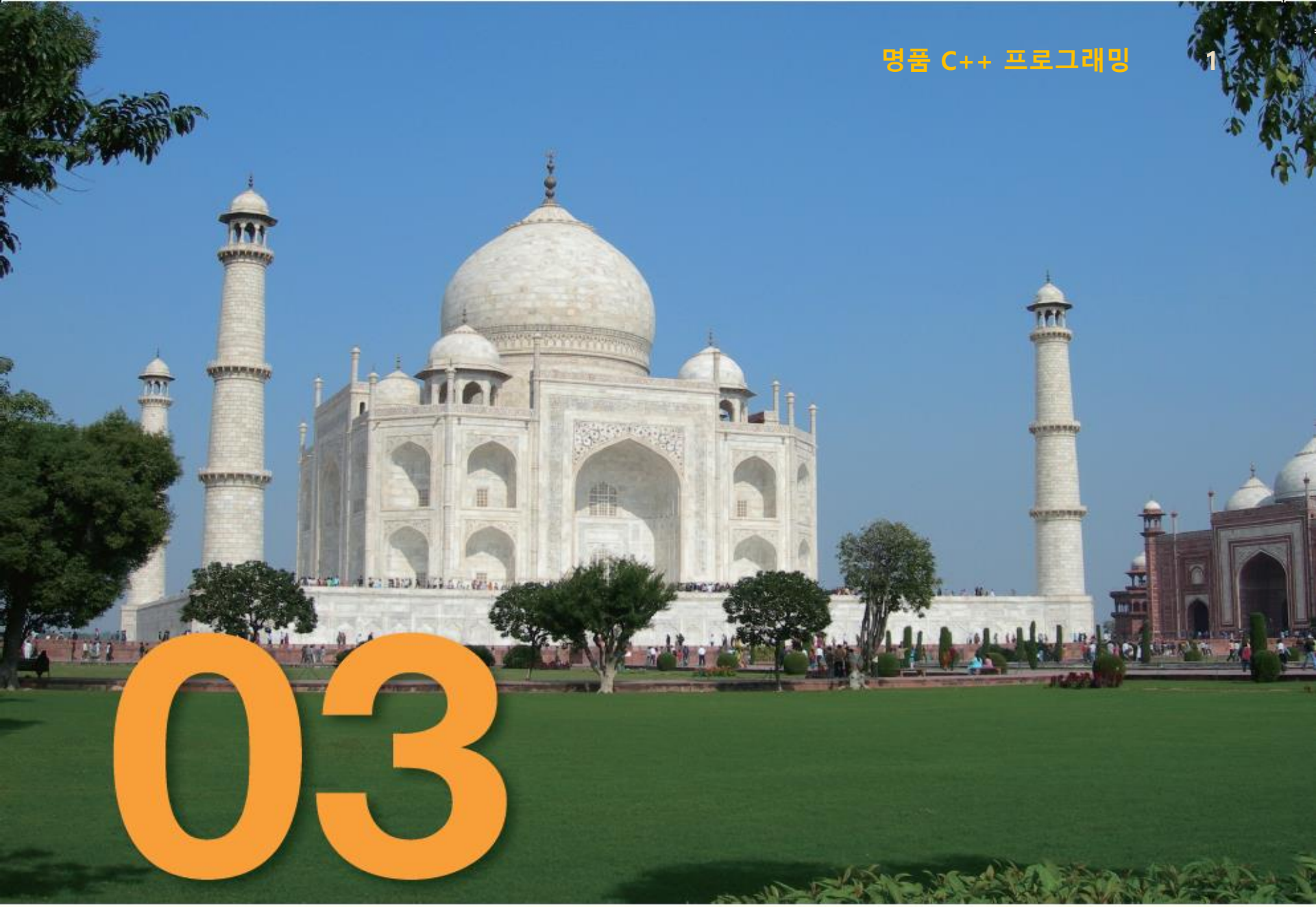


03

클래스와 객체

학습 목표

1. 실세계의 객체와 C++ 객체에 대해 이해한다.
2. C++ 클래스를 작성할 수 있다.
3. 객체를 생성하고 활용할 수 있다.
4. 생성자와 소멸자를 알고 작성할 수 있다.
5. private, protected, public 접근 지정자를 이해한다.
6. 인라인 함수의 목적을 이해하고 활용할 수 있다.
7. C++ 구조체를 작성하고, 클래스와의 차이점을 안다.
8. 헤더 파일과 cpp 파일을 분리하여 C++ 프로그램을 작성할 수 있다.

세상의 모든 것이 객체이다.

- 세상 모든 것이 객체

TV

의자

책

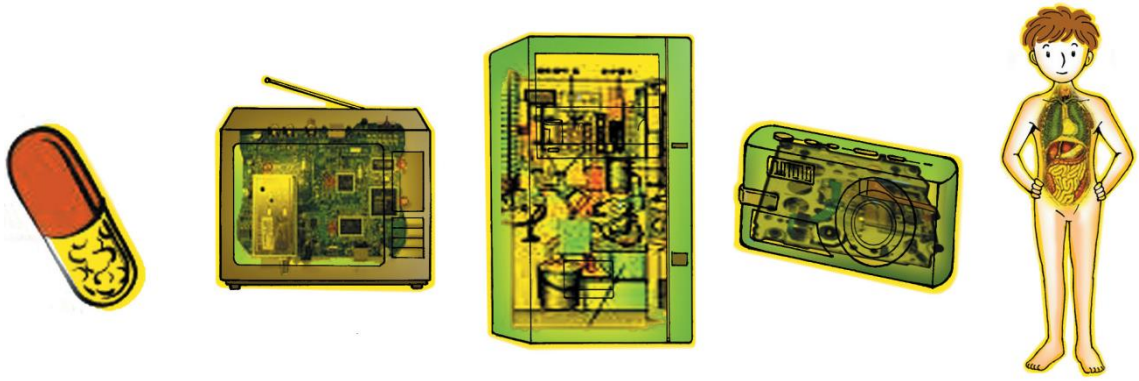
집

카메라

컴퓨터

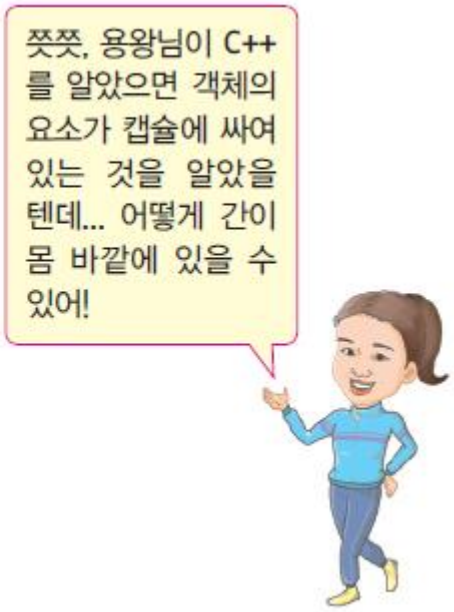
객체는 캡슐화된다.

- 캡슐화(encapsulation)
  - 객체의 본질적인 특성
  - 객체를 캡슐로 싸서 그 내부를 보호하고 볼 수 없게 함
    - 캡슐에 든 약은 어떤 색인지 어떤 성분인지 보이지 않고, 외부로부터 안전
  - 캡슐화 사례



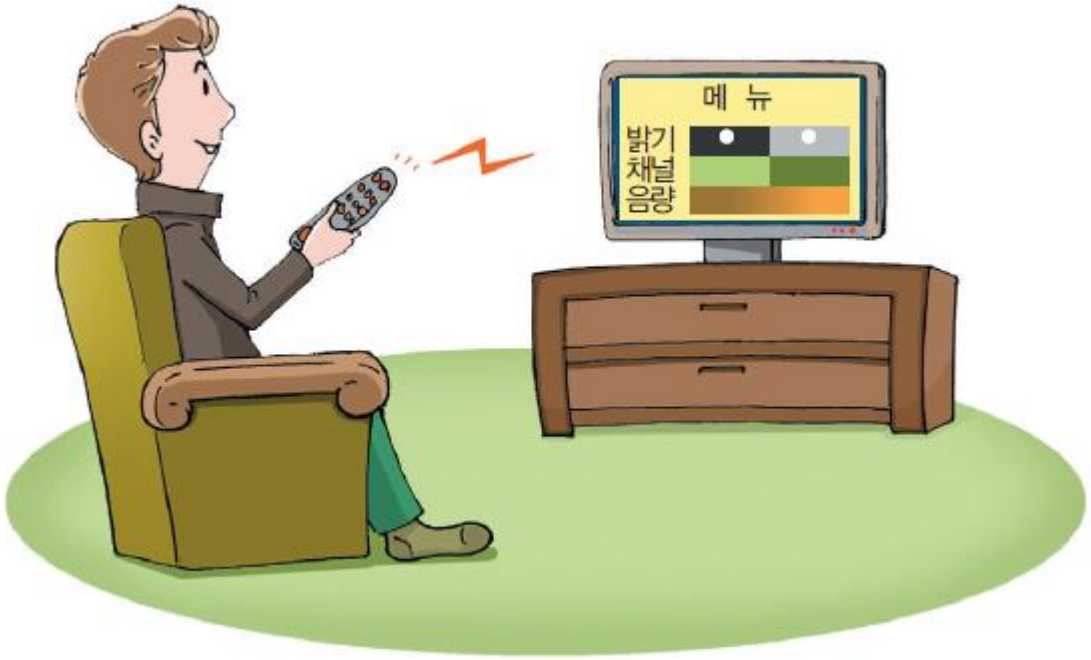
  - 캡슐화의 목적
    - 객체 내 데이터에 대한 보안, 보호, 외부 접근 제한

토끼의 간과 객체의 캡슐화



객체의 일부 요소는 공개된다.

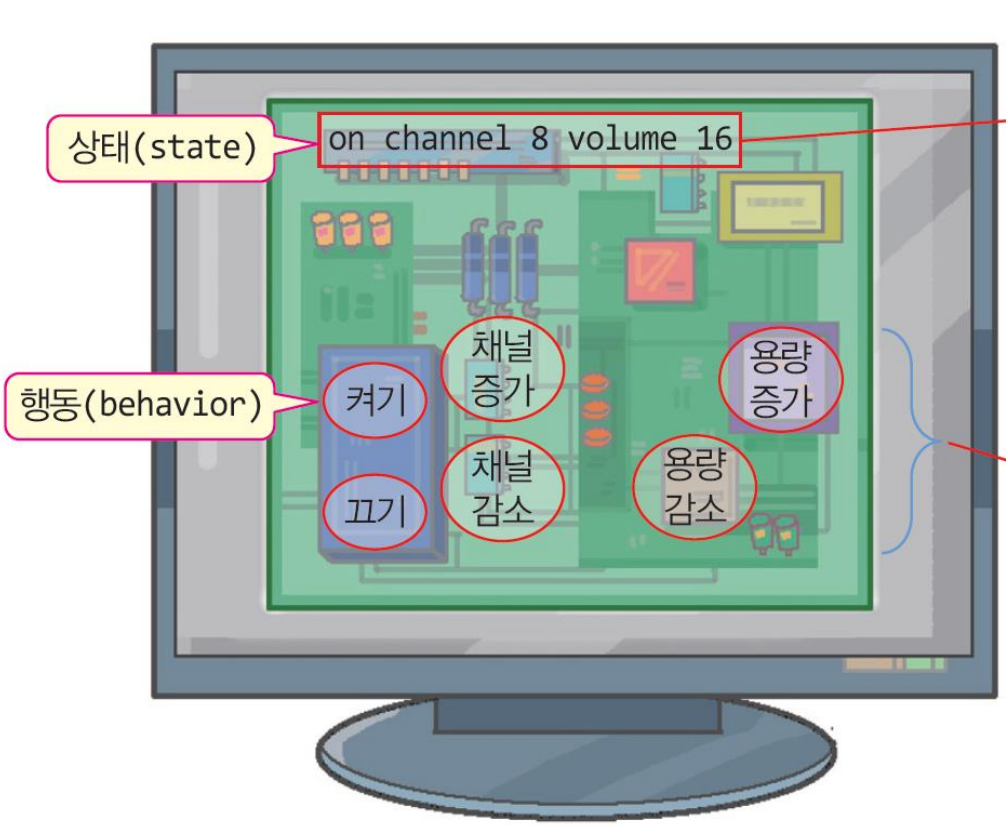
- 객체의 일부분 공개
  - 외부와의 인터페이스(정보 교환 및 통신)를 위해 객체의 일부분 공개
  - TV 객체의 경우, On/Off 버튼, 밝기 조절, 채널 조절, 음량 조절 버튼 노출. 리모콘 객체와 통신하기 위함



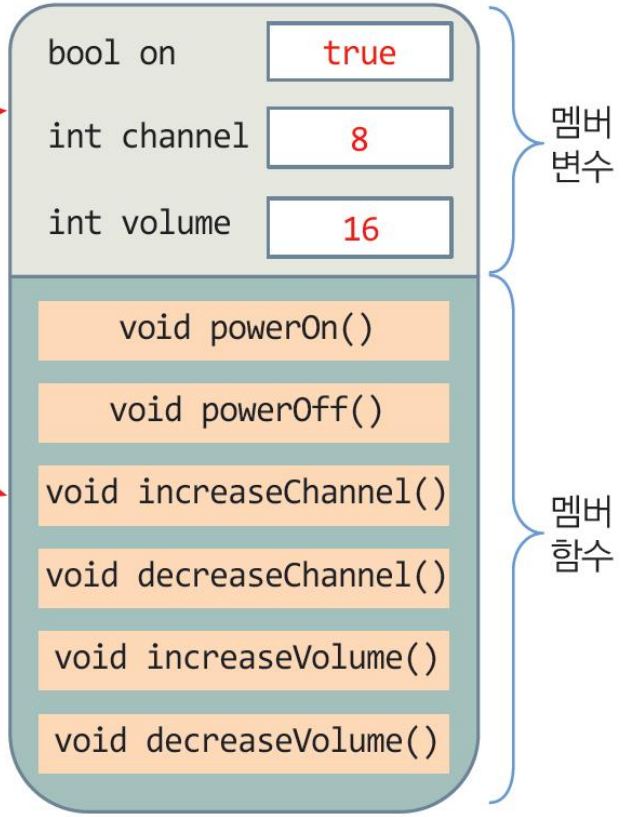
C++ 객체는 멤버 함수와 멤버 변수로 구성된다.

- 객체는 상태(state)와 행동(behavior)으로 구성
- TV 객체 사례
  - 상태
    - on/off 속성 – 현재 작동 중인지 표시
    - 채널(channel) – 현재 방송중인 채널
    - 음량(volume) – 현재 출력되는 소리 크기
  - 행동
    - 켜기(power on)
    - 끄기(power off)
    - 채널 증가(increase channel)
    - 채널 감소(decrease channel)
    - 음량 증가(increase volume)
    - 음량 줄이기(decrease volume)

TV와 C++로 설계된 TV 객체



실세계의 TV 객체

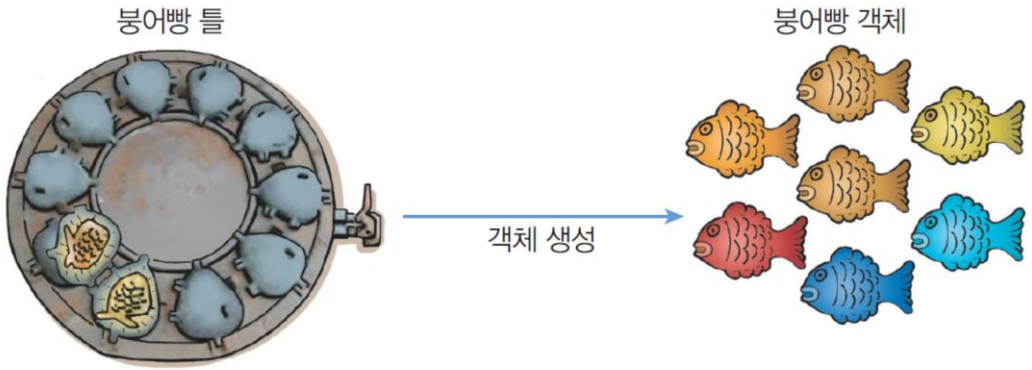


C++로 설계한 TV 객체

C++클래스와 C++객체

- 클래스
  - 객체를 만들어내기 위해 정의된 설계도, 틀
  - 클래스는 객체가 아님. 실체도 아님
  - 멤버 변수와 멤버 함수 선언
- 객체
  - 객체는 생성될 때 클래스의 모양을 그대로 가지고 탄생
  - 멤버 변수와 멤버 함수로 구성
  - 메모리에 생성, 실체(instance)라고도 부름
  - 하나의 클래스 틀에서 찍어낸 여러 개의 객체 생성 가능
  - 객체들은 상호 별도의 공간에 생성

클래스와 객체 관계



(a) 붕어빵 틀과 붕어빵 객체들

```
class TV {
   bool on;
   int channel;
   int volume;
public:
   void powerOn() { ... }
   void powerOff() { ... }
   void increaseChannel() { ... }
   void decreaseChannel() { ... }
   void increaseVolume() { ... }
   void decreaseVolume() { ... }
};
```

TV 클래스

객체 생성

객체들은 서로 별도의 영역에 멤버들을 만든다.

TV 객체 1	
bool on	true
int channel	8
int volume	16
void powerOn()	
void powerOff()	
void increaseChannel()	
void decreaseChannel()	
void increaseVolume()	
void decreaseVolume()	

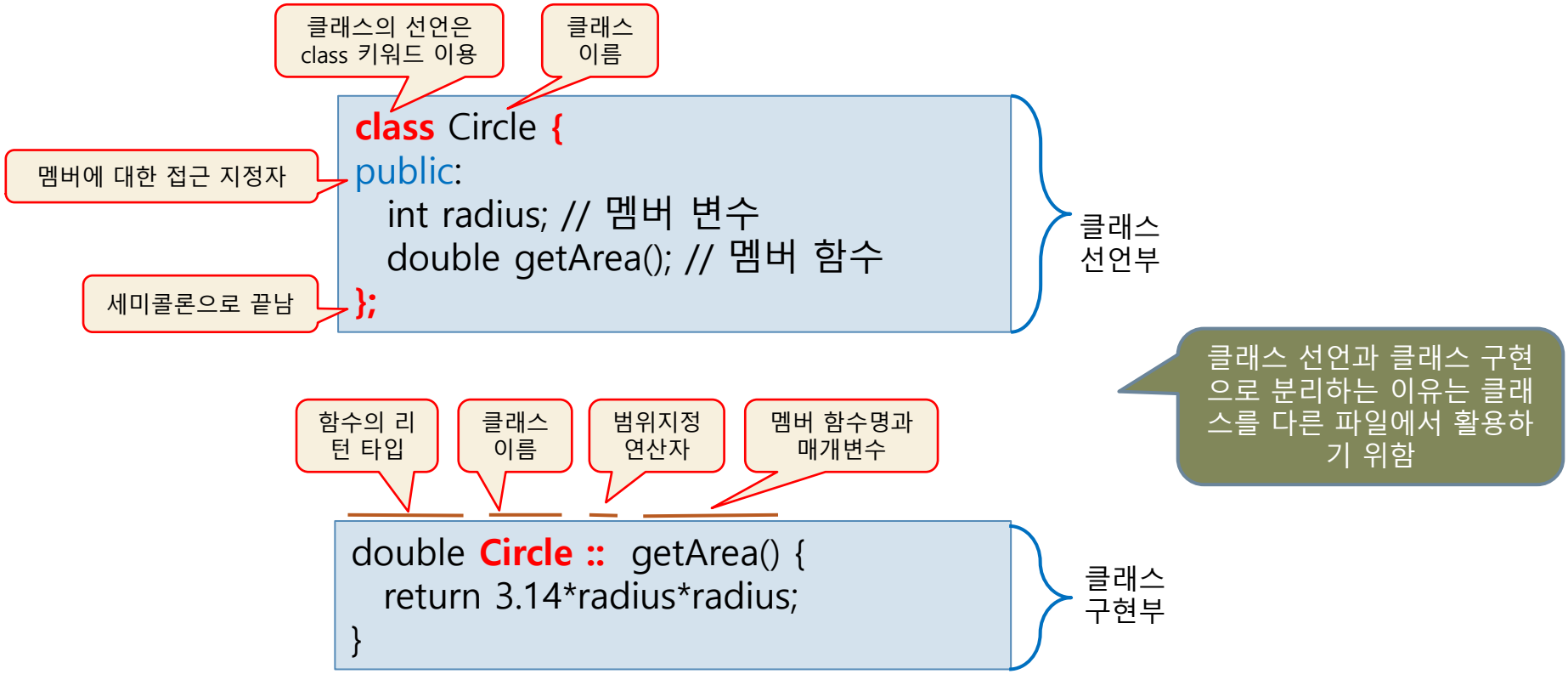
TV 객체 2	
bool on	false
int channel	13
int volume	5
void powerOn()	
void powerOff()	
void increaseChannel()	
void decreaseChannel()	
void increaseVolume()	
void decreaseVolume()	

(b) C++로 표현한 TV 클래스와 TV 객체들

C++ 클래스 만들기

- 클래스 작성
  - 멤버 변수와 멤버 함수로 구성
  - 클래스 선언부와 클래스 구현부로 구성
- 클래스 선언부(class declaration)
  - class 키워드를 이용하여 클래스 선언
  - 멤버 변수와 멤버 함수 선언
    - 멤버 변수는 클래스 선언 내에서 초기화할 수 없음
    - 멤버 함수는 원형(prototype) 형태로 선언
  - 멤버에 대한 접근 권한 지정
    - private, public, protected 중의 하나
    - 디폴트는 private
    - public : 다른 모든 클래스나 객체에서 멤버의 접근이 가능함을 표시
- 클래스 구현부(class implementation)
  - 클래스에 정의된 모든 멤버 함수 구현

클래스 만들기 설명


클래스의 선언은 class 키워드 이용
클래스 이름
멤버에 대한 접근 지정자
세미콜론으로 끝남
class Circle {
public:
int radius; // 멤버 변수
double getArea(); // 멤버 함수
};
클래스 선언부
클래스 선언과 클래스 구현으로 분리하는 이유는 클래스를 다른 파일에서 활용하기 위함
함수의 리턴 타입
클래스 이름
범위지정 연산자
멤버 함수명과 매개변수
double Circle :: getArea() {
return 3.14*radius*radius;
}
클래스 구현부

예제 3-1 Circle 클래스의 객체 생성 및 활용


```
#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; // donut 객체의 반지름을 1로 설정
    double area = donut.getArea(); // donut 객체의 면적 알아내기
    cout << "donut 면적은 " << area << endl;

    Circle pizza;
    pizza.radius = 30; // pizza 객체의 반지름을 30으로 설정
    area = pizza.getArea(); // pizza 객체의 면적 알아내기
    cout << "pizza 면적은 " << area << endl;
}
```

Circle 선언부

Circle 구현부

객체 donut 생성

donut의 멤버 변수 접근

donut의 멤버 함수 호출

```
donut 면적은 3.14
pizza 면적은 2826
```

객체 생성 및 활용 설명

- 객체 이름 및 객체 생성

```
Circle  donut; // 이름이 donut 인 Circle 타입의 객체 생성
```

객체의 타입. 클래스 이름

객체 이름

- 객체의 멤버 변수 접근

```
donut.radius = 1; // donut 객체의 radius 멤버 값을 1로 설정
```

객체 이름

객체 이름과 멤버 사이에 . 연산자

멤버 변수

- 객체의 멤버 함수 접근

```
double area = donut.getArea(); //donut 객체의 면적 알아내기
```

객체 이름

객체 이름과 멤버 사이에 . 연산자

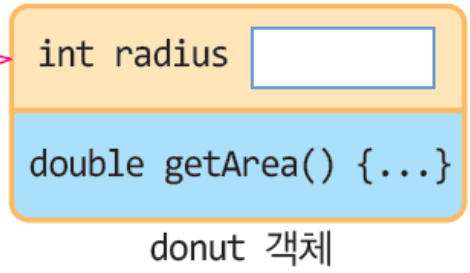
멤버 함수 호출

객체 이름과 생성, 접근 과정

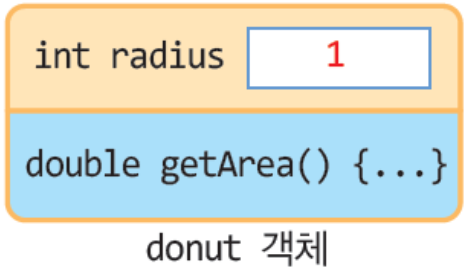

```
(1) Circle donut;
```

객체 이름

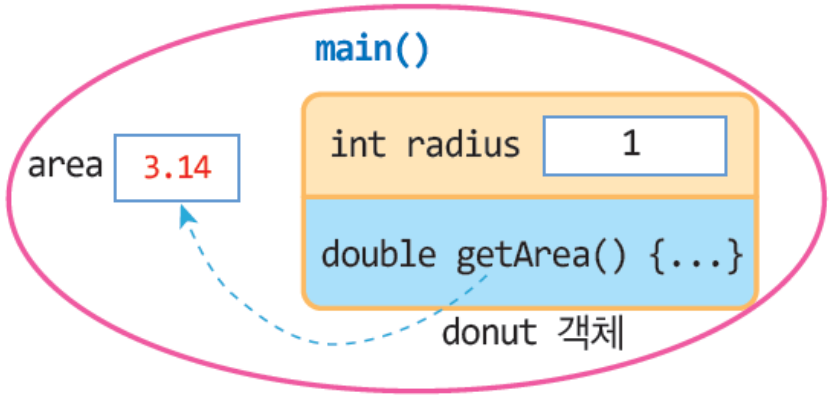
객체가 생성되면
메모리가 할당된다.




```
(2) donut.radius = 1;
```




```
(3) double area = donut.getArea();
```



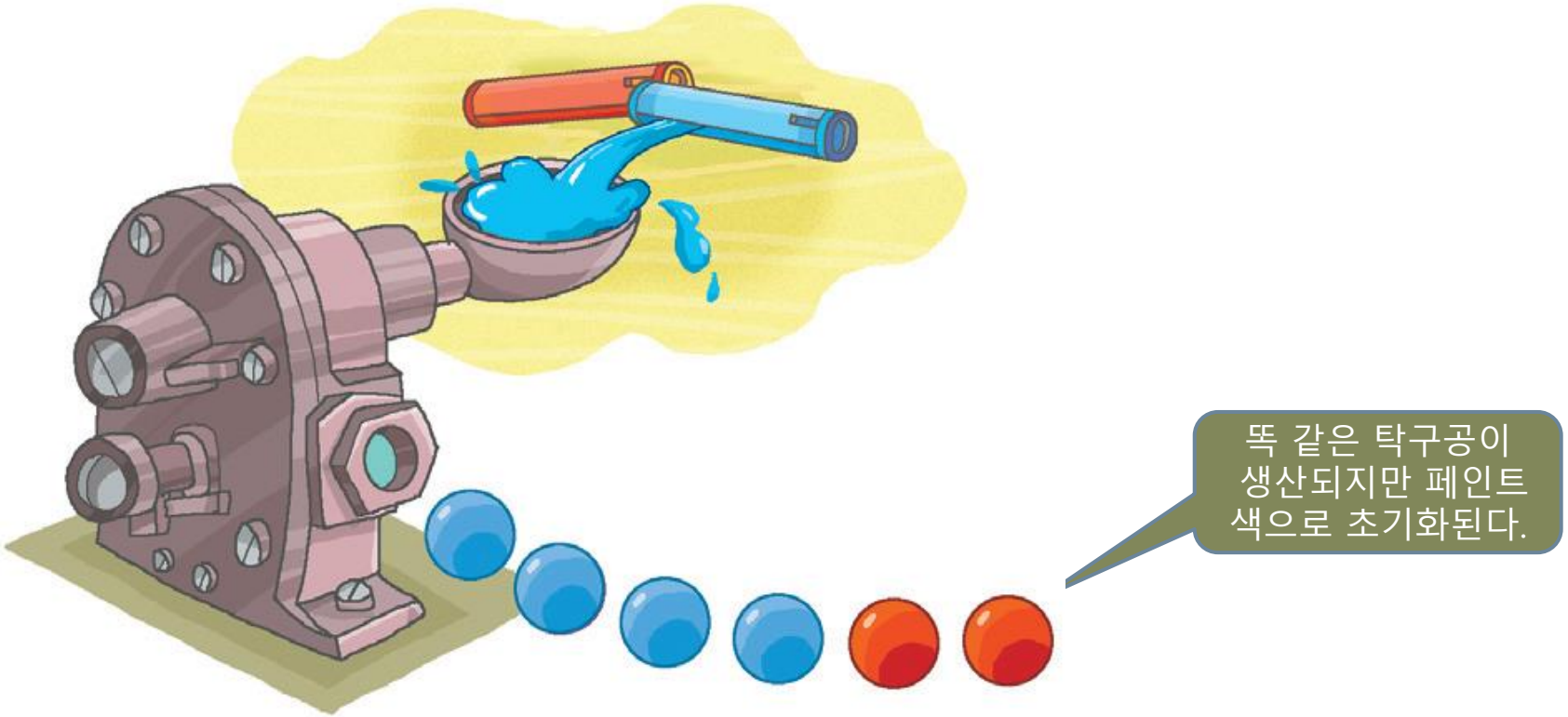
예제 3-2(실습) – Rectangle 클래스 만들기

다음 main() 함수가 잘 작동하도록 너비(width)와 높이(height)를 가지고 면적 계산 기능을 가진Rectangle 클래스를 작성하고 전체 프로그램을 완성하라.

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

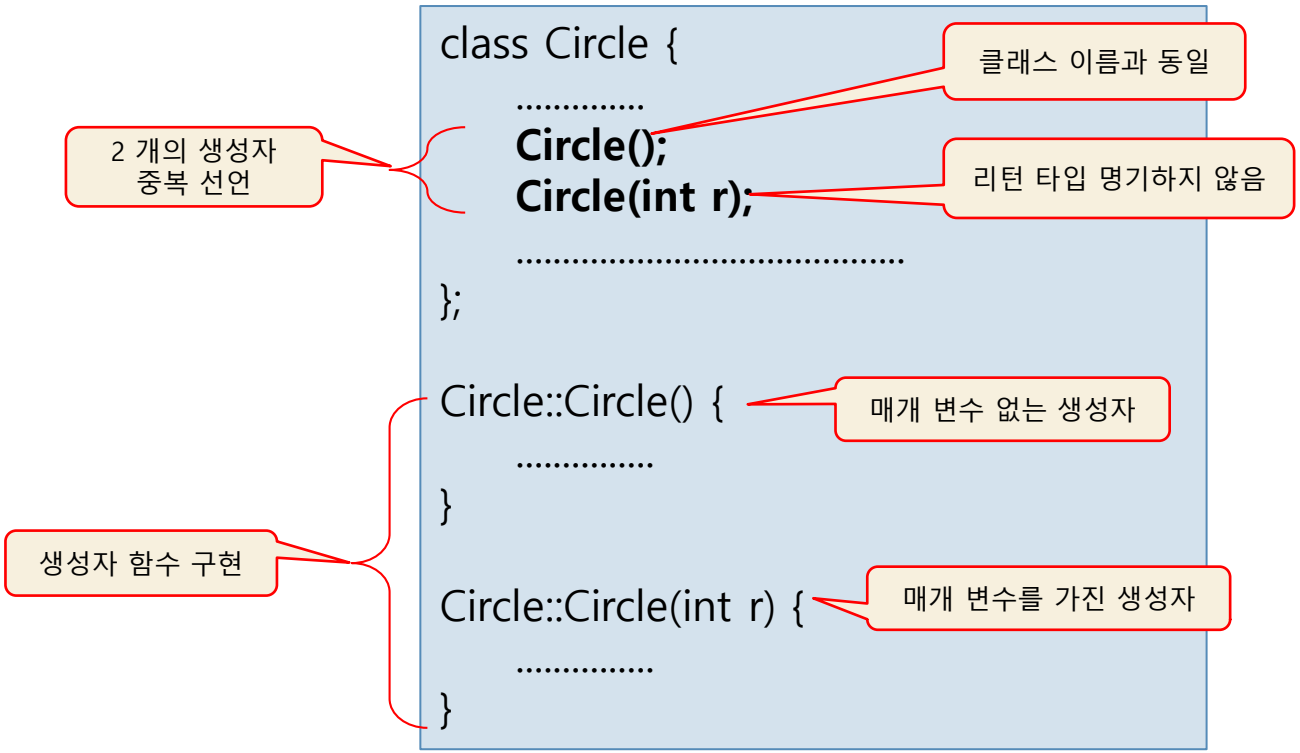
```
사각형의 면적은 15
```

탁구공 생산 장치와 생성자



생성자

- 생성자(constructor)
  - 객체가 생성되는 시점에서 자동으로 호출되는 멤버 함수
  - 클래스 이름과 동일한 멤버 함수



생성자 함수의 특징

- 생성자의 목적
  - 객체가 생성될 때 객체가 필요한 초기화를 위해
    - 멤버 변수 값 초기화, 메모리 할당, 파일 열기, 네트워크 연결 등
- 생성자 이름
  - 반드시 클래스 이름과 동일
- 생성자는 리턴 타입을 선언하지 않는다.
  - 리턴 타입 없음. void 타입도 안됨
- 객체 생성 시 오직 한 번만 호출
  - 자동으로 호출됨. 임의로 호출할 수 없음. 각 객체마다 생성자 실행
- 생성자는 중복 가능
  - 생성자는 한 클래스 내에 여러 개 가능
  - 중복된 생성자 중 하나만 실행
- 생성자가 선언되어 있지 않으면 기본 생성자 자동으로 생성
  - 기본 생성자 – 매개 변수 없는 생성자
  - 컴파일러에 의해 자동 생성

예제 3–3 2 개의 생성자를 가진 Circle 클래스


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

class Circle {
public:
    int radius;
    Circle(); // 매개 변수 없는 생성자
    Circle(int r); // 매개 변수 있는 생성자
    double getArea();
};

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

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

double Circle::getArea() {
    return 3.14*radius*radius;
}
```

Circle(); 자동 호출

30

Circle(30); 자동 호출

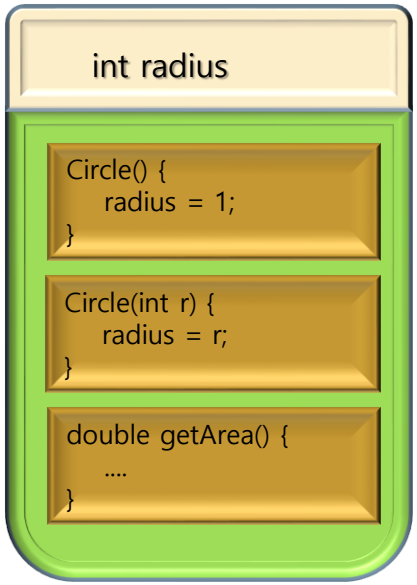
```
int main() {
    Circle donut; // 매개 변수 없는 생성자 호출
    double area = donut.getArea();
    cout << "donut 면적은 " << area << endl;

    Circle pizza(30); // 매개 변수 있는 생성자 호출
    area = pizza.getArea();
    cout << "pizza 면적은 " << area << endl;
}
```

```
반지름 1 원 생성
donut 면적은 3.14
반지름 30 원 생성
pizza 면적은 2826
```

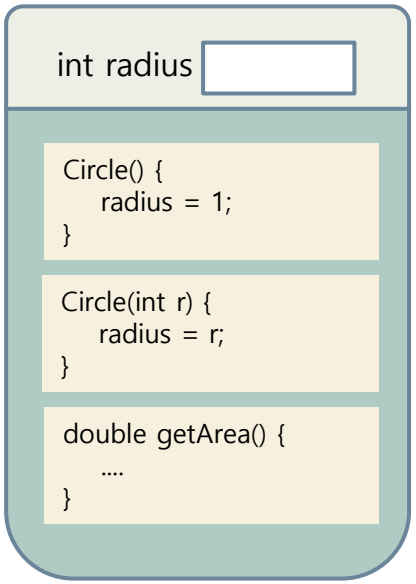
객체 생성 및 생성자 실행 과정

Circle donut;



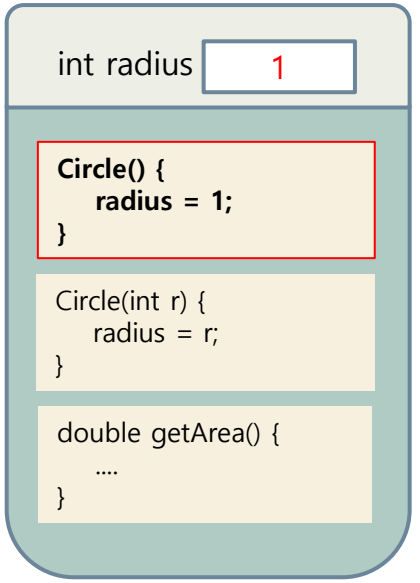
Circle 클래스

❶ 객체 공간 할당



donut 객체

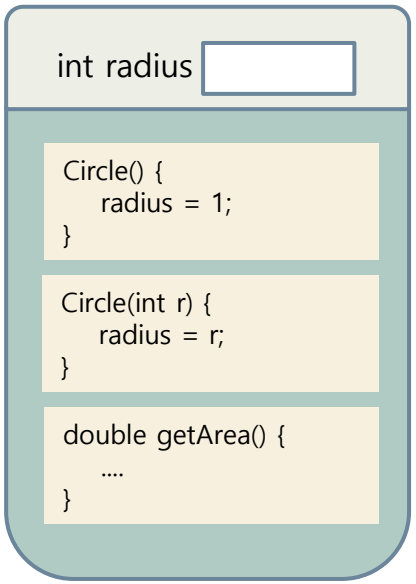
❷ 생성자 실행



donut 객체

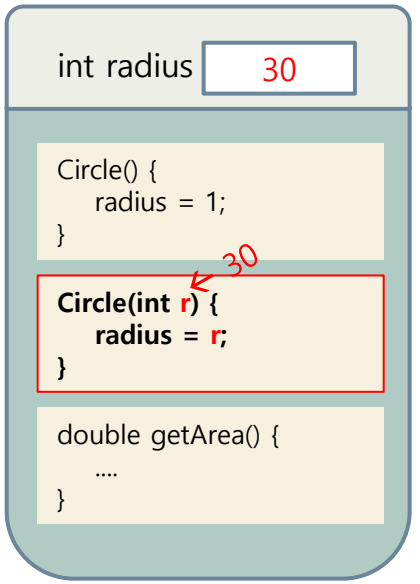
Circle pizza(30);

❶ 객체 공간 할당



pizza 객체

❷ 생성자 실행



pizza 객체

생성자가 다른 생성자 호출(위임 생성자)

- 여러 생성자에 중복 작성된 코드의 간소화
  - 타겟 생성자와 이를 호출하는 위임 생성자로 나누어 작성
    - 타겟 생성자 : 객체 초기화를 전담하는 생성자
    - 위임 생성자 : 타겟 생성자를 호출하는 생성자, 객체 초기화를 타겟 생성자에 위임

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

여러 생성자에 코드 중복

위임 생성자

타겟 생성자

```
Circle::Circle() : Circle(1) { } // Circle(int r)의 생성자 호출

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

호출

r에 1 전달

간소화된 코드

예제 3-4 생성자에서 다른 생성자 호출 연습(위임 생성자 만들기)

예제 3-3을 수정하여 객체 초기화를 전담하는 타겟 생성자와 타겟 생성자에 개체 초기화를 위임하는 위임 생성자로 재작성하라.

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

class Circle {
public:
    int radius;
    Circle(); // 위임 생성자
    Circle(int r); // 타겟 생성자
    double getArea();
};

Circle::Circle() : Circle(1) { } // 위임 생성자

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

double Circle::getArea() {
    return 3.14*radius*radius;
}
```

호출

호출. r에 1 전달

```
int main() {
    Circle donut; // 매개 변수 없는 생성자 호출
    double area = donut.getArea();
    cout << "donut 면적은 " << area << endl;

    Circle pizza(30); // 매개 변수 있는 생성자 호출
    area = pizza.getArea();
    cout << "pizza 면적은 " << area << endl;
}
```

```
반지름 1 원 생성
donut 면적은 3.14
반지름 30 원 생성
pizza 면적은 2826
```

다양한 생성자의 멤버 변수 초기화 방법


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

(1) 생성자 코드에서 멤버 변수 초기화

```
Point::Point() { x = 0; y = 0; }
Point::Point(int a, int b) { x = a; y = b; }
```

(2) 생성자 서두에 초깃값으로 초기화

```
Point::Point() : x(0), y(0) { // 멤버 변수 x, y를 0으로 초기화
}
Point::Point(int a, int b)    // 멤버 변수 x=a로, y=b로 초기화
    : x(a), y(b) {            // 콜론(:) 이하 부분을 밑줄에 써도 됨
}
```

(3) 클래스 선언부에서 직접 초기화

```
class Point {
    int x=0, y=0; // 클래스 선언부에서 x, y를 0으로 직접 초기화
public:
    ...
};
```

예제 3-5 멤버변수의 초기화와 위임 생성자 활용

다음 Point 클래스의 멤버 x, y를
생성자 서두에 초기값으로 초기화하고
위임 생성자를 이용하여 재작성하라.

```
class Point {
    int x, y;
public:
    Point();
    Point(int a, int b);
};
Point::Point() { x = 0; y = 0; }
Point::Point(int a, int b) { x = a; y = b; }
```

```
(0, 0)
(10, 20)
```

기본 생성자

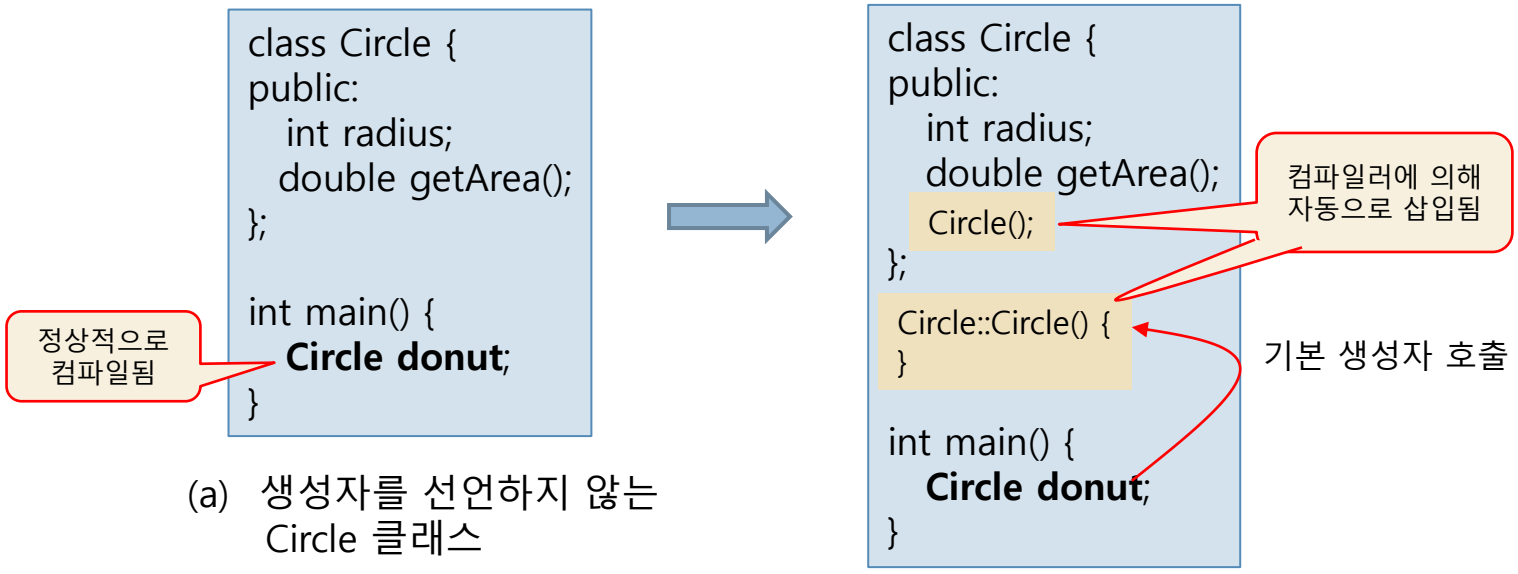
1. *생성자는 꼭 있어야 하는가?*
   - 예. C++ 컴파일러는 객체가 생성될 때, 생성자 반드시 호출
2. *개발자가 클래스에 생성자를 작성해 놓지 않으면?*
   - 컴파일러에 의해 기본 생성자가 자동으로 생성

- 기본 생성자란?
  - 클래스에 생성자가 하나도 선언되어 있지 않은 경우, 컴파일러가 대신 삽입해주는 생성자
  - 매개 변수 없는 생성자
  - 디폴트 생성자라고도 부름

```
class Circle {
   .....
   Circle(); // 기본 생성자
};
```

기본 생성자가 자동으로 생성되는 경우

- 생성자가 하나도 작성되어 있지 않은 클래스의 경우
  - 컴파일러가 기본 생성자 자동 생성



(a) 생성자를 선언하지 않는 Circle 클래스

(b) 컴파일러에 의해 기본 생성자 자동 삽입

기본 생성자가 자동으로 생성되지 않는 경우

- 생성자가 하나라도 선언된 클래스의 경우
  - 컴파일러는 기본 생성자를 자동 생성하지 않음

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

Circle::Circle(int r) {
    radius = r;
}

int main() {
    Circle pizza(30);
    Circle donut;
}
```

Circle 클래스에 생성자가 선언되어 있기 때문에, 컴파일러는 기본 생성자를 자동 생성하지 않음

호출

컴파일 오류.
기본 생성자 없음

예제 3-6(실습) – Rectangle 클래스 만들기

다음 main() 함수가 잘 작동하도록 Rectangle 클래스를 작성하고 프로그램을 완성하라. Rectangle 클래스는 width와 height의 두 멤버 변수와 3 개의 생성자, 그리고 isSquare() 함수를 가진다.

```
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;
}
```

```
rect1은 정사각형이다.
rect3는 정사각형이다.
```