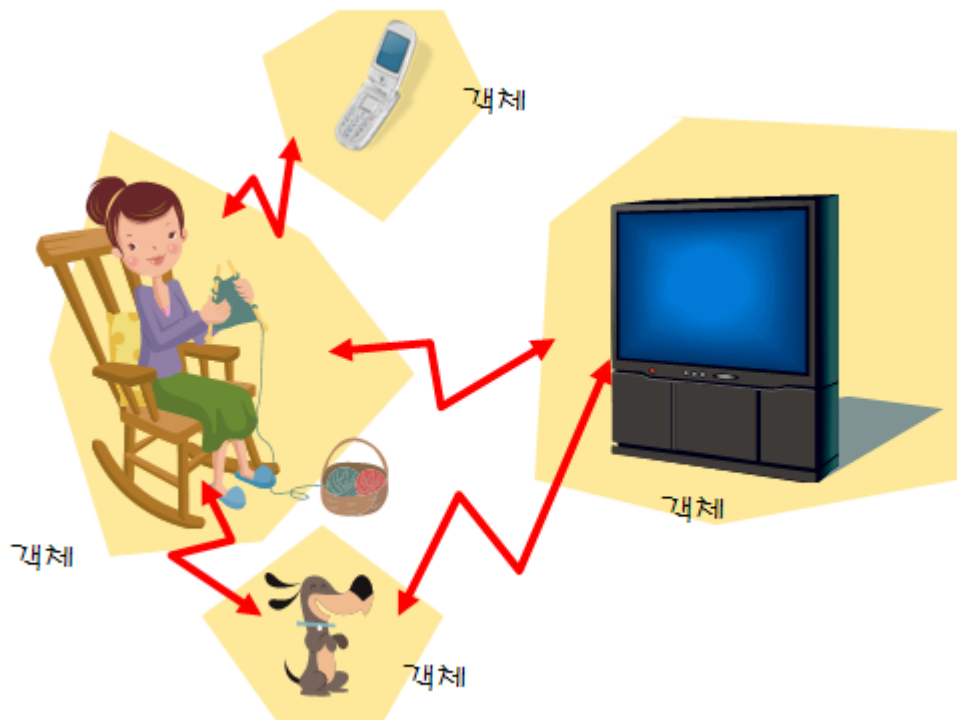




*C++ Espresso*

## 제6장 생성자와 소멸자





# 이번 장에서 학습할 내용



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

객체가 생성될  
때 초기화를  
담당하는  
생성자에 대하여  
살펴봅니다.





# 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; // 디폴트생성자호출
```

```
    return 0;
```

```
}
```

C:\Windows\system32\cmd.exe

디폴트 생성자 호출  
계속하려면 아무 키나 누르십시오 . . .



c1

speed	0
gear	1
color	"white"

## 생성자의 외부 정의

\* 이전에 실습한 코드를 수정해보자 ~

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

class Car {
private:
    int speed; // 속도
    int gear; // 기어
    string color; // 색상

public:
    Car();
};

int main()
{
    Car c1; // 디폴트생성자호출
    return 0;
}
```

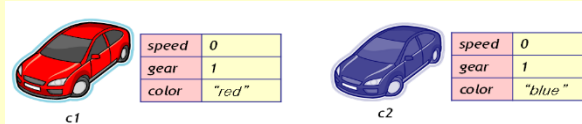
```
Car::Car()
{
    cout << "디폴트 생성자 호출" << endl;
    speed = 0;
    gear = 1;
    color = "white";
}
```

# 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;
        }
        void print()
        {
            cout << "===== " << endl;
            cout << "속도: " << speed << endl;
            cout << "기어: " << gear << endl;
            cout << "색상: " << color << endl;
            cout << "===== " << endl;
        }
};
```



```
int main()
```

```
{
    Car c1(0, 1, "red"); // 생성자 호출
    Car c2(0, 1, "blue"); // 생성자 호출
    c1.print();
    c2.print();
    return 0;
}
```

```
C:\Windows\system32\cmd.exe

=====
속도: 0
기어: 1
색상: red
=====
속도: 0
기어: 1
색상: blue
=====
계속하려면 아무 키나 누르십시오 . . .
```



# 생성자의 중복 정의

- 생성자도 메소드이므로 **중복 정의**가 가능하다.

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

class Car {
private:
    int speed; // 속도
    int gear; // 기어
    string color; // 색상

public:
    Car();
    Car(int s, int g, string c);
};
```

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

Car::Car(int s, int g, string c)
{
    cout << "매개변수가 있는 생성자 호출" << endl;
    speed = s;
    gear = g;
    color = c;
}

int main()
{
    Car c1;
    Car c2(100, 0, "blue");
    return 0;
}
```

C:\Windows\system32\cmd.exe

```
디폴트 생성자 호출
매개변수가 있는 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

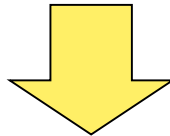
    Car c3(100, 3, "white"); // ③생성자 호출
    Car c4 = Car(0, 1, "blue");
    // ④이것은 먼저 임시 객체를 만들고 이것을 c4에 복사
    return 0;
}
```



# 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")
```

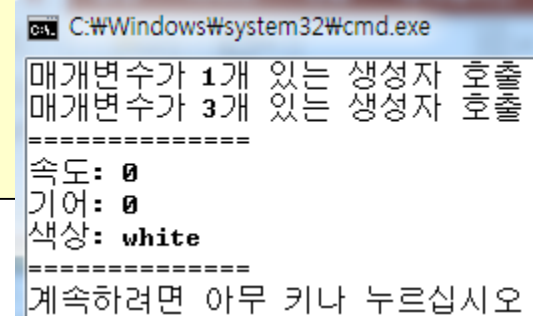
```
{  
    speed = s;  
    gear = g;  
    color = c;  
}
```

-> 디폴트 생성자를 정의한 것과 같은 효과를 낸다.

## 생성자에서 다른 생성자 호출하기

```
#include<iostream>
#include<string>
using namespace std;
class Car {
private:
    int speed;
    int gear;
    string color;
public:
    Car(int s, int g, string c);
    Car(string c);
    void print();
};
Car::Car(int s, int g, string c)
{
    cout << "매개변수가 3개 있는 생성자 호출"
    << endl;
    speed = s;
    gear = g;
    color = c;
}
```

```
Car::Car(string c)
{
    cout << "매개변수가 1개 있는 생성자 호출"
    << endl;
    new (this) Car(0, 0, c);
}
void Car::print()
{
    cout << "=====" << endl;
    cout << "속도: " << speed << endl;
    cout << "기어: " << gear << endl;
    cout << "색상: " << color << endl;
    cout << "=====" << endl;
}
int main()
{
    Car c1("white");
    c1.print();
    return 0;
}
```



```
C:\Windows\system32\cmd.exe
매개변수가 1개 있는 생성자 호출
매개변수가 3개 있는 생성자 호출
=====
속도: 0
기어: 0
색상: white
=====
계속하려면 아무 키나 누르십시오 . . .
```



## 중간 점검 문제

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





# 소멸자

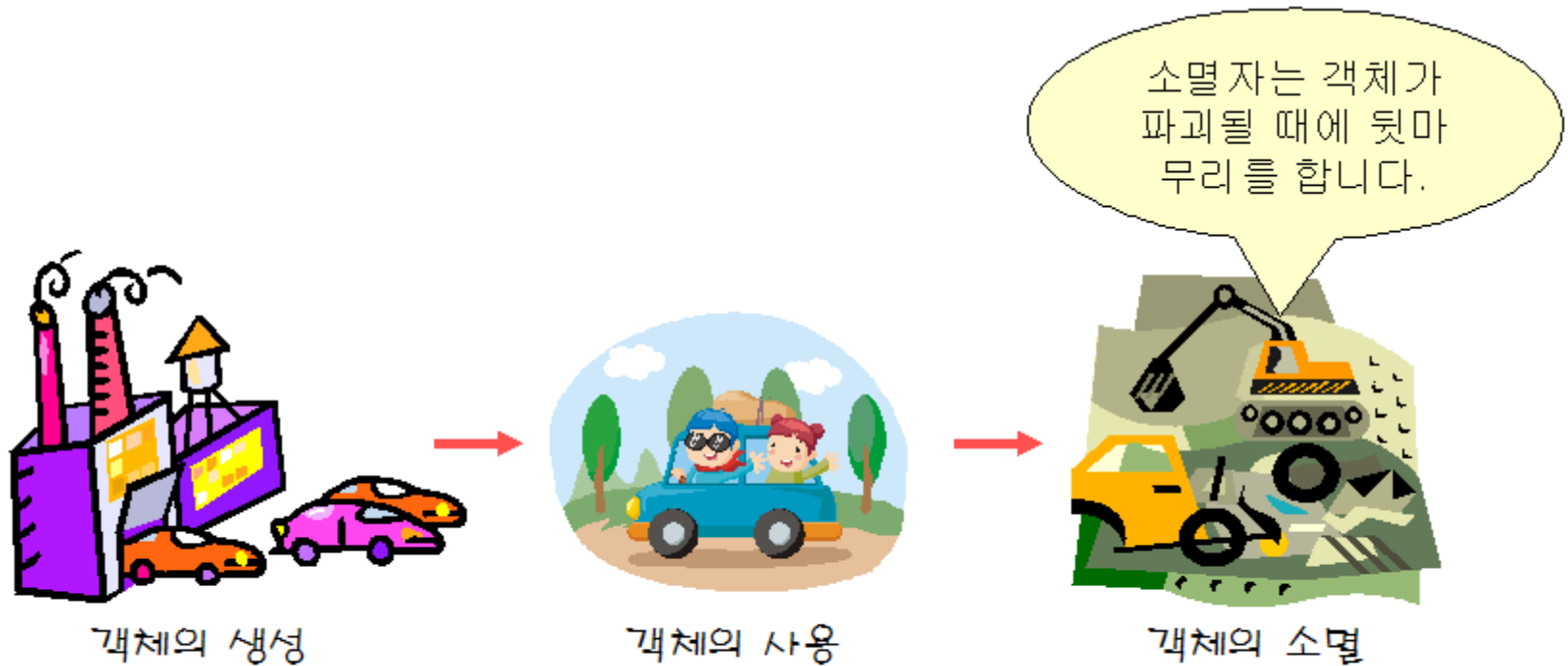


그림 10.4 소멸자의 개념



## 소멸자의 특징

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

```
class Car  
{
```

```
public:
```

```
    ~Car()
```

```
{
```

```
    ...
```

```
}
```

```
};
```

소멸자



# 소멸자



```
class Car {  
private:  
    int speed;           // 속도  
    int gear;            // 주행거리  
    string color;        // 색상
```

public:

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

생성자

```
    ~Car()  
    {  
        cout << "소멸자 호출" << endl;  
    }
```

소멸자

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

C:\Windows\system32\cmd.exe

```
생성자 호출  
소멸자 호출  
계속하려면 아무 키나 누르십시오 . . .
```



## 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?



## 멤버 초기화 목록

- 멤버 변수를 간단히 초기화할 수 있는 형식
- 이전에 작성한 코드를 수정해봅시다 ~

```
Car(int s, int g, string c) : speed(s), gear(g), color(c) {  
    cout << "생성자 호출" << endl;  
}
```



# 상수 멤버의 초기화

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

상수 멤버의 초기화는  
이렇게,



# 객체 멤버의 경우

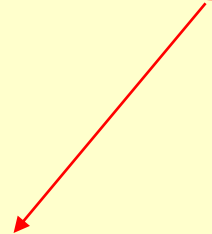


```
#include <iostream>
#include <string>
using namespace std;
```

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

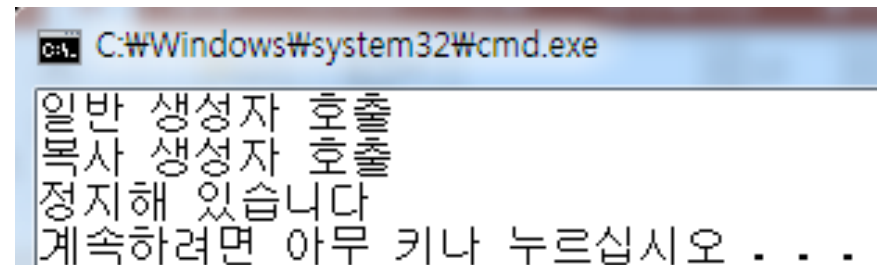
```
class Rectangle
{
    Point p1, p2;
public:
    Rectangle(int x1, int y1, int x2, int y2) : p1(x1, y2), p2(x2, y2)
    {
    }
};
```

생성자 호출



## 예제

- 다음 실행결과가 출력되는 프로그램을 만들어보자.
- Car 클래스 : 전용변수(speed, gear, color)  
    생성자(3개의 매개변수를 가진 일반 생성자, 복사 생성자)  
    speed값을 반환하는 접근자
- isMoving 함수 : 반환형(void), 전달받는매개변수(Car obj)  
    기능(객체 obj의 speed값이 0보다 크면 “움직이고 있습니다 “ 출력  
    그렇지 않으면 “정지해 있습니다 “ 출력)
- main 함수 : Car객체 “c” 생성(speed(0), gear(1), color( “white” ))  
    isMoving 함수 호출(객체 c를 매개변수로 전달)



```
C:\Windows\system32\cmd.exe
일반 생성자 호출
복사 생성자 호출
정지해 있습니다
계속하려면 아무 키나 누르십시오 . . .
```



## 예제(정답)

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  class Car {
5      int speed;
6      int gear;
7      string color;
8  public:
9      Car(int s, int g, string c)
10     {
11         cout << "일반 생성자 호출" << endl;
12         speed = s;
13         gear = g;
14         color = c;
15     }
16     Car(const Car &obj): speed(obj.speed), gear(obj.gear), color(obj.color)
17     {
18         cout << "복사 생성자 호출" << endl;
19     }
20     int getSpeed()
21     {
22         return speed;
23     }
24 };

25 void isMoving(Car obj)
26 {
27     if (obj.getSpeed() > 0)
28         cout << "움직이고 있습니다" << endl;
29     else
30         cout << "정지해 있습니다" << endl;
31 }
32 int main()
33 {
34     Car c(0, 1, "white");
35     isMoving(c);
36     return 0;
37 }
```





## 중간 점검 문제

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

