面向对象方法与C++程序设计



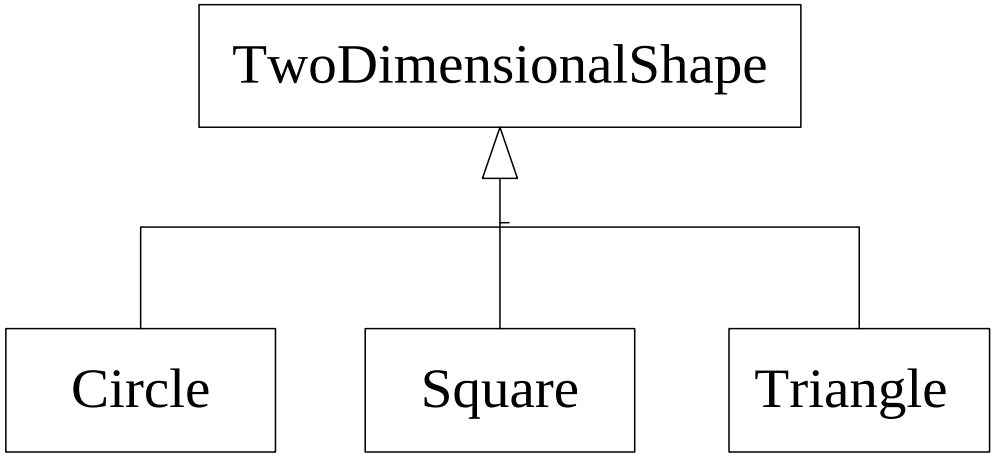
第5章　　多态

大连理工大学
主讲人-赵小薇

类族



- 二维图形派生关系：



比如现在有一个test函数用来测试平面图形的信息，如下：

```
void test( TwoDimensionalShape & t ){
    t.print();                                          // 输出平面图形信息
    cout<<"面积为"<<t.getArea()<<endl;     // 输出平面图形面积
}
```

该类族应该怎样设计呢?



TwoDimensionalShape类的getArea函数该如何定义如下：

```
virtual double getArea()
    { return 0;}
```

函数的返回值设定为0，实际上这个函数的返回值是无法确定的。二维图形本身就是一个抽象的概念，其面积也是无法求的，将面积的值设定为0，其实是一种不得已的做法。

纯虚函数与抽象类



- C++语言为这种无法具体化的抽象函数提供了一种声明的方式：

```
virtual double getArea() = 0;
```

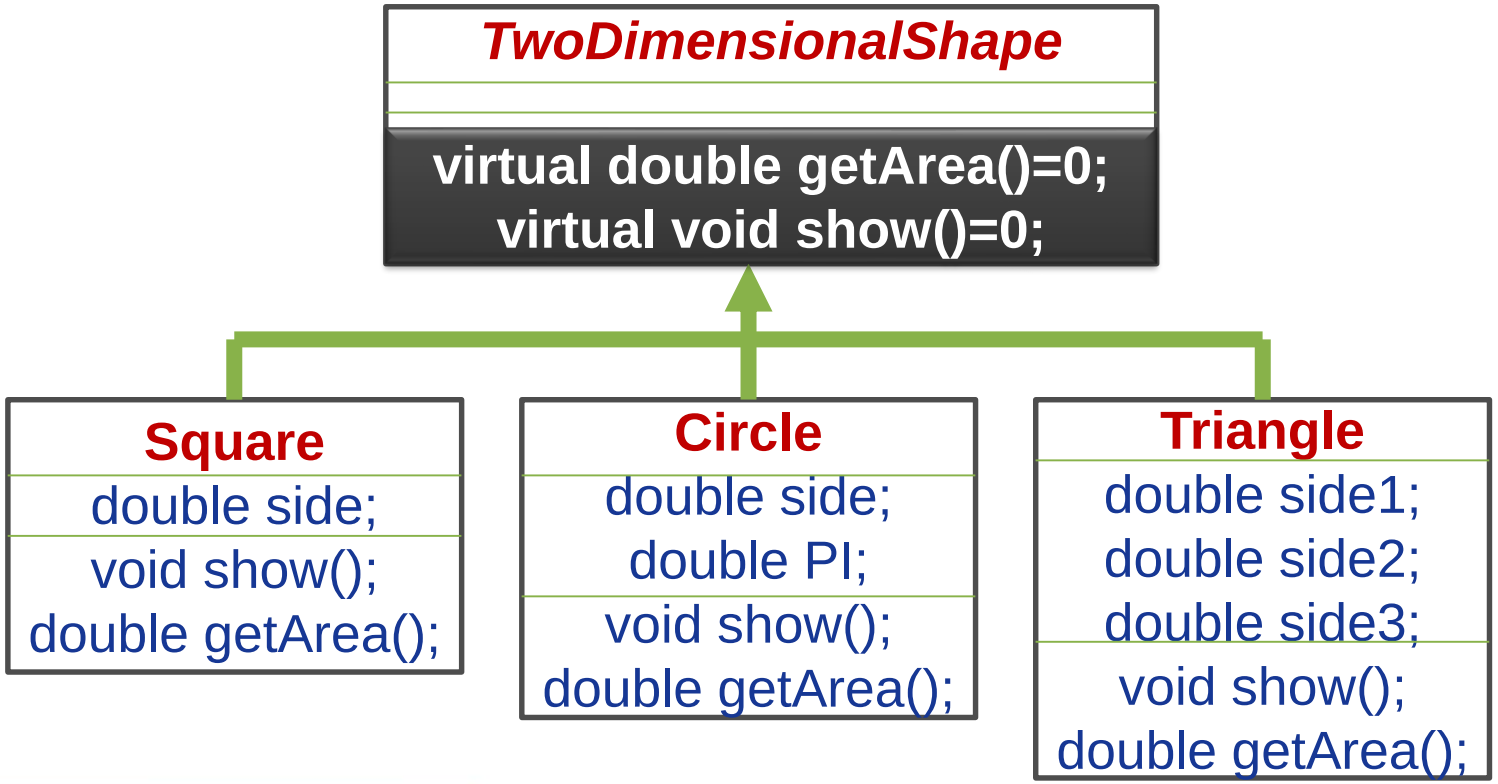
- 此函数没有函数体，只给出函数的原形，并在后面加上“=0”，这样就把getArea函数声明为一个**纯虚函数**（pure virtual function）。纯虚函数的一般形式是：

```
virtual 函数类型 函数名（参数列表）= 0;
```

- 注意，纯虚函数是没有函数体的，与下面的函数具有本质上的区别：

```
virtual void getArea(){ }
```

- 利用虚函数和纯虚函数设计二维图形类族。
- 所有的图形都具有getArea函数能够计算面积；都具有show函数输出图形信息。



用test函数测试二维图形

```
void test(TwoDimensionalShape & t){
    t.show();
    cout<<"面积为"<<t.getArea()<<endl;
}
// 主函数
void main(){
    Square s(10);
    Circle c(10);
    test(s);
    test(c);
}
```

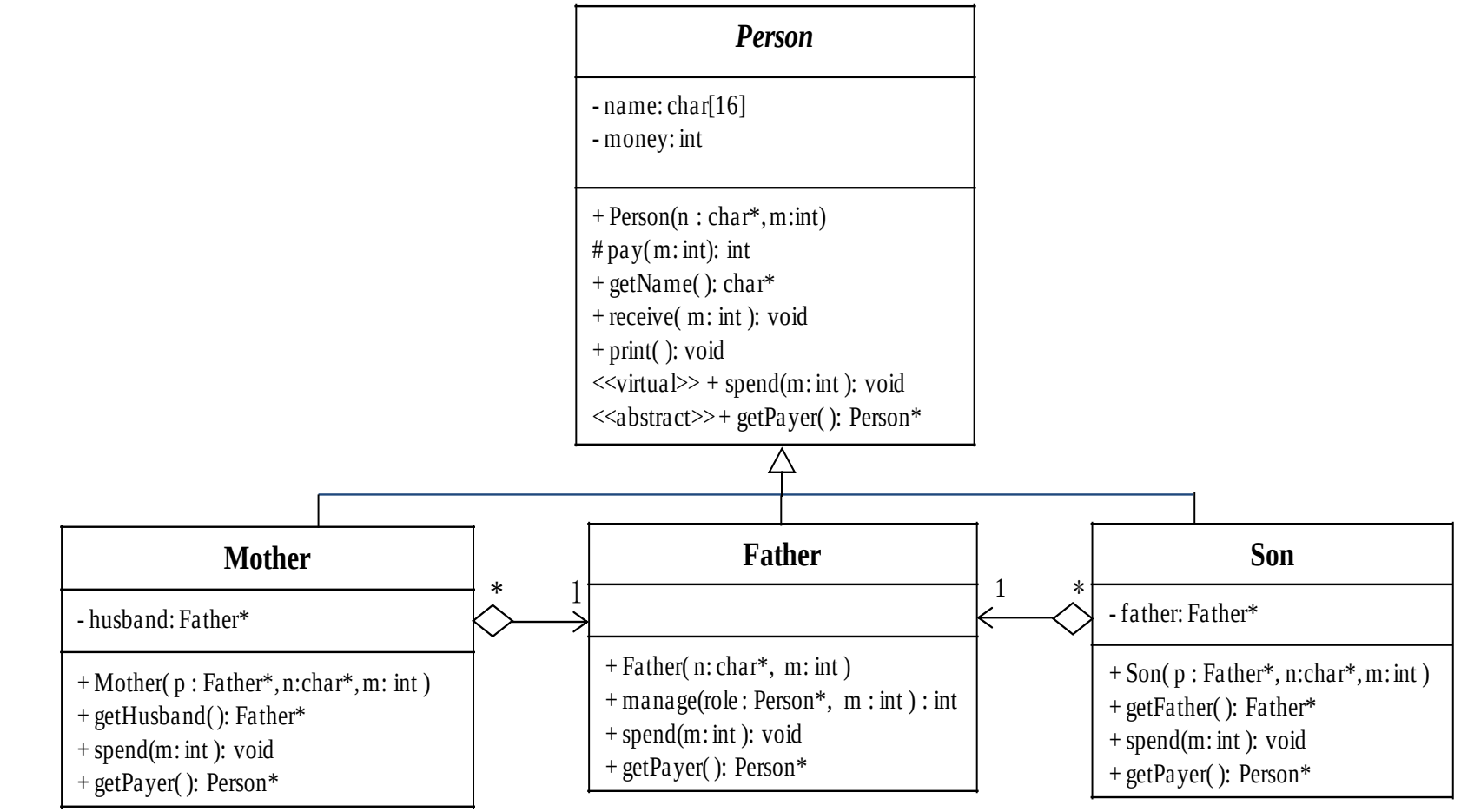
输出结果：
这是边长为10的正方形
面积为100
这是半径为10的圆形
面积为314

如果二维图形又派生出了椭圆形，用于测试的test函数需要需改吗？这有何意义？

案例分析



- 本章将继续探讨父亲给儿子钱的问题。
- 每个人都可以花自己的钱进行日常消费，所以可以为Person类添加消费spend函数，父亲和儿子类都可以继承。
- 由于父亲和儿子在家庭中担任的角色不同，所以他们消费的具体项目是不一样的。这样，对于继承结构中的同名函数，不同的派生类会产生不同的行为。
- 可以使用虚函数来实现动态多态，以完成这样一种不同对象对同一指令产生不同相应的效果。所以可以将基类Person中的spend函数设置为虚函数（virtual），在Father和Son中分别重新定义这个函数以实现各自不同的功能。


Person
- name: char[16]
- money: int
+ Person(n : char*, m:int)
# pay(m: int): int
+ getName( ): char*
+ receive(m: int): void
+ print( ): void
<<virtual>> + spend(m: int): void
<<abstract>> + getPayer( ): Person*
Mother
- husband: Father*
+ Mother(p : Father*, n:char*, m: int)
+ getHusband( ): Father*
+ spend(m: int): void
+ getPayer( ): Person*
Father
+ Father(n: char*, m: int)
+ manage(role : Person*, m : int) : int
+ spend(m: int): void
+ getPayer( ): Person*
Son
- father: Father*
+ Son(p : Father*, n:char*, m: int)
+ getFather( ): Father*
+ spend(m: int): void
+ getPayer( ): Person*
*
1
1
*

Person类



```
class Person{                                   //Person类定义
private:
        char name[16];                          //姓名字符串
        int money;                              //持有钱数
protected:
        int pay(int m);                         //支付m钱
public:
        Person(char *n,int m);                  //构造函数
        char * getName();                       //返回name
        void receive(int m);                    //接收m钱
        void print();                           //输出函数
        virtual void spend(int m);              //消费函数
};
```

Father类与Son类



```
class Son;                                  //Son类声明
class Father:public Person{                 //Father类定义
public:
        Father(char *n,int m);              //构造函数
        int manage(Son *role,int m);        //授权管理
        void spend(int m);                  //消费函数
};
class Son:public Person{                    //Son类定义
private:
        Father * father;                        //父亲
public:
        Son(Father *p,char *n,int m);       //构造函数
        Father * getFather();               //返回Father
        void spend(int m);                  //消费函数
};
```

本章小结



1．继承实现了代码复用，使派生类与基类相似，多态实现了不同的派生类彼此之间又有不同，派生类与基类的不同。

2．基类中的某些功能本身并没有具体的作用，一般把这种函数设计为纯虚函数，它的存在是为了提供给多个派生类一个统一的访问接口，具体功能是在派生类中根据实际对象的要求而实现的。

3. 利用继承和多态技术，可以使程序代码更加简洁可以实现不同类型的对象针对同一指令产生不同的响应。

C++语言的其他知识



- 模板
- 异常处理
- 输入输出操作
- Windows操作