面向对象程序设计(OOP)

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

C++(继承和派生)

福州大学·软件学院·软件工程系
王灿辉(wangcanhui@fzu.edu.cn)

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

概述

- 面向对象程序设计最重要的特性：

抽象、类、对象、**封装**

继承与派生

多态性、重载、动态绑定

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

概述

- 继承最主要的目的就是**复用**别人的代码。派生类可以从基类继承所有的特性，而且可以根据自己的需要改造基类的特性或增加新的特性。
- C++的类库就是代码复用的一个现成的实例。

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

概述

- 继承是面向对象程序设计最重要的机制之一。所谓继承(inheritance)就是利用已有的数据类型定义出新的数据类型。
- 被继承的类称为:**基类**(Base class)或超类(superclass)或父类，定义出的新类称为**派生类**(deriver class)或子类(subclass)

Evaluation Only.
Created with Aspose.Slides for Java 22.7
Copyright 2004-2022 Aspose Pty Ltd.

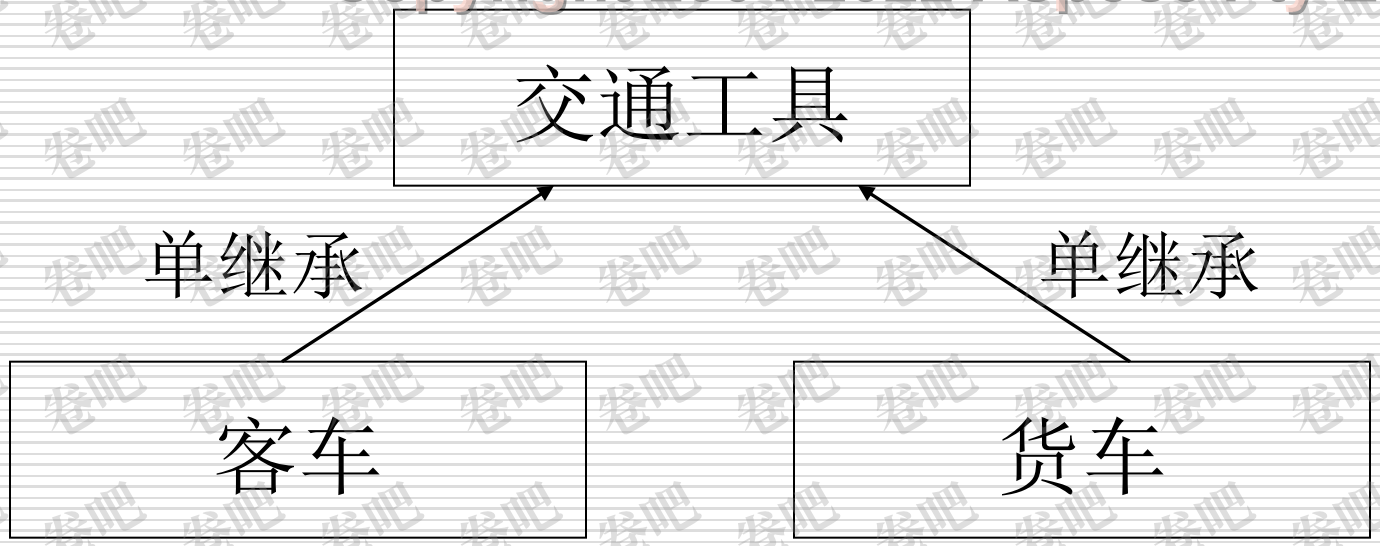
Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

概述：单继承和多继承

- 一个派生类既可以从一个基类派生，也可以从多个基类派生。从一个基类派生称为单继承，从多个基类派生称为多(重)继承

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

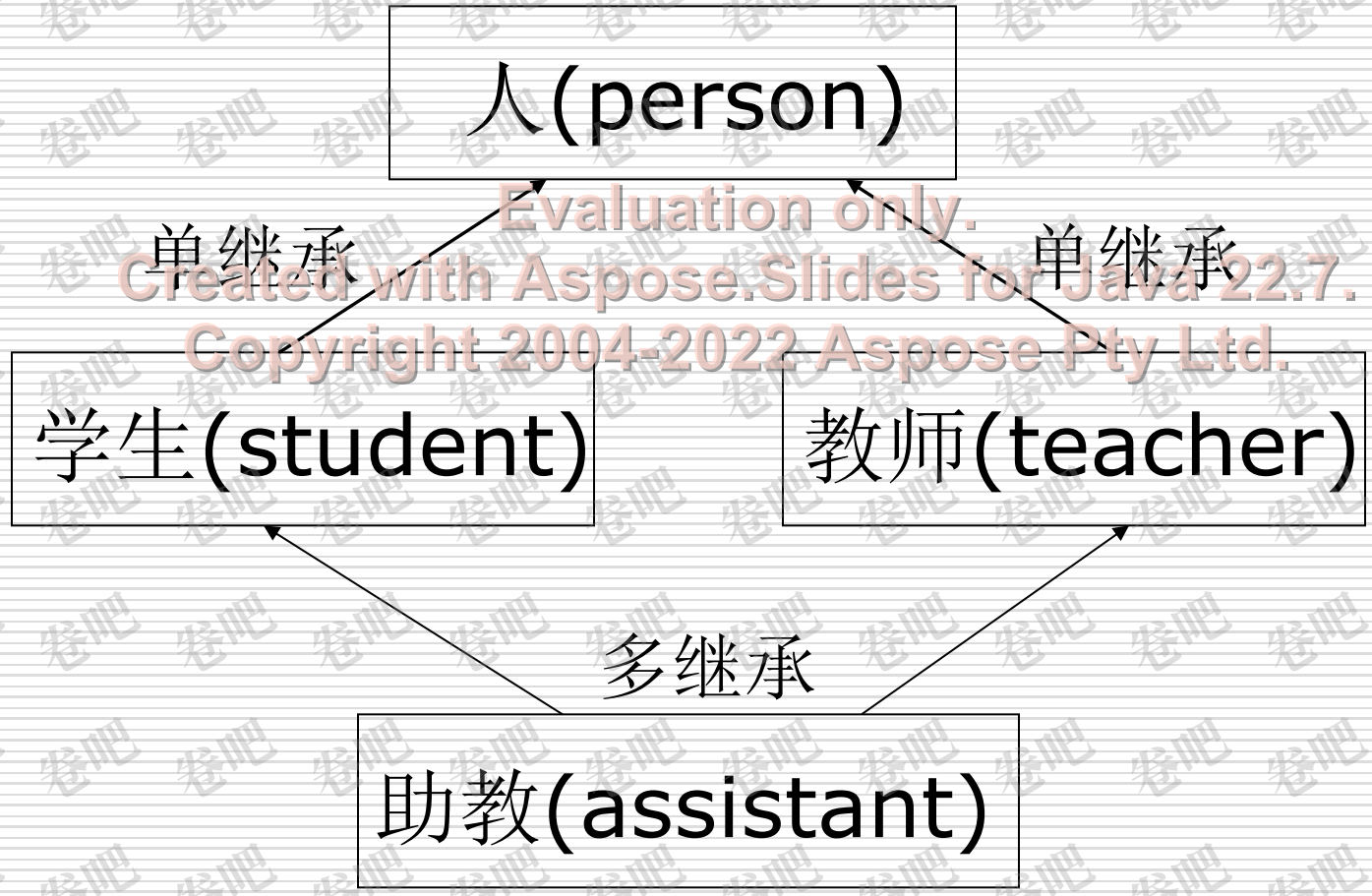


类继承图中的箭头表示："从...派生"

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

概述：单继承和多继承



Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

返回

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

概述

- 派生类既可以对基类的性质进行扩充，又可以对基类进行限制，从而得到更加灵活、适用的可重用模块，大大缩短了程序的开发时间(这正是OO所追求的最主要目标之一)。
- 派生类是基类的一种特例，**派生类和基类的关系是一种的(a kind of)关系**。
- 派生类是基类的具体化，基类是派生类的抽象。

Evaluation Only.
Created with Aspose.Slides for Java 21.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

派生类的定义

- 单继承的定义形式如下：

```
class 派生类名:[访问方式] 基类名
{//基类名必须已定义过！
    派生类新增加的成员
};
```

- 访问方式可以是：public(struct缺省)、protected、private(类缺省)三者之一。

Evaluation Only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

派生类的定义

- 多(重)继承的定义形式如下：

```
class 派生类名:[访问方式] 基类名1,
[访问方式] 基类名2, ……
{//所有基类名必须已定义过！
   派生类新增加的成员
};
```

- 一个类不能多次作为另一个类的直接基类，但可以多次作为间接基类

Evaluation Only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

类的访问(控制)方式

- public(公有):类的外部(当然包含派生类中)均可以访问
- protected(保护):派生类中可以访问，类的外部不能访问
- private(私有):除定义该成员的类外均不能访问(包括派生类中)

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

类的访问(控制)方式

- protected(保护):主要用在对访问权限犹疑不定的时候使用。
- 数据成员极少设置为保护成员；当对函数成员的访问权限难于决定时，可以考虑设置为“保护”权限。

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

公有、保护、私有继承

- 派生类通过访问方式对基类成员的访问方式进行改造。

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

访问控制 / 继承方式	Public	Protected	Private
Public	Public	Protected	不可访问
Protected	Protected	Protected	不可访问
Private	Private	Private	不可访问

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

派生类的生成过程

- 派生类的生成经历三个步骤：
  1、吸收基类成员(除基类的构造、析构函数外的所有数据和函数成员)
  2、改造基类成员(通过访问方式、数据和函数成员的覆盖)
  3、添加新成员(尤其是构造和析构函数)
- 此外：**用户定义的new和赋值运算符及友元关系也不能继承**

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2001-2021 王旭辉. All Rights Reserved.

第6章：继承和派生

公有、保护、私有继承实例

- 无论哪种继承方式，派生类都不能访问基类的私有成员(这正是OO的思想所在)
- 在类的外部(如：在主函数中)只能访问类的公有(数据或函数)成员。不能访问类的私有成员也不能访问类的保护成员(在类的外部，保护成员和私有成员完全一样)。

Evaluation Only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

- 公有继承：基类的保护、公有成员仍然是派生类的保护、公有成员

公有继承实例

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

```
class Base {
private:    int pri;  //私有成员
protected:  int pro;  //保护成员
public:     int pub; //公有成员
            Base(void) {...}
            void show(void) {...}
};
class Derived:public Base {...};
void main() {...}
```

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

公有、保护、私有继承实例

- 保护继承：基类的保护、公有成员全部变成派生类的保护成员

保护继承实例

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

```
class Base {
private:   int pri;  //私有成员
protected: int pro;  //保护成员
public:    int pub; //公有成员
           Base(void) {...}
           void show(void) {...}
};
class Derived: protected Base {...};
void main() {...}
```

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

公有、保护、私有继承实例

- 私有继承：基类的保护、公有成员全部变成派生类的私有成员，实际上是终止了基类成员的继续派生！

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

私有继承实例

```
class Base {
private:     int pri;   //私有成员
protected:   int pro;  //保护成员
public:      int pub; //公有成员
             Base(void) {...}
             void show(void) {...}
};
class Derived: Base {...}; //默认为：private
void main() {...}
```

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

公有、保护、私有继承实例

- 在派生类的基础上进一步派生服从同样的规则，即：私有成员派生类不能访问，保护成员派生类可以访问，公有成员外部可以访问。例如：

[在保护继承基础上进行公有派生实例]

[在私有继承基础上进行公有派生实例]

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

派生类的构造和析构函数

- 派生类没有继承基类的构造和析构函数。
- 派生类必须对新增的（普通和对象）成员初始化，还需要初始化基类的成员。
- 派生类的构造函数的定义格式为：

  派生类名(参数表):需初始化的基类名i(参数表i)…{构造函数体}

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

派生类的构造和析构函数

- 派生类构造函数的执行顺序(**熟记**)：

  1、调用基类的构造函数(按继承时说明从左到右的顺序)

  2、(如果有的话)调用子对象的构造函数(按类中说明的顺序)

  3、执行派生类构造函数的函数体

- 执行析构函数的顺序与此正好相反

Evaluation Only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

派生类的构造和析构函数

```
class Derived:public Base2,Base1,protected Base
{//注意:Base1(缺省)为私有继承,而不是公有继承!
   int di;          //普通私有成员
   Myclass obj; //私有(对象)成员
public:
   Derived(int a,int b,int c,int d)
      :obj(b),Base1(c),Base2(d) {
      //自动调用Base的无参构造函数(必须存在)
      di=a; //……
}//Base2-Base1-Base-Myclass-Derived
```

完整实例

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

派生类的拷贝构造函数

- 派生类的拷贝构造函数的一般格式如下：

```
Derived(const Derived &d):Base(d)
{//用派生类的引用去初始化基类的引用合法！
  //正常的派生类的拷贝构造函数体
}//Base:基类，Derived:派生类
```

- 派生类的拷贝构造函数实例

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

基类和派生类指针

- 派生类的成员函数将覆盖基类的所有同名成员函数(含重载的同名函数)。

```
class Base {
  ……
  void show(void) const{…}
  void show(member_type mtype) const {…}
  void sum(void) const {…}
};
class Derived:public Base //派生类，公有继承 {
  ……
  void show(void) const {…}
};
void main() {…}
```

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

基类和派生类指针

- 派生类常先执行基类的同名函数再执行其他功能。

Evaluation only.
Created with Aspose.Slides for Java 22.7
Copyright 2004-2022 Aspose Pty Ltd.

```
class Derived:public Base //派生类，公有继承 {
    ......
    void show(void) const {
        Base::show(pri_mem);
        //必须用Base::限定，否则出错！
        ......
    }
};
```

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

基类和派生类指针

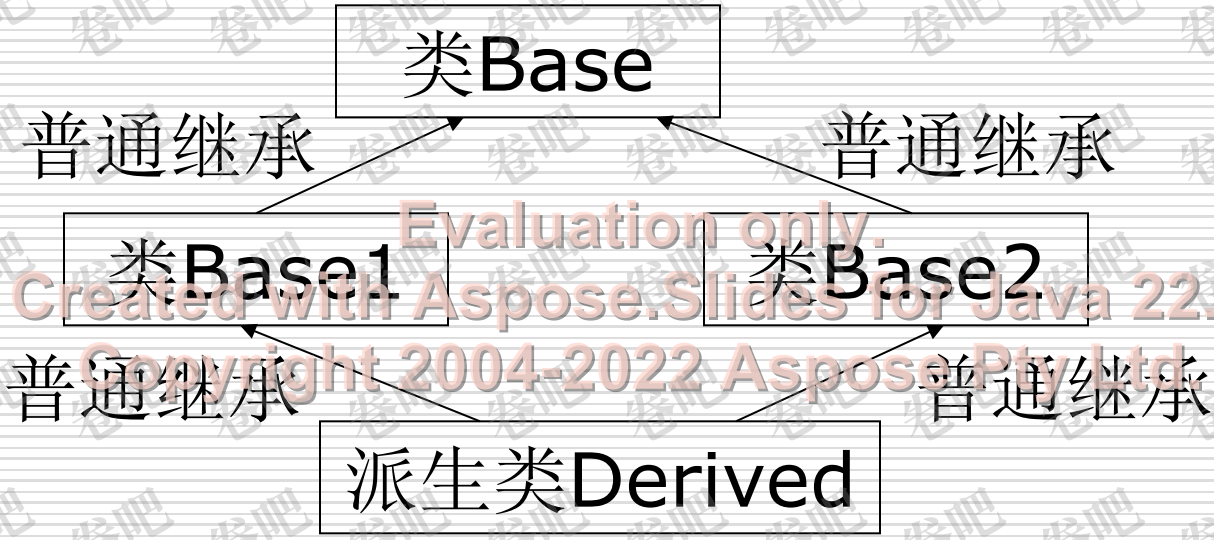
- 不能通过基类对象(指针)访问派生类新增的(公有)成员；基类未被覆盖的函数可以直接调用；被覆盖的函数必须加类名进行限定。
- 可以通过基类和派生类的指针访问基类和派生类。可以通过对象、指针和引用访问基类和/或派生类的成员。

Evaluation Only.
Created with Aspose.Slides for Java 22.7
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

多层(次)继承



Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

- 如果类Base有一保护成员bi，则派生类将有两个类Base的成员bi，分别是：Base1::bi和Base2::bi

完整实例

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

多层(次)继承

- 多层(次)继承不能出现循环！按派生顺序调用构造函数，按相反的顺序调用析构函数。一个类可以在多层继承中多次作为其间接基类。
- 如果基类构造函数需要参数，那么所有派生类都必须向上传递参数，而不管派生类自己是否需要参数。

Evaluation Only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

虚基类机制

- 为避免在派生类中访问的二义性，将直接基类的共同基类设置为虚基类，即在基类的访问方式前加上关键字virtual，声明虚基类的格式如下：

  class 派生类名:virtual 访问方式 基类名
  {//声明派生类成员}

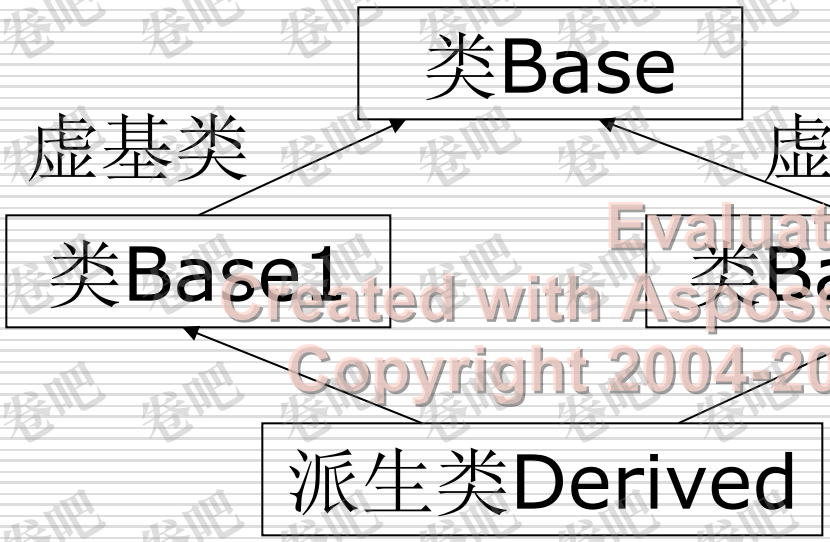
- 以虚方式继承照样继承基类的对象，只是对再次继承它的派生类有影响。

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

虚基类(注意与虚函数的区别)



- 当Base1和Base2有相同成员，可以提到Base，然后设为虚基类。

Evaluation only.
Created with Aspose.Slides for Java 22.7
Copyright 2004-2022 Aspose Pty Ltd.

- 虚基类(在访问方式前加virtual)虽然被一个派生类间接地多次继承，但派生类只继承一份该基类的成员。完整实例

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

虚基类机制下构造函数的执行顺序

- 虚基类机制下构造函数的执行顺序：

  1、直接基类中的虚基类的构造函数在非虚基类之前调用

  2、多个虚基类按继承时说明从左到右的顺序执行

  3、若虚基类由非虚基类派生而来，则仍先调用基类构造函数，再按派生类中的构造函数的执行顺序调用

  4、执行派生类构造函数的函数体

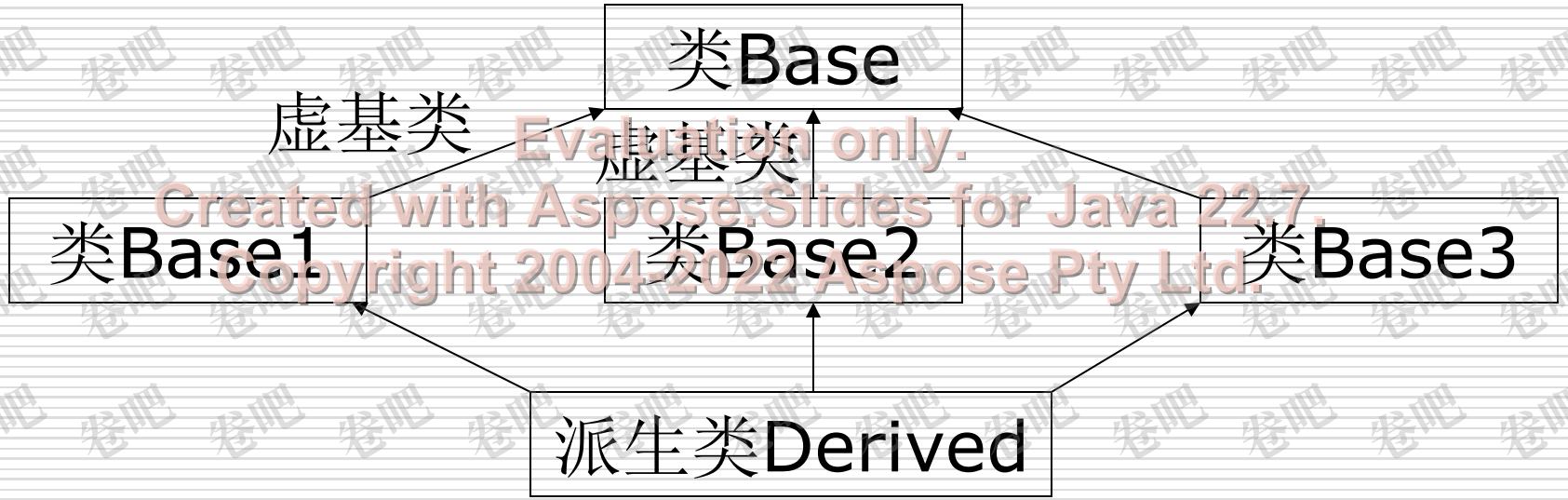
- 执行析构函数的顺序与此正好相反

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

虚基类



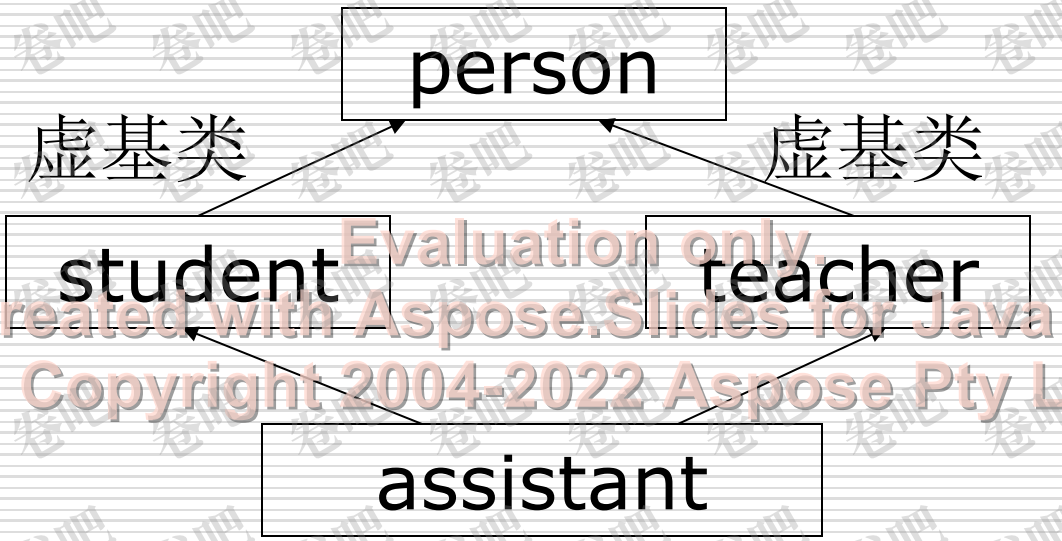
Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

- 必须所有的直接派生类都声明为虚基类，否则仍存在多份间接基类的成员。

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

虚基类及其派生类的构造函数



Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

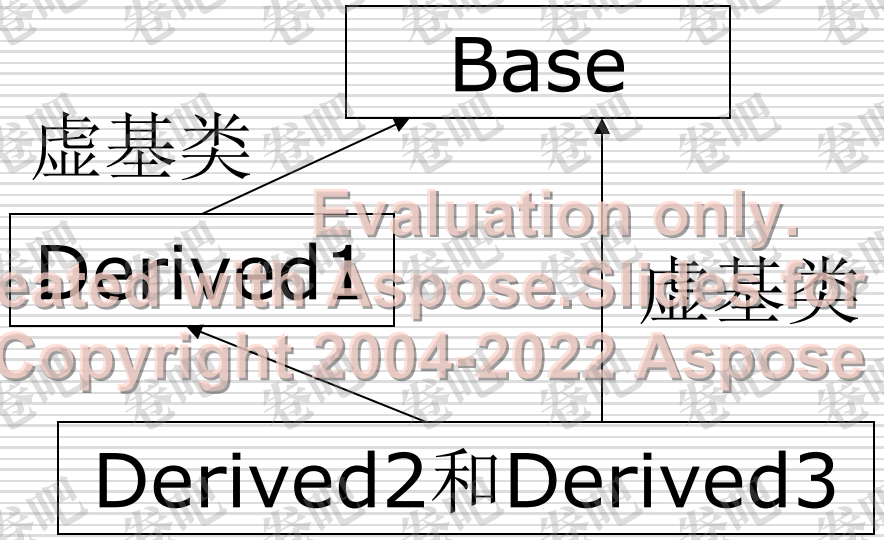
- 对于非虚基类，在派生类的构造函数中初始化间接基类是不允许的，而对于虚基类，则必须在派生类中对虚基类初始化。

虚基类初始化实例

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

虚基类及其派生类的构造函数



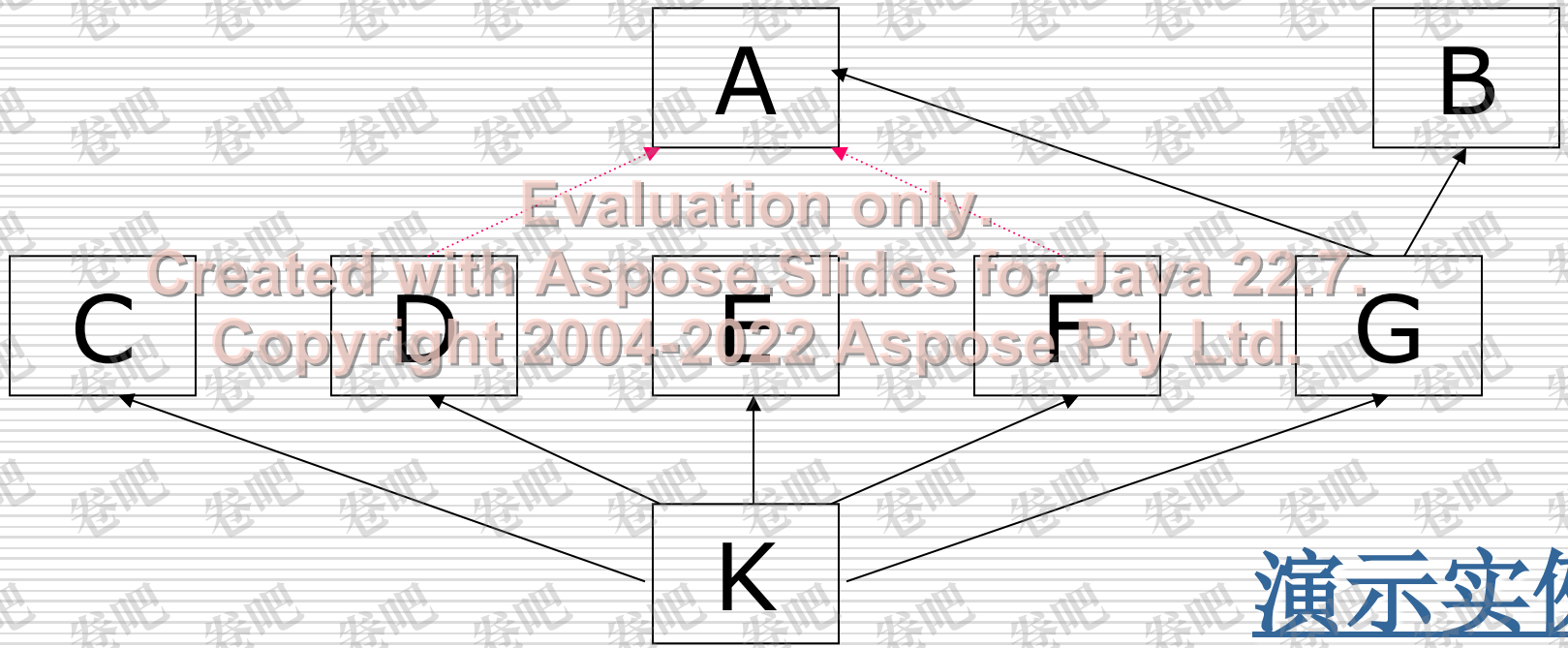
Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

- 类Base有一成员函数show(),Derived2对其覆盖，Derived3没有覆盖该函数。
- 用各自对象调用该成员函数如何？

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

构造函数的执行顺序



Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

演示实例

- 类K中按顺序有三个对象:H、I和J，则构造函数执行顺序为:A-C-D-E-F-A-B-G-H-I-J-K。

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

授权访问

- 在特定情况下可以用访问声明(格式为：基类名::成员;)恢复一个或几个被继承成员的初始访问属性，但标准C++不能使用访问声明来提升或降低某个成员的访问状态(VC++可以改变保护、公有成员的访问状态)。 授权访问实例

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

类指针的转换

```
#include <iostream>
using namespace std;
class Ca { /*.....*/ };
class Cb { /*.....*/ };
void main() {
  Ca CaObj,*pa=&CaObj;
  Cb CbObj,*pb=&CbObj;
  //pa=pb; //不能直接赋值，错误！
  pb=(Cb *)pa; //不安全！何时正确？
}
```

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

类指针的转换

```
#include <iostream>
using namespace std;
class Base {/*…*/};
class Deriver:public Base {/*…*/};
int main() {
    Deriver d;
    Base b=d;&r=d,*p=&d;//全部正确！
}
```

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

基类和派生类指针

- 一个类(结构)对象指针指向的是此对象存储区的起始地址，通过强制类型转换可以将指向某一类(结构)的指针转换成指向另一类(结构)的指针。一般说来这种转换是不安全的。
- 两个成员相同，顺序相同的结构指针的强制转换实例(不能直接赋值)
- 两个成员相同，但顺序不同的结构指针的强制转换实例(不能直接赋值)

Evaluation only.
Created with Aspose.Slides for Java 22.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

基类和派生类指针

- 一种类型的指针不能指向另一种类型的对象，但派生类例外。在C++中规定基类指针可以指向(公有继承的)派生类，不过只能访问基类的成员(此时有基类和派生类同名的成员不必加类名限定自动访问基类的成员)。但不能使用派生类的指针访问基类的对象(不安全)。

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

基类和派生类指针

- 在C++中，若为公有派生，把派生类的地址赋值给基类指针是安全的，不需要强制类型转换。原因是C++的继承采用了复制继承的方式，派生类中包含了原基类中的全部成员并且结构和顺序都与原基类相同。派生类的对象在内存中的布局形如：

  基类中的所有成员+派生类中新增的成员

  因基类指针本身的类型并没有改变，所以通过其访问的是派生类对象成员的基类部分。

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

基类和派生类指针

- 实际上：不仅可以将派生类对象的地址赋值给基类指针，甚至允许把派生类对象赋值给基类对象，或者用派生类对象初始化基类的引用。
- 基类指针的这个特点(即类型兼容原则)是C++实现多态性的重要基础。
- 如果是私有或保护派生，把派生类的地址赋值给基类指针是非法的！

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

基类和派生类指针

```
class Base { /*......*/ };
class Derived:public Base {/*...*/};
void main() {
   Base bObj,*pb=&bObj;
   Derived dObj,*pd=&dObj;
   pb=&dObj;
   //基类指针可以指向(公有继承的)派生类
   pd=(Derived *)&bObj;
   //将基类地址赋值给派生类指针需要使用
   //强制类型转换，但这种转换是不安全的。
}
```

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

基类和派生类指针

- 如果想通过基类型指针来访问在派生类中定义的成员，必须先把该基类指针转换为派生类型的指针，例如：

Evaluation Only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

```
((Derived *)bp)->showDerived();
```

在类型转换运算外部的圆括号是必须的。尽管这种指针类型的转换在技术没有任何错误，但最好避免这样做！

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

基类和派生类指针

- 相反，如果要将基类地址赋值给派生类指针，需要使用强制类型转换，但这种转换是不安全的。因为这个指针可能用来访问并未实际定义的成员(基类并不总有所有派生类中定义的成员)。**此时必须由程序员自己保证程序的正确性(安全性)。**
- 基类和派生类指针使用实例

Evaluation Only.
Created with Aspose.Slides for Java 22.7
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

虚基类和派生类指针

- 由于虚基类只有一个拷贝，所以一个派生类的地址可以直接赋值给虚基类的指针，并且不需要进行强制类型转换。
- 相反，将虚基类的指针赋值给派生类的指针，即使进行强制转换还是会错误！
- 虚基类和派生类指针使用实例

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

综合应用实例

- 先建一个“交通工具”类，然后派生出一个“卡车”类
- 先建一个“点(Point)”类，然后派生出一个“圆(Circle)”类和“矩形(Rectangle)” 类等(但不合理)

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

总结

- 类可以将数据和函数封装在一起，但不可以滥用类的封装功能，不要把类当成火锅，什么东西都往里扔（封装）。
- 类的设计是以数据为中心，还是以行为为中心？

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

总结

- C++的“继承”特性可以提高程序的可复用性。正因为“继承”太有用、太容易用，才要防止乱用“继承”。
- 如果类A和类B毫不相关，不可以为了使B的功能更多些而让B继承A的功能（虽然技术上可以！）。

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 灰灿工作室 All Rights Reserved.

第6章：继承和派生

总结

- 若在逻辑上B是A的“一种”（a kind of），则允许B继承A的功能。如男人（Man）是人（Human）的一种，男孩（Boy）是男人的一种。那么类Man可以从类Human派生，类Boy可以从类Man派生。

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

总结

- 若在逻辑上A是B的“一部分”（a part of），则不允许B继承A的功能，而是要用A和其它东西组合出B。例如眼（Eye）、鼻（Nose）、口（Mouth）、耳（Ear）是头（Head）的一部分，所以类Head应该由类Eye、Nose、Mouth、Ear组合而成，不是派生而成。

Evaluation Only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

第6章：继承和派生

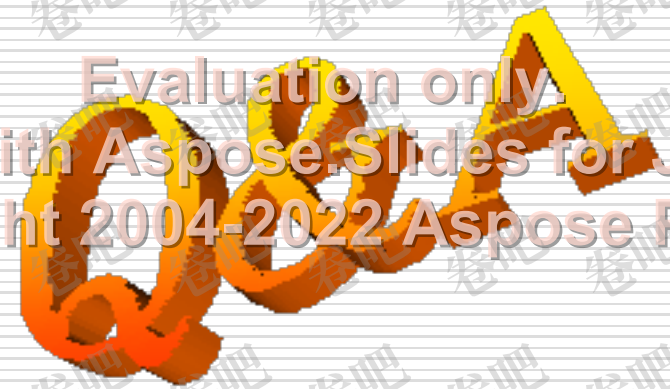
总结

```
class Head {
public:
  void Look(void){m_eye.Look();}
  //……
private:
  Eye    m_eye;
  Nose   m_nose;
  Mouth m_mouth;
  Ear    m_ear;
  //……
};
```

Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

Copyright © 2005-2016 王灿辉. All Rights Reserved.

本章内容讲授到此结束！



Evaluation only.
Created with Aspose.Slides for Java 22.7.
Copyright 2004-2022 Aspose Pty Ltd.

福州大学·软件学院·软件工程系
王灿辉(wangcanhui@fzu.edu.cn)

Copyright © 2005-2016 王灿辉. All Rights Reserved.