Lecture 13

Object-Oriented Programming IX

Protected and Private Derivation

Prof. Hyeong-Seok Ko
Seoul National University
Graphics & Media Lab

Contents

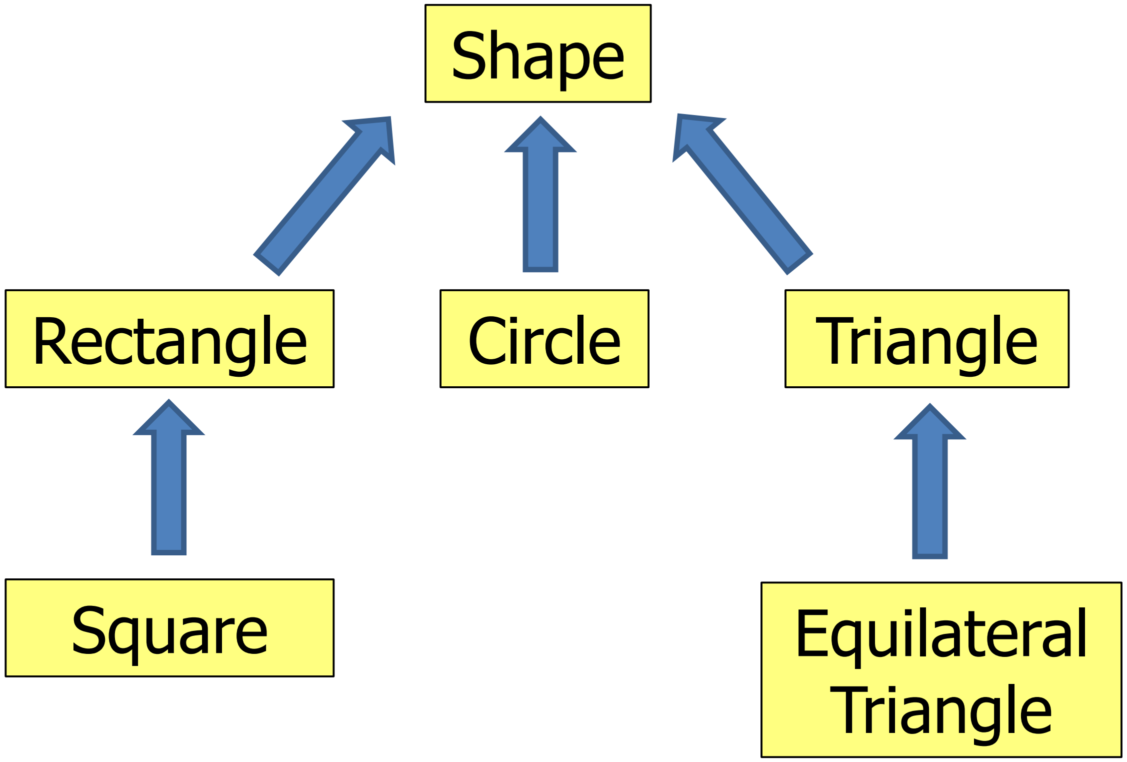
- Inheritance (15.1)
- Base and derived class (15.2.1, 15.2.3)
- Protected members (15.2.2)
- Public, protected, private inheritance (15.2.5)

Inheritance

- Models relationships among types.
  - Hierarchy in the real world is reflected to the code.

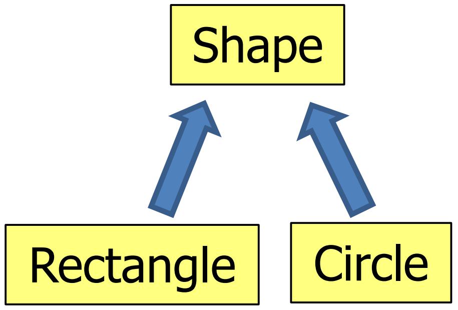
- Shares what is common and specializes only what is inherently different.
  - Results in less amount of coding.

Class Hierarchy



An Example

- Think about Shape, Rectangle, Circle
  - Rectangle or Circle is a Shape.
  - Each Shape has its own color and needs to be drawn.



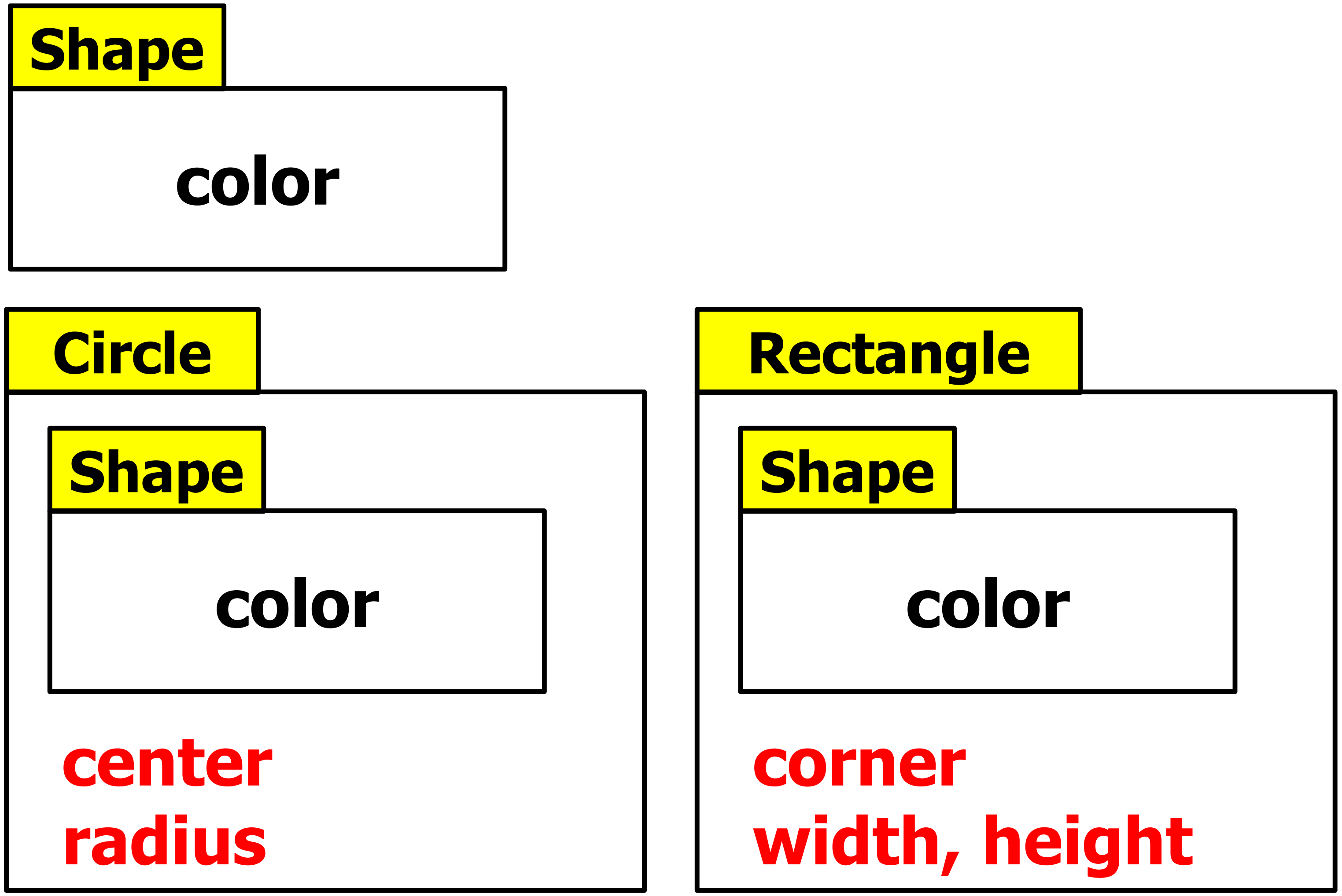
```
class Shape {
public:
   Shape();
   virtual void draw(Image& img) const = 0;

protected:
   unsigned char color[3];
};

class Rectangle : public Shape {
public:
   void draw(Image& img) const;
   Vec2  corner;
   float width, height;
};

class Circle : public Shape {
public:
   void draw(Image& img) const;
   Vec2  center;
   float radius;
};
```

Inherited Datafields



Protected Members

- Can be thought of as a blend of `private` and `public`
  - Inaccessible to users of the class
  - Accessible from the classes derived from this class.
    - Accessible from the body of the member functions of the derived classes.

Protected Members

- Example

- Inaccessible to users of the class
- Accessible from the classes derived from this class

```
class Base {
public :
   int public_var;
protected :
   int protected_var;
private :
   int private_var;
};

class Derived : public Base {
public :
   void set_zero() {
      public_var = protected_var = 0;  // It is OK.
      private_var = 0;                // Compilation Error !!!
                                      // cannot access private member declared in class 'Base'
   }
   void func(Base& base) {
      base.protected_var = 1;         // Compilation Error !!!
      // The derived class has no special access to protected members of base type objects.
   }
};

void main() {
   Derived derived;
   derived.public_var = 0;       // It is OK.
   derived.protected_var = 0;    // Compilation Error.
   derived.private_var = 0;      // Compilation Error, too.
   // cannot access protected or private member declared in class 'Base'
}
```

Public, Protected, Private Inheritance

- public Inheritance
  - Members of the base retain their access levels
    - public -> public
    - protected -> protected
- protected Inheritance
  - Public and protected members of the base class are protected members in the derived class
    - public -> protected
    - protected -> protected
- private Inheritance
  - All the members of the base class are private in the derived class
    - public -> private
    - protected -> private

Public, Protected, Private Inheritance

- Example

```
class Base {
public :
   int public_var;
protected :
   int protected_var;
private :
   int private_var;
};

class Public_Derived : public Base {
   // public_var is public
   // protected_var is protected
   // private_var is not accessible
};

class Protected_Derived : protected Base {
   // public_var, protected_var are protected
   // private_var is not accessible
};

class Private_Derived : private Base {
   // public_var, protected_var are private
   // private_var is not accessible
};
```