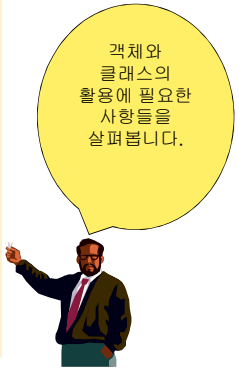
# 제7장 클래스의 활용




## 이번 장에서 학습할 내용

- 객체와 연산자
  - 대입연산자는 사용할 수 있다
  - 다른 연산자를 사용하려면?
- 객체와 함수
  - 객체가 함수의 매개변수로 전달
  - 객체의 레퍼런스가 매개변수로 전달
  - 함수가 객체를 반환
  - 객체의 포인터가 매개변수로 전달
- this
  - 객체 자신의 주소를 반환
- 정적 멤버
  - 정적 멤버 변수
  - 정적 멤버 함수
- 기타
  - 임시 객체
  - const






## 7.1 객체와 연산자

- 객체에 할당 연산자(=)를 사용할 수 있는가?

```
class Car
{
        ... //생략
};

int main()
{
        Car c1(0, 1, "white");
        Car c2(0, 1, "red");
        c1 = c2;            // 어떻게되는가?
        return 0;
}
```

OK!!!
디폴트 대입연산자
컴파일러가 제공
c2 객체의 모든
내용이 c1으로 복사됨



## 객체와 연산자

- 객체에 비교 연산자(==)를 사용할 수 있는가?

```
int main()
{
        Car c1(0, 1, "white");
        Car c2(0, 1, "red");
        if( c1 == c2 ){
                cout << "같습니다" << endl;
        }
        else {
                cout << "같지않습니다" << endl;
        }
        return 0;
}
```

NO !!!
연산자 중복이 되어
있지 않으면 오류!

- 다른 연산자들은 사용할 수 있을까?

## 중간 점검 문제

1. = 연산자를 이용하여서 하나의 객체를 다른 객체에 할당할 수 있는가 ?
2. == 연산자를 이용하여서 하나의 객체와 다른 객체를 비교할 수 있는가?

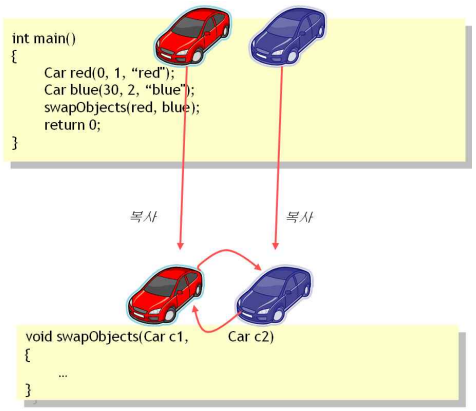


## 7.2 객체와 함수

① 객체가 함수의 매개 변수로 전달되는 경우
② 함수가 객체를 반환하는 경우
③ 객체의 포인터가 함수의 매개 변수로 전달되는 경우
④ 객체의 레퍼런스가 함수의 매개 변수로 전달되는 경우



## 객체가 함수의 매개 변수로 전달






## 객체가 함수의 매개 변수로 전달

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

class Car {
       int speed;
       int gear;
       string color;
 public:
      Car(int s=0, int g=1, string c="white"): speed(s), gear(g), color(c) { }
     void print() {
         cout << "속도: " << speed << " 기어: "
                  << gear << " 색상: " << color << endl;
     }
};
```

## 객체가 함수의 매개 변수로 전달

```
void swapObjects(Car c1, Car c2)
{
        Car tmp;
        tmp = c1; c1 = c2;   c2 = tmp;
}

int main()
{
        Car red(0, 1, "red");
        Car blue(30, 2, "blue");

        swapObjects(red, blue);
        red.print();
        blue.print();
        return 0;
}
```

복사 생성자 호출!!!

c1은 red의 복사본
c2는 blue의 복사본

복사된 객체들을 교환!

속도: 0 기어: 1 색상: red
속도: 30 기어: 2 색상: blue



## 함수가 객체를 반환

```
...// 전과동일
Car createCar()
{
        Car tmp(0, 1, "metal");
        return tmp;
}

int main()
{
        Car c;
        c.print();
        c = createCar();
        c.print();

        return 0;
}
```

반환시 복사 생성자 호출!!!

임시 객체에 저장
임시 객체를 c에 복사

속도: 0 기어: 1 색상: white
속도: 0 기어: 1 색상: metal



## 객체의 포인터가 함수에 전달

```
int main()
{
    Car red(0, 1, "red");
    Car blue(0, 1, "blue");
    swapObjects(&red, &blue);
    return 0;
}
```

주소 1234560

주소 1234570

p1

p2

객체의 주소 값이
p1과 p2에 복사됨

```
void swapObjects(Car *p1,    Car *p2)
{
    ...
}
```



## 객체의 포인터가 함수에 전달

```
...// 전과동일
void swapObjects(Car *p1, Car *p2)
{
        Car tmp;

        tmp = *p1;
        *p1 = *p2;
        *p2 = tmp;
}

int main()
{
        Car red(0, 1, "red");
        Car blue(0, 1, "blue");
        swapObjects(&red, &blue);
        red.print();
        blue.print();

        return 0;
}
```

객체 red와 blue의
주소가 p1과 p2에
복사됨

대입 연산자를
이용한 객체 복사

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

## 객체의 참조자가 함수에 전달

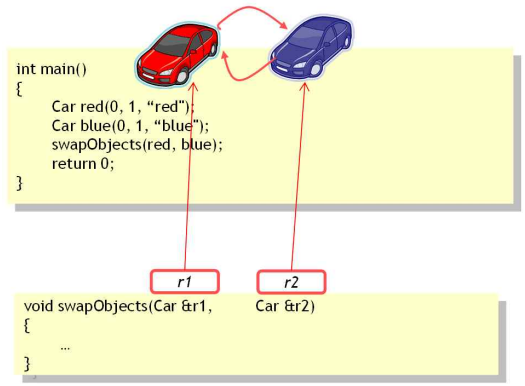



그림 7.5 객체의 참조자를 전달하는 경우



## 객체의 포인터가 함수에 전달

```
...// 전과동일
void swapObjects(Car &r1, Car &r2)
{
        Car tmp;

        tmp = r1;
        r1 = r2;
        r2 = tmp;
}
int main()
{
        Car red(0, 1, "red");
        Car blue(30, 2, "blue");

        swapObjects(red, blue);
        red.print();
        blue.print();
        return 0;
}
```

r1과 r2는 객체 red와 blue의 별명!!! 새로운 객체가 만들어지지 않음 (복사생성자 호출않음)

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



## 중간 점검 문제

1. 함수 안에서 매개 변수로 전달받은 객체의 내용을 수정하려면 매개 변수를 어떤 타입으로 선언하여야 하는가?
2. 매개 변수로 포인터와 레퍼런스를 사용하는 경우를 비교하여 보자.
3. 복소수 클래스에서 레퍼런스형 매개 변수를 사용할 수 있는 함수는?



## 7.3 this 포인터

- this는 현재 코드를 실행하는 객체를 가리키는 포인터

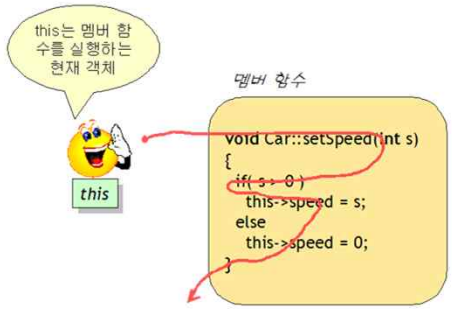



그림 10.2 this 포인터

## this를 사용하는 예

```
void Car::setSpeed(int speed)
{
    if( speed > 0 )
        this->speed = speed; // speed는 매개 변수, this->speed는 멤버 변수
    else
        this->speed = 0;
}

// 생성자
Car::Car(int s) {
    this->setSpeed(s); // 멤버 함수임을 강조(없어도 됨)
    this->gear = 1;
    this->color = "white";
}
```

객체에서 호출됨.
myCar.setSpeed(100);

setSpeed() 함수
안에서 호출한 객체의
주소를 알려면???



## 예제

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

class Car {
        int speed; // 속도
        int gear; // 기어
        string color; // 색상
public:
        Car(int s=0, int g=1, string c="white") : speed(s), gear(g), color(c) { }
        int getSpeed()      { return speed;  }
        void setSpeed(int speed)  {
                this->speed = (speed>0) ? speed : 0;
        }
        void print() {
                cout << "속도: " << speed << " 기어: " << gear << " 색상: " << color;
        }
```



## 예제

```
        void isFaster(Car *p)
        {
                if( this->getSpeed() > p->getSpeed() )
                        this->print();
                else
                        p->print();
                cout << "의 자동차가 더 빠름" << endl;
        }
};

int main()
{
        Car c1(0, 1, "blue");
        Car c2(100, 3, "red");
        c1.isFaster(&c2);
        return 0;
}
```

속도: 100 기어: 3 색상: red의 자동차가 더 빠름



## 중간 점검 문제

1. this 포인터는 무엇을 가리키는가?

2. this 포인터가 꼭 필요한 경우는?

# 7.4 정적 멤버

- 인스턴스 변수(instance variable): 객체마다 하나씩 있는 변수
- 정적 변수(static variable): 모든 객체를 통틀어서 하나만 있는 변수

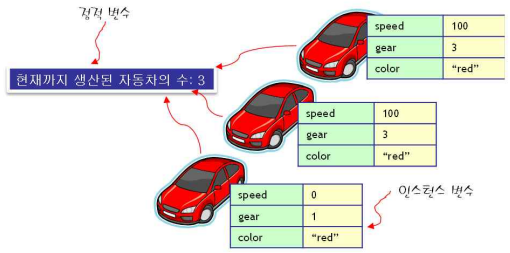



그림 7.6 정적 멤버



# 정적 멤버 변수

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

class Car {
        int speed;
        int gear;
        string color;
public:
        static int count;
        Car(int s=0, int g=1, string c="white")
            : speed(s), gear(g), color(c) {
                count++;
        }
        ~Car() {
                count--;
        }
};
```

정적 변수의 선언



# 정적 멤버 변수

```
int Car::count = 0;                    //①

int main()
{
        cout <<"지금까지 생성된 자동차수= " << Car::count << endl; //②

        Car c1(100, 0, "blue");
        Car c2(0, 0, "white");
        cout <<"지금까지 생성된 자동차수= " << Car::count << endl; //

        Car c3(0, 0, "red");
        cout <<"지금까지 생성된 자동차수= " << c1.count << endl; //
        cout <<"지금까지 생성된 자동차수= " << c2.count << endl; //

        return 0;
}
```

정적 변수의 정의

```
지금까지 생성된 자동차 수 = 0
지금까지 생성된 자동차 수 = 2
지금까지 생성된 자동차 수 = 3
지금까지 생성된 자동차 수 = 3
```



# 정적 멤버 함수

- 정적 멤버 함수는 static 수식자를 멤버 함수 선언에 붙인다.
- 클래스 이름을 통하여 호출
- 정적 멤버 함수도 클래스의 모든 객체들이 공유

- Java에서 삼각함수를 사용하려면?
  - Java에서는 일반 함수가 없다. 모든 함수가 클래스의 멤버함수.
  - Math 클래스에 다양한 함수를 넣음 ➔ static 멤버 함수로
  - Math 객체가 필요 없이 함수 호출 가능

## 정적 멤버 함수

```
class Car {
        ...
public:
        static int count;        // 정적변수의 선언

        ...
        // 정적 멤버 함수
        static int getCount() {
                return count;
        }
};
int Car::count=0;        // 정적 변수의 정의

int main()
{
        Car c1(100, 0, "blue");
        Car c2(0, 0, "white");
        int n = Car::getCount();
        cout <<"지금까지 생성된 자동차수= " << n << endl;
        return 0;
}
```

```
지금까지 생성된 자동차 수 = 2
계속하려면 아무 키나 누르십시오 . . .
```



## 주의할 점

- 정적 멤버 함수에서 멤버 변수들은 사용할 수 없다.
- 정적 멤버 함수에서 일반 멤버 함수를 호출하면 역시 오류

```
class Car {
        int speed;
        ...
public:
        int getSpeed() {
                return speed;
        }
        static int break() {
                int s = getSpeed();     // 오류: 일반 멤버 함수는 호출할 수 없음
                speed = 0;              // 오류: 일반 멤버 변수는 접근할 수 없음
                return s;
        }
};
```

정적 멤버 함수에서 일반 멤버는 사용할 수 없다.



## 중간 점검 문제

1. 정적 변수는 어떤 경우에 사용하면 좋은가?

2. 정적 변수나 정적 멤버 함수를 사용할 때, 클래스 이름을 통하여 접근하는 이유는 무엇인가?
3. 정적 멤버 함수 안에서 일반 멤버 함수를 호출할 수 없는 이유는 무엇인가?



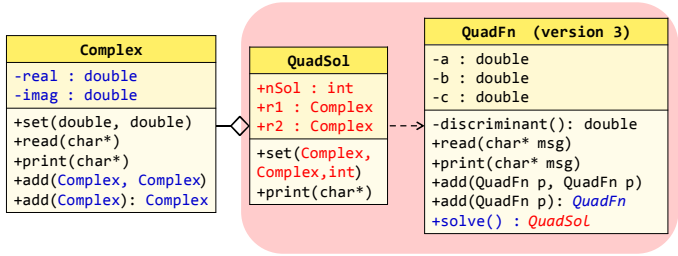
## 7.5 클래스와 클래스 간의 관계

- 사용(use): 하나의 클래스가 다른 클래스를 사용한다.
- 포함(has-a): 하나의 클래스가 다른 클래스를 포함한다.
- 상속(is-a): 하나의 클래스가 다른 클래스를 상속한다.

| 관계 | UML Symbol | 의미 | 예 |
|---|---|---|---|
| inheritance | | is-a | A book is a printed resource. |
| aggregation | | has-a | A book has a publisher. |
| dependency | | use | A book uses several fonts. |

# 사용 관계

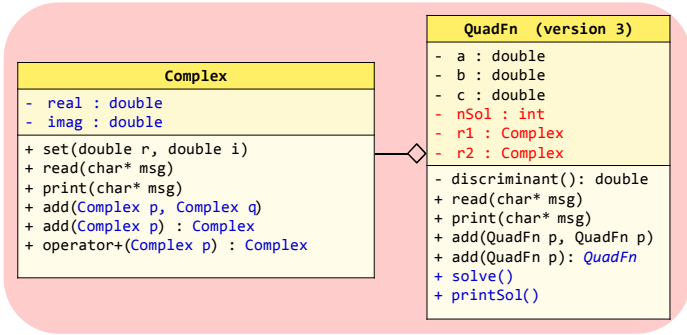
- 클래스 A의 멤버 함수에서 클래스 B의 멤버 함수들을 호출






# 포함 관계

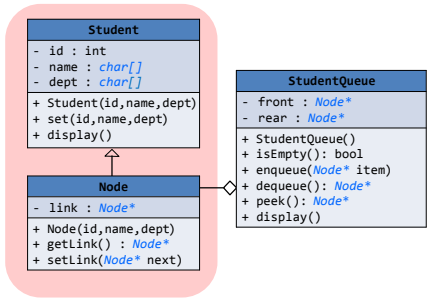
- 하나의 객체 안에 다른 객체들이 포함






# 상속 관계

- 다른 클래스를 상속해서 새로운 클래스를 만듦






# 중간 점검 문제

1. 사용 관계와 포함 관계는 어떻게 다른가?
2. 사용 관계와 포함 관계의 예를 더 들어보자.

## 7.6 기타

- 멤버 변수에 const를 붙이는 경우

```
class Car
{
        const int serial;
        string color;
        ...
public:
        Car(int s, string c) : serial(s) { color = c; }
}
```

이 멤버 변수의 값을 변경할 수 없다.

- 멤버 함수에 const를 붙이는 경우

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

이 함수 안에서는 멤버 변수의 값을 변경할 수 없다.



## const 수식어

- 객체에 const를 붙이는 경우

```
int main()
{
        const Car c1(0, 1, "yellow");
        c1.setSpeed();  // 오류!
        return 0;
}
```

이 객체를 통해서는 멤버 변수의 값을 변경할 수 없다.



## 중간 점검 문제

1. 객체 선언시에 const가 붙으면 어떤 의미인가?

2. 멤버 변수 getSpeed()에 const를 붙여보라. 어떤 의미인가?



## 임시 객체

- 수식의 계산 도중에 중간 결과를 저장하기 위하여 임시적으로 만들어지는 객체

```
int main()
{
  string s1 = "Hello ";
  string s2 = "World";
  const char* p = (s1+s2).c_str();        // ①
  cout << p;

  return 0;
}
```

임시 객체가 생성된다.

- 생성자와 소멸자도 물론 호출된다.
  - 복사생성자

# 임시 객체

- 함수가 객체를 반환하는 경우에도 생성

*temp_obj.cpp*

```
class Car {
...
};
Car createCar()
{
  Car tmp(0, 1, "metal");
  return tmp;
}

int main()
{
  createCar().print();
  return 0;
}
```

반환된 임시객체를 통하여 멤버 함수 호출



# 객체들의 배열

- 객체 들의 배열 생성
  - Car objArray[3];  // 디폴트 생성자 3회
  - objArray[0].speed = 0;// 멤버 변수 접근
  - objArray[1].speedUp();// 멤버 함수 호출

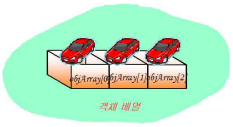


그림 10.8 객체 배열

- 객체 들의 배열 초기화
  - 기본적으로 디폴트 생성자 사용
  - 다른 방법?

```
Car   objArray[3] = {
        Car(0, 1, "white"),
        Car(0, 1, "red"),
        Car(0, 1, "blue"),
};
```

객체 별로 생성자를 호출할 수 있다.



# 예제

```
void main()
{
        Car objArray[3] = {
                Car(0, 1, "white"),
                Car(0, 1, "red"),
                Car(0, 1, "blue"),
        };
        for(int i=0; i< 3; i++)
                objArray[i].print();
}
```

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



# Q & A