 받는다.

```
Car(Car& obj);

Car(const Car& obj);
```

복사 생성자

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

class Car {
	int speed; // 속도
	int gear; // 기어
	string color; // 색상
public:
	// 첫 번째 생성자
	Car(int s, int g, string c) {
		speed = s;
		gear = g;
		color = c;
	}
	void printInfo()
	{
		cout << "속도: " << speed << endl;
		cout << "기어: " << gear << endl;
		cout << "색상: " << color << endl;
	}
};
```

복사 생성자

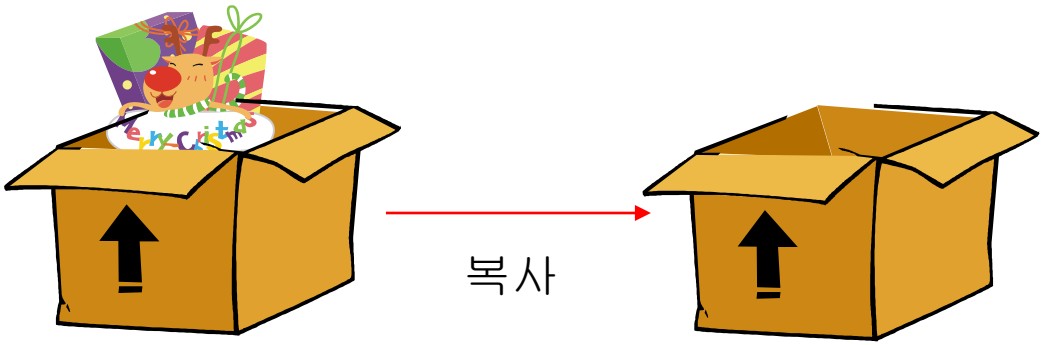
```
int main()
{


    Car c1(0, 1, "yellow");
    Car c2(c1);        // 복사 생성자 호출

    c1.printInfo();
    c2.printInfo();
    return 0;
}
```

복사 생성자 호출

```
속도: 0
기어: 1
색상: yellow
속도: 0
기어: 1
색상: yellow
계속하려면 아무 키나 누르십시오 . . .
```

얕은 복사 문제

- 멤버의 값만 복사하면 안되는 경우가 발생한다.
- 얕은 복사(shallow copy) 문제



예제

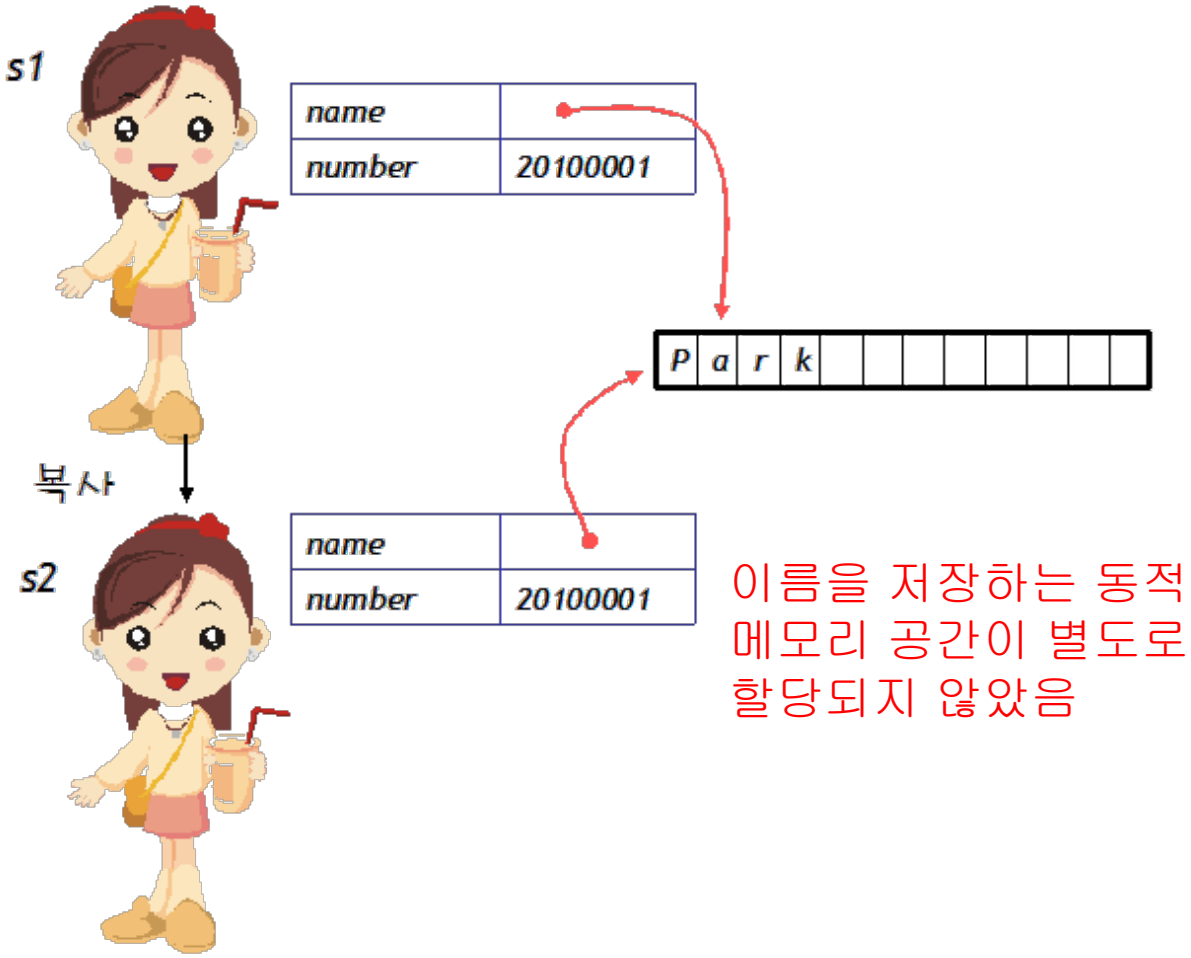
```
#include <iostream>
#include <string>
using namespace std;

class Student {
    char *name; // 이름
    int number;
public:
    // 첫 번째 생성자
    Student(char *pn, int n) {
        name = new char[strlen(pn)+1];
        strcpy(name, pn);
        number = n;
    }
    ~Student() {
        delete [] name;
    }
    void setName(char *pn)
    {
        delete[] name;
        name = new char[strlen(pn)+1];
        strcpy(name, pn);
    }
```

예제

```
    void printInfo()
    {
        cout << "이름: " << name << " ";
        cout << "학번: " << number << endl;
    }
};
int main()
{
    Student s1("Park", 20100001);
    Student s2(s1);  // 복사 생성자 호출

    s1.printInfo();
    s2.printInfo();
    s1.setName("Kim");

    s1.printInfo();
    s2.printInfo();
    return 0;
}
```

```
이름: Park 학번: 20100001
이름: Park 학번: 20100001
이름: Kim 학번: 20100001
이름: Kim 학번: 20100001.
```

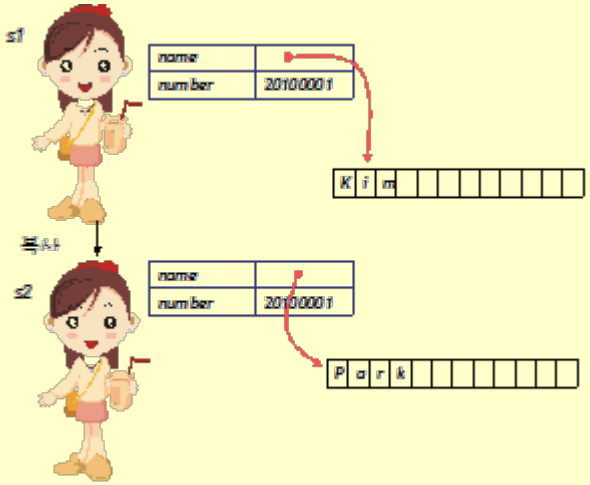
문제점



이름을 저장하는 동적 메모리 공간이 별도로 할당되지 않았음

깊은 복사

```
class Student {


        ....
    Student(Student& s) {
        name = new char[strlen(s.name)+1];
        strcpy(name, s.name);
        number = s.number;
    }
        ....
};
```




```
이름: Park 학번: 20100001
이름: Park 학번: 20100001
이름: Kim 학번: 20100001
이름: Park 학번: 20100001
```

복사 생성자가 호출되는 경우

- 기존의 객체의 내용을 복사하여서 새로운 객체를 만드는 경우
- 객체를 값으로 매개 변수로 전달하는 경우
- 객체를 값으로 반환하는 경우

복사 생성자

예제

```
class Student {
    char *name; // 이름
    int number;
public:
    // 첫 번째 생성자
    Student(char *pn, int n) {
        name = new char[strlen(pn)+1];


        strcpy(name, pn);
        number = n;
    }
    Student(Student& s) {
        name = new char[strlen(s.name)+1];
        strcpy(name, s.name);
        number = s.number;
    }
    ~Student() {
        delete [] name;
    }
```

예제

```
    char *getName()
    {
        return name;
    }
    int getNumber()
    {
        return number;
    }


};
void displayStudent(Student obj)
{
    cout << "이름: " << obj.getName() << endl;
    cout << "학번: " << obj.getNumber() << endl;
}

int main()
{
    Student s1("Park", 20100001);
    displayStudent(s1);
    return 0;
}
```

복사 생성자 호출

실행 결과

```
이름: Park
학번: 20100001
계속하려면 아무 키나 누르십시오 . . .
```

중간 점검 문제

1. 복사 생성자는 언제 사용되는가?
2. 얕은 복사와 깊은 복사의 차이점은 무엇인가?

디폴트 멤버 함수

- 디폴트 생성자
- 디폴트 소멸자
- 디폴트 복사 생성자
- 디폴트 할당 연산자

자동으로 추가된다.

pointer를 사용하면 copy constructor, assign constructor, 소멸자 만들어야함

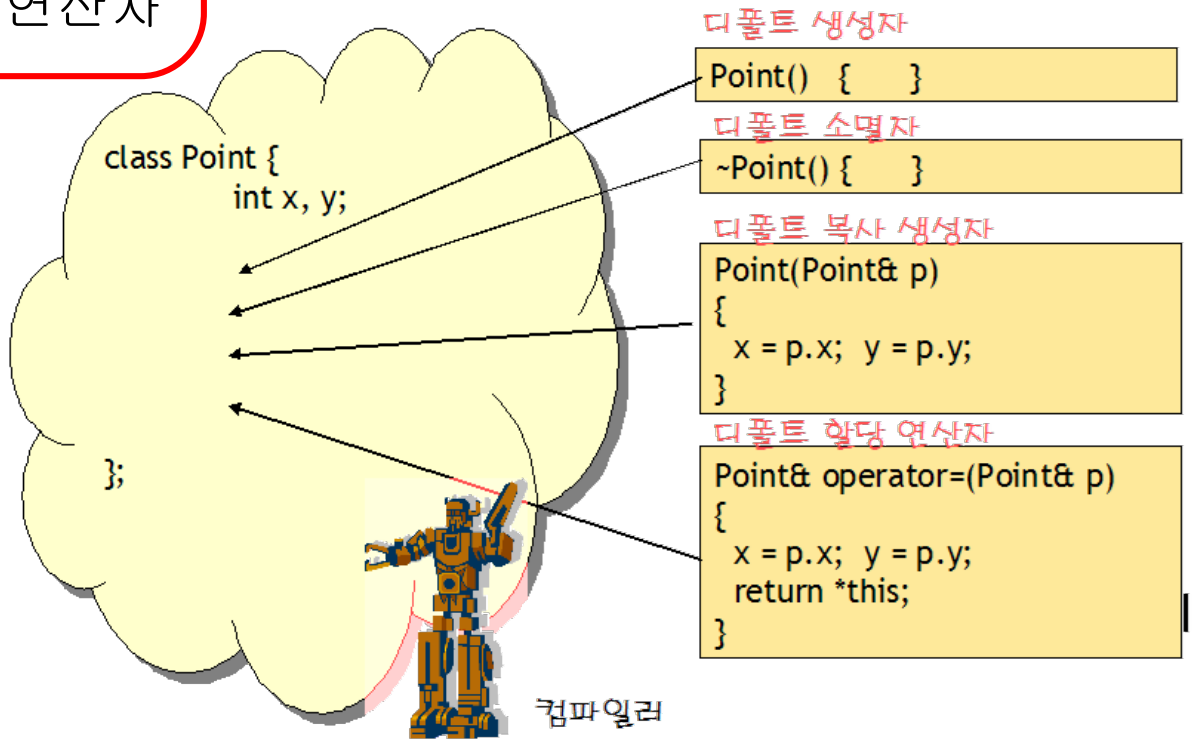
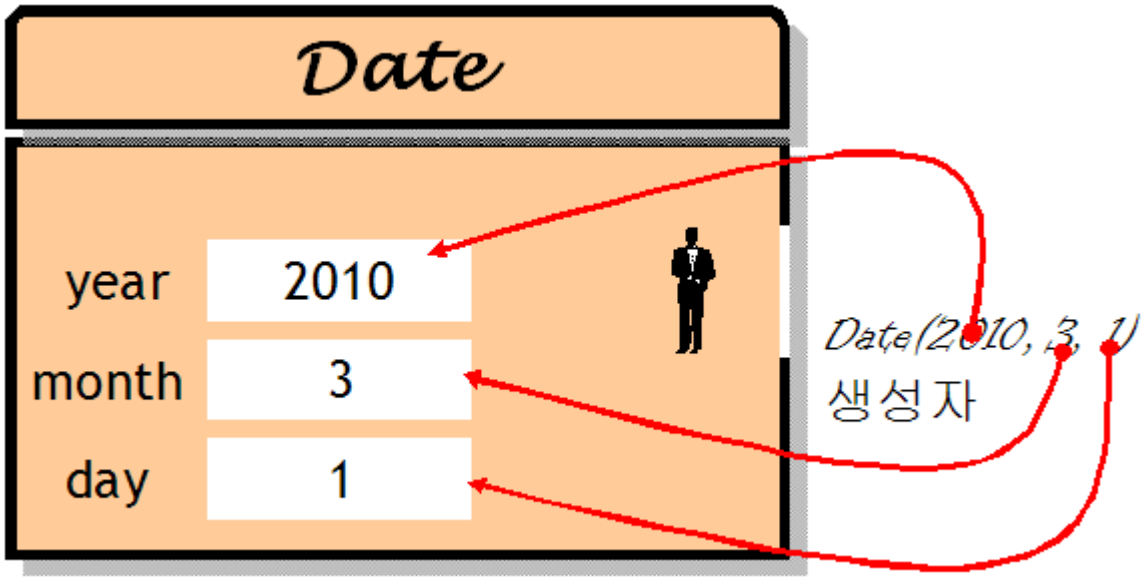


그림 10.7 디폴트 멤버 함수의 추가

예제

- Date 클래스에 생성자와 소멸자를 추가



예제

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

class Date {
private:
    int year;
    int month;
    int day;
public:
    Date();                                          // 디폴트 생성자
    Date(int year);                                  // 생성자
    Date(int year, int month, int day);              // 생성자
    void setDate(int year, int month, int day);      // 멤버 함수
    void print();                                    // 멤버 함수
};

Date::Date()          // 디폴트 생성자
{
    year = 2010;
    month = 1;
    day = 1;
}
```

예제

```
Date::Date(int year) // 생성자
{
    setDate(year, 1, 1);
}
Date::Date(int year, int month, int day) // 생성자
{
    setDate(year, month, day);
}
void Date::setDate(int year, int month, int day)
{
    this->month = month;                    // this는 현재 객체를 가리킨다.
    this->day = day;
    this->year = year;
}
void Date::print()
{
    cout << year << "년 " << month <<  "월 " << day << "일" << endl;
}
```

예제

```
int main()
{
    Date date1(2009, 3, 2);    // 2009.3.2
    Date date2(2009);                    // 2009.1.1
    Date date3;                          // 2010.1.1
    date1.print();
    date2.print();
    date3.print();
    return 0;
}
```

2009년 3월 2일
2009년 1월 1일
2010년 1월 1일

예제

- Time 클래스에 생성자와 소멸자를 추가

예제

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

class Time {
private:
	int hour; // 0 - 23
	int minute; // 0 - 59
	int second; // 0 - 59
public:
	Time();
		// 생성자
	Time(int h, int m, int s);
	void setTime(int h, int m, int s);
	void print();
};
// 첫 번째 생성자
Time::Time()
{
	setTime(0, 0, 0);
}
```

예제

```
// 두 번째 생성자
Time::Time(int h, int m, int s)
{
    setTime(h, m, s);
}

// 시간 설정 함수
void Time::setTime(int h, int m, int s)
{
    hour = ((h >= 0 && h < 24) ? h : 0); // 시간 검증
    minute = ((m >= 0 && m < 60) ? m : 0); // 분 검증
    second = ((s >= 0 && s < 60) ? s : 0); // 초 검증
}

// "시:분:초" 의 형식으로 출력
void Time::print()
{
    cout << hour << ":" << minute << ":" << second << endl;
}
```

예제

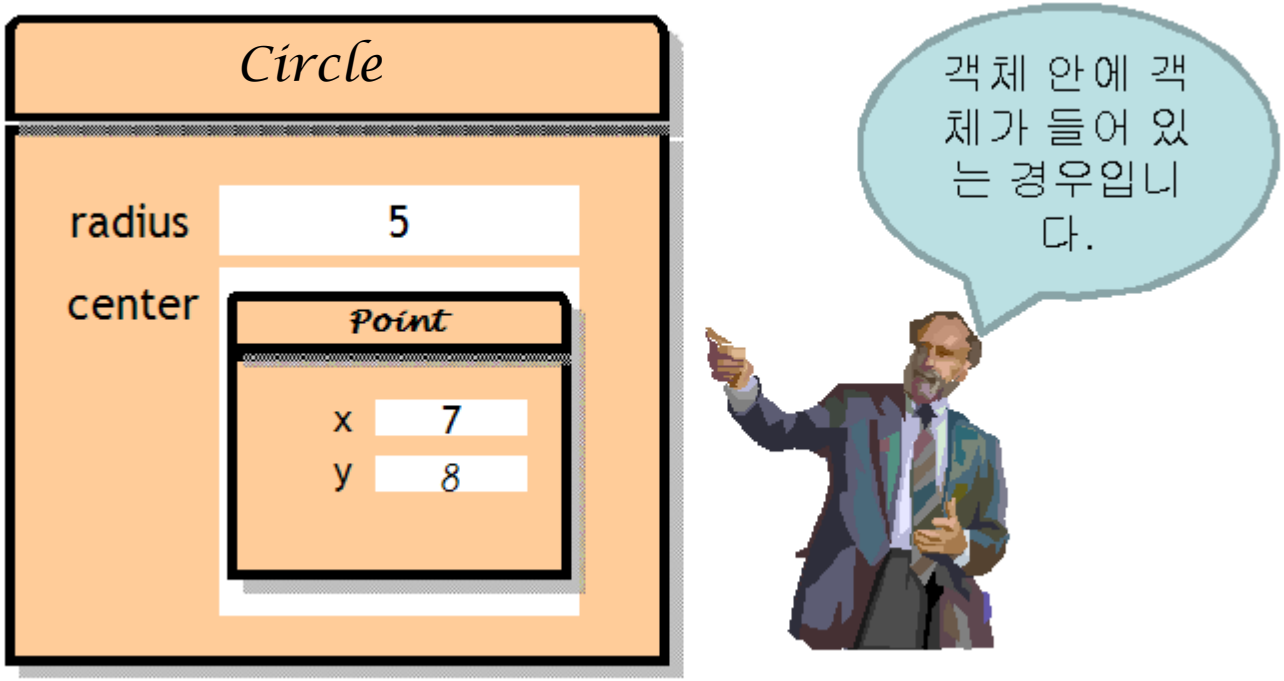
```
int main()
{
    Time time1;

    cout << "기본 생성자 호출 후 시간:  ";
    time1.print();

    // 두 번째 생성자 호출
    Time time2(13, 27, 6);
    cout << "두번째 생성자 호출 후 시간: ";
    time2.print();

    // 올바르지 않은 시간으로 설정해본다.
    Time time3(99, 66, 77);
    cout << "올바르지 않은 시간 설정 후 시간: ";
    time3.print();

    return 0;
}
```

예 제

기본 생성자 호출 후 시간: 0:0:0
두번째 생성자 호출 후 시간: 13:27:6
올바르지 않은 시간 설정 후 시간: 0:0:0

예제

- Circle 객체 안에 Point 객체가 들어 있는 경우



예제

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

class Point {
private:
	int x;
	int y;
public:
	Point();
	Point(int a, int b);
	void print();
};

Point::Point() : x(0), y(0)
{
}

Point::Point(int a, int b) : x(a), y(b)
{
}
```

예제

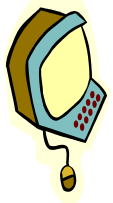
```
void Point::print()
{
    cout << "( " << x << ", " << y << " )\n";
}

class Circle {
private:
    int radius;
    Point center; // Point 객체가 멤버 변수로 선언되어 있다.
public:
    Circle();
    Circle(int r);
    Circle(Point p, int r);
    Circle(int x, int y, int r);
    void print();
};

// 생성자
Circle::Circle(): radius(0), center(0, 0)
{
}
```

예제

```
Circle::Circle(int r) : radius(r), center(0, 0)
{
}
Circle::Circle(Point p, int r) : radius(r), center(p)
{
}
Circle::Circle(int x, int y, int r) : radius(r), center(x, y)
{
}
void Circle::print()
{
    cout << "중심: ";
    center.print();
    cout << "반지름: " << radius << endl << endl;
}
```

예 제

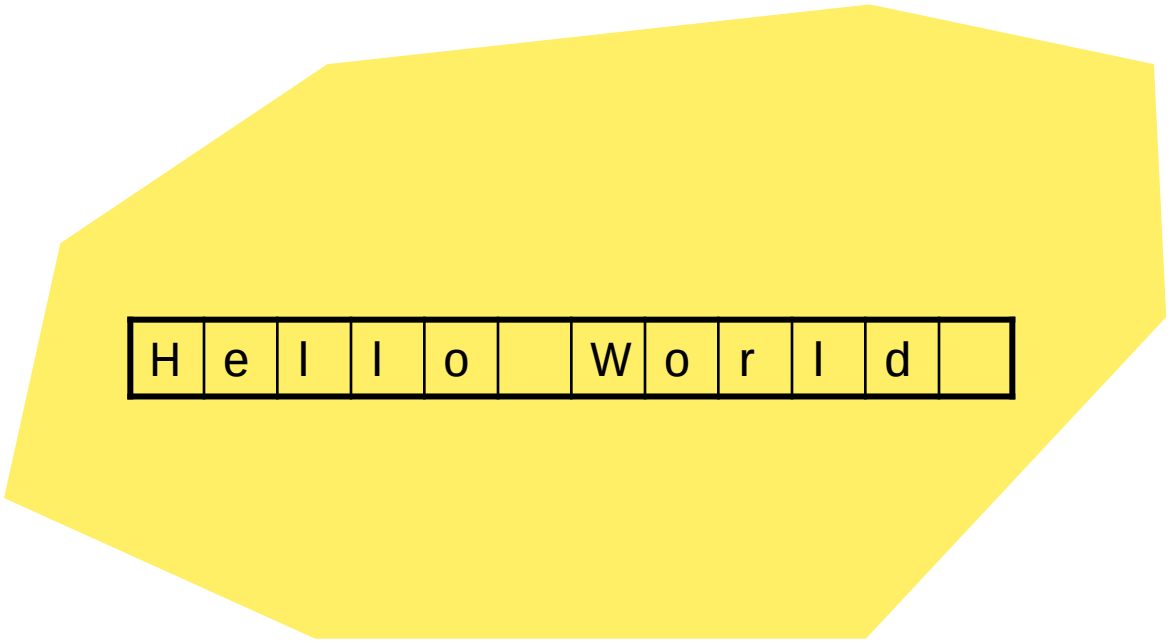
```
int main()
{
    Point p(5, 3);

    Circle c1;
    Circle c2(3);
    Circle c3(p, 4);
    Circle c4(9, 7, 5);

    c1.print();
    c2.print();
    c3.print();
    c4.print();
    return 0;
}
```



```
중심: ( 0, 0 )
반지름: 0
중심: ( 0, 0 )
반지름: 3
중심: ( 5, 3 )
반지름: 4
중심: ( 9, 7 )
반지름: 5
```

예제

- 문자열을 클래스로 작성해보자.



예제

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

class MyString
{
private:
    char *pBuf;                 //동적으로 할당된 메모리의 주소값 저장

public:
    MyString(const char *s=NULL);
    MyString(MyString& s);
    ~MyString();

    void print();      // 문자열을 화면에 출력
    int getSize();     // 문자열의 길이 반환
};
```

예제

```
// 생성자
MyString::MyString(const char *s)
{
    if( s == NULL )
    {
        pBuf = new char[1];
        pBuf[0] = NULL;
    }
    else
    {
        pBuf = new char[::strlen(s)+1];
        strcpy(pBuf, s);
    }
}
// 복사 생성자
MyString::MyString(MyString &s)
{
    pBuf = new char[s.getSize()+1];
    strcpy(pBuf, s.pBuf);
}
```

예제

```
// 소멸자
MyString::~MyString()
{
    if ( pBuf )
        delete [] pBuf;
}

void MyString::print()
{
    cout << pBuf << endl;
}

int MyString::getSize()
{
    return strlen(pBuf);
}
```

예제

```
int main() {

    MyString str1;
    MyString str2("Hello");
    MyString str3 = "World!";
    MyString str4(str3);

    str1.print();
    str2.print();
    str3.print();
    str4.print();

    return 0;
}
```

```
Hello
World!
World!
```

중간 점검 문제

1. MyString 클래스에 두 개의 문자열을 합하는 멤버 함수인 add(String& s)를 추가하여 보자.

2. MyString 클래스에서 변환 생성자를 이용하여서 묵시적인 변환이 일어나지 못하도록 explicit를 다음과 같이 생성자에 추가한 후에 코드를 다시 컴파일하여 보자. 컴파일 오류가 발생하는 부분은 어디인가?

```
explicit MyString(const char *s=NULL);
```

Q & A