Java编程思想_面向对象之继承

课程概要

学习目标

能够理解继承的概念和好处

能够说出继承的优缺点和使用场景

能够定义类的继承关系并正确使用

理解方法重写的概念，并能够重写父类方法

能够说出方法重写和方法重载的区别

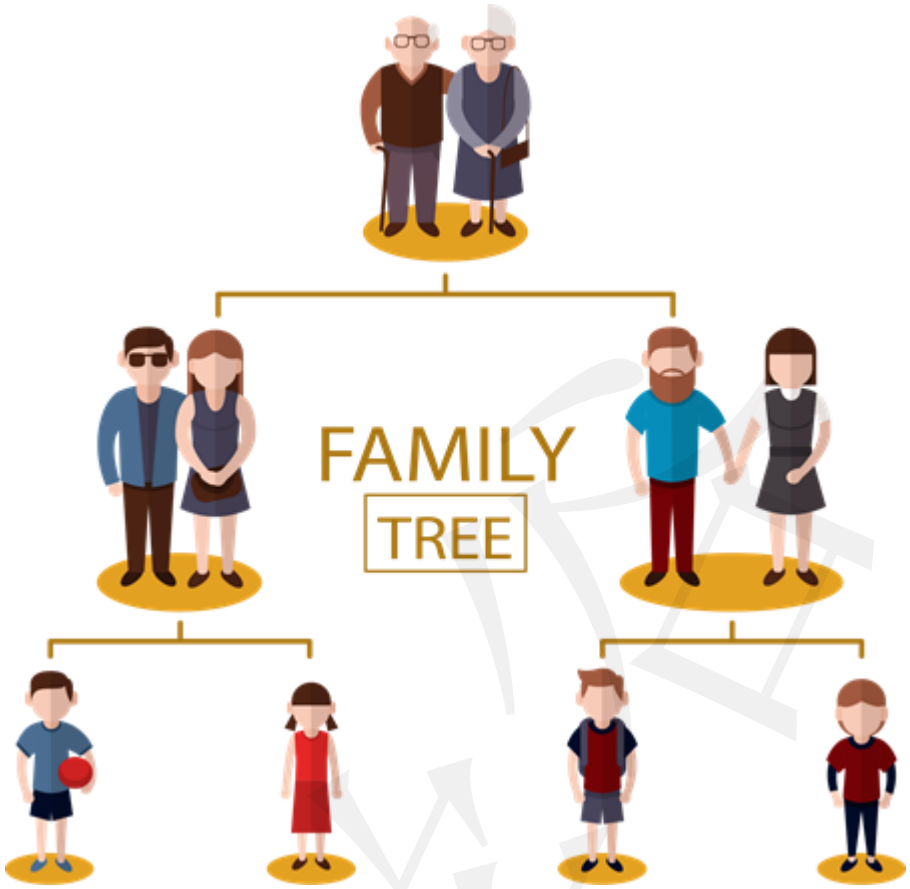
能够正确使用Java的四种访问权限修饰符

能够说出Java中类的继承的特点

继承概述

继承的概念

泛指把前人的作风、文化、知识、财产等接受过来



Java中的继承

通过扩展一个类来建立另外一个类的过程，叫做继承（inheritance）

通俗地说，所谓继承，就是让类与类之间产生父子关系。

所有的类都直接或间接的继承自：`java.lang.Object` 被继承的类叫做**父类**（基类、超类） 继承的类叫做**子类**（派生类）

格式（extends）

```
class 父类 {
    // ...
}
class 子类 extends 父类 {
    // ...
}
```

子类继承父类之后有什么效果?

子类拥有了父类的**非私有**成员（成员变量、成员方法）

```java
public class Child extends Parent {

}
```

```java
public class Test {
    public static void main(String[] args) {
        Child c = new Child();
        c.setName("小黑"); // 调用父类方法
        // c.age; // 报错：找不到该属性
        System.out.println(c.getName()); // 输出：小黑
    }
}
```

```java
public class Parent {
    public Parent() {
    }
    private String name;

    private int age;

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
}
```

父亲的身体是属于他自己的，而他的房子、车子等财产却是可以通过继承得到的， 所以，在Java中，父类的**私有成员**不能被继承。

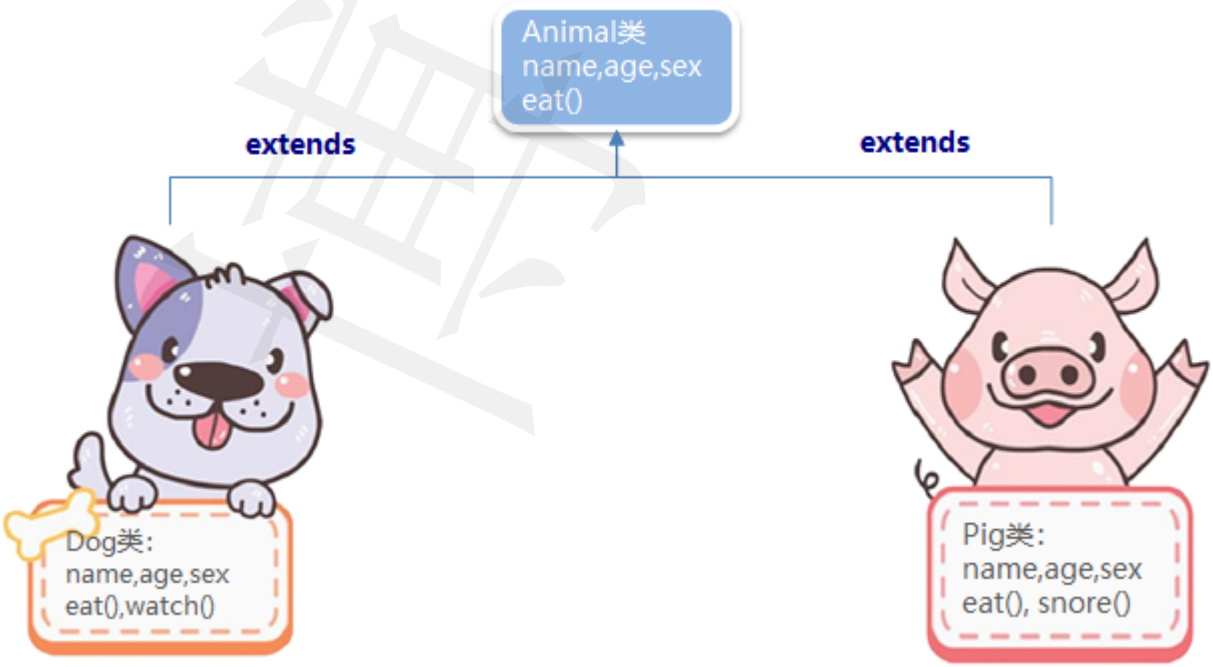
继承的使用场景

子类通过继承，拥有了父类的非私有成员，这是开发中常见的做法，

那么，继承还有哪些使用场景呢?

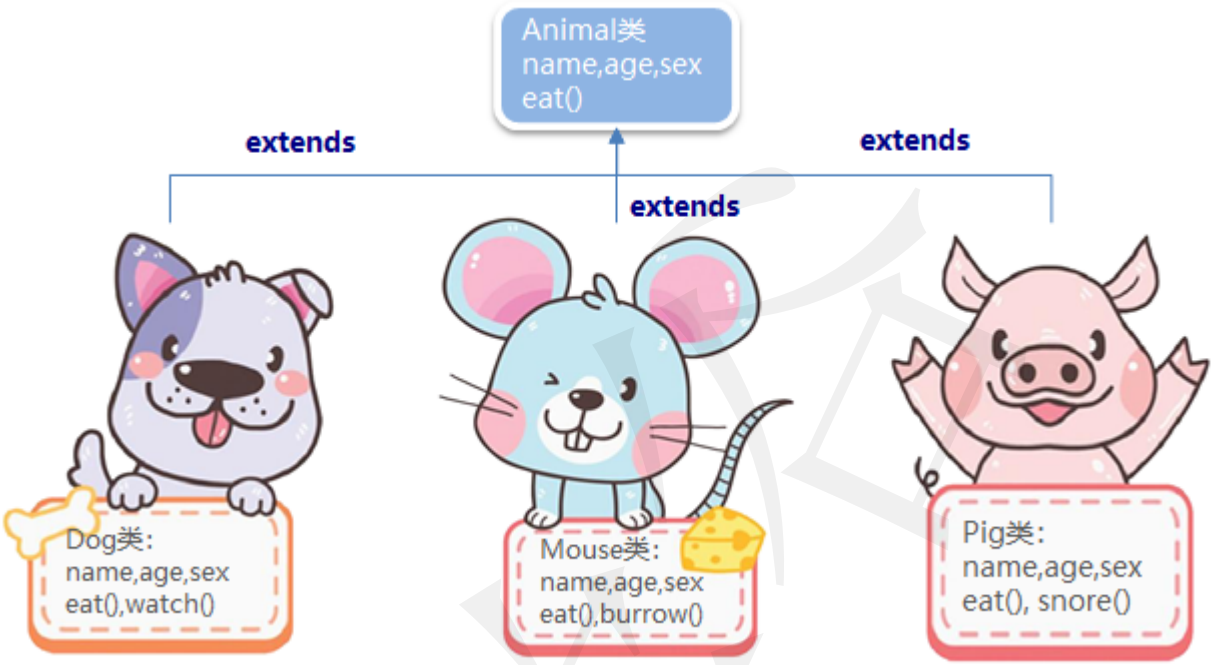
向上抽取：

多个类中存在相同的属性和行为时，可以将这些内容提取出来放到一个新类中，让这些类和新类产生父子关系，实现代码复用。



向下扩展：

当需要扩展已有的类的功能时，可以通过继承已有的类，在子类中添加新功能或重新实现已有功能，对父类（已有的类）没有影响。



案例：定义继承关系的动物类并使用

需求：

分别定义Dog类、Mouse类、Pig类，它们共有的属性有：name、age、sex，共有的行为有：eat()，Dog类和Mouse类特有的属性为coatColor（毛色），三者特有的行为分别是：watch()，burrow()，snore()

分析：

A：定义Dog类，属性和行为：

name、age、sex、coatColor ;eat(),watch()

B：定义Pig类，属性和行为：

name、age、sex; eat(),snore()

C：定义测试类，分别创建两种动物的对象并使用

D：抽取Dog类和Pig类共性内容，定义到类Animal中：

name, age, sex, eat()

E：让Dog类和Pig类继承Animal类，删掉重复内容

F：定义Mouse类，继承Animal类，特有的属性和行为：

burrow()

G：在测试类中创建Mouse的对象并使用

效果：

```
"C:\Program Files\Java\jdk-11.0.1\bin\java.exe" ...
黑灰色哈士奇在拆家
橙黄色Jerry会打洞
小猪佩琪在打鼾

Process finished with exit code 0
```

继承的优缺点

◆ 功能复用	直接将已有的属性和行为继承过来，实现了功能的复用，节省了大量的工作
◆ 便于扩展新功能	在已有功能的基础上，更容易建立、扩充新功能
◆ 结构清晰，简化认识	同属于一个继承体系的相关类，他们之间结构层次清晰，简化了人们对代码结构的认识
◆ 易维护性	不同类之间的继承关系，让这些事物之间保持一定程度的一致性，大大降低了维护成本
◆ 打破了封装性	父类向子类暴露了实现细节，打破了父类对象的封装性
◆ 高耦合性	类与类之间紧密的结合在一起，相互依赖性高

程序设计的追求

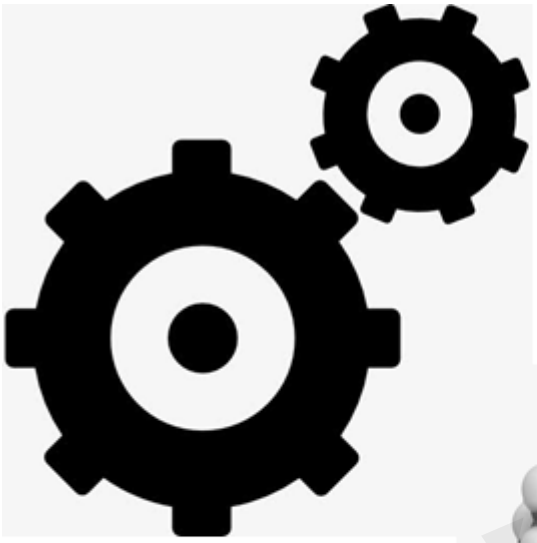
低耦合，高内聚

耦合

两个（或更多）模块相互依赖于对方

内聚

模块内部结构紧密，独立性强



齿轮，两个紧密的齿轮分离开之后，整个系统可能无法运转；

氨基酸分子：氨基酸是蛋白质的主要构成，氨基酸从蛋白质分离之后，还可以自由的和其它氨基酸再组成新的蛋白质，因为它内部结构完整；

类似的例子还有：U盘：随意拔插，因为其内部结构完整，不依赖任何一台特定的设备

继承关系中类成员的使用

继承关系中子父类成员变量的使用

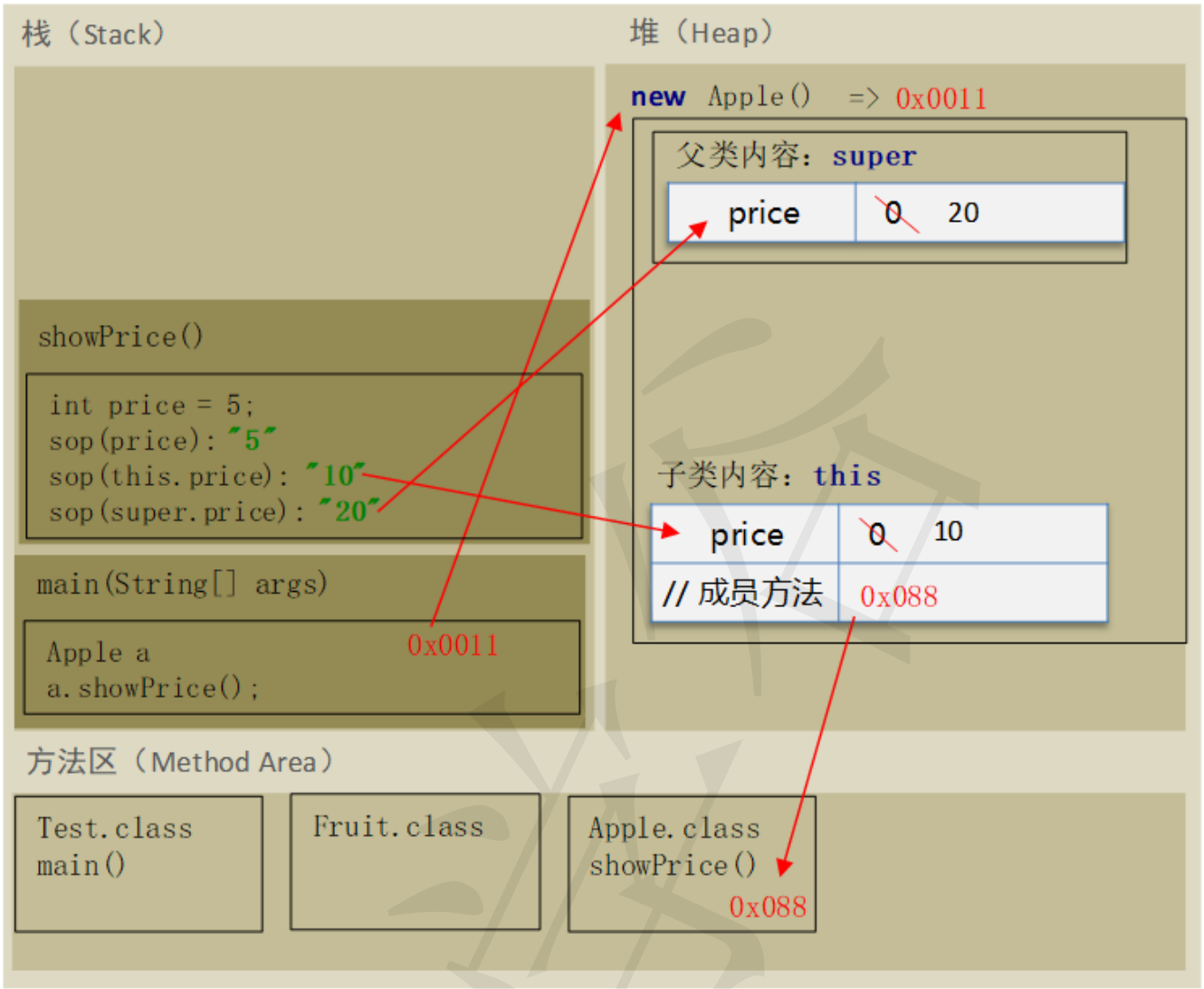
需求：子父类中定义了同名的成员变量，如何使用?

分析：

A：子类中定义成员变量int price，默认值为10 B：父类中定义成员变量int price，默认值为20 C：子类成员方法中定义局部变量int price，值为5 D：在子类成员方法中分别输出三个price的值

图解

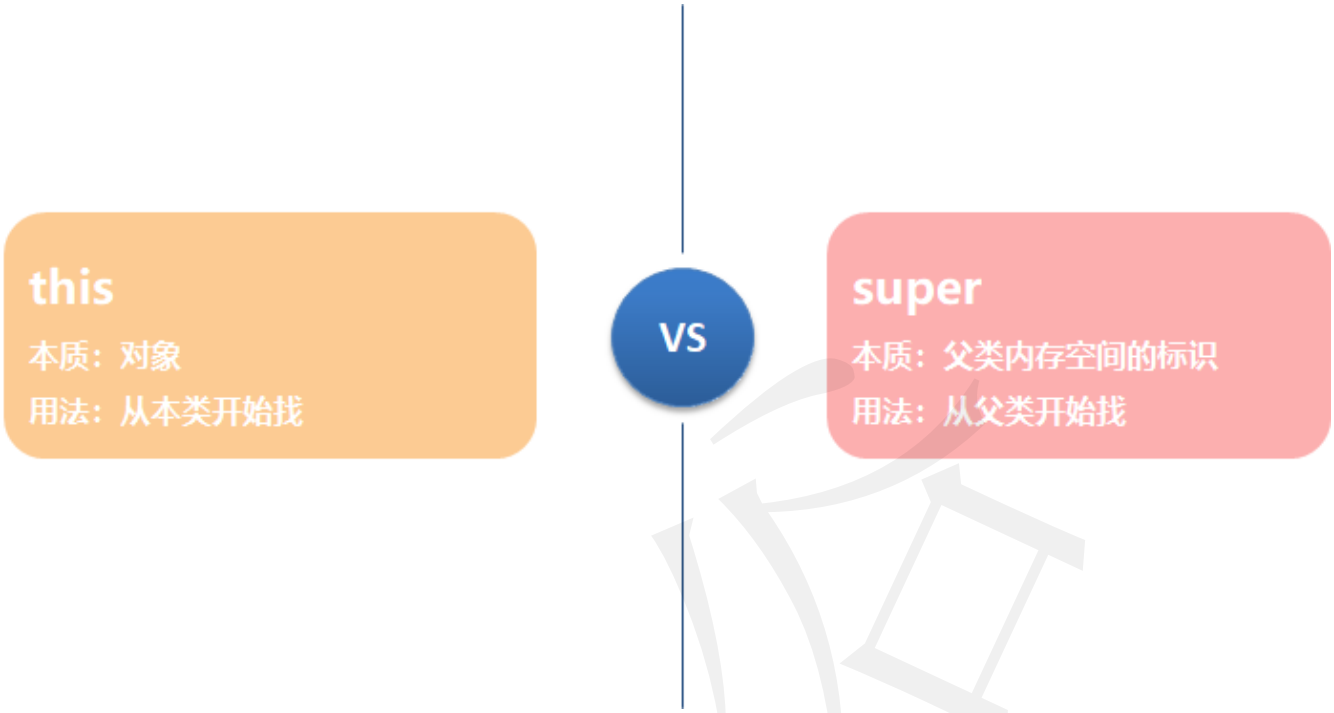
```
public class Test {
    public static void main(String[] args) {
        Apple a = new Apple();
        a.showPrice();
    }
}
public class Fruit {
    int price = 20;
}
public class Apple extends Fruit {
    int price = 10;
    public void showPrice() {
        int price = 5;
        System.out.println(price);
        System.out.println(this.price);
        System.out.println(super.price);
    }
}
```



结论

1. 查找变量的原则： 就近原则
2. 查找变量的顺序： 局部变量□成员变量□父类□更高的父类...Object
3. 访问父类变量的方式： super.父类变量名;
4. super： 当前对象父类的引用（父类内存空间的标识）
5. 对象初始化顺序： 先初始化父类内容，再初始化子类内容

this和super的区别

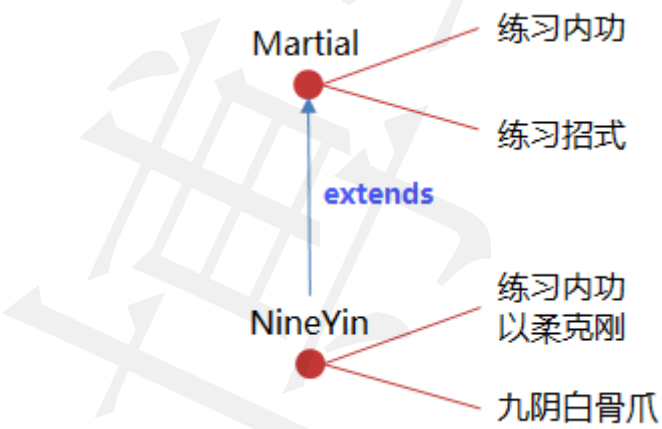


继承关系中子父类成员方法的使用

需求：子父类中定义了同名的成员方法，如何使用?

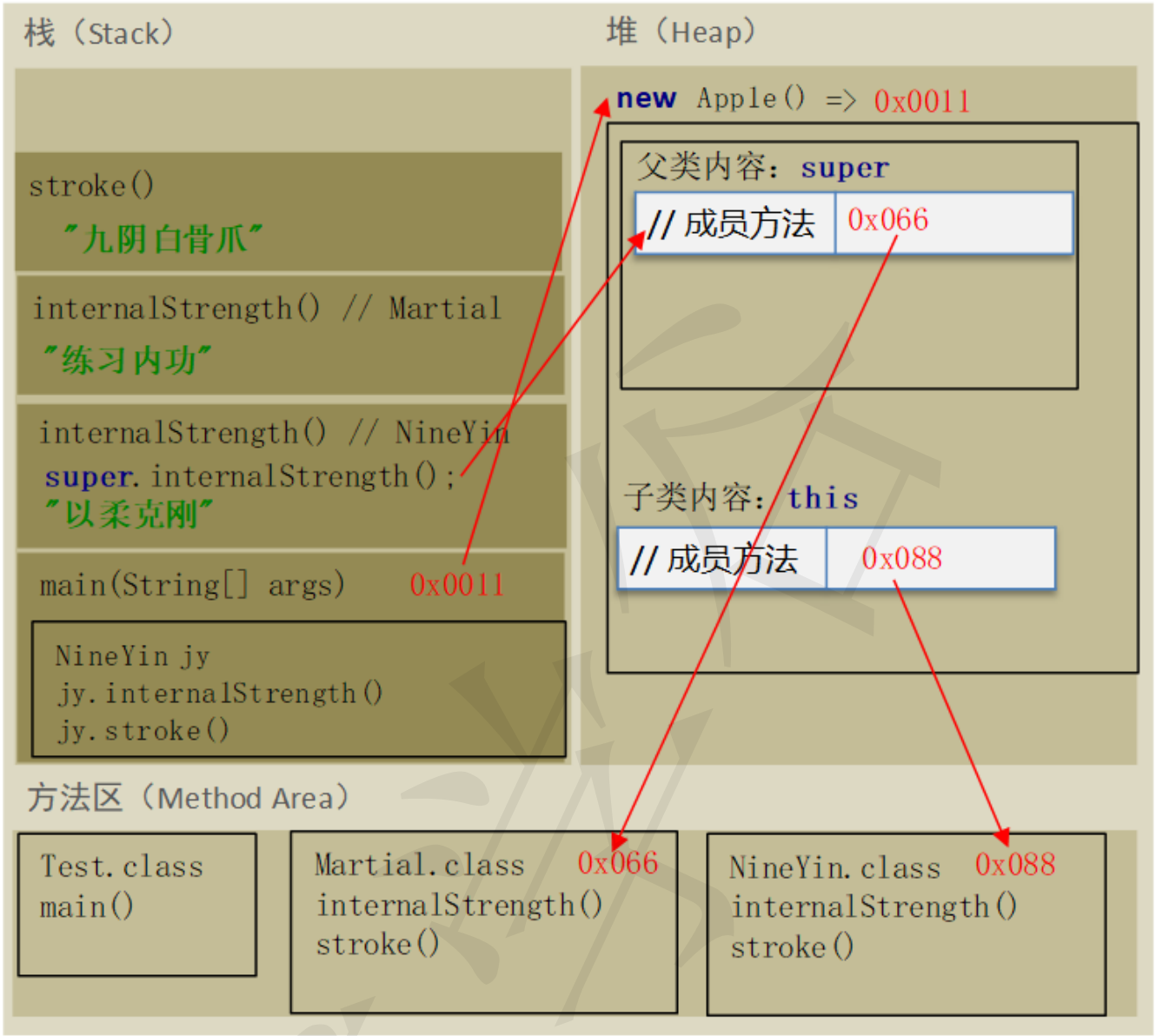
分析：

A：天下武功，无非是内功和招式。 定义武功类Martial ，定义练习内功和招式的成员方法：internalStrength(), stroke() B：九阴真经，讲究以柔克刚，绝招是九阴白骨爪。 定义九阴真经类NineYin，继承Martial类 C：九阴真经的修炼，不仅要练习基本内功，还要能够以柔克刚，需要扩展父类方法；简单的招式已经不足为用，必须有九阴白骨爪这样的大招才能制胜，需要重新实现父类方法 D：定义测试类，创建NineYin对象并使用



图解

```
public class Test {
    public static void main(String[] args) {
        NineYin ny = new NineYin();
        ny.internalStrength();
        ny.stroke();
    }
}
public class NineYin extends Martial {
    public void internalStrength() {
        super.internalStrength(); // 不仅要练习基本的内功
        System.out.println("以柔克刚");// 还要能够以柔克刚
    }
    public void stroke() {
        // 简单的招式已经不足为用
        // 必须有九阴白骨爪这样的大招，才能制胜
        System.out.println("九阴白骨爪");
    }
}
public class Martial {
    public void internalStrength() {
        System.out.println("练习内功");
    }
    public void stroke() {
        System.out.println("练习招式");
    }
}
```



结论

1. 查找方法的原则： 就近原则
2. 查找方法的顺序： 本类□父类□更高的父类...Object
3. 访问父类方法的方式： super.父类方法名();
4. 定义重名方法的前提： 父类功不能完全满足现实需求，扩展父类功能 父类功能已过时，重新实现父类功能

继承关系中子父类构造方法的使用

需求：创建对象时，构造方法是如何被调用的?

分析:

A：定义父类Person，在默认无参构造中输出语句 B：定义子类Worker，继承Person，在默认无参构造中输出语句
C：定义测试类，创建子类Worker对象

结论:

1. 创建子类对象时，优先调用父类构造方法

2. 子类构造方法的第一行，隐含语句super()，用于调用父的类默认无参构造

```
public class Person {
    public Person() {
        System.out.println("person");
    }
}
public class Worker extends Person {
    public Worker() {
        super(); // 调用父类默认无参构造
        System.out.println("worker");
    }
}
```

```
"C:\Program Files\Java\jdk-11.0.1\bin\java.exe" ...
person
worker

Process finished with exit code 0
```

需求：父类不存在默认无参构造方法怎么办?

分析:

子类创建对象时，必须先初始化该对象的父类内容，若父类中不存在默认无参构造，须手动调用父类其它构造。

```
public class Person {
    public Person(String name) {
        System.out.println("person" + name);
    }
}
public class Worker extends Person {
    // 方法声明报错，提示父类没有默认无参构造
    public Worker() {
        super("小黑");
        System.out.println("worker");
    }
}
```

方法重写

方法重写（Override）

```java
public class Martial {
    public void internalStrength() {
        System.out.println("练习内功");
    }
    public void stroke() {
        System.out.println("练习招式");
    }
}
```

```java
public class NineYin extends Martial {
    public void internalStrength() {
        super.internalStrength();
        System.out.println("以柔克刚");
    }
    public void stroke() {
        System.out.println("九阴白骨爪");
    }
}
```

定义：

子类中出现和父类方法定义相同的方法的现象

解释：

方法重写也叫方法的复写、覆盖 方法名、参数列表、返回值类型都相同

注意事项：

父类私有方法无法重写 子类方法访问权限不能小于父类方法 子类不能比父类方法抛出更大的异常（了解）

使用场景：

扩展父类功能 父类功能过时，重新实现父类功能

Java中的访问权限修饰符

	本类	本包	子类	其它类
private	✔			
默认	✔	✔		
protected	✔	✔	✔	
public	✔	✔	✔	✔

案例：Java中四种访问权限演示

```
public class Person {
        public void showPublic() {
            System.out.println("showPublic");
        }

        private void showPrivate() {
            System.out.println("showPrivate");
        }

        void showDefault() {
            System.out.println("showDefault");
        }

        protected void showProtected() {
            System.out.println("showProtected");
        }
}
```

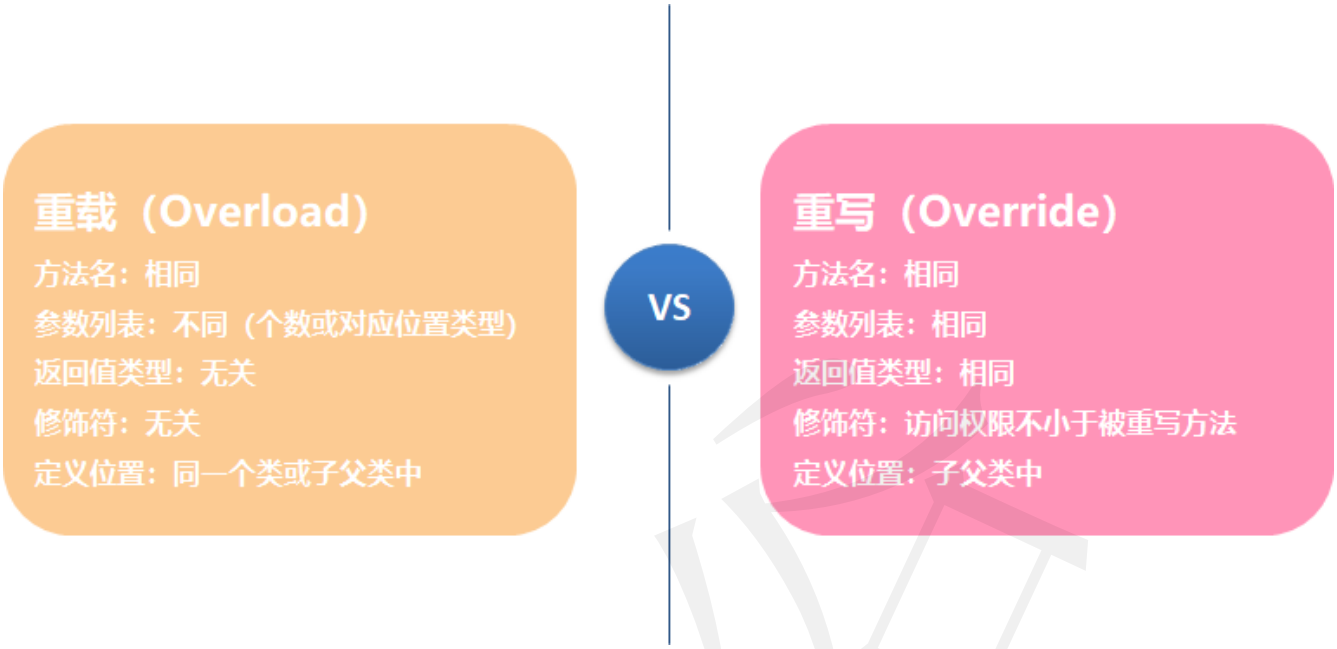
需求：

在不同的包、子父类中定义成员并在测试类中尝试访问

分析：

A. 在父类Person中分别定义四种权限修饰的方法 B. 在本包下创建测试类，在main方法中测试四个方法 C. 在其它包下创建类Student，并测试四个方法 D. 使Student类继承Person类，并测试四个方法

方法重写和方法重载的区别



Java中继承的特点

单继承

Java只支持类的单继承，但是支持多层（重）继承

Java支持接口的多继承，语法为： 接口A extends 接口B,接口C,接口D...

```
public class Fruit { // 水果类
}

public class Apple extends Fruit { // 苹果类
}

public class Orange extends Fruit { // 橘子类
}


public class Fuji extends Apple { // 富士苹果类
}

public class GreenApple extends Apple { // 青苹果类
}
```

私有成员不能继承

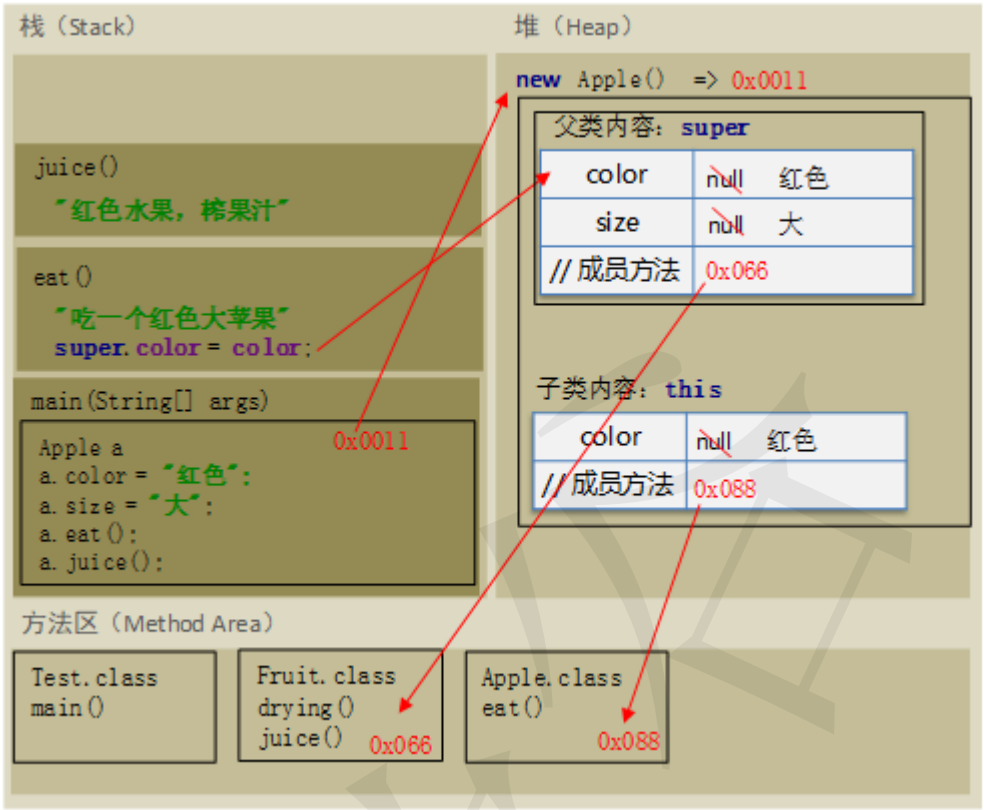
只能继承父类的非私有成员（成员变量、成员方法）

```
public class Fruit { // 水果类
    private String pri;
    public void test() {
        System.out.println(0);
    }
}

public class Apple extends Fruit { // 苹果类
}

public class Test { // 测试类
    public static void main(String[] args) {
        Apple a = new Apple();
        // System.out.println(a.pri); // 报错
        a.test();
    }
}
```

构造方法不能继承

构造方法用于初始化本类对象。

创建子类对象时，需要调用父类构造初始化该对象的父类内容，若父类构造可以被继承，该操作会造成调用的混乱。



继承体现了“is a”的关系

子类符合“is a（是一个）”父类的情况下，才使用继承，其它情况不建议使用