Похідні класи. Наслідування

Derived classes. Iheritance

Б.Страуструп:

Похідні типи дають *простий, гнучкий і ефективний* апарат для:

- задання класові альтернативного інтерфейсу (повторне використання коду)
- визначення класу шляхом додавання нових можливостей до базового класу без перепрограмування чи перекомпілювання

Синтаксис наслідування

```
class  ідентифікатор : список наслідування
{
    оголошення і визначення членів класу;
};
```

список наслідування:

специфікатори наслідування ідентифікатор базового класу ,

специфікатори наслідування ідентифікатор базового класу ,

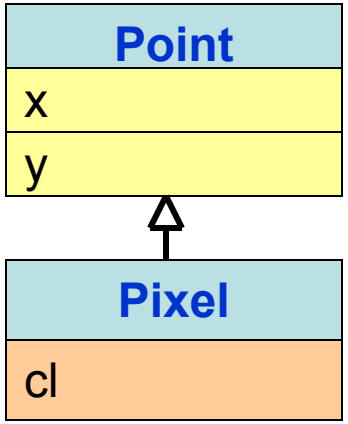
. . .

специфікатори наслідування ідентифікатор базового класу

специфікатор наслідування:

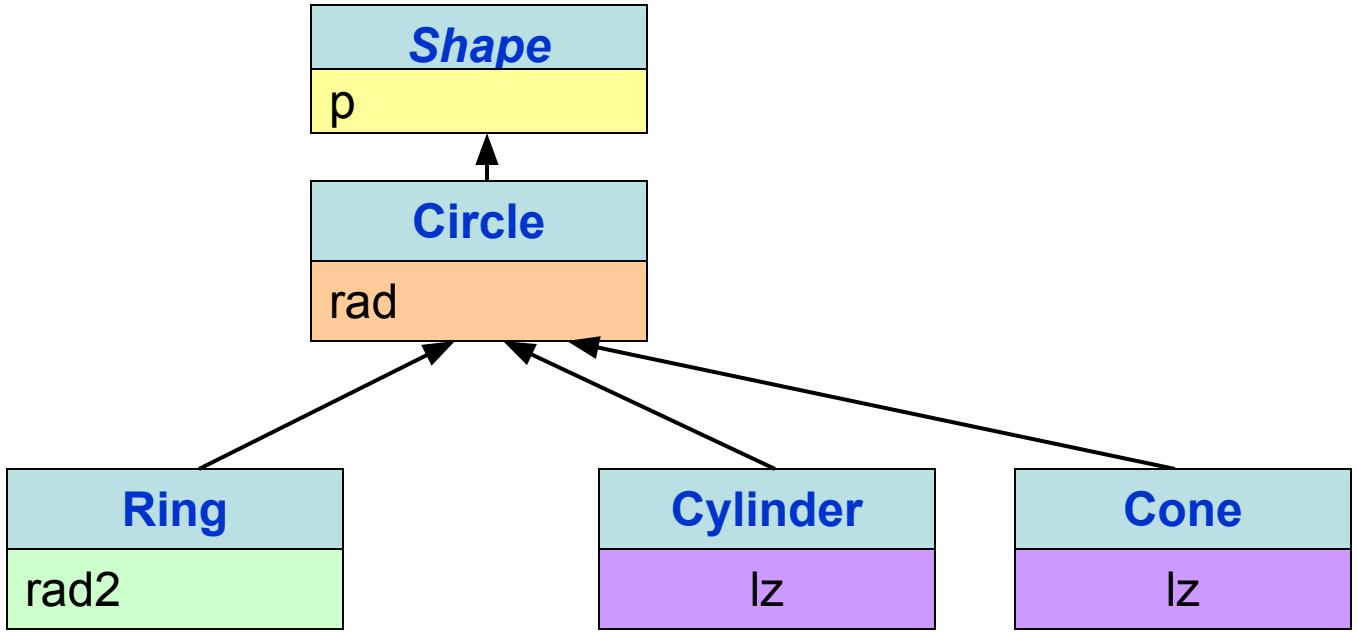
```
public
protected
private
virtual
```

Діаграми класів



Базовий клас (base class)
тип з деякими даними і методами

Похідний клас (derived class)
подібний на базовий, але має нові дані та фунціональність



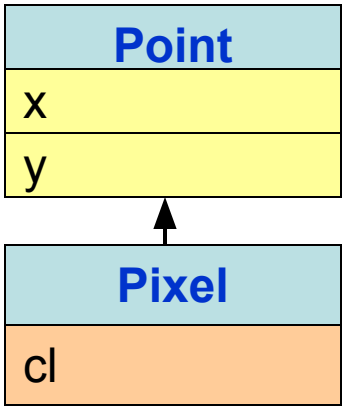
2 види наслідування C++

- **Наслідування реалізації**
  - всі члени крім стандартних базових типів стають членами похідного типу
  - функціональність базового і похідного типів може відрізнятись при однакових назвах методів
- **Наслідування інтерфейсів**
  - базові класи – абстрактні (pure methods)
  - у похідному – реалізація чистих методів
  - фактично можна використати лише функціональність, яка реалізована у похідному типі
  - немає спеціальної конструкції для інтерфейсу

Наслідування реалізації

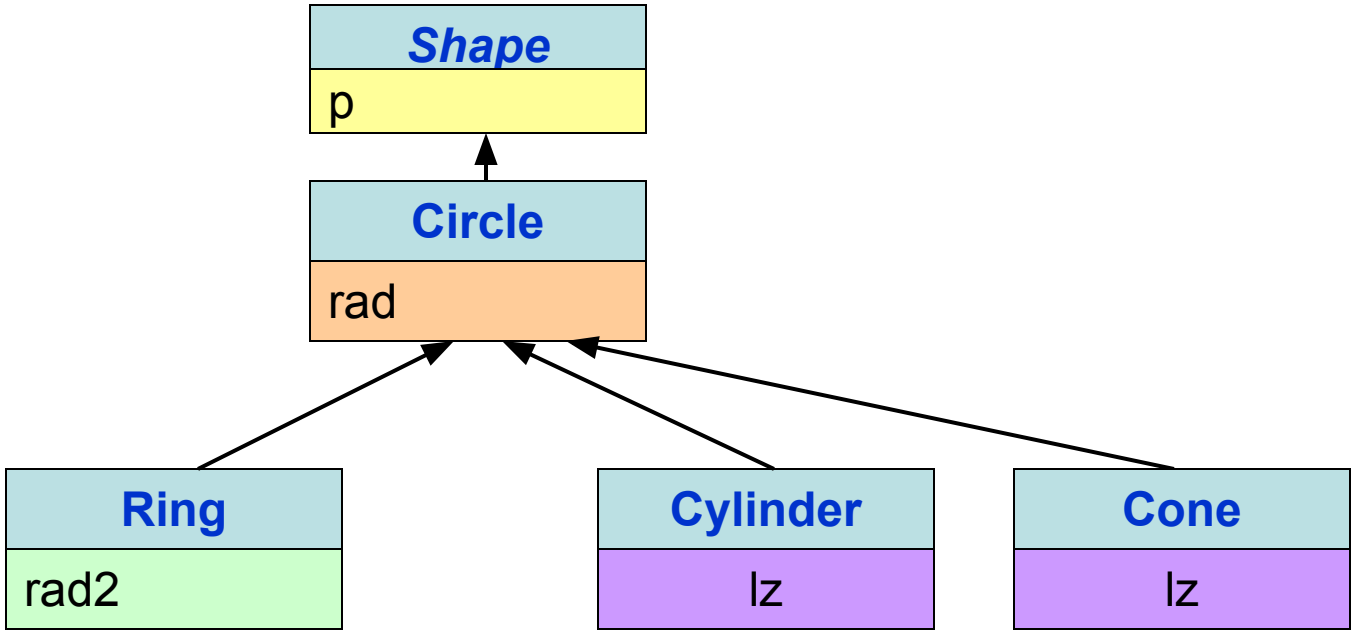
- всі члени (дані і методи) крім стандартних (к-тори, д-тор, присвоєння) базових типів стають членами похідного типу
- специфікатори керують лише доступом до членів
- виклик конструкторів усіх базових типів згідно списку наслідування зліва направо
- виклик деструкторів базових типів у порядку, зворотньому до виклику конструкторів
- функціональність базових типів у похідних типах може змінюватися за рахунок перевизначення методів і поліморфізму
- усунення неоднозначності виклику методів:
  - задання області видимості конкретного класу
  - приведення типу

Діаграми класів



Базовий клас
(base class)

Похідний клас
(derived class)



class Shape

```
class Shape{
protected:
   Point p;
public:
   Shape(const Point& _p):p(_p){}

   virtual  void print(){ cout<<"Shape: p="<<p<<endl; }
   virtual  float perimeter() =0;
   virtual ~Shape(){}
};
```

class Circle, class Ring

```
class Circle: public Shape{
protected:
   int rad;
public:
   Circle(const Point& pp, int r): Shape(pp), rad(r){}
   void print(){cout<<"Circle: p="<<p<<",
                    rad="<<rad<<endl; }
   float perimeter(){ return 2*PI*rad;}
};

class Ring: public Circle{
protected:
   int rad2;
public:
   Ring(const Point& pp, int r1, int r2)
      :Circle(pp,r1), rad2(r2){}
   void print(){cout<<"Ring: p="<<p<<", rad1="<<rad<<",
                    rad2="<<rad2<<endl; }
   float perimeter(){ return 2*PI*(rad+rad2);}
};
```

class Cylinder, class Cone

```
class Cylinder: public Circle{
   int lz;
public:
   Cylinder(const Point& p, int r, int l)
      :Circle(p,r), lz(l){}
   void print(){...; Circle::print(); ... }
   float square(){ return perimeter()*lz;}
};
class Cone: public Circle {
   int lz;
public:
     Cone(const Point& p,int r,int h)
      :Circle(p,r), lz(h){}
     void print(){...; Circle::print(); ....  }
     float square(){
        return perimeter()*sqrt(float(lz*lz+rad*rad))/2;
};
```

Діаграма об'єктів