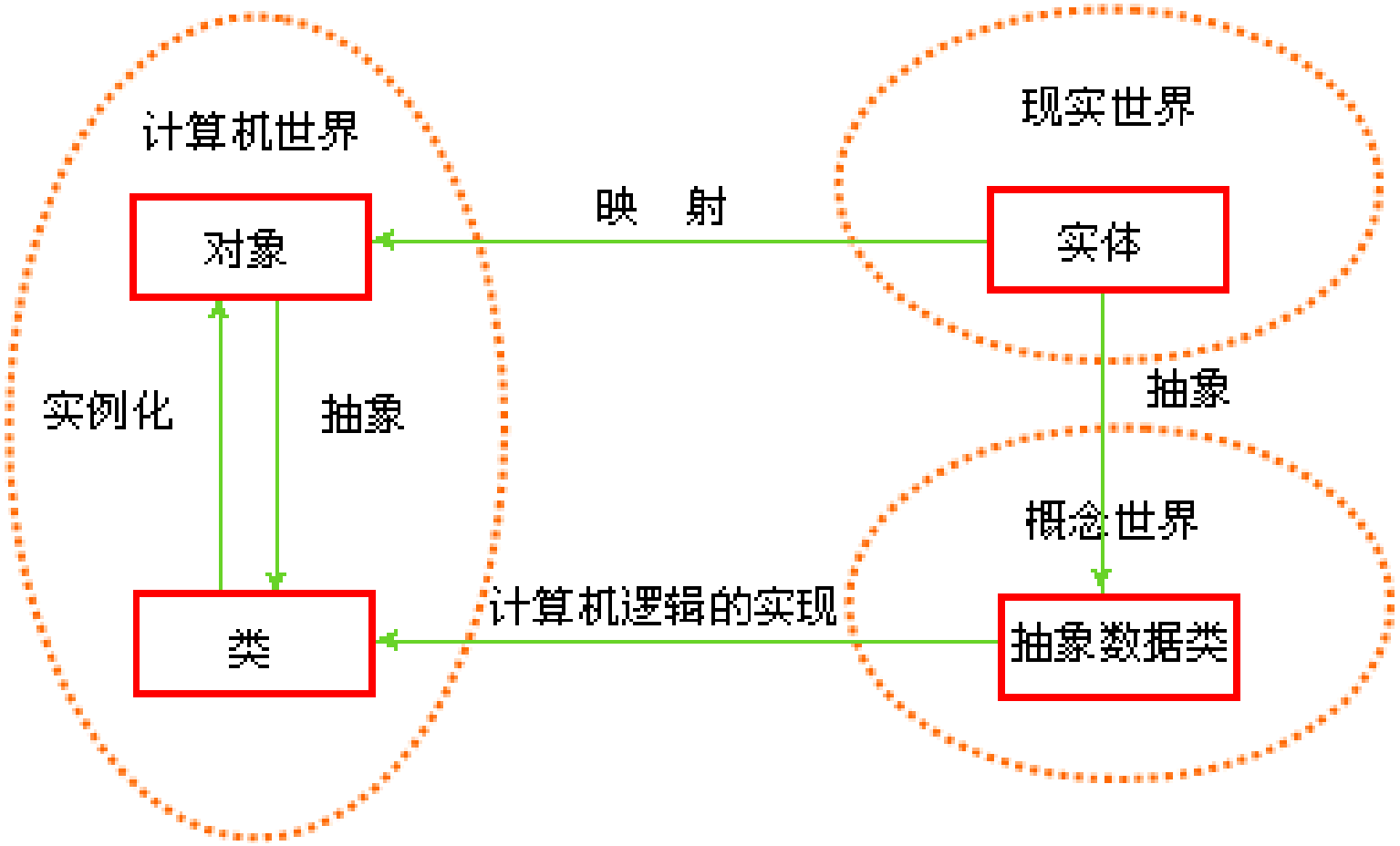
面向对象方法与C++程序设计

第2章　　类与对象

大连理工大学
主讲人-赵小薇

理解对象





对象的定义



1. 在定义类的同时直接定义对象

```
class  Clock
{
private :
    int hour, minute, second;    //关于时间的数据；
public :
    void setClock(int h, int m, int s){…}
    void showClock ( ){…}
} clock1,clock2;
```

对象的定义



2.在定义类的同时直接定义对象，并且不给类命名

```
class
{
private :
    int hour, minute, second;          //关于时间的数据；
public :
    void setClock(int h, int m, int s){…}
    void showClock ( ){…}
} clock1,clock2;
```

此处无类名

对象的定义



3.定义类以后，在使用对象之前再定义对象

```
class Clock
{
private :
    int hour, minute, second;              //关于时间的数据；
public :
    void setClock(int h, int m, int s){…}
    void showClock ( ){…}
};
Clock clock1,clock2;
```

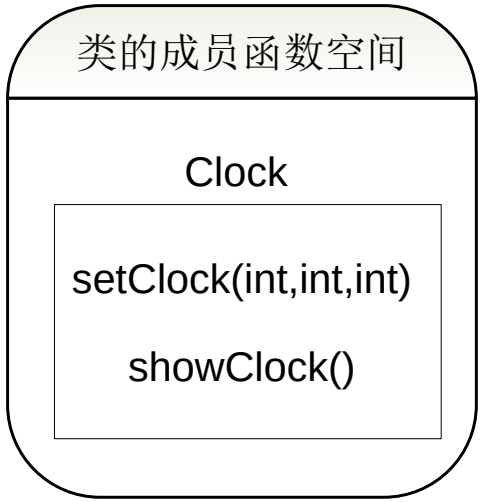
另外一条语句定义；
这种方式最常用

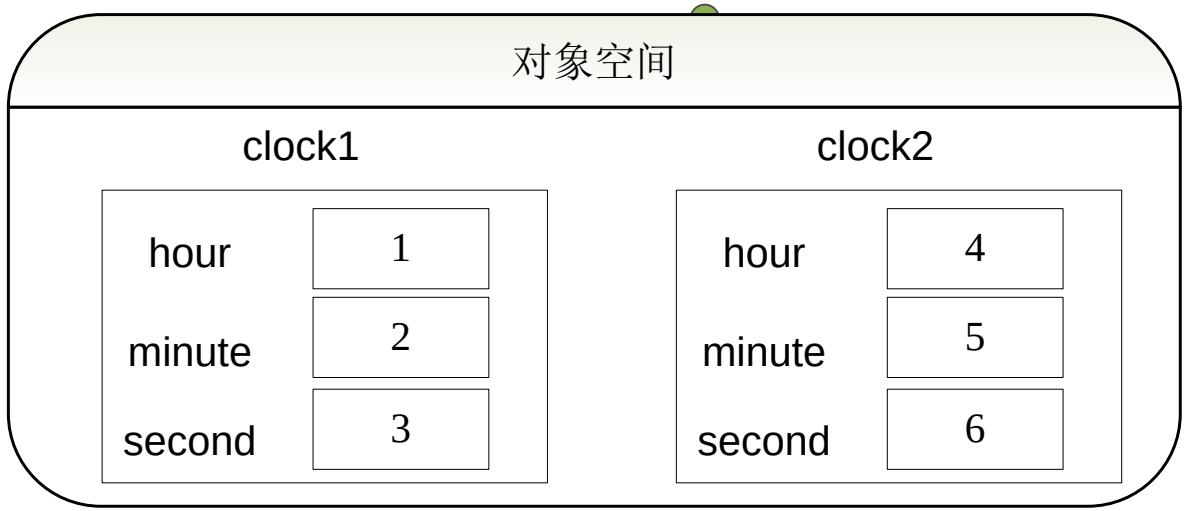
对象的存储空间



```
Clock clock1,clock2;
clock1.setClock(1,2,3);
clock2.setClock(4,5,6);
```

每个对象有各自的空间





对象的访问



三种访问方式

```
Clock clock1;
clock1.setClock(1,2,3);  //对象名直接访问
Clock *p;                //p为指向Clock类型的指针
p=&clock1;               //p为指向clock1对象
Clock & clock3=clock2;  //clock3为对clock2的引用
```

举例

计算两点之间的距离

```
class Point
{
public:
double distance(Point & p) {
    return sqrt((p.x-x)*(p.x-
     x)+(p.y-y)*(p.y-y));   }
void  setX(double i){x=i;}
void  setY(double j){y=j;}
private:
  double x;
  double y;
};
```

```
void main(){
   Point p1,p2;
   p1.setX(2);p1.setY(2);
   p2.setX(5);p2.setY(6);
  cout<<p1.distance(p2);
}
```

distance参数必须引用吗？

distance怎样调用？

p.x 对象名.私有成员？

举例代码分离



头文件point.h包含类定义：

```
//point.h
class Point
{
public:
double distance(Point & p);
void  setX(double i);
          void  setY(double j);
private:
  double x;
  double y;
};
```

源文件point.cpp包含类实现：

```
//point.cpp
#include<iostream>
#include<cmath>
#include"point.h"
using namespace std;
double Point::distance(Point &
p){
return sqrt((p.x-x)*(p.x-x)+(p.y-
y)*(p.y-y));    }
void  Point::setX(double i){x=i;}
void  Point::setY(double j){y=j;}
```

源文件main.cpp对类的使用：

```
//ch3_2.cpp
#include<iostream>
#include<cmath>
#include"point.h"
using namespace std;
void main(){
  Point p1,p2;
  p1.setX(2);p1.setY(2);
  p2.setX(5);p2.setY(6);
  cout<<p1.distance(p2);
}
```

把类的设计、实现和使用完全分离开来，程序结构更加合理，便于项目的团队研发。