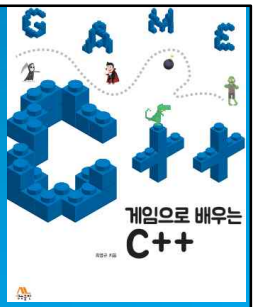
게임으로 배우는 C++

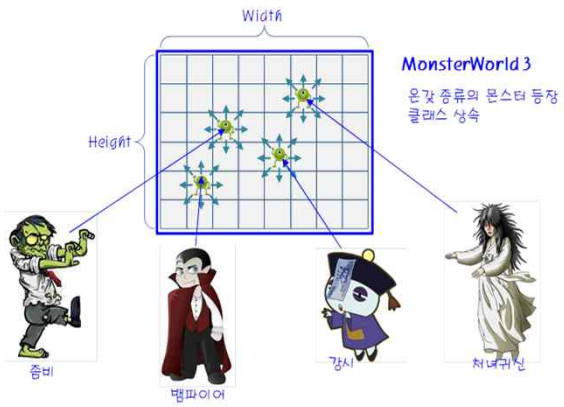


CHAPTER

09 상속

몬스터 월드3: 세상의 모든 귀신

- Monster World 3
  - 다양한 종류의 몬스터가 등장함
  - 클래스의 상속을 이용해 다양한 몬스터 클래스를 구현



2

9장 학습 목표

- 클래스 상속의 개념을 이해한다.
- 상속과 protected 멤버의 의미를 이해한다.
- 여러 가지 상속 방법을 이해한다.
- 상속에서 생성자, 소멸자의 호출 순서를 이해한다.
- 멤버 함수의 재정의를 이해하고 활용할 수 있도록 한다.
- 다중 상속의 문제점을 이해한다.

9.1 클래스의 상속

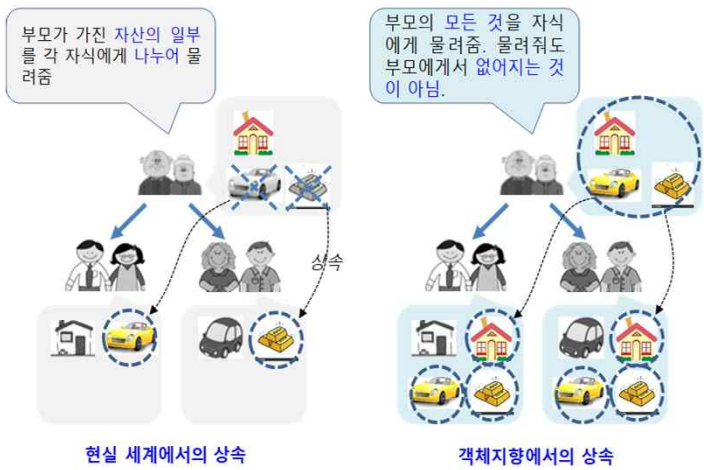
- 상속이란?
- 현실 세계와 객체지향에서의 상속 차이
- 클래스의 상속
- 예: 다양한 운송 수단들의 상속 관계
- "상속"과 "포함 "

상속이란?

- 현실 세계에서의 상속
  - 흙수저와 금수저
- 객체지향에서의 상속
  - 기존에 어떤 클래스가 잘 만들어져 있으면 이를 상속받아 새로운 클래스를 만드는 것
  - 객체지향에서 상속을 이용하는 가장 큰 이유는 복잡한 기능의 클래스를 빨리 쉽게 만들기 위함
    - 예) MFC를 이용한 윈도우 프로그래밍
- 상속하는 클래스와 상속받는 클래스
  - 부모(parent) 클래스, 수퍼(super) 클래스, 기반(base) 클래스
  - 자식(child) 클래스, 서브(sub) 클래스, 파생(derived) 클래스

5

현실 세계와 객체지향에서의 상속 차이



6

클래스의 상속

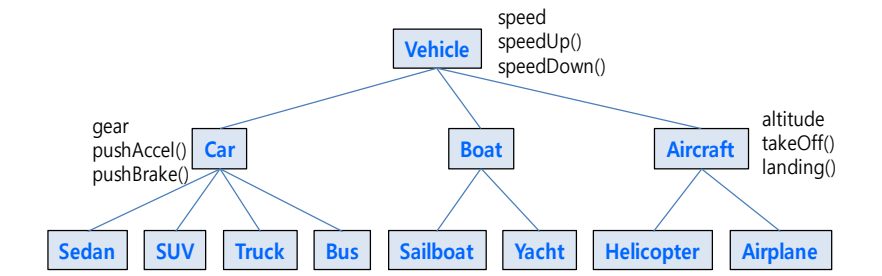
- 상속을 이용한 파생 클래스

```
class 파생클래스명 : public 기반클래스명
{
    // 새로운 특징들 (새로운 멤버 변수 또는 멤버 함수)
};
```

  - 문자 ':'는 상속을 나타낸다.
  - ':' 다음의 public은 private나 protected가 될 수 있다.
  - 대부분 public을 사용. 이 부분을 빠트리면 private로 인식.
  - 파생 클래스에는 기반 클래스의 모든 멤버가 포함됨.
  - 파생 클래스도 다른 클래스의 기본 클래스가 될 수 있음.
  - 하나의 클래스는 여러 클래스로 상속할 수 있다.

7

예: 다양한 운송 수단들의 상속 관계



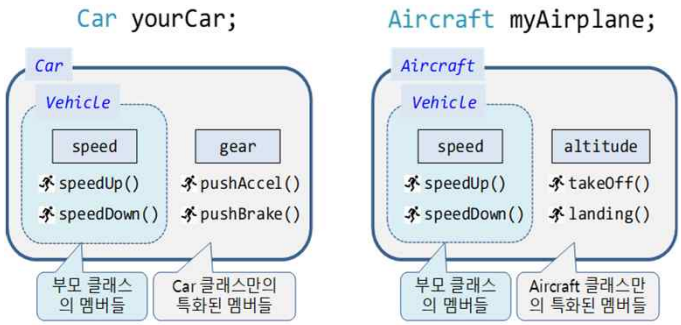

```
class Vehicle {
protected:
  int   speed;
public:
  void speedUp() {...}
  void speedDown(){...}
};
```

```
class Car
  : public Vehicle{
  int   gear;
public:
  void pushAccel(){...}
  void pushBrake(){...}
};
```

```
class Aircraft
  : public Vehicle{
  int   altitude;
public:
  void takeOff() {...}
  void landing() {...}
};
```

8

- Car와 Aircraft의 내부



- 상속을 사용하지 않으면?

```
class Car {
  int   speed;
  int   gear;
public:
  void speedUp(){...}
  void speedDown(){...}
  void pushAccel(){...}
  void pushBrake(){...}
};
```

```
class Aircraft {
  int   speed;
  int   altitude;
public:
  void speedUp(){...}
  void speedDown(){...}
  void takeOff() {...}
  void landing() {...}
};
```

"상속"과 "포함 "

- "상속"을 사용하는 경우
  - "is a" 또는 "is a kind of"의 관계
  - "포유류"는 "동물"의 일종이다.
  - "트럭"은 "자동차"의 일종이다.
  - "배"는 "탈 것"의 일종이다.
  - "좀비"는 "괴물"의 일종이다.

- "포함"을 사용하는 경우
  - "has a"관계
  - "자동차"는 "4개의 타이어"를 가지고 있다.
  - "자동차"는 "소유주"가 있다.
  - "선분"은 "2개의 점"으로 표현된다(가지고 있다).
  - "원"은 중심을 나타내는 "점"을 가지고 있다.

9.2 상속의 방법과 접근 지정자

- 상속과 접근 지정자
- 부모 클래스를 상속받는 방법들

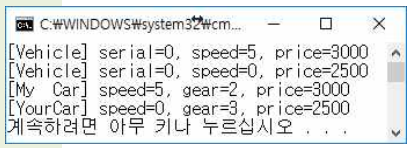
상속과 접근 지정자

- protected
  - 현재 클래스와 이 클래스의 모든 자식 클래스에서 접근
  - 보호 멤버 (예: 집안의 보물)
  - 상속을 사용하지 않으면 private와 동일

```
class Vehicle {
    int  serial;
protected:
    int  speed;
public:
    int  price;
    ...
};
```

```
class Car : public Vehicle {
   int   gear;
public:
    Car(int g=0) : gear(g) {}
    void pushAccel() { speed += 5; }
    void pushBrake() { speedDown(); }
    void setGear(int g) { gear = g; }
    void printCarInfo(char* msg="car"){
         printf("%s", msg);
         //printf(" serial=%d", serial);
         printf(" speed=%d,", speed);
         printf(" gear=%d,", gear);
         printf(" price=%d\n", price);
    }
};
```


```
void main() {
    Car myCar(2), yourCar(3);
    // myCar.serial = 20170118;
    // yourCar.speed = 20;
    myCar.pushAccel();
    myCar.price = 3000;
    yourCar.price = 2500;
    myCar.print();
    yourCar.print();
    myCar.printCarInfo ("[My Car]");
    yourCar.printCarInfo("[YourCar]");
}
```



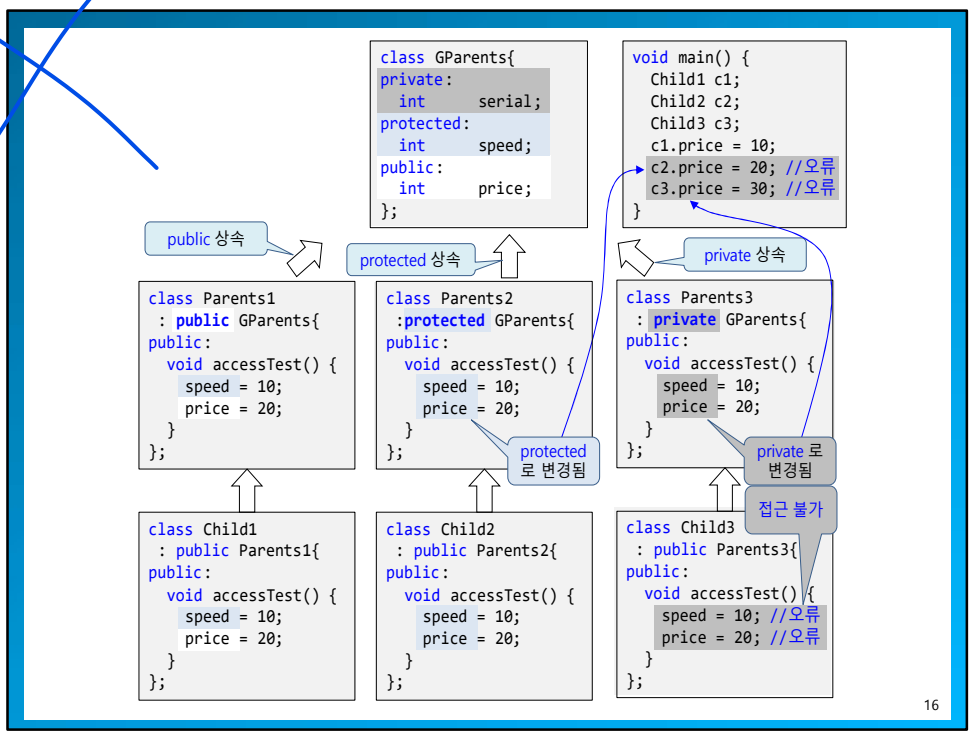
	설명	파일	줄
1	error C2248: 'Vehicle::speed' : protected 멤버('Vehicle' 클래스에서 선언)에 액세스할 수 없습니다.	vehiclemain.cpp	38

	설명	파일	줄
1	error C2248: 'Vehicle::serial' : private 멤버('Vehicle' 클래스에서 선언)에 액세스할 수 없습니다.	vehiclemain.cpp	27
2	error C2248: 'Vehicle::serial' : private 멤버('Vehicle' 클래스에서 선언)에 액세스할 수 없습니다.	vehiclemain.cpp	37

부모 클래스를 상속받는 방법들

부모 클래스의 접근 지정자	private 상속	protected 상속	public 상속
private	자식에서 직접 접근 불가 private 멤버가 됨	자식에서 직접 접근 불가 private 멤버가 됨	자식에서 직접 접근 불가 private 멤버가 됨
protected	자식에서 직접 접근 가능 **private 멤버로 변함**	자식에서 직접 접근 가능 protected 멤버	자식에서 직접 접근 가능 protected 멤버
public	자식에서 직접 접근 가능 **private 멤버로 변함**	자식에서 직접 접근 가능 **protected 멤버로 변함**	자식에서 직접 접근 가능 public 멤버

15



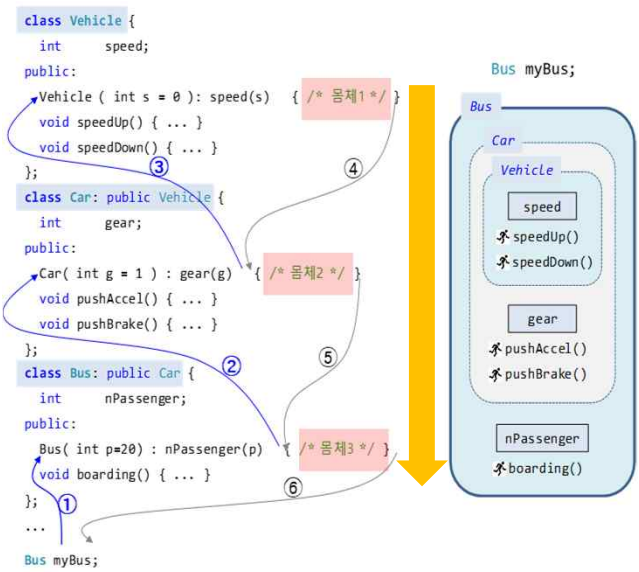
16

9.3 상속에서의 생성자와 소멸자

- 상속에서 생성자의 호출 순서
- 부모 클래스 생성자의 명시적 호출
- 상속에서 소멸자의 호출 순서

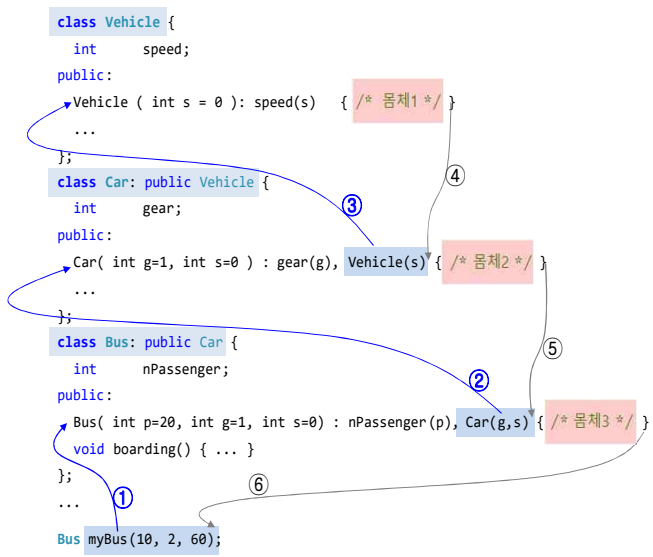
17

상속에서 생성자의 호출 순서

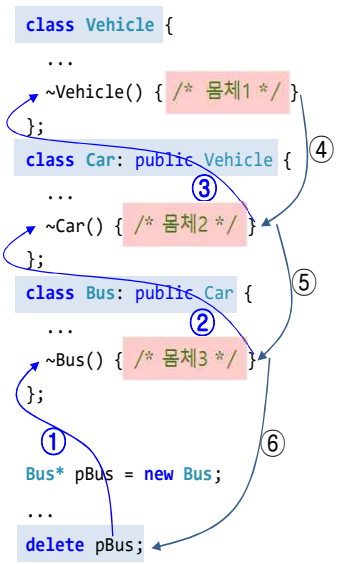


18

부모 클래스 생성자의 명시적 호출



상속에서 소멸자의 호출 순서



9.4 멤버의 재정의

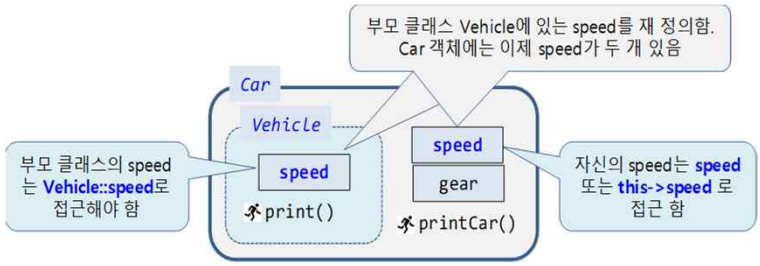
- 멤버 변수의 재정의
- 멤버 함수의 재정의(overriding)
- 멤버 함수의 탐색 순서
- 전역 함수(일반 함수) 의 호출 방법

멤버 변수의 재정의

- 바람직하지 않음
  - 부모 클래스의 멤버 변수와 동일한 이름의 멤버를 자식에 선언

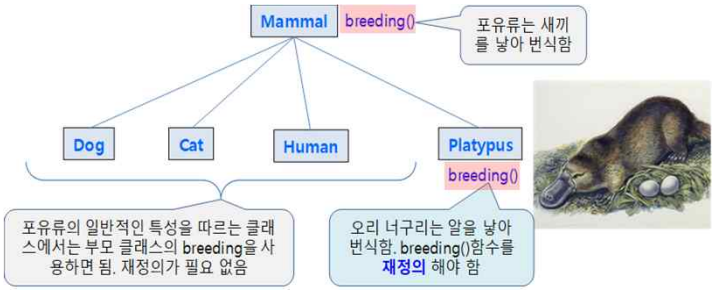
```
class Vehicle {
        int     speed;
        ...
};
```

```
class Car : public Vehicle {
        int     speed;
        int     gear;
        ...
};
```



멤버 함수의 재정의(overriding)

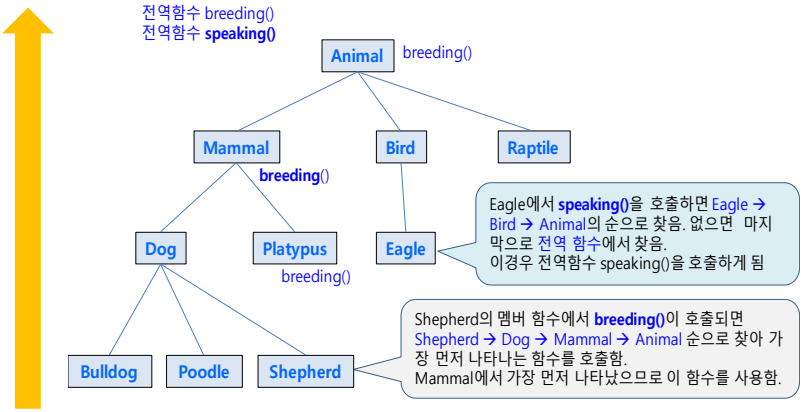
- 부모 클래스의 멤버 함수를 다시 구현 ➔ 매우 중요!
  - 함수 원형 동일.
  - cf. 중복 정의: 같은 이름, 매개변수가 다른 함수
- 재정의가 필요한 예
  - 오리너구리의 "번식"함수



23

멤버 함수의 탐색 순서

(1) 먼저 그 클래스에 이 함수가 있는지를 찾는다.
(2) 만약 없으면 찾을 때 까지 계속 부모 클래스로 올라간다.
(3) 모든 부모 클래스에 없으면 마지막으로 전역함수에서 찾는다.



24

전역 함수(일반 함수) 의 호출 방법

- 범위 연산자 ::

```
Animal::breeding();    // Animal의 breeding 함수를 호출
::breeding();          // 전역 함수 breeding()을 호출
```

- **상속되지 않는 함수들**
  - 부모 클래스의 private 멤버 함수
  - 생성자. 소멸자.
  - 복사 생성자. 대입 연산자 중복 함수.

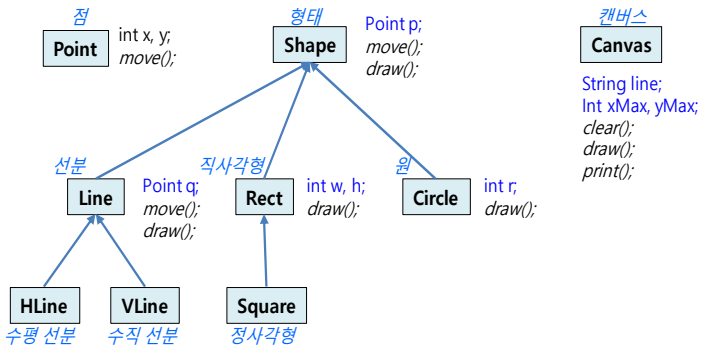
9.5 응용: 그래픽 에디터

- 그래픽 에디터
- 클래스 설계
- 클래스 구현

그래픽 에디터

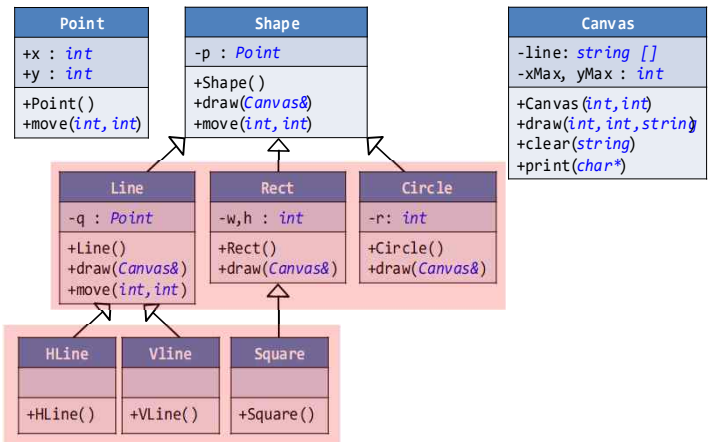
- 2차원 그래픽 원형(primitives) 클래스들
  - 선, 직사각형, 원 정사각형, 수평선, 수직선, 정사각형 등

  ➔ 상속으로 구현

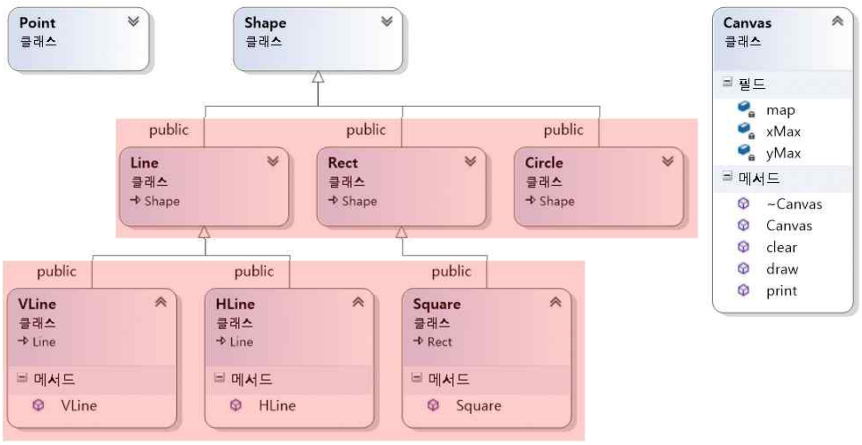


클래스 설계

- UML 클래스 다이어그램

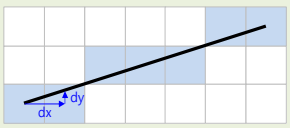


- Visual Studio의 클래스 다이어그램



클래스 구현

```
struct Point {
    int x, y;       // 점의 좌표
    Point(int xx=0, int yy=0):x(xx),y(yy){}
    void move(int dx, int dy){x+=dx; y+=dy;}
};
class Shape {
protected:
    Point p;
public:
    Shape(int x=0, int y=0):p(x,y) {}
    void draw(Canvas& canvas, string color="★"){...}
    void move(int dx, int dy) { p.move(dx, dy); }
};
class Line : public Shape{
    Point q;
public:
    Line(int x1=0, int y1=0, int x2=0, int y2=0)
                    : Shape(x1,y1), q(x2,y2) { }
    void draw(Canvas& canvas, string color = "선") { ... }
    void move(int dx, int dy) { p.move(dx, dy); q.move(dx,dy); }
};
struct HLine : public Line {
    HLine(int x = 0, int y = 0, int len = 0) : Line(x, y, x + len, y) { }
};
struct VLine : public Line {
    VLine(int x = 0, int y = 0, int len = 0) : Line(x, y, x, y + len) { }
};
```



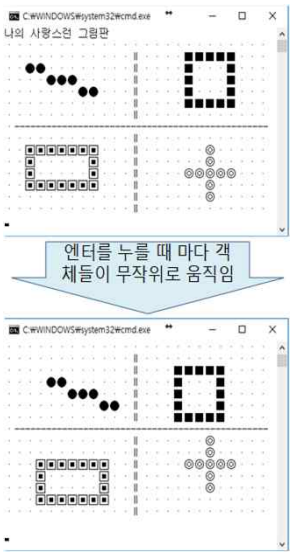
클래스 구현

```
class Rect : public Shape {
    int w, h;
public:
    Rect(int x=0, int y=0, int ww=0, int hh=0)
          : Shape(x, y), w(ww), h(hh) { }
    void draw(Canvas& canvas, string color = "■") {...}
};
class Square : public Rect {
public:
    Square(int x=0, int y=0, int w=0) : Rect(x,y,w,w){ }
};

class Circle : public Shape {
    int r;
public:
    Circle(int x=0, int y=0, int rr=0)
          : Shape(x, y), r(rr) { }
    void draw(Canvas& canvas, string color = "◎") {...}
};
```


클래스 구현

```
void main()
{
    Canvas myCanvas(25,15);
    Line   l(2, 2, 8, 4);
    Rect   r(2, 9, 6, 3);
    HLine  h(1, 7, 23);
    VLine  v(12, 1, 13);
    Square s(17, 1, 4);
    Circle c(19, 11, 2);
    do {
          myCanvas.clear();
          l.draw(myCanvas, "●");
          r.draw(myCanvas, "▣");
          v.draw(myCanvas, "ǁ");
          h.draw(myCanvas, "--");
          s.draw(myCanvas, "■");
          c.draw(myCanvas);
          myCanvas.print("나의 사랑스런 그림판");
          l.move(rand()%3-1, rand()%3-1);
          r.move(rand()%3-1, rand()%3-1);
          s.move(rand()%3-1, rand()%3-1);
          c.move(rand()%3-1, rand()%3-1);
    } while (getchar() != 'q');
}
```

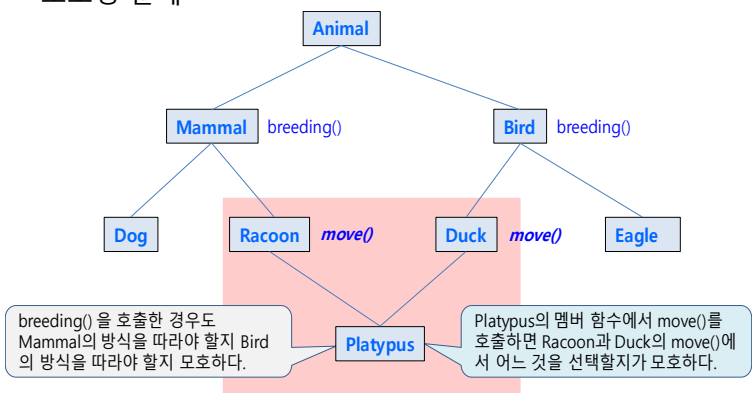


9.6 다중 상속 (multiple inheritance)

33

다중 상속 (multiple inheritance)

- 여러 개의 부모로부터 직접 상속 받는 것
  - C++에서는 다중 상속을 허용: 예) Platypus
  - 모호성 문제

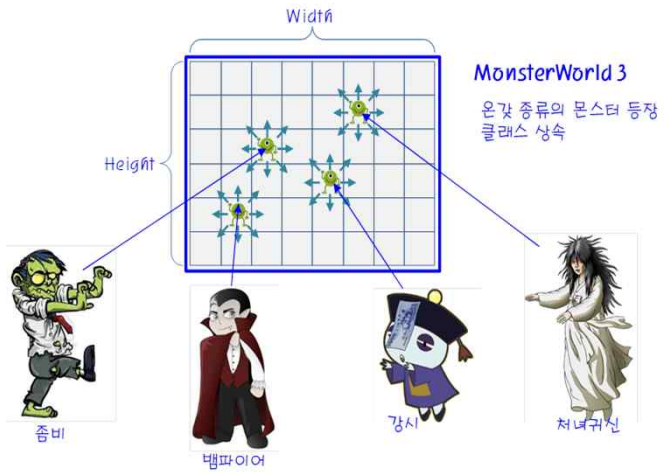


34

9.7 응용: MonsterWorld 3

- Monster World 3: 세상의 모든 귀신
- Monster별 특징
- 클래스 다이어그램
- 각종 몬스터 클래스 구현
- MonsterWorld클래스의 Play() 함수
- 고찰

Monster World 3: 세상의 모든 귀신



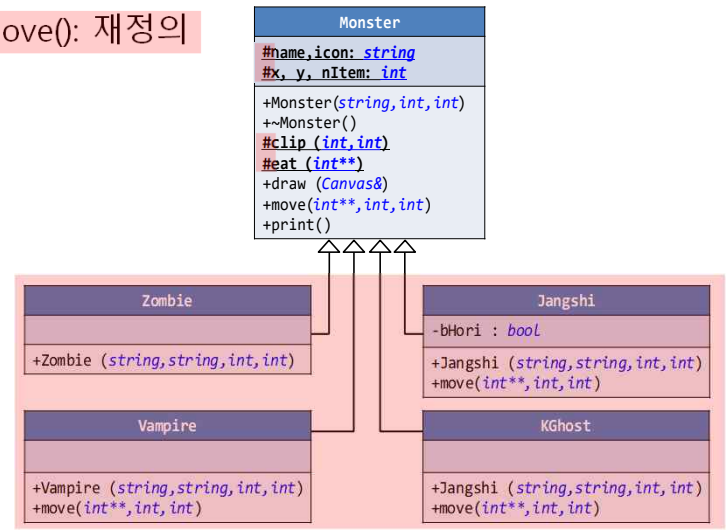
Monster별 특징

몬스터 종류	특징
좀비(Zombi)	정신없이 돌아다닌다. - 현재 위치에서 인접한 8방향으로 무작위로 움직인다.
뱀파이어(Vampire):	약간은 정신을 차리고 돌아다닌다. - 상하좌우로 인접한 4방향으로만 움직인다.
강시(Jiangshi):	한쪽(또는 반대) 방향으로만 열심히 달린다. - 좌우로 움직이거나 상하로만 움직인다. - 한꺼번에 여러 칸을 움직일 수 있다.
처녀귀신(KGhost, Korean Unmarried Woman ghost)	갑자기 나타난다. 즉, 공간 이동을 할 수 있다. - 현재 좌표에 상관없이 다음에 어디든 갈 수 있다.

➔ 상속을 이용해 구현

클래스 다이어그램

- move(): 재정의



각종 몬스터 클래스 구현

```
class Zombie : public Monster{
public:
  Zombie(string n="허접좀비", string i="§", int x=0, int y=0)
         : Monster(n, i, x, y) {}
  ~Zombie() { cout << " Zombie"; }
};
```

```
class KGhost : public Monster{
  ...
  void move(int** map, int maxx, int ma
    x = rand() % maxx;
    y = rand() % maxy;
    ...
  }
};
```

```
class Vampire : public Monster{
  ...
  void move(int** map, int maxx, int maxy) {
    int dir = rand() % 4;
    if (dir == 0) x--;
    else if (dir == 1) x++;
    else if (dir == 2) y--;
    else y++;
```

```
class Jiangshi : public Monster{
  bool    bHori;
public:
  Jiangshi(string n="대륙강시", string i="↔", int x=0, int y=0,
           bool bH=true) : Monster(n, i, px, py), bHori(bH) {}
  ...
  void move(int** map, int maxx, int maxy) {
    int dir = rand() % 2;
    int jump = rand() % 2 + 1;
    ...
  }
};
```


클래스 구현

```
#include "MonsterWorld.h"
#include "VariousMonsters.h"
#include <time.h>
void main()
{
        srand((unsigned int)time(NULL));
        int w = 16, h = 8;
        MonsterWorld game(w, h);
        game.add(new Zombie("허접한좀비", "§", rand()%w, rand()%h));
        game.add(new Vampire("뱀파이어짱", "★", rand()%w, rand()%h));
        game.add(new KGhost("어쩌다귀신", "♥", rand()%w, rand()%h));
        game.add(new Jiangshi("못먹어도고", "↔", rand()%w,rand()%h, true));
        game.add(new Jiangshi("못먹어세로", "↕", rand()%w,rand()%h, false));
        game.play(500, 10);
        printf("------게임 종료-------------------\n");
}
```

MonsterWorld클래스의 Play() 함수

- 모든 객체가 동일하게 움직임. Why?
- MonsterWorld의 play()함수

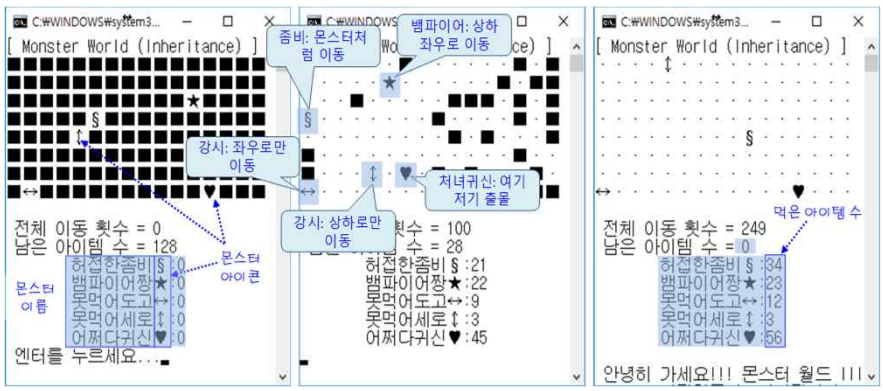
```
50              for (int k = 0; k < nMon ; k++)
51                      pMon[k]->move(world.Data(), xMax, yMax);
```

- 수정 → 더 이상 for문을 사용할 수 없음. ㅠㅠ


```
((Zombie*)pMon[0])->move(world.Data(), xMax, yMax);
((Vampire*)pMon[1])->move(world.Data(),xMax, yMax);
((KGhost*)pMon[2])->move(world.Data(), xMax, yMax);
((Jiangshi*)pMon[3])->move(world.Data(),xMax,yMax);
((Jiangshi*)pMon[4])->move(world.Data(),xMax,yMax);
```

- 해결방안 → 실행시간 다형성 (다음장에서 학습함)

실행 결과



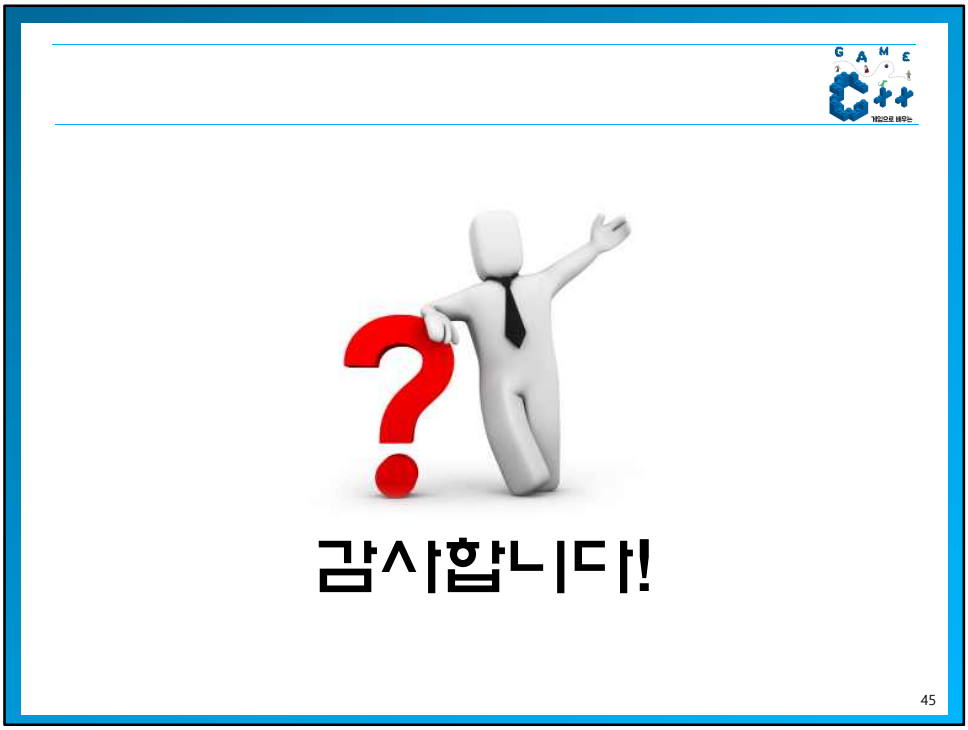
고찰

- 다양한 몬스터 클래스를 구현하고 동작시키는 것은 생각보다 어렵지 않고, 약간 재미있다. 상속을 사용하면 다양한 클래스를 손쉽게 구현할 수 있다.
- 강제 형 변환을 이용하는 부분을 수정하려면?
- 객체의 소멸에도 문제가 있는데...
- 메모리의 누수(leakage)도 발생하고...

43

9장 요약문제, 연습문제, 실습문제

44


감사합니다!
45