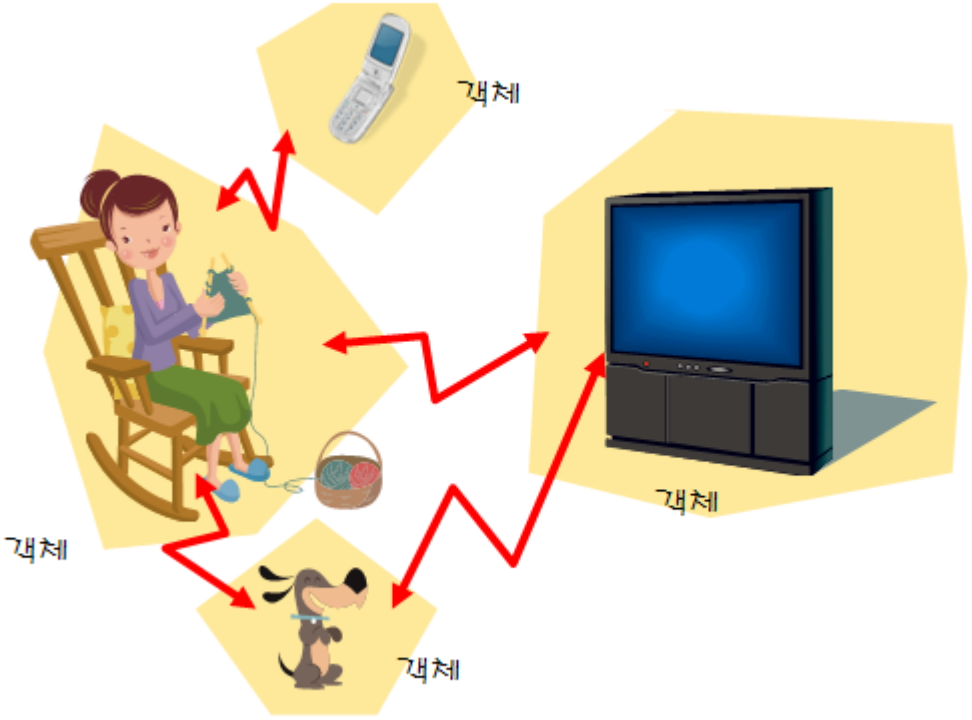
C++ Espresso

제8장 상속



Up-Down 게임

- 컴퓨터가 1에서 1000까지의 숫자 중 하나를 정하고, 10번 안에 컴퓨터가 생각한 수를 맞추는 게임

```
C:₩Windows₩system32₩cmd.exe
1번째 숫자 입력: 500
그것보다 작습니다
2번째 숫자 입력: 250
그것보다 작습니다
3번째 숫자 입력: 125
그것보다 작습니다
4번째 숫자 입력: 62
그것보다 작습니다
5번째 숫자 입력: 30
그것보다 큽니다
6번째 숫자 입력: 45
그것보다 작습니다
7번째 숫자 입력: 37
그것보다 큽니다
8번째 숫자 입력: 42
정답입니다!!
계속하려면 아무 키나 누르십시오 . . .
```

< random값 생성 방법 >

- #include<time.h>
- srand((unsigned)time(NULL)
  //랜덤 시드 초기화
- rand() % M + N
  //0+N ~ (M-1) + N 의 난수 발생

예)

rand() % 5 : 0 ~ 4

rand() % 100 : 0 ~ 99

rand() % 100 + 1 : 1 ~ 100

Up-Down 게임 정답

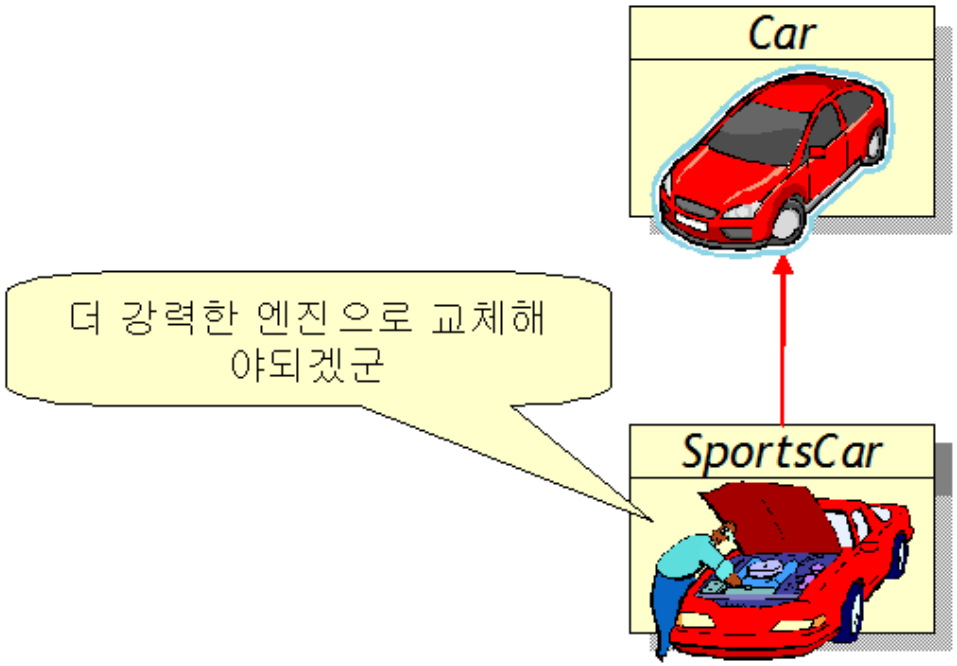
```
#include<iostream>
#include<time.h>
using namespace std;
int main()
{
  int comNum, playerNum;

  srand((unsigned)time(NULL));

  comNum = rand() % 1000 + 1;

  for (int i = 0; i < 10; i++)
  {
    cout << i + 1 << "번째 숫자 입력: ";
    cin >> playerNum;

    if (playerNum == comNum)
    {
      cout << "정답입니다!!" << endl;
      return 0;
    }
    else if (playerNum > comNum)
      cout << "그것보다 작습니다" << endl;
    else
      cout << "그것보다 큽니다" << endl;
  }
  cout << "실패!!" << endl;
  cout << "정답은 " << comNum << endl;
  return 0;
}
```

이번 장에서 학습할 내용

- 재정의(오버라이딩)
- 다중 상속

재정의

- 재정의(overriding): 자식 클래스가 필요에 따라 상속된 멤버 함수를 다시 정의하는 것



예제

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

class Car {
public:
        int getHP()
        {
                return 100;
        }
};

class SportsCar : public Car {
public:
        int getHP()
        {
                return 300;
        }
};
```

// 100마력반환

재정의

// 300마력반환

예제

```
int main()
{
        SportsCar sc;
        cout << "마력: " << sc.getHP() << endl;
        return 0;
}
```

```
마력: 300
계속하려면 아무 키나 누르십시오 . . .
```

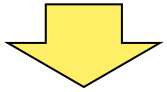
원래의 함수를 호출하려면

```
int main()
{
    SportsCar sc;
    cout << "마력: " << sc.Car::getHP() << endl;    // 100이 출력된다.
    return 0;
}
```

```
마력: 100
계속하려면 아무 키나 누르십시오 . . .
```

재정의의 조건

```
class Animal {
        void makeSound()
        {
        }
};
```



재정의가 아님!!

```
class Dog : public Animal {
        int makeSound()
        {
        }
};
```

- 부모 클래스의 멤버 함수와 동일한 시그니처를 가져야 한다.
- 즉 멤버 함수의 이름, 반환형, 매개 변수의 개수와 데이터 타입이 일치하여야 한다.

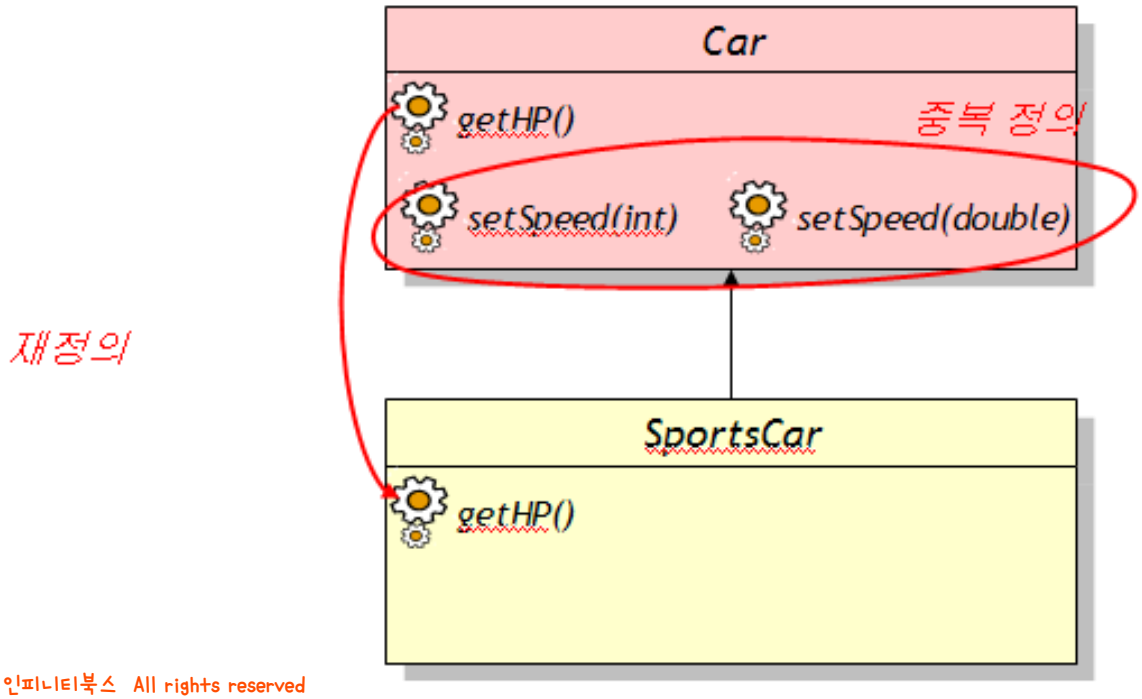
재정의와 중복 정의

- 중복 정의(Overroading)

: 같은 이름의 멤버 함수를 여러 개 정의하는 것

- 재정의(Overriding)

: 부모 클래스에 있던 상속받은 멤버 함수를 다시 정의하는 것



멤버 변수 재정의

```
class Car {
public:
    int speed = 80;
    string color;
};
class SportsCar : public Car {
public:
    int speed = 100;
    string color;
};
```

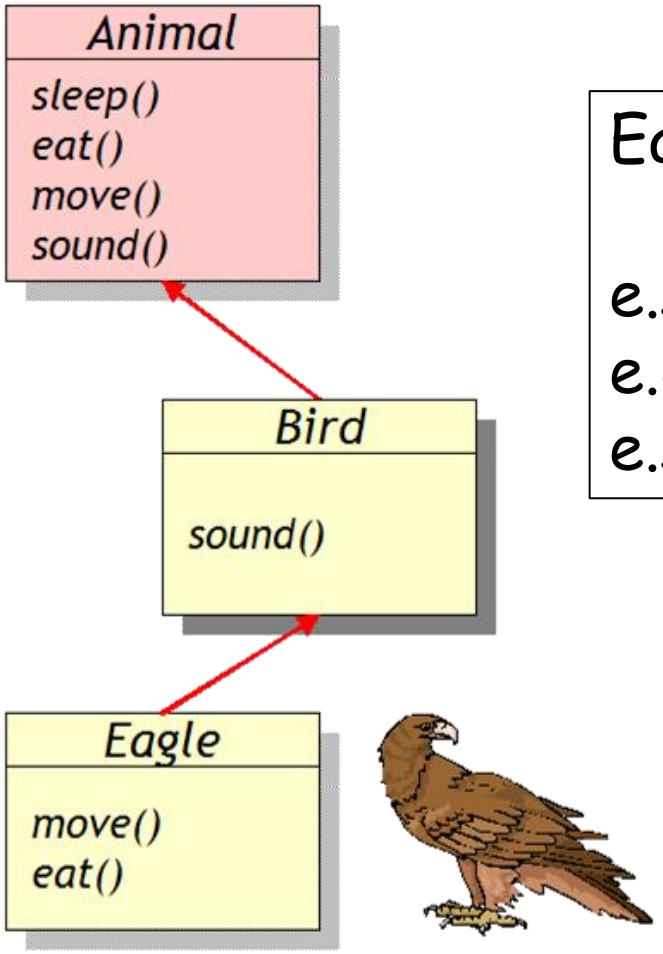
```
int main()
{
    SportsCar sc;
    cout << "스피드: " << sc.speed << endl;
    cout << "스피드: " << sc.Car::speed << endl;
    return 0;
}
```

Q. 출력결과는?

```
C:₩Windows₩system32₩cmd.exe
스피드: 100
스피드: 80
계속하려면 아무 키나 누르십시오 . . .
```

- 멤버변수 재정의
  : 가능하지만 혼란을 일으킴!

재정의된 멤버 함수의 호출 순서



Eagle e;

e.sleep(); → Animal의 sleep() 호출
e.eat(); → Eagle의 eat() 호출
e.sound(); → Bird의 sound() 호출

부모 클래스의 멤버 호출

```
class ParentClass {
public:
    void print() {
        cout << "부모클래스의 print() 멤버함수" << endl;
    }
};

class ChildClass : public ParentClass {
            int data;
public:

    void print() { //멤버함수오버라이딩
        ParentClass::print();
        cout << "자식클래스의print() 멤버함수" << endl;
    }
};
int main()
{
    ChildClass obj;
    obj.print();
            return 0;
}
```

부모 클래스의 함수 호출!

```
부모 클래스의 print() 멤버 함수
자식 클래스의 print() 멤버 함수
계속하려면 아무 키나 누르십시오 . . .
```

상속의 3가지 유형

```
class 자식클래스 : public 부모클래스
{
    ….
}
```

	Public으로 상속	Protected로 상속	Private로 상속
부모클래스의 public 멤버	public	protected	Private
부모클래스의 protected 멤버	protected	protected	Private
부모클래스의 private 멤버	접근 불가	접근 불가	접근 불가

예제 : main에서 가능한 출력문은?

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

class ParentClass {
private:
        int x;
protected:
        int y;
public:
        int z;
};
class ChildClass1 : public ParentClass
{
};

class ChildClass2 : protected ParentClass
{
};

class ChildClass3 : private ParentClass
{
};
```

```
int main()
{
    ChildClass1 obj1;
    ChildClass2 obj2;
    ChildClass3 obj3;

    cout << obj1.x << endl;
    cout << obj1.y << endl;
    cout << obj1.z << endl;

    cout << obj2.x << endl;
    cout << obj2.y << endl;
    cout << obj2.z << endl;

    cout << obj3.x << endl;
    cout << obj3.y << endl;
    cout << obj3.z << endl;
    return 0;
}
```

중간 점검 문제

1. 자식 클래스가 필요에 따라 상속된 멤버 함수를 다시 정의하는 것

 : 재정의(Overriding)

2. 부모 클래스에 public 변수 x가 있다. 부모 클래스를 protected로 상속받은 자식 클래스 에서는 x의 접근 권한이 무엇으로 변경되는가?

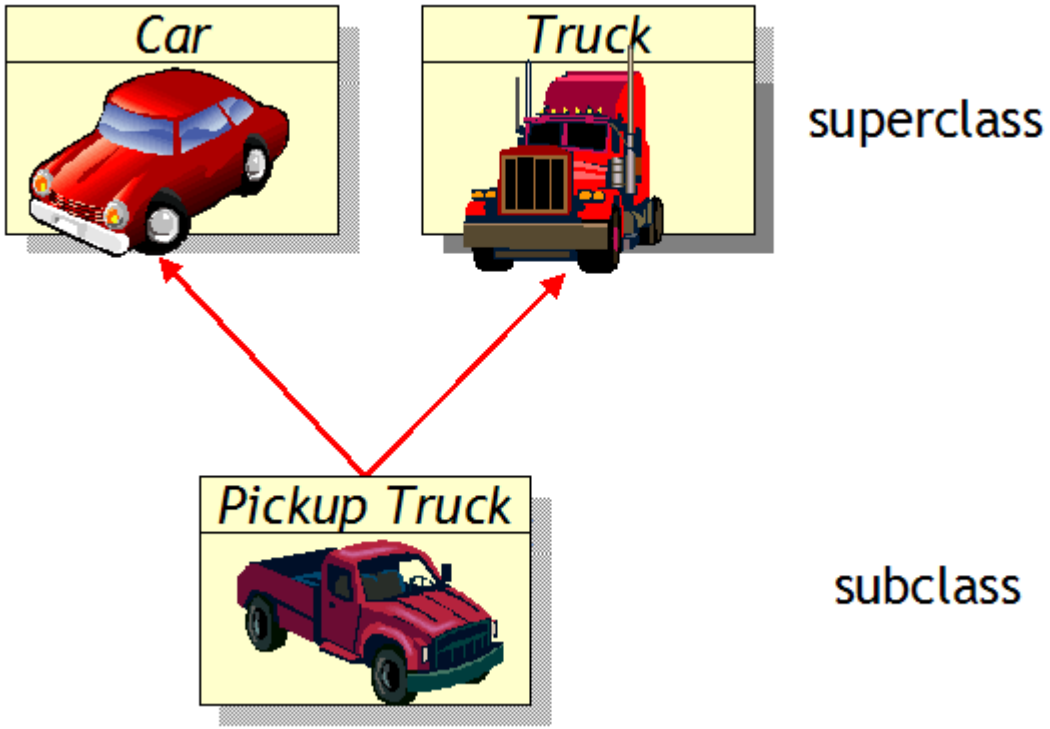
 : protected

3. 부모 클래스에 protected 변수 y가 있다. 부모 클래스를 private으로 상속받은 자식 클래스에서는 y의 접근 권한이 무엇으로 변경되는가?

 : private

다중 상속

```
class Sub : public Sup1, public Sup2
{
        ...// 추가된 멤버
        ...// 재정의된 멤버
}
```



예제

```
1  #include<iostream>
2  using namespace std;
3  class PassangerCar {
4  public:
5      int seats;
6      PassangerCar() { seats = 4; }
7  };
8  class Truck {
9  public:
10     int payload; // 적재 하중
11     Truck() { payload = 10000; }
12 };
13 class PickUp : public PassangerCar, public Truck {
14 public:
15     int tow_capability; //견인능력
16     PickUp() { tow_capability = 30000; }
17 };
```

```
18 int main()
19 {
20     PickUp myCar;
21
22     cout << myCar.seats << endl;
23     cout << myCar.payload << endl;
24     cout << myCar.tow_capability << endl;
25     return 0;
26 }
```

```
C:₩Windows₩system32₩cmd.exe
4
10000
30000
계속하려면 아무 키나 누르십시오 . . .
```

다중 상속의 문제점

```
1   #include<iostream>
2   using namespace std;
3   class SuperA {
4   public:
5       void sub() {  cout << "SuperA의 sub()" << endl; }
6   };
7   class SuperB {
8   public:
9       void sub() { cout << "SuperB의 sub()" << endl;  }
10  };
11  class Sub : public SuperA, public SuperB {};
```

```
12  int main()
13  {
14      Sub obj;
15      obj.sub();
16      return 0;
17  }
```

Q. 어떤 sub()가 호출될까?

```
1>------ 빌드 시작: 프로젝트: ConsoleApplication1, 구성: Debug Win32 ------
1>  main.cpp
1>c:\users\king\documents\visual studio 2015\projects\consoleapplication1\consoleapplication1\main.cpp(15): error C2385: 'sub' 액세스가 모호합니다.
1>  c:\users\king\documents\visual studio 2015\projects\consoleapplication1\consoleapplication1\main.cpp(15): note: 기본 'SuperA'의 'sub'일 수 있습니다.
1>  c:\users\king\documents\visual studio 2015\projects\consoleapplication1\consoleapplication1\main.cpp(15): note: 또는 기본 'SuperB'의 'sub'일 수 있습니다.
========== 빌드: 성공 0, 실패 1, 최신 0, 생략 0 ==========
```

```
C:\Windows\system32\cmd.exe
SuperB의 sub()
계속하려면 아무 키나 누르십시오 . . .
```

```
12  int main()
13  {
14      Sub obj;
15      obj.SuperB::sub();
16      return 0;
17  }
```

실습예제 #1

- 클래스는 총 3개(Person, Employee, Manager)
- Employee클래스는 Person클래스를 상속받는다
- Manager클래스는 Person과 Employee 클래스를 모두 상속받는다

- Person 클래스의 print()는 다음과 같다

```
void print() {  cout << "Person의 print()" << endl;  }
```

- Employee, Manager클래스에서는 Person의 print()를 재정의한다

- main함수는 다음과 같다

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

- 출력결과는 다음과 같다

```
C:₩Windows₩system32₩cmd.exe
Person의 print()
Employee의 print()
Manager의 print()
계속하려면 아무 키나 누르십시오 . . .
```

실습예제#1 정답

```
1   #include<iostream>
2   using namespace std;
3   class Person {
4   public:
5       void print() {  cout << "Person의 print()" << endl;  }
6   };
7   class Employee : public Person {
8   public:
9       void print() {
10          Person::print();
11          cout << "Employee의 print()" << endl;
12      }
13  };
14  class Manager : public Person, public Employee {
15  public:
16      void print() {
17          Employee::print();
18          cout << "Manager의 print()" << endl;
19      }
20  };
```

Q & A