计算机科学技术学院
School of Computer Science and Technology

Java 程序设计

第 6 章 接口与实现

导读

- **主要内容**
  - 接口、实现接口、理解接口
  - 接口回调、接口与多态
  - 接口变量做参数、面向接口编程
- **重点和难点**
  - 重点：接口的理解；抽象类和接口的区别
  - 难点：抽象类和接口的异同点

接口概述

Java 不支持多重继承

```
class People {
    private int averHeight = 166;
}
class ChinaPeople extends People {
    int height;
    public int getHeight() {
       return height;
    }
}
```

6.1 接口

- 使用关键字 interface 来定义一个接口。接口的定义和类的定义很相似，分为接口的声明和接口体

```
interface  Printable {
    final  int MAX=100;
    void add();
    float sum(float x, float y);
}
```

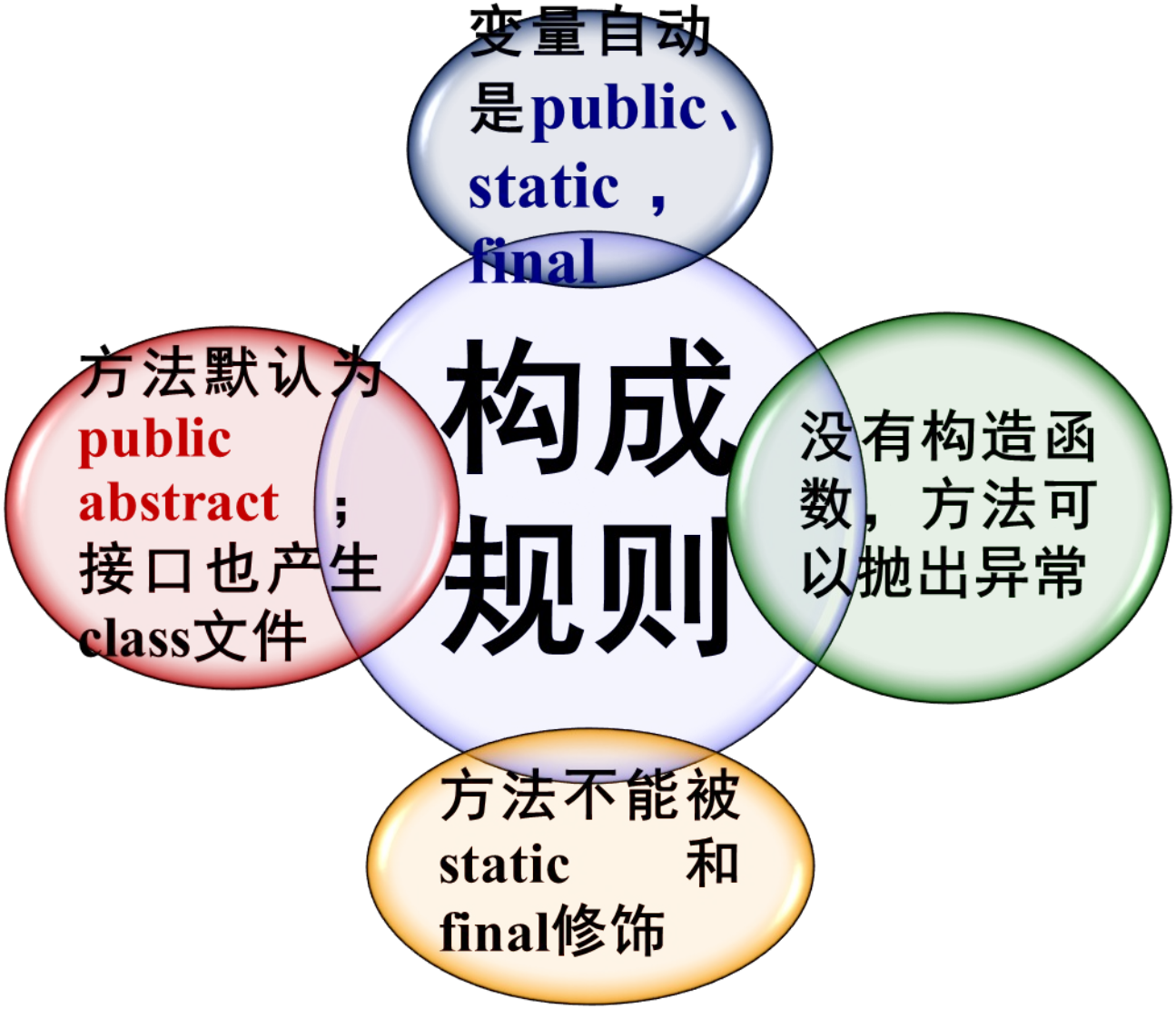
① 接口声

- 接口通
- 格式：

② 接口体

- 接口体 ……进行**方法的声明**，不许提供方法的实现，所以，方法的定义没有方法体，且用分号“；”结尾。如：

接口的变量和方法的构成规则



6.2 实现接口

- 类通过关键字 implements 实现一个或多个接口

  如： class A implements Printable, Addable

- 一个类实现了某个接口，必须重写该接口所有方法
- 接口可以通过继承产生新的接口
- Java 提供的接口都在相应的包中，通过 import 语句不仅可以引入包中的类，也可以引入包中的接口，

  例如： import java.io.*;

接口例题

Chinese 类、 English 类和 SayHello 接口，而且 Chinese 和 English 类都实现了 SayHello 接口

6.3 接口的 UML 图

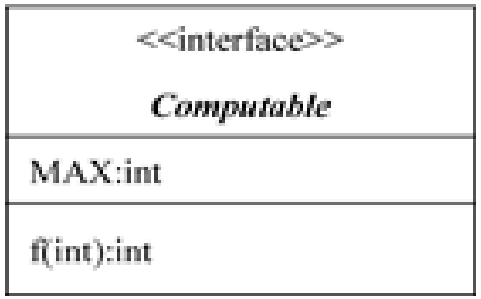


图 6.2 接口 UML 图

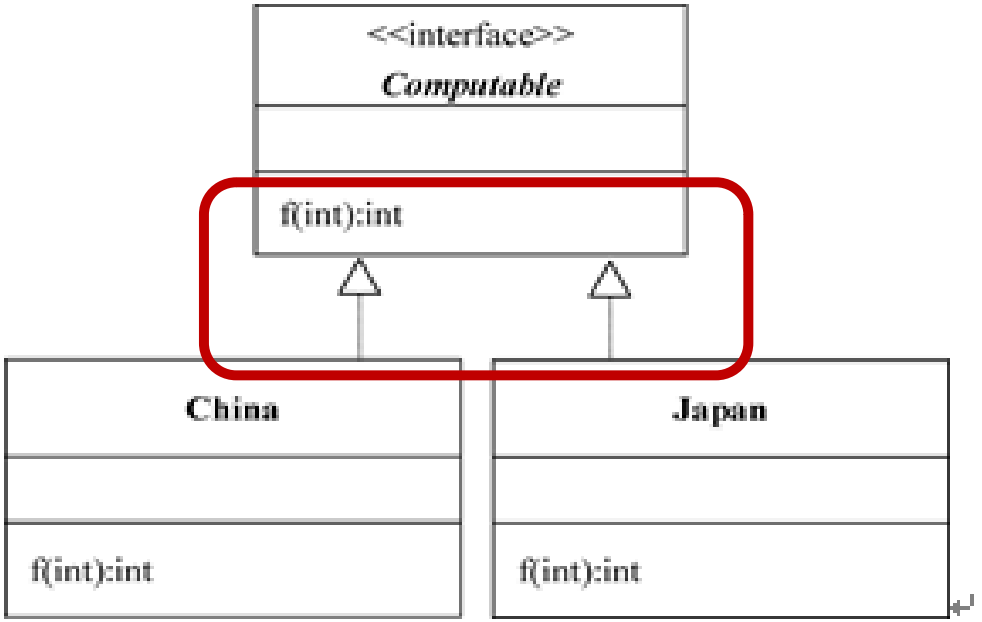


图 6.3 实现关系的 UML 图

6.4 接口回调

- **接口回调**：可以把实现某一接口的类创建的对象的引用赋给该接口声明的接口变量中，那么该接口变量就可以调用被类重写的接口方法。实际上，当接口变量调用被类重写的接口方法时，就是通知相应的对象调用这个方法

6.4 接口回调

```
Com com;   // 声明接口对象
ImpleCom obj = new ImpleCom(); // 实现接口子类对象
com = obj; // 接口回调
```

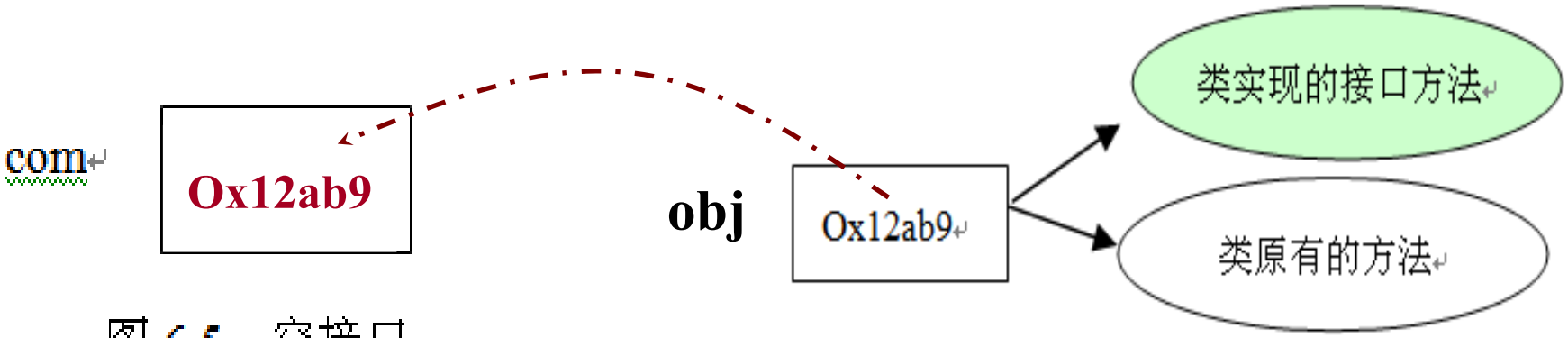


图 6.5 空接口

图 6.6 对象调用方法的内存模型

6.5 理解接口

理解的关键点是：

（1）接口可以抽象出重要的行为标准，该行为标准用抽象方法来表示

（2）可以把实现接口的类的对象的引用赋值给接口变量，该接口变量可以调用被该类实现的接口方法，即体现该类根据接口里的行为标准给出的具体行为

6.5 理解接口

例3中，要求 MotorVehicles 类（机动车）的子类 Taxi（出租车）和 Bus （公共汽车）必须有名称为 brake 的方法（有刹车功能），但额外要求 Taxi 类有名字为 controlAirTemperature 和 charge 的方法（有空调和收费功能），即要求 Taxi 实现两个接口，要求客车类有名字为 charge 的方法（有收费功能），即要求 Bus 只实现一个接口

6.6 接口与多态

- 可以通过在接口中声明若干个 abstract 方法，表明这些方法的重要性，方法体的内容细节由实现接口的类去完成
- 使用接口进行程序设计的核心思想是使用接口回调，即接口变量存放实现该接口的类的对象的引用，从而接口变量就可以回调类实现的接口方法

6.7 接口参数

- 如果一个方法的**参数是接口类型**，我们就可以将任何实现该接口的类的实例的**引用传递**给该接口参数，那么接口参数就可以**回调**类实现的接口方法
- 例5

```
中国人习惯问候语：你好,吃饭了吗?
英国人习惯问候语:你好,天气不错
```

图 6.10 接口与参数

6.8 abstract 类与接口的比较

- 接口和 abstract 类的比较如下：

①abstract 类和接口都可以有 abstract 方法

②接口中只可以有常量，不能有变量；而 abstract 类中即可以有常量也可以有变量

③abstract 类中可以有非 abstract 方法，接口不可以

6.9 面向接口编程

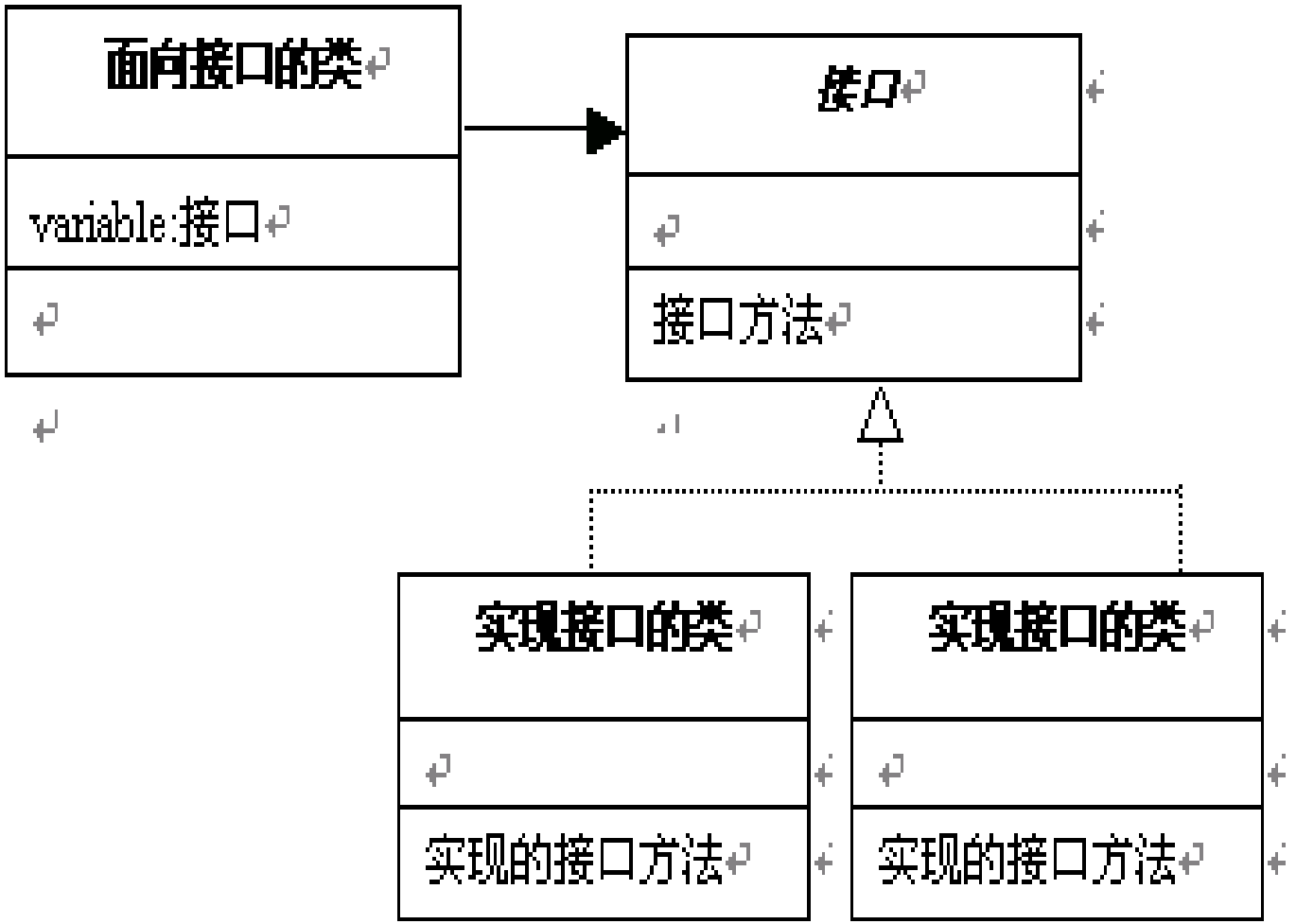


图 6.11 UML 类图

接口和抽象类的比较

- 当准备为组件提供多态性时，以下建议将有助于在抽象类和接口之间做出正确的选择

目标	选择
创建多个组件版本	抽象类
设计小而简练的功能块	接口
设计大型功能单元	抽象类

6.10 应用举例

- 设计一个广告牌，希望所设计的广告牌可以展示许多公司的广告词

1．问题的分析
2．设计接口
3．设计广告牌类

```
黑土集团的广告词如下:
***************
劳动是爹
土地是妈
***************
白云有限公司的广告词如下:
@@@@@@@@@@@@@@@@@@@@@@@@@@
飞机中的战斗机，哎yes!
@@@@@@@@@@@@@@@@@@@@@@@@@@
```

图 6.12 体现“开-闭”原则

总结

- 接口的接口体中只可以有常量和 abstract 方法
- 接口也是引用型数据类型
- 当接口变量中存放了实现接口的类的对象的引用后，接口变量就可以调用类实现的接口方法，该过程称为接口回调
- 和子类体现多态类似，接口产生的多态就是指不同的类在实现同一个接口时可能具有不同的实现方式

接口 - 扩展

- 接口的定义语法：

```
[public] interface 接口名称 [extends 父接口名列表] {
    ......
    [public][static][final] 数据类型 成员变量名 = 常量;
    [public][abstract] 返回值的数据类型 方法名(参数表);
    [public] static 返回值的数据类型 方法名(参数表) {
                方法体
    }
    [public] default 返回值的数据类型 方法名(参数表) {
                方法体
    }
}
```

接口 - 扩展

接口的实现与引用

- [public][static][final] 数据类型　成员变量名 = 常量;

```
interface Person
{
    int age;
    public int age1 = 25;
}

class ClassTest implements Person
{   }
```

接口 - 扩展

接口的实现与引用

- [public][abstract] 返回值的数据类型 方法名 (参数表) ;

```
interface Person  {
    int age;
    public int age1 = 25;
    int add(int a, int b);
}

class ClassTest implements Person
{   }
```

接口 - 扩展

接口的实现与引用

- [public] static 返回值的数据类型 方法名 (参数表) {
  方法体 }

```
interface Person  {
    int age;
    public int age1 = 25;
    static int sub(int a, int b)  {
        return a -b;
    }
}

class ClassTest implements Person
{   }
```

ClassTest 也包含此方法呢？

接口 - 扩展

接口的实现与引用

- [public] default 返回值的数据类型 方法名 (参数表) {
  方法体 }

```
interface Person  {
    int age;
    public int age1 = 25;
    default int mul(int a, int b)  {
        return a*b;
    }
}

class ClassTest implements Person
{   }
```

ClassTest 也包含此方法呢？

接口 - 扩展

接口的实现与引用

```
interface InterA
{
    static String name = " 张三 ";
    static int age = 20;
}

interface InterB
{
    static String name = " 李四 ";
    static float salary = 8000f;
}
```

```
class TestClass implements InterA, InterB
{

}
```

接口 - 扩展

接口的实现与引用 —— default 方法

```
interface InterA
{
    static String name = " 张三 ";
    static int age = 20;
    default int add(int a, int b) {return a + b + 1;}
}

interface InterB
{
    static String name = " 李四 ";
    static float salary = 8000f;
    default int add(int a, int b) {return a + b + 2;}
}
```

```
class TestClass implements
InterA, InterB
{
}
```

接口 - 扩展

接口的实现与引用 —— abstract 方法

```
interface InterA
{
    static String name = " 张三 ";
    static int age = 20;
    int add(int a, int b);
}

interface InterB
{
    static String name = " 李四 ";
    static float salary = 8000f;
    int add(int a, int b);
}
```

```
class TestClass implements
InterA, InterB
{
}
```

接口 - 扩展

接口的实现与引用

```
interface InterA
{
    static String name = " 张三 ";
}

class ClassC
{
    static String name = " 李四 ";
}
```

```
class TestClass extends ClassC implements InterA {
}
```

接口 - 扩展

接口的实现与引用

```
interface InterA
{
    default int add(int a, int b) {return a + b + 1;}
}

class ClassC
{
    default int add(int a, int b) {return a + b + 2;}
}
```

```
class TestClass extends ClassC implements InterA {
}
```

作业

- 习题 6 ：　　2,3,4