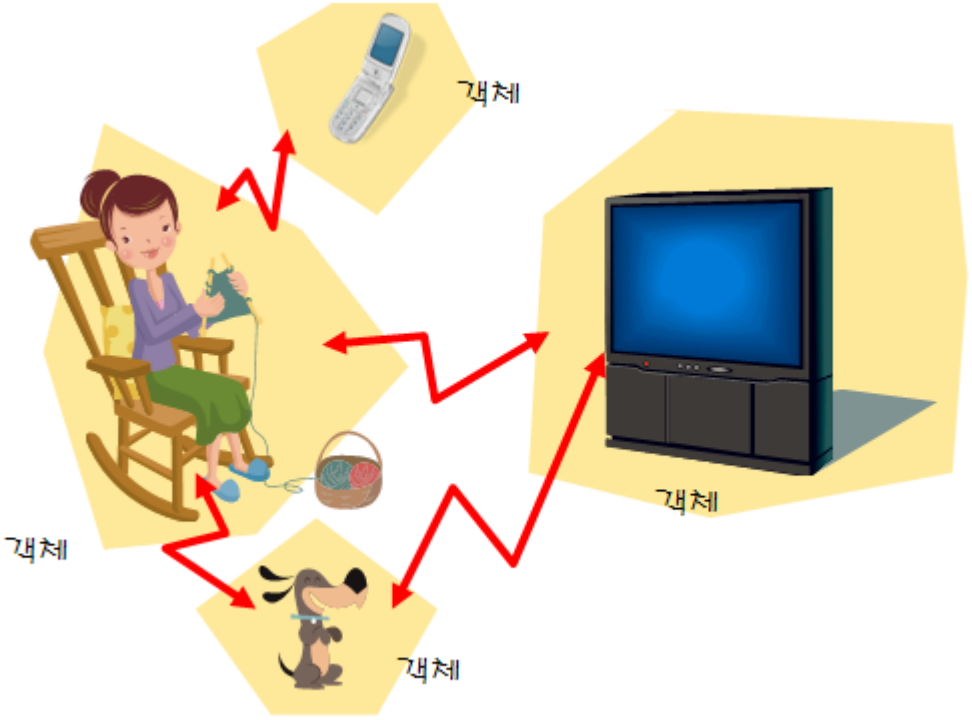
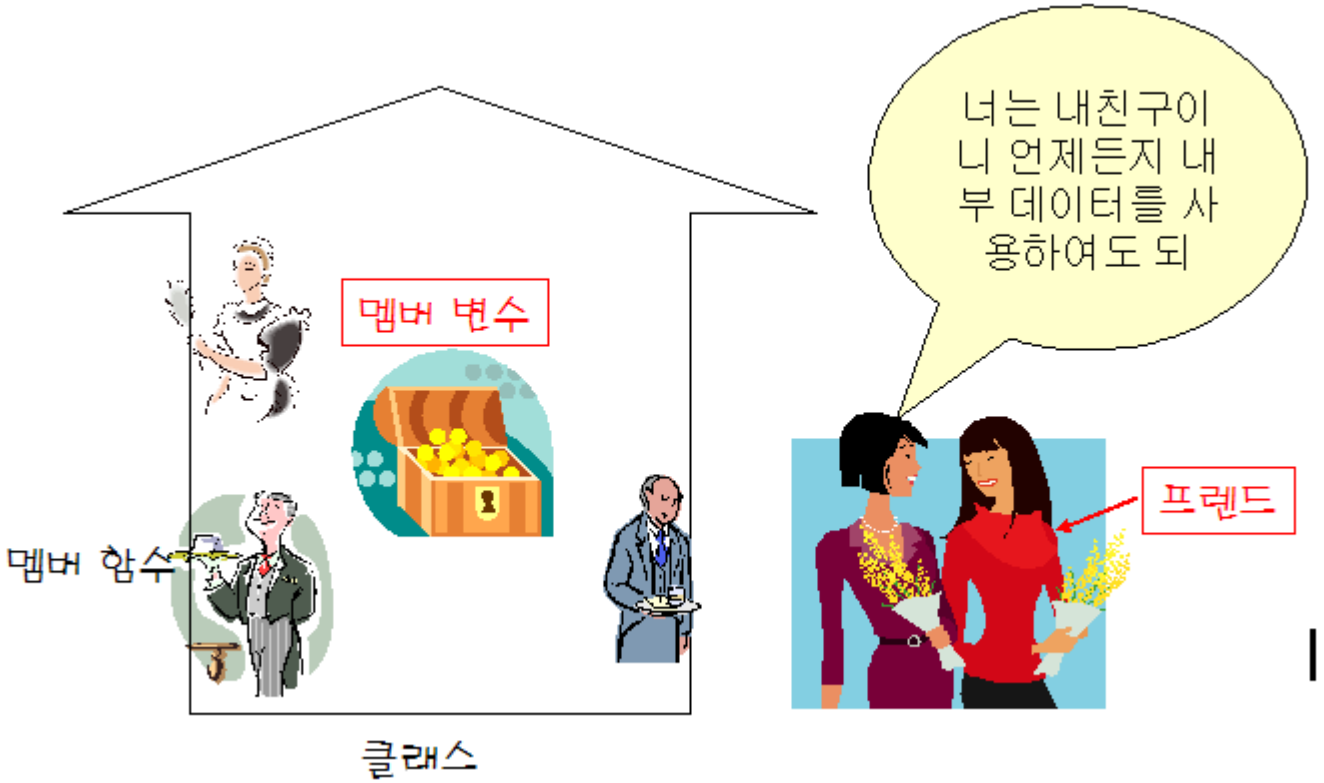
C++ Espresso

제10장 프렌드



프렌드 함수

- *프렌드 함수(friend function):* 클래스의 내부 데이터에 접근할 수 있는 특수한 함수



프렌드 함수 선언 방법

- 프렌드 함수의 원형은 클래스 내부에 포함
- 하지만 멤버 함수는 아니다.
- 프렌드 함수의 본체는 외부에서 따로 정의
- 프렌드 함수는 클래스 내부의 모든 멤버 변수를 사용 가능

```
class MyClass
{
        friend void sub();
        ....
};
```

프렌드 함수

실습예제 1

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

class Company{
    int sales, profit;
public:
    Company() :sales(100), profit(500) {}
};

int main()
{
    Company c1;
    cout << c1.profit << endl;
    return 0;
}
```

```
C:₩Windows₩system32₩cmd.exe
500
계속하려면 아무 키나 누르십시오 . . .
```

실습예제 2

```
class Company{
    int sales, profit;
    friend void sub(Company &c);
public:
    Company() :sales(100), profit(500) {}
};
void sub(Company &c)
{
    cout << c.profit << endl;
}
```

```
int main()
{
    Company c1;
    sub(c1);
    return 0;
}
```

프렌드 클래스

- 클래스도 프렌드로 선언할 수 있다.
- (예) Manager의 멤버들은 Employee의 전용 멤버를 직접 참조할 수 있다.

```
class Employee {
    int salary;
    // Manager는 Employee의 전용 부분에 접근할 수 있다.
    friend class Manager;
    // ...
};
```

프렌드 클래스

프렌드 함수의 용도

- 두 개의 객체를 비교할 때 많이 사용된다.

① 일반 멤버 함수 사용

```
if( obj1.equals(obj2) )
{
        …
}
```

② 프렌드 함수 사용

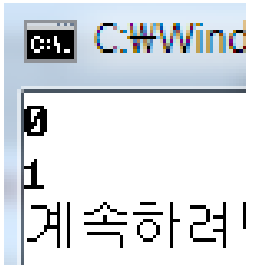
```
if( equals(obj1, obj2) )
{
        …
}
```

실습예제 3

```
class Date
{
    friend bool equals(Date d1, Date d2);
    int year, month, day;
public:
    Date(int y, int m, int d)
    {
        year = y;
        month = m;
        day = d;
    }
};
bool equals(Date d1, Date d2)
{
    return d1.year == d2.year && d1.month == d2.month && d1.day == d2.day;
}
```

```
int main()
{
    Date d1(1994, 11, 30);
    Date d2(2002, 10, 30);
    Date d3(1994, 11, 30);

    cout << equals(d1, d2) << endl;
    cout << equals(d1, d3) << endl;
}
```



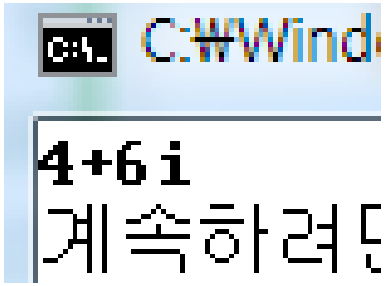
멤버변수에 접근 가능!

실습예제 4

```
class Complex{
    double re, im;
public:
    friend Complex add(Complex, Complex);
    Complex(double r, double i)
    {
        re = r; im = i;
    }

    void Output(){
        cout << re << "+"
            << im << "i" << endl;
    }
};

Complex add(Complex c1, Complex c2)
{
    return Complex(c1.re + c2.re, c1.im + c2.im);
}
```




```
void main()
{
    Complex com1(1, 2), com2(3, 4);
    Complex com3 = add(com1, com2);
    com3.Output();
}
```