Module 8: Polymorphism

Dr. Tran Minh Triet

Acknowledgement

- Slides
  - Course CS202: Programming Systems

    Instructor: MSc. Karla Fant,

    Portland State University
  - Course CS202: Programming Systems

    Instructor: Dr. Dinh Ba Tien,

    University of Science, VNU-HCMC
  - Course DEV275: Essentials of Visual Modeling with UML 2.0

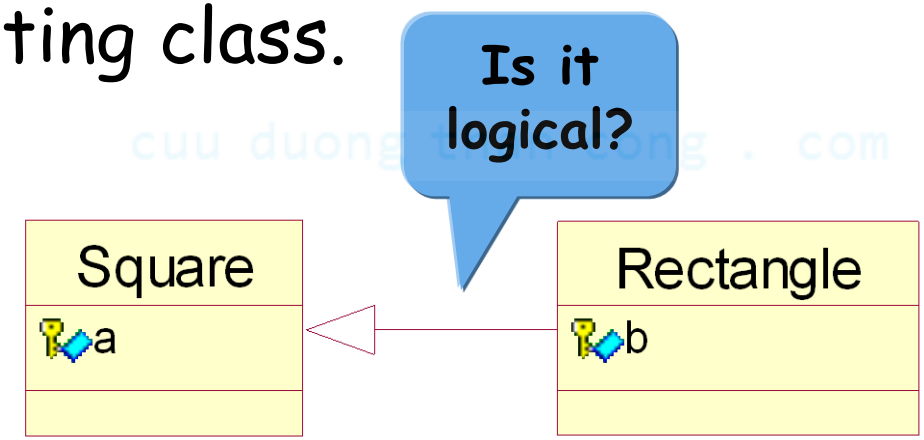
    IBM Software Group

Outline

- IS-A relationship
- Initialization of a pointer to base class
- Static binding
- Typecasting of a pointer to an object
- Virtual functions and polymorphism
- Pure virtual functions
- Abstract class

IS-A relationship

- Do not implement inheritance when you only want to use some attributes or behaviors of an existing class.



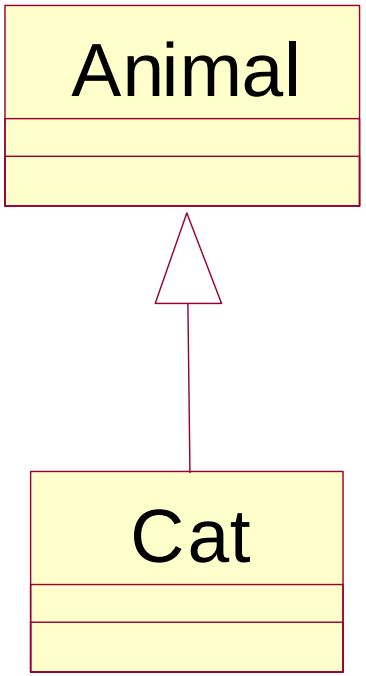
- The inheritance is only applied when there is a "**IS-A**" relationship between classes
  - E.g. Dog is an animal. Or, employer is an employee.

An example

- Cat "is an" animal.

```
class Animal
{
   …
};

class Cat: public Animal
{
   …
};
```



A pointer to base class

- A pointer to base class can be assigned with the address of an object of the derived class.
- For example:

```
Animal      *pAni;
Cat         c;
pAni = &c; //OK
```

Implicit type conversion in inheritance

- It is normal to pass a derived class variable to a function with an argument of base class data type.
- The compiler will do an implicit conversion.

```
Animal::Process(const Animal &a);
int main()
{
   Cat c;
   Animal::Process(c); //OK
}
```

Initialization of a pointer to base class

```
A*  pA;
```

pA can be instantiated:

```
pA = new X(…);
```

where X is A or X is a class derived from A

Initialization of a pointer to base class

```
A*    pA;
```

```
pA = new A(...);
```

```
pA = new B(...);
```

```
pA = new C(...);
```

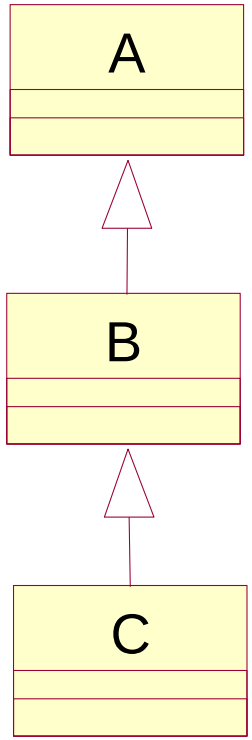
Initialization of a pointer to base class

```
B*    pB;
```

```
pB = new A(...);
```

```
pB = new B(...);
```

```
pB = new C(...);
```



Initialization of a pointer to base class

You are requested to create a polygon

```
CPolygon*  pShape;
```

You can draw a triangle

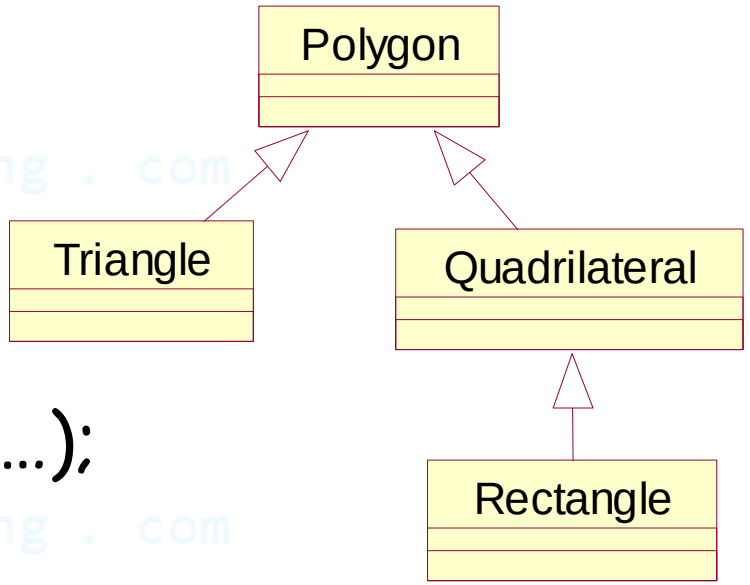
✓ `pShape = new Triangle(...);`

or a quadrilateral

✓ `pShape = new Quadrilateral(...);`

or a rectangle

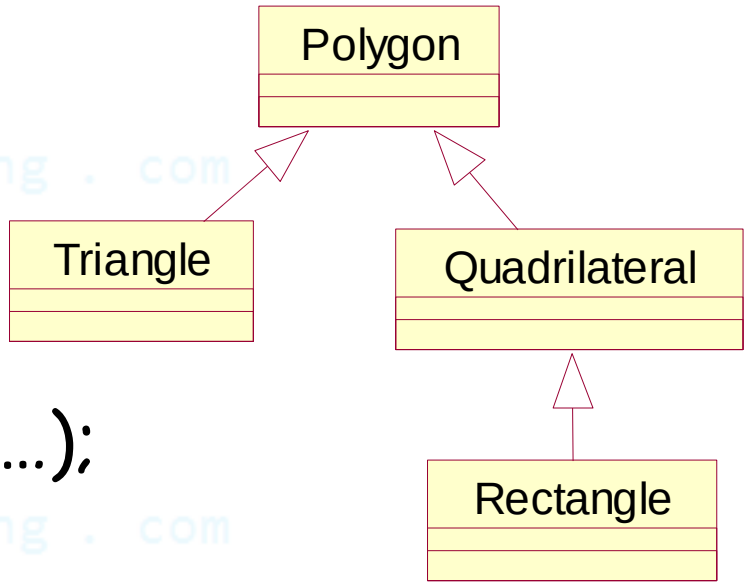
✓ `pShape = new Rectangle(...);`



Initialization of a pointer to base class

You are requested to create a quadrilateral

```
CQuadrilateral* pShape;
```



You can draw a rectangle

✓ `pShape = new Rectangle(...);`

or a quadrilateral

✓ `pShape = new Quadrilateral(...);`

You cannot draw a triangle

~~`pShape = new Triangle(...);`~~

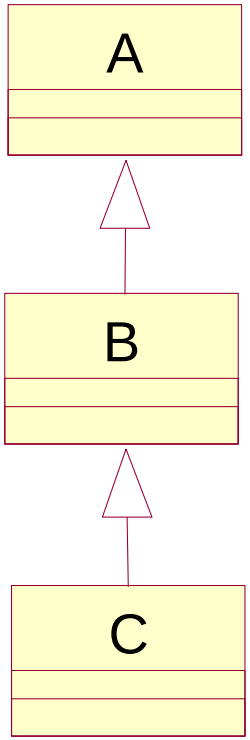
You cannot draw a general polygon (e.g. pentagon)

~~`pShape = new Polygon(...);`~~

Static binding

Consider the following situation:

- class A has void Print()
- class B also has void Print()
- class C has void Print() too




```
int main() {
   C    varC;
   B    varB;
   varB.Print();          // Print() of B
   varC.Print();          // Print() of C
   varC.B::Print();       // Print() of B
}
```

Static binding

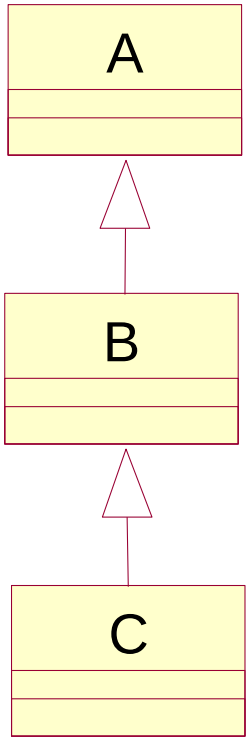
- Another example:

```
int main()
{
   A    varA;
   B    varB;
   C    varC;
   A    *var1, *var2;

   var1 = &varC;
   var2 = &varB;
   var1->Print();    // Print() of A
   var2->Print();    // Print() of A
}
```

var1 is formally a pointer to class A

var1 is actually pointed to an object of class C



Typecasting of pointer to an object

Formally, pA is a pointer to an object of class A

```
A*  pA;
```

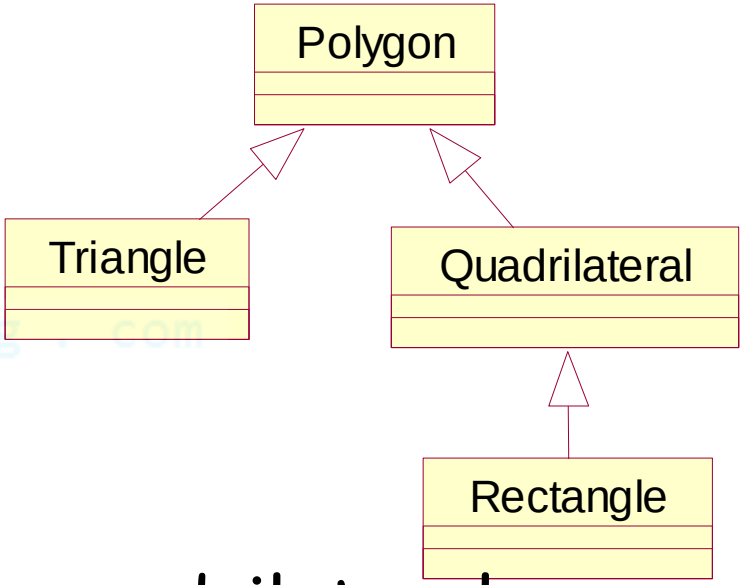
Actually, pA is currently pointer to an object of class X

```
pA = new X(…);
```

pA can be typecasted to a pointer to an object of class Y where Y is X or Y is a base class of X

```
Y* pY;
pY = (Y*)pA;
```

Typecasting of pointer to an object



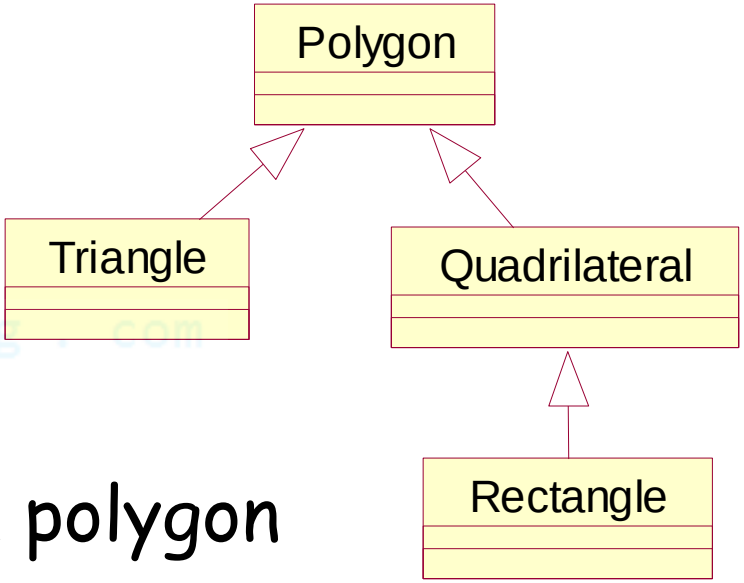
pShape is formally a pointer to a quadrilateral

```
CQuadrilateral* pShape;
```

Actually, pShape is a pointer to a rectangle

```
pShape = new Rectangle(...);
```

Typecasting of pointer to an object



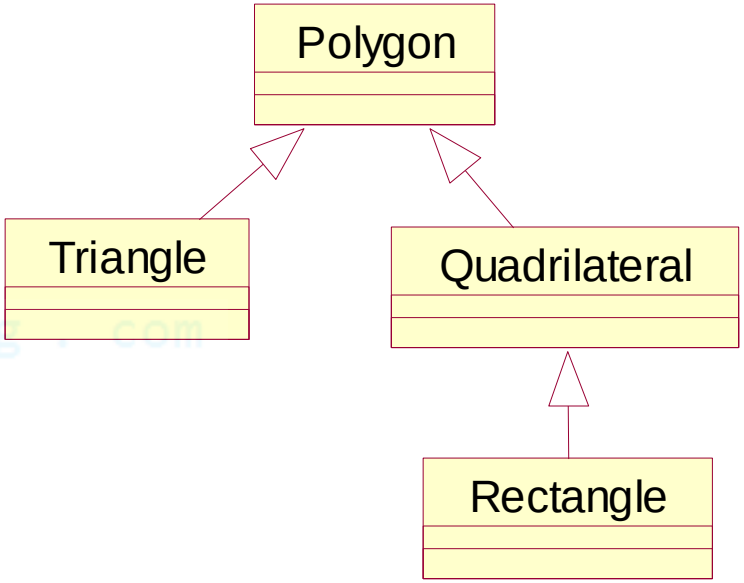
pShape is currently pointed to a polygon

✓ Polygon* p = (Polygon*)pShape;

pShape is currently pointed to a rectangle

✓ Rectangle* p = (Rectangle *)pShape;

Typecasting of pointer to an object



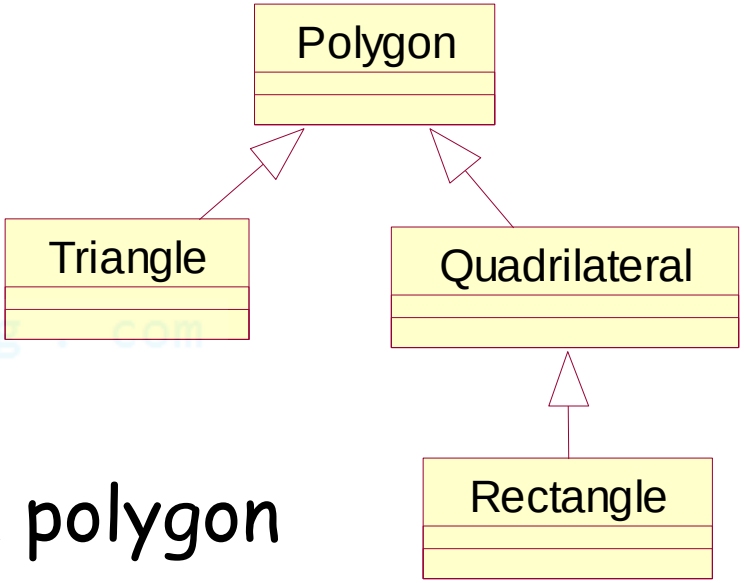
We no longer need this object

```
delete pShape;
```

Now, pShape is re-instantiated as a pointer to a quadrilateral

```
pShape = new Quadrilateral(...);
```

Typecasting of pointer to an object



pShape is currently pointed to a polygon

✓ Polygon* p = (Polygon*)pShape;

We are not sure if pShape is currently pointed to a rectangle

~~Rectangle* p = (Rectangle *)pShape;~~

Method invocation

pA -> F(…);

F(…) must be a member function of the *current* formal type of pA

The actual behavior of F(…) should correspond to the *actual* type of pA

Method Invocation

```
class  Polygon
{
public:
   double      Surface();
   double      Draw();
}

class  Triangle:
      public Polygon
{
public:
   double      Surface();
   double      Draw();
}
```

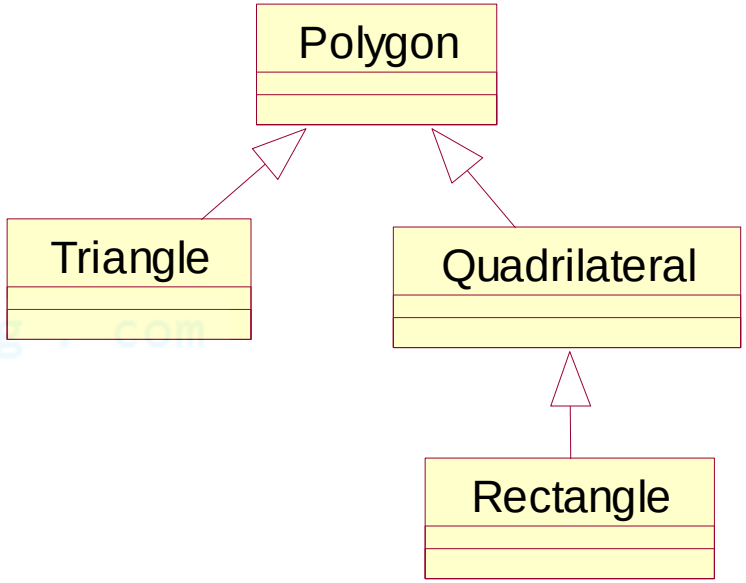
```
class  Quadrilateral :
      public Polygon
{
public:
   double      Surface();
   double      Draw();
}
class  Rectangle:
      public Quadrilateral
{
public:
   double      Surface();
   double      Draw();
}
```

Method Invocation

```
Polygon         myPolygon;
myPolygon.Draw();
```

```
Triangle        myTriangle;
myTriangle.Draw();
```

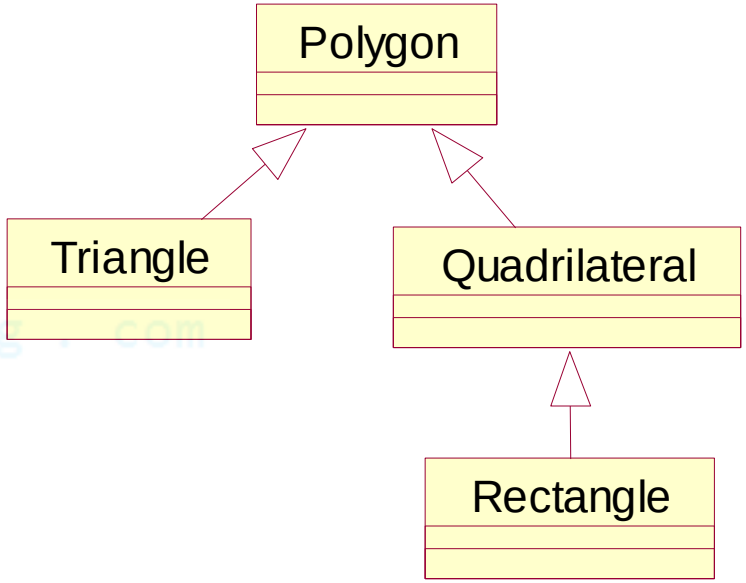
```
Polygon* pPolygon;
pPolygon = new Triangle(...);
pPolygon->Draw();
```



Method Invocation

```
Quadrilateral*      pPolygon;
pPolygon = new Rectangle(…);
pPolygon->Draw();

…
// IN CLASS DEMO
```



Virtual Method Invocation

```
class  Polygon
{
public:
virtual double Surface();
virtual double Draw();
}

class  Triangle:
      public Polygon
{
public:
virtual double Surface();
virtual double Draw();
}
```

```
class  Quadrilateral :
      public Polygon
{
public:
virtual double Surface();
virtual double Draw();
}
class  Rectangle:
      public Quadrilateral
{
public:
virtual double Surface();
virtual double Draw();
}
```