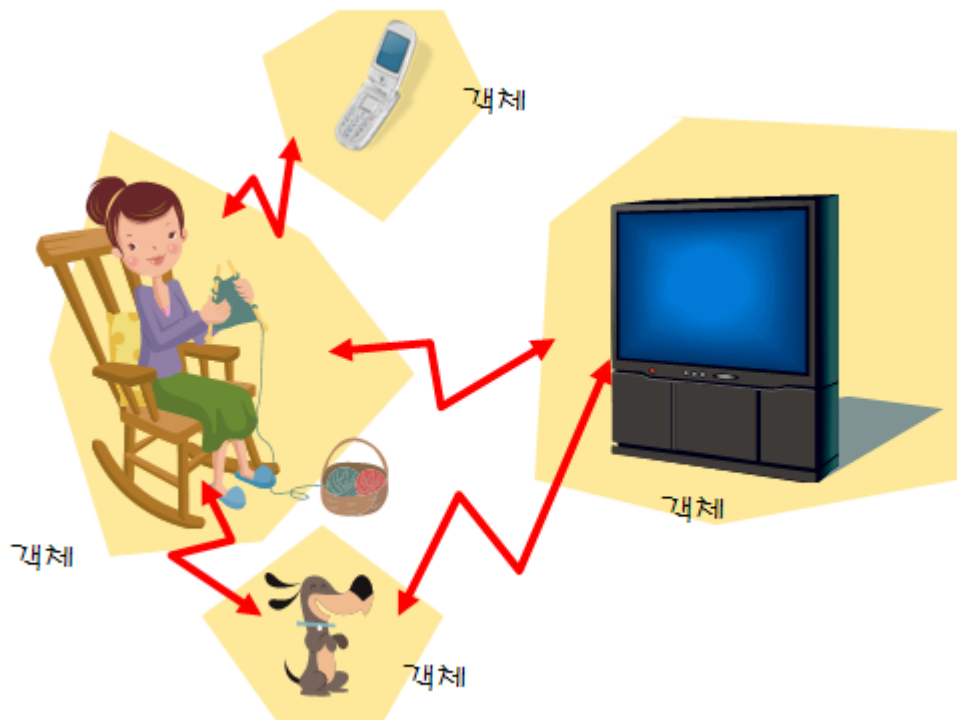





C++ Espresso

제9-1장 다형성, 형변환





이번 장에서 학습할 내용

- 
- 다형성
 - 객체 포인터
 - 형변환

다형성은
객체들이 동일한
메시지에 대하여
서로 다르게
동작하는 것
입니다.





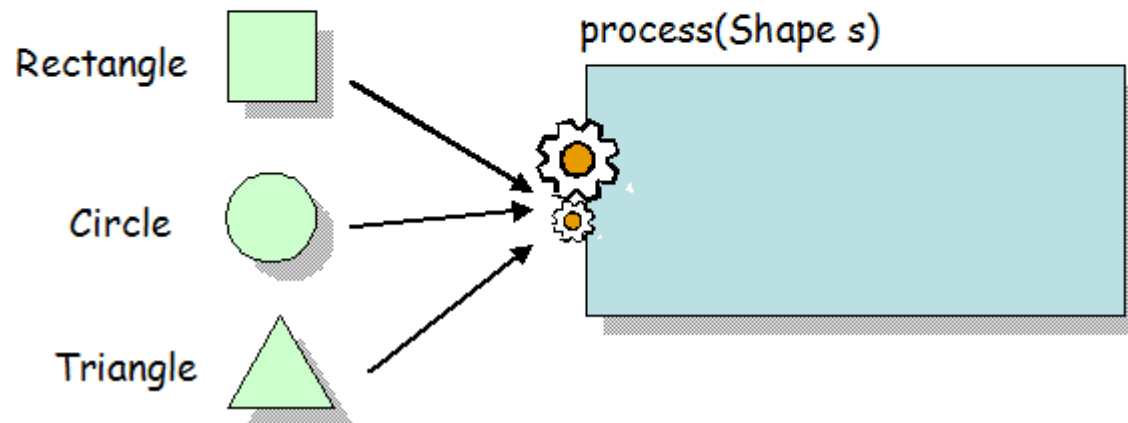
다형성이란?

- 다형성(polymorphism)이란 객체들의 타입이 다르면 똑같은 메시지가 전달되더라도 서로 다른 동작을 하는 것
- 다형성은 객체 지향 기법에서 하나의 코드로 다양한 타입의 객체를 처리하는 중요한 기술이다.





다형성이란?





상속과 객체 포인터

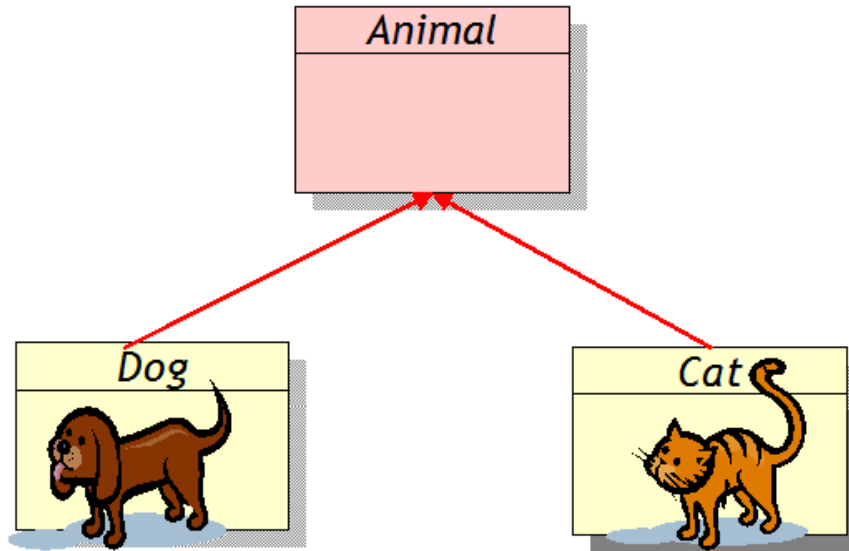


그림 9.2 상속 계층도

Animal 타입
포인터로 Dog
객체를 참조하니
틀린 거 같지만
올바른 문장!!

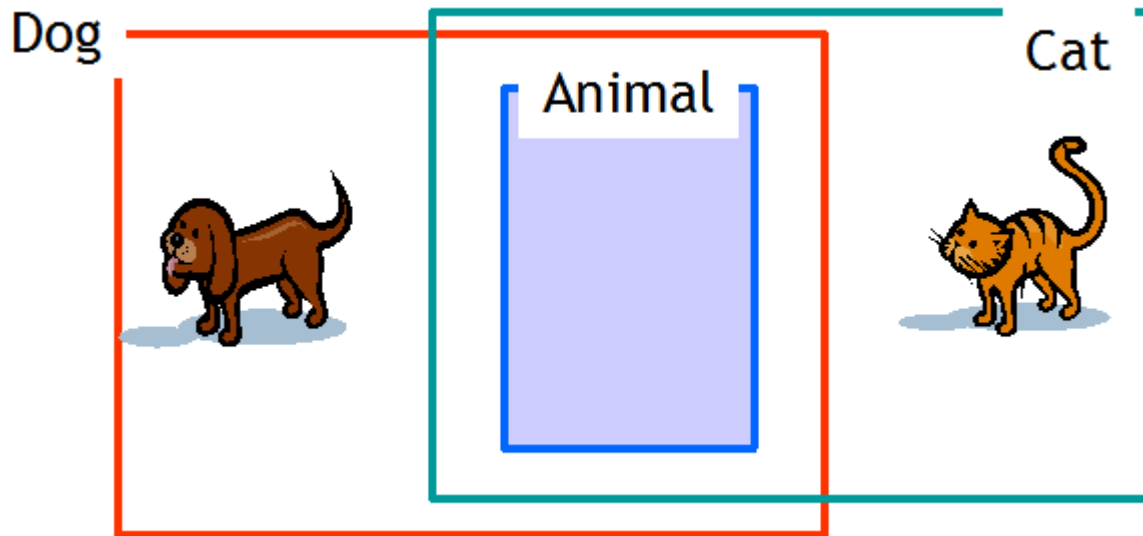
```
Animal *pa = new Dog();
```





왜 그럴까?

- 자식 클래스 객체는 부모 클래스 객체를 포함하고 있기 때문이다.





객체 포인터

- 포인터 연산의 가능성 여부를 판단할 때, **포인터의 자료형**을 기준으로 판단! (실제 가리키는 객체의 자료형을 기준으로 판단하지 않음)
- 즉, **포인터 형**에 해당하는 클래스의 멤버에만 접근이 가능!

```
Animal *pa = new Dog();
```



객체 포인터의 형변환

- 먼저 객체 포인터의 형변환을 살펴보자.

객체 포인터의 형변환

상향 형변환(**up**casting):

자식 클래스 타입을 부모 클래스타입으로 변환

하향 형변환(**down**casting):

부모 클래스 타입을 자식 클래스타입으로 변환



도형 예제

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

class Shape {
protected:
    int x, y;

public:
    void setOrigin(int x, int y) {
        this->x = x;
        this->y = y;
    }
    void draw() {
        cout << "Shape Draw";
    }
};
```

```
class Rectangle : public Shape {
private:
    int width, height;
public:
    void setWidth(int w) {
        width = w;
    }
    void setHeight(int h) {
        height = h;
    }
    void draw() {
        cout << "Rectangle Draw";
    }
};
```



상향 형변환(up-casting)

- 자식 클래스의 객체를 부모 클래스의 포인터가 가리키는 것
- Shape *ps = new Rectangle();
- ps->setOrigin(10, 10);

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

class Shape {
protected:
    int x, y;

public:
    void setOrigin(int x, int y) {
        this->x = x;
        this->y = y;
    }
    void draw() {
        cout << "Shape Draw";
    }
};
```

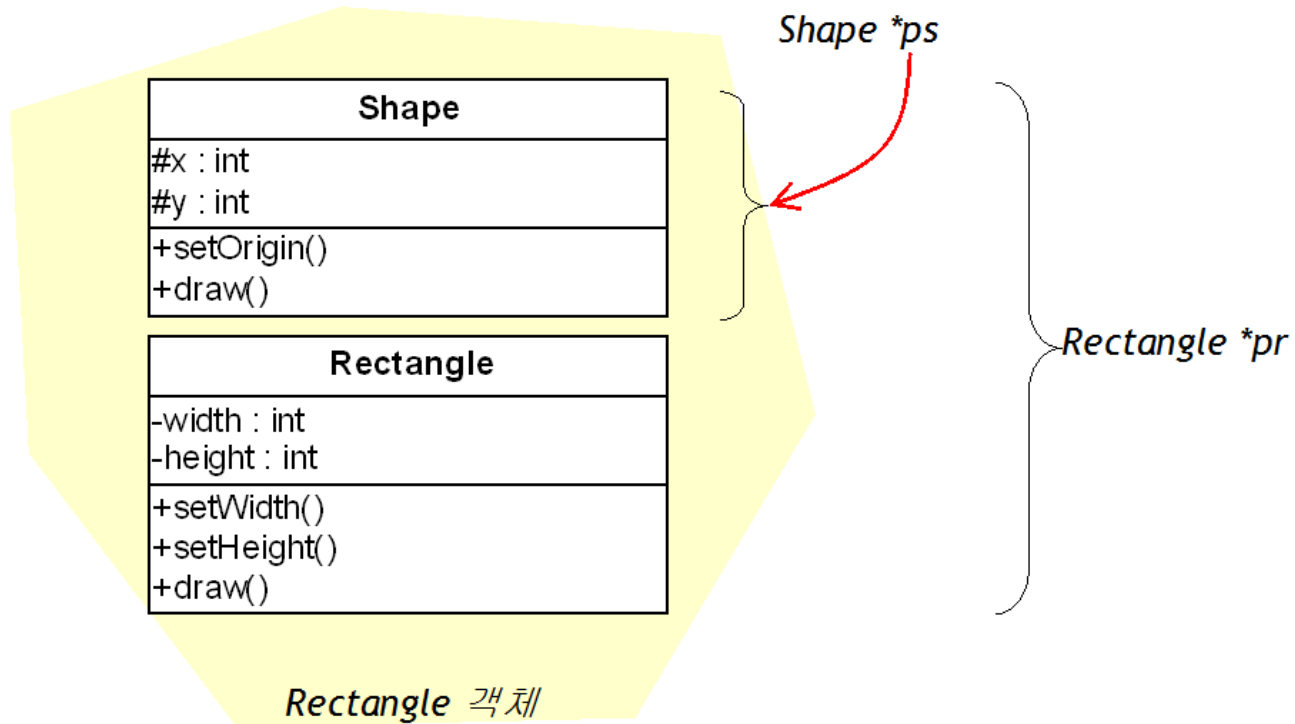
```
class Rectangle : public Shape {
private:
    int width, height;
public:
    void setWidth(int w) {
        width = w;
    }
    void setHeight(int h) {
        height = h;
    }
    void draw() {
        cout << "Rectangle Draw";
    }
};
```

Q. 이 코드는 가능할까?
ps -> setHeight(20);

NO!!!!!!



상향 형변환(up-casting)



Rectangle 객체를 Shape 포인터로 가리키면 Shape에 정의된 부분밖에 가리키지 못한다.



객체 포인터 가능성 판단

```
class First{  
public:  
    void FirstFunc(){ cout << "FirstFunc" << endl; }  
};  
class Second: public First{  
public:  
    void SecondFunc(){ cout << "SecondFunc" << endl; }  
};  
class Third: public Second{  
public:  
    void ThirdFunc(){ cout << "ThirdFunc" << endl; }  
};
```

```
int main(){  
    Third* tptr=new Third();  
    Second* sptr=tptr;  
    First* fptr=sptr;  
  
    tptr->FirstFunc() ;      ( O )  
    tptr->SecondFunc();     ( O )  
    tptr->ThirdFunc();      ( O )  
  
    sptr->FirstFunc();       ( O )  
    sptr->SecondFunc();     ( O )  
    sptr->ThirdFunc();      ( X )  
  
    fptr->FirstFunc();       ( O )  
    fptr->SecondFunc();     ( X )  
    fptr->ThirdFunc();      ( X )  
}
```



하향 형변환(down-casting)

- 부모 클래스 포인터가 가리키는 객체를 자식 클래스의 포인터로 가리키는 것
- 다운캐스팅은 업 캐스팅과 달리 명시적으로 타입 변환 지정
- ```
Shape *ps = new Rectangle();
((Rectangle *) ps)->setWidth(100);
```

\* ps가 setWidth()에 접근하기 위해서는  
다운캐스팅이 필요!!



## 예제

```
#include<iostream>
#include<string>
using namespace std;
class Base{
public:
 int baseMember1;
 string baseMember2;
 void showBase(){
 cout << "Base Function" << endl;
 }
};
class Derived : public Base {
public:
 int derivedMember1;
 string derivedMember2;
 void showDerived(){
 cout << "Derived Function" << endl;
 }
};
```

```
int main()
{
 1) Base *b = new Derived();

 2) Derived *d1;
 3) d1 = (Derived*)b;

 4) d1->showBase();
 5) d1->showDerived();

 return 0;
}
```

C:\Windows\system32\cmd.exe

```
Base Function
Derived Function
계속하려면 아무 키나 누르십시오 . . .
```



# 비교

## <업캐스팅>

- 부모 형식에서 자식 형식을 사용하겠다는 것.
- 특정 객체가 하위 클래스의 형에서 상위 클래스의 형으로 캐스팅 되는 것
- 형만 정확하다면 묵시적으로 캐스팅
- 캐스팅 후에 자식클래스에만 정의된 메소드에는 접근 할 수 없다.

## <다운캐스팅>

- 자식 형식에서 부모 형식을 사용하겠다는 것.
- 업캐스팅 한 것을 다시 원래의 형으로 되돌려주는 작업
- 명시적으로 원래의 형을 지정해 주어야 한다.
- 원래의 형이 무엇인지 잘 알고 사용해야 한다.



# 함수의 매개 변수

- 함수의 매개 변수는 자식 클래스보다는 부모 클래스 타입으로 선언하는 것이 좋다.

```
void move(Shape& s, int sx, int sy)
{
 s.setOrigin(sx, sy);
}
int main()
{
 Rectangle r;
 move(r, 0, 0);

 Circle c;
 move(c, 10, 10);
 return 0;
}
```

모든 도형을 받을 수 있다.



Q. 부모 클래스 TV와 자식 클래스 ColorTV가 있을 때,  
다음 중 업캐스팅과 다운 캐스팅을 찾으시오.

```
TV *p, tv;
ColorTV *q, ctv;
```

업 : ㄴ  
다운 : ㄷ

- ㄱ. p=&tv;
- ㄴ. p=&ctv;
- ㄷ. q=(ColorTV\*)&tv;
- ㄹ. q=&ctv;



## 중간 점검 문제

1. 부모 클래스 포인터 변수는 자식 클래스 객체를 참조할 수 있는가?

: Yes

2. 부모 클래스 포인터로 자식 클래스에만 정의된 함수를 호출할 수 있는가?

: No!!





# 실습(main은 뒤에)

```
class Shape {
protected:
 int x, y;
public:
 void setOrigin(int x, int y) {
 this->x = x;
 this->y = y;
 cout << "Shape: " << this->x
 << ", " << this->y << endl;
 }
 void draw() {
 cout << "Shape Draw" << endl;
 }
};
```

```
class Rectangle : public Shape {
private:
 int width, height;
public:
 void setWidth(int w) {
 width = w;
 cout << "Rectangle width: " << width << endl;
 }
 void setHeight(int h) {
 height = h;
 cout << "Rectangle height: " << height << endl;
 }
 void draw() {
 cout << "Rectangle Draw" << endl;
 }
};
```



## 실습

Q. 다음과 같이 출력 결과가 나오도록 main함수를 완성하세요 ^^  
(빈칸만 완성하면 됨)

C:\Windows\system32\cmd.exe

```
Shape: 10, 10
Shape Draw
Rectangle width: 100
Rectangle height: 200
Rectangle Draw
계속하려면 아무 키나 누르십시오 . . .
```

```
int main()
{
 Shape *ps = new Rectangle();
 ps->setOrigin(10, 10);
 ps->draw();

 ((Rectangle *)ps)->setWidth(100);
 ((Rectangle *)ps)->setHeight(200);
 ((Rectangle *)ps)->draw();

 delete ps;

 return 0;
}
```



# Q & A

