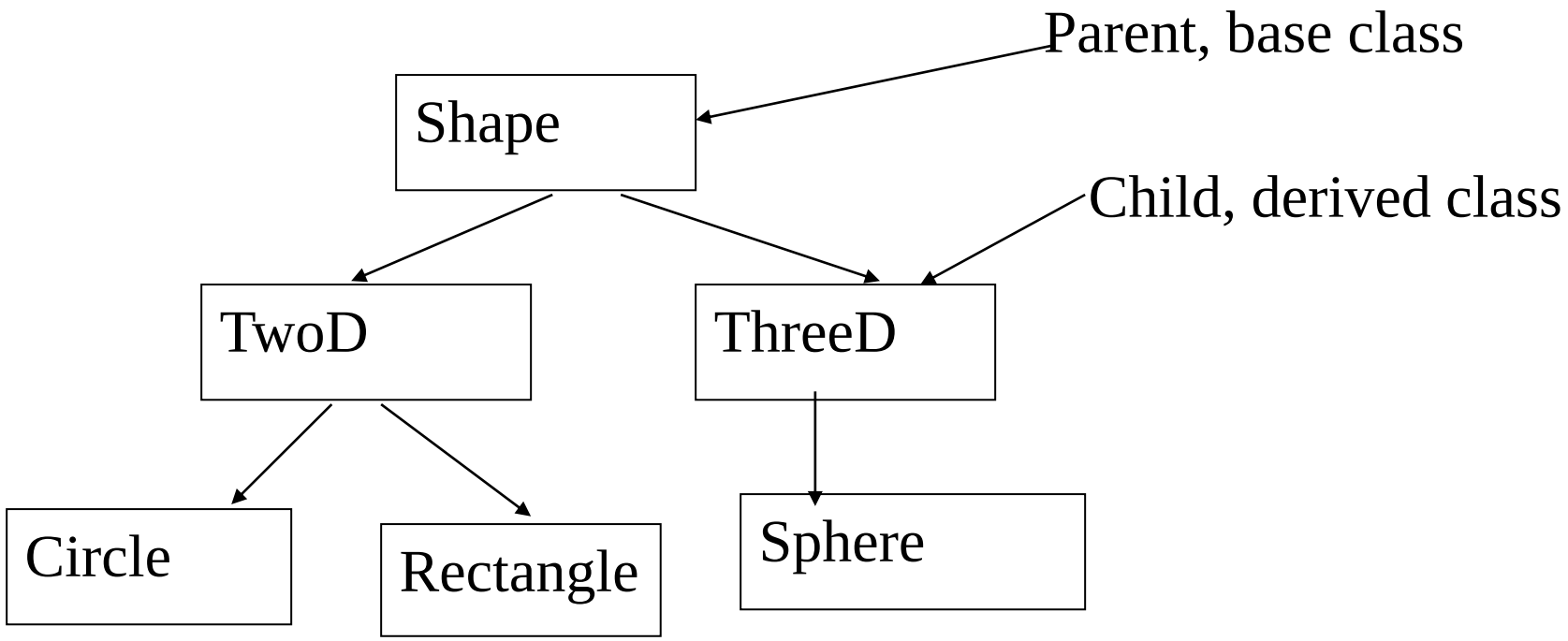
Lecture 4

Inheritance -Part 2

University of Anbar
College of Computer Science and Information Technology
Department of Computer Science
Object Oriented Programming
Second Class
Dr. Ruqayah R. Al-Dahhan

Inheritance in C++



How can we code our objects like this?

```
#include <iostream>
Using namespace std;
const static double  PI =  3.141592654;

class Shape{  // base class
 private:
  char * label;  // a string label  for the  shape
};
class TwoD : public  Shape{  // two dimensional  shapes
 private:
  double  x, y;
};
class ThreeD : public Shape{  // three dimensional shapes
 private:
  double x, y, z;
};
class Circle : public TwoD{  // Circle  extends TwoD
 public:
  double area(){ return  PI * radius * radius; }
 private:
  double radius;
};
class Rectangle : public TwoD{
 public:
  double area(){ return  width * height; }
 private:
  double width, height;
};

class Sphere : public ThreeD{
 public:
  double  volume(){ return 3.0 / 4.0  * PI * radius * radius *
      radius; }
 private:
  double radius;
};
int main(){
  Sphere s1;
  Circle  c1;
  Shape *shptr;
  shptr = & s1;
  shptr = & c1;
  return 0;
}
```

Notes

- No output - this is only a skeleton code.
- The public inheritance relations between the classes.
- Shape is the base class, and TwoD and ThreeD are derived from it.
- We can group information (and avoid having to duplicate code) in a way that reflects our application and the real things or ideas that our code models.
- Note that in main, a Shape pointer can point to any object that is derived from it.

Inheritance

For the parent (of class Circle) we might write:

```
class TwoD : public Shape{
 public:
    void print(){cout<< x;}
     //..
 protected:
    double x,y;
    //..
};
```

`protected:` data of the class is accessible to the derived classes (and friends), but not any other part of the program.

Inheritance

For the parent class of Circle we write:

```
class TwoD:public Shape{
 public:
   void print(){cout<<…;}
    //..
 protected:
   double x,y;
   //..
};
```

For the derived class we write:

```
class Circle:public TwoD{
 public:
   double area();
 private:
   double radius;
};
```

`public print()` and `protected x` and `y,` are also part of `Circle` - inherited from the parent.

Inheritance

```
class Circle:public TwoD{
 public:
   double area();
 private:
   double radius;
};
```

This is called **public inheritance**.

`public` and `protected` data of the parent class are inherited in the derived class and have the same access type. (We can have private and protected inheritance as well. Public inheritance is the most commonly used.)

```
#include <iostream>
Using namespace std;
const static double  PI =  3.141592654;

class Shape{  // base class
 private:
  char * label;  // a string label  for the  shape
};
class TwoD : public  Shape{  // two dimensional  shapes
 protected:
  double  x, y;
};
class ThreeD : public Shape{  // three dimensional shapes
 protected:
  double x, y, z;
};
class Circle : public TwoD{  // Circle  extends TwoD
 public:
  double area(){ return  PI * radius * radius; }
 private:
  double radius;
};
class Rectangle : public TwoD{
 public:
  double area(){ return  width * height; }
 private:
  double width, height;
};
class Sphere : public ThreeD{
 public:
  double  volume(){ return 4.0 / 3.0  * PI * radius * radius *
     radius; }
 private:
  double radius;
};
int main(){
  Sphere s1;
  Circle  c1;
  cout << "size of  s1 is " << sizeof(s1) << endl;
  cout << "size of  c1 is " << sizeof(c1) << endl;
  return 0;
}
```

Notes

- **Output:**

size of s1 is 36

size of c1 is 28

36 = 4 (char *) + 3 * 8 (double) + 8 (double)

28 = 4 (char*) + 2 * 8 (double) + 8 (double)

Circles inherit label and x and y

Spheres inherit label, x,y and z