윈도우 프로그래밍

5. 클래스(2)

2018. 4.6.

심미나 교수

목 차

I. 정보은닉

객체지향프로그래밍의 특징

정보은닉과 캡슐화

- **정보은닉**
  - 객체는 자신의 데이터와 함수를 외부에 공개하거나 숨길 수 있음
    - 예를 들어, "A클래스의 정보(멤버변수)는 A내 함수외에 B나 C클래스에서 접근제한"
  - 만약 외부에 객체의 데이터는 숨기고 데이터를 처리하는 멤버함수만 공개한다면 데이터의 잘못된 수정을 막을 수 있어서 프로그램의 안정성을 높일 수 있음

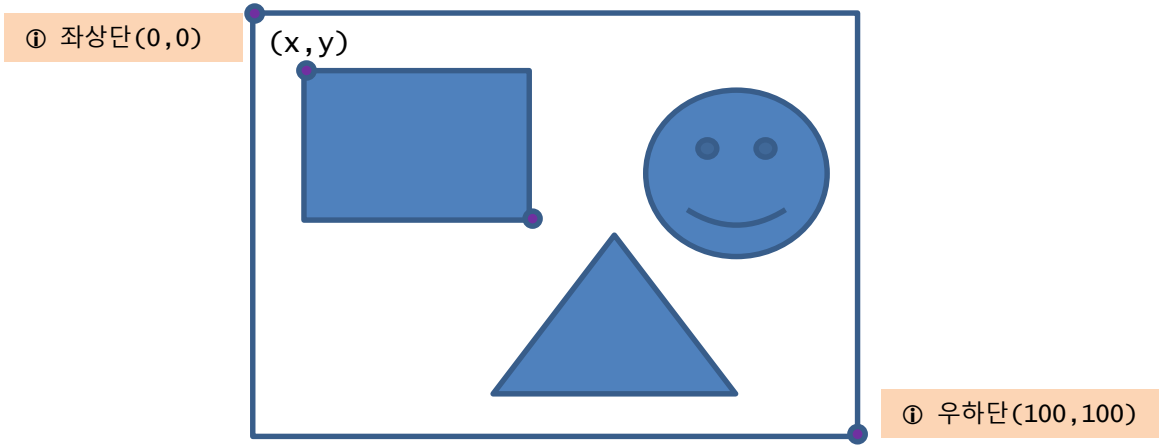
- **캡슐화**
  - 데이터와 해당 데이터를 처리할 수 있는 함수들을 결합하여 하나의 단위로 묶는 것
  - 캡슐화를 통해서 비로소 클래스가 프로그램의 부품처럼 사용될 수 있음
    - 캡슐화가 잘 안될경우, 독립성을 명확하게 유지하기 어려움

정보은닉

정보은닉의 이해

- **정보은닉**
  - 멤버변수의 외부접근을 허용할 경우, 잘못된 값이 저장되는 문제 발생 가능
  - 따라서, (private 선언을 통해) 멤버변수의 외부접근을 막는 것

  - (예시) 그림판
    - 그림판의 좌 상단(0,0)과 우 하단(100,100)의 범위 내에서 그림을 그려야 함
    - 하나의 그림에서 좌표의 좌우정보는 서로 바뀌어 저장되면 안됨



정보은닉

정보은닉의 이해

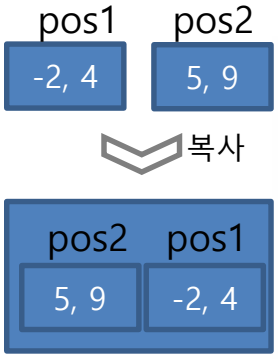
- **그림판의 구현 - 좌표 클래스, 직사각형 클래스**
  - Point의 멤버변수에는 0~100 이외의 값이 들어가지 못하도록 막을 수 없음
  - Rectangle의 멤버변수에는 좌우정보가 서로 바뀌어 저장되지 못하도록 막을 수 없음

ⓘ 정보은닉 실패

```
class Point
{
public:
    int x;    // x좌표의 범위는 0이상 100이하
    int y;    // y좌표의 범위는 0이상 100이하
};
```

```
class Rectangle
{
public:
    Point upLeft;
    Point lowRight;
public:
    void ShowRecInfo()
    {
        cout<<"좌 상단: "<<'['<<upLeft.x<<", ";
        cout<<upLeft.y<<']'<<endl;
        cout<<"우 하단: "<<'['<<lowRight.x<<", ";
        cout<<lowRight.y<<']'<<endl<<endl;
    }
};
```

ⓘ 정보은닉 실패



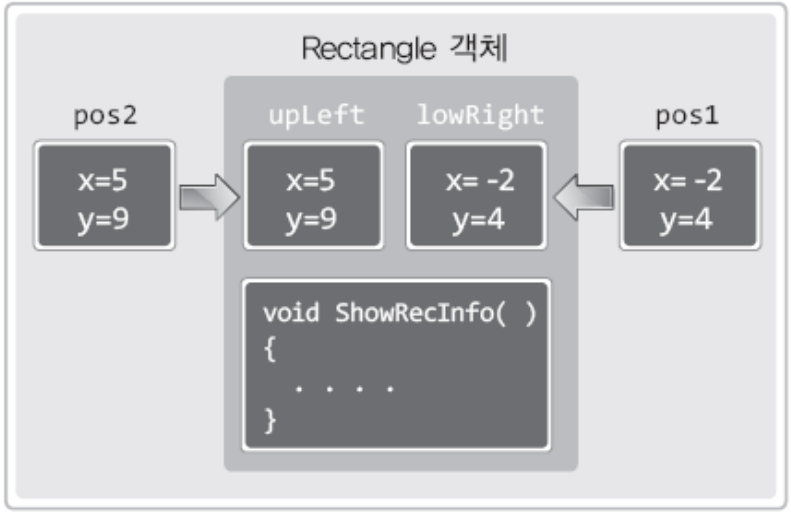

```
int main(void)
{
    Point pos1={-2, 4};
    Point pos2={5, 9};
    Rectangle rec={pos2, pos1};
    rec.ShowRecInfo();
    return 0;
}
```

정보은닉

정보은닉의 이해

- 직사각형(Rectangle) 객체의 이해
  - 클래스의 객체도 다른 객체의 멤버가 될 수 있음

```
int main(void)
{
    Point pos1={-2, 4};
    Point pos2={5, 9};
    Rectangle rec={pos2, pos1};
    rec.ShowRecInfo();
    return 0;
}
```



정보은닉

정보은닉의 구현

- 좌표(Point) 클래스의 정보은닉
  - 클래스의 멤버변수를 private으로 선언하고, 해당 변수에 접근하는 함수를 별도정의
  - 이로써 안전한 형태로의 멤버변수 접근을 유도함 즉, '정보은닉' 구현

ⓘ 정보은닉

```
class Point
{
private:
    int x;
    int y;
public:
    bool InitMembers(int xpos, int ypos);
    int GetX() const;
    int GetY() const;
    bool SetX(int xpos);
    bool SetY(int ypos);
};
```

(1)

(2)

ⓘ 엑세스함수

ⓘ GetX() 함수는 x값을 반환함. GetY() 함수는 y값을 반환함
ⓘ 이들은 객체안에서 멤버변수의 값이 변경되지 않도록 함

ⓘ 엑세스함수는 정보은닉으로 인하여 추가됨

(2)

ⓘ 제어함수 SetX()

```
bool Point::SetX(int xpos)
{
    if(0>xpos || xpos>100)
    {
        cout<<"벗어난 범위의 값 전달"<<endl;
        return false;
    }
    x=xpos;
    return true;
}
```

ⓘ x값 유입의 유일한 경로를 SetX함수가 되도록 함

ⓘ 엑세스 함수 정의 규칙
ⓘ AAA 변수 선언 시,
- GetAAA() 함수 만듬; 값 반환용
- SetAAA() 함수 만듬: 값 저장용

정보은닉

정보은닉의 구현

- **직사각형(Rectangle) 클래스의 정보은닉**
  - 클래스의 멤버변수를 private으로 선언하고, 해당 변수에 접근하는 함수를 별도정의
  - 이로써 안전한 형태로의 멤버변수 접근을 유도함 즉, '정보은닉' 구현

```
class Rectangle
{
(1) private:
        Point upLeft;
        Point lowRight;
public:
(2)     bool InitMembers(const Point &ul, const Point &lr);
        void ShowRecInfo() const;
};
```

ⓘ 정보은닉

ⓘ 엑세스함수

```
(2) bool Rectangle::InitMembers(const Point &ul, const Point &lr)
{
    if(ul.GetX()>lr.GetX() || ul.GetY()>lr.GetY())
    {
        cout<<"잘못된 위치정보 전달"<<endl;
        return false;
    }
    upLeft=ul;
    lowRight=lr;
    return true;
}
```

ⓘ 오류정보 알림

ⓘ 값 저장

ⓘ 좌상단, 우하단이 바뀌는 것을 차단

정보은닉의 구현

- const 함수
  - 정보은닉 구현 시, 프로그램의 안정성 위해 멤버함수에 const 선언
    1. const함수 내에서는 동일 클래스에 선언된 멤버변수 값을 변경하지 못함
    2. const함수는 const가 아닌 함수를 호출할 수 없음
    3. const로 상수화된 객체를 대상으로는 const 멤버함수만 호출이 가능함

```
int GetX() const;
int GetY() const;
void ShowRecInfo() const;
```

ⓘ 값변경불가

```
int GetNum()
{
    return num;
}

void ShowNum() const
{
    cout<<GetNum()<<endl;   //컴파일에러 발생
}
```

ⓘ 호출불가

```
void InitNum(const EasyClass &easy)
{
    num = easy.GetNum();   //컴파일에러 발생
}
```

IV. 실습

실습4-04

교재 - 예제 10-3.(p.363)

private 멤버를 다루기 위한 멤버함수 추가하기

실습4-05

교재 - 예제 10-4.(p.365)

인라인 함수 사용하기

감사합니다

mnshim@sungkyul.ac.kr