目标

面向对象编程（一）

- 理解继承及其特点
- 运用JAVA程序实现继承
- 理解多态及其特点设计
- 掌握访问修饰符

继承

父类/子类是相对的

父类

交通工具


自行车


小汽车


畜力车


卡车


奔驰


宝马

子类

继承的特点

- Java使用的是单一继承
- 继承的双方必须满足 is-a 关系
- 具有层次结构
- 子类继承了父类的属性和方法

继承的优点

- 代码的可重用性
- 父类的属性和方法可用于子类
- 可以轻松地自定义子类
- 设计应用程序变得更加简单

继承的JAVA实现

```
class Vehicles { //父类
protected String name = “bmw";   /**存储交通工具的名称. */
protected String color = "Red";   /** 存储颜色信息.*/
Vehicles() {}                                  /** 构造器.*/
void showDetail() { //显示父类汽车的详细信息
      //…….
   }
}
```

```
class Car extends Vehicles {//子类
Car() { } /**构造器. */
void show() {/** 显示子类 Car 的信息. */
                //…..
   }
}
```

构造器调用顺序

在实例化子类对象的时候，系统会先调用父类构造器，再调用子类构造器。

- 要调用父类构造器，使用关键字 super
- 即使我们没有显示的去调用父类的构造器，系统也会默认在子类构造器的第一行自动调用父类的一个空参数的构造器。

关键字 super

- 调用父类构造器的语法为：
  super() 或 super(参数列表)；

- super () 方法始终指向调用类的父类构造器
- 子类调用父类的构造器是，必须在子类构造器的第一行代码调用。

- super.方法() 可以在子类中调用父类的构造器以外的其他方法。

调用父类构造器

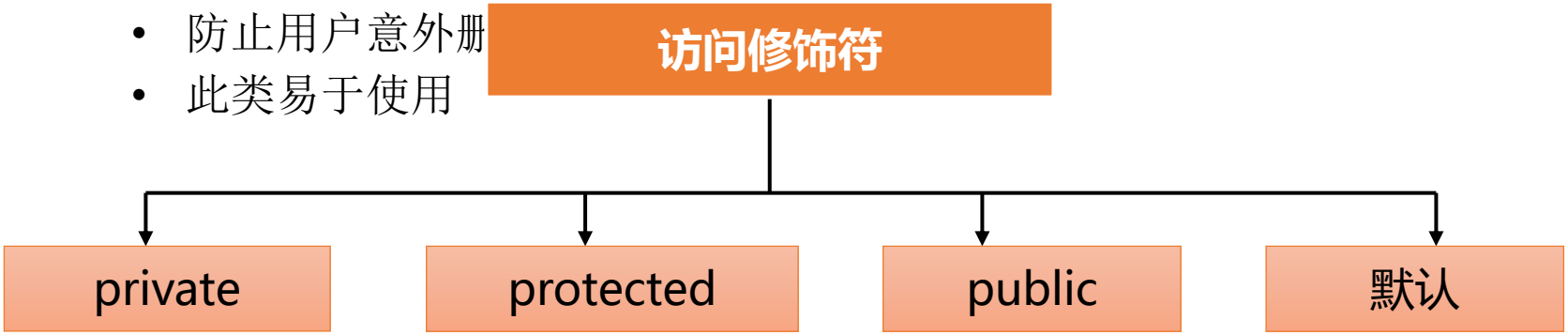
- 关键字 super 调用父类构造示例

```
class FictionAuthor extends Author {
        FictionAuthor(String name, String type) {
            super(name);
            storytype = type;
            System.out.println("从 FictionAuthor 类输出");
            System.out.println("小说类型为" + storytype);
    }
}
```

调用父类构造器

访问修饰符

- 信息隐藏是 OOP 最重要的功能之一，也是使用访问修饰符的原因
- 信息隐藏的原因包括：
  - 对任何实现细节所作的更改不会影响使用该类的代码
  - 防止用户意外册
  - 此类易于使用

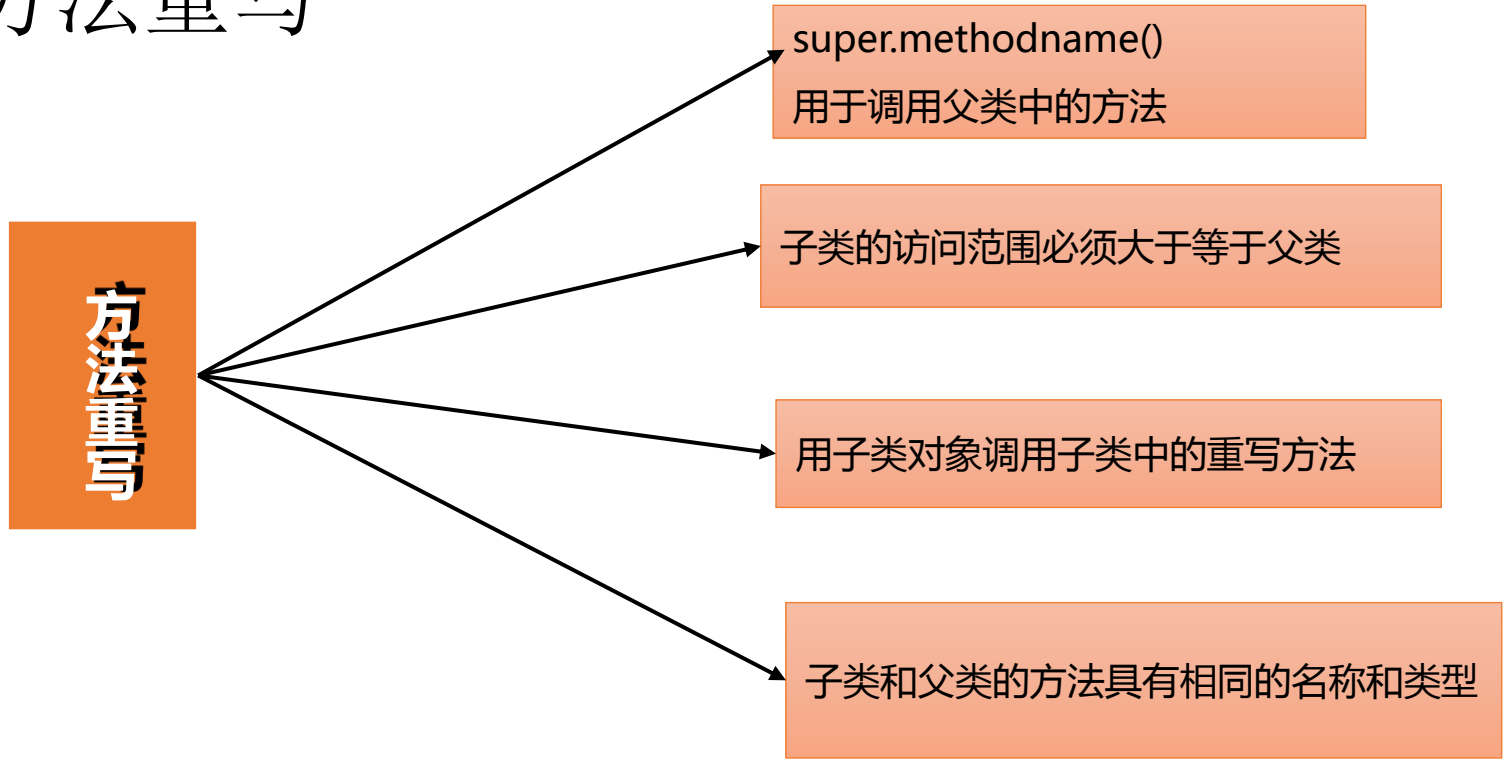


访问修饰符

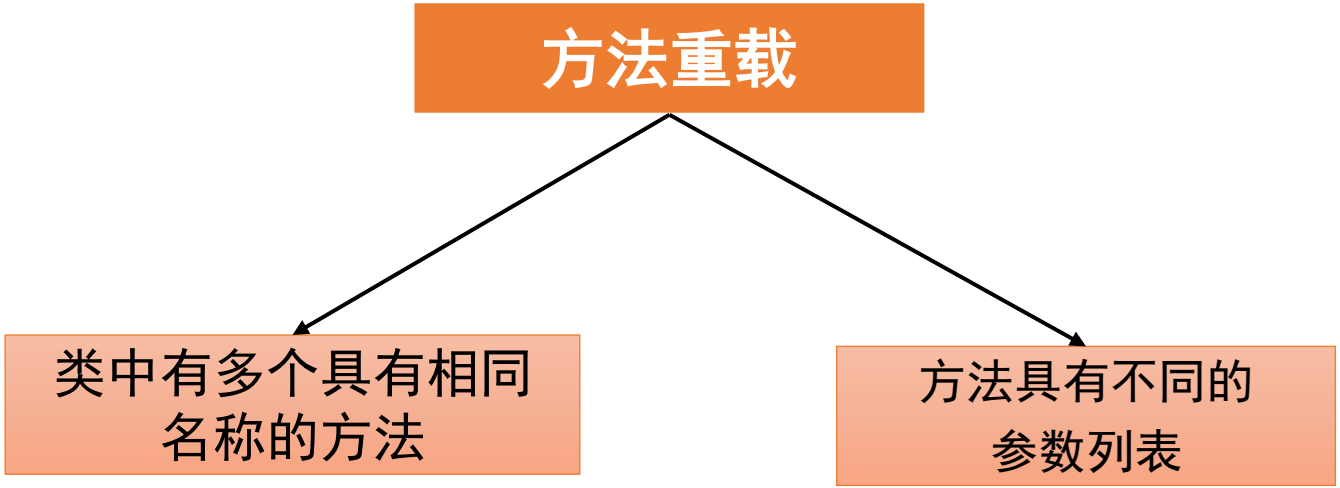
- 访问修饰符可访问性

修饰符	类自身	同包内	子类	所有其他类
private	Yes			
默认	Yes	Yes		
protected	Yes	Yes	Yes	
public	Yes	Yes	Yes	Yes

方法重写


方法重写
super.methodname()
用于调用父类中的方法
子类的访问范围必须大于等于父类
用子类对象调用子类中的重写方法
子类和父类的方法具有相同的名称和类型

方法重载



构造器重载

- 构造器重载和一般的方法重载类似。
-

```
public class Employee {
    private int id;

    public  Employee(){
        this(1); //调用自己的其他构造器
    }
    /** 构造器重载 */
    public  Employee(int id){
        this.id=id;
    }

    //getter方法
    public int getId() {
        return id;
    }
}
```

instanceof运算符

```
public class TestInstanceof {
    public static void main(String[] args) {
        Employee e1 = new Employee();

        if(e1 instanceof Employee){
            System.out.println("e1是Employee类的对象");
        }else{
            System.out.println("e1不是Employee类的对象");
        }

    }
}
```

Object类

- Java.lang.Object类是所有类的父类。
- 如果一个类的声明没有明确地指出继承哪个类，则编译器默认地

方法	说明
boolean equals(Objectobj)	将当前对象实例与给定的对象进行比较，检查它们是否相等
void finalize() throws Throwable	当垃圾回收器确定不存在对该对象的更多引用时，由对象的垃圾回收器调用此方法。通常被子类重写
String toString()	返回此对象的字符串表示
void wait() throws InterruptedException	使当前线程进入等待状态

Object类的常用方法 toString()方法

toString方法将一个对象转换为String。
如果我们需要输出各种有用的信息就必须覆盖toString方法。

当发生自动的字符串转换时，编译器会调用这个方法。
例如System.out.println()调用：

```
Employee emp = new Employee();
System.out.println(emp);
```

等同于

```
Employee emp = new Employee();
System.out.println(emp.toString());
```

```
package com.igeekhome;
    public class Employee {
    private int id;
    public int getId() { return id; }
    public void setId(int id) { this.id = id;}

    public boolean equals(Object obj) {
        if(this == obj) { return true; }
        if ((obj != null) && (obj instanceof Employee)) {
                Employee emp = (Employee) obj;
                if (this.getId() == emp.getId()) {
                        return true;
            }
        }
        return false ;
    }
    public int hashCode() {return (id << 5);}
}
```

相等。

枚举

- 枚举是用一组有限的符号名称来表示一组属性值，例如一年就四个季节，我们只要定义四个常量就行了。

- Java SE 5.0开始，定义枚举类使用关键字enum，其基本语法如下面一个例子所示：

```
public enum Season{
        SPRING, SUMMER, AUTUMN, WINTER
}
```

总结

- 封装、继承和多态是面向对象的主要特征
- 继承可提高代码的重用性，使用extends关键字来实现。除了构造方法之外，父类的所有方法和属性都被子类的对象继承
- 访问修饰符用于确定访问类成员的方式
- Java.lang.Object类是所有类的父类。
- 枚举是用一组有限的符号名称来表是一些属性值

400-133-0510

www.igeekhome.com