课程回顾

1 构造方法

```
构造方法书写时，有哪些特点？
    public 没有返回值部分，方法名与类同名{

    }
如何调用？
    其他类，创建对象，new 构造方法()
    同一个类，出现多个构造方法，this()
构造方法编写时机：构造方法中实现属性赋值

构造方法重载问题？
必须要保证方法形参列表不一样（个数、类型、顺序不一样）
    举例：两个构造方法不形成重载！！！！
    public Person(int a,int b){

    }

    public Person(int age,int height){

    }
```

2 封装

```
oop三大特征：
    封装    继承    多态

封装：保证属性输入值，值合法！！！！
封装实现步骤：
1.private 修饰属性
2.设置值 setter    获取值 getter
```

3 访问修饰符 ===== 理解

```
public：
protected:
private:
默认的：啥都不写，就是默认的
```

回顾案例: 模拟动物世界

属性都封装、体重和年龄必须正数

无参构造方法、全参构造方法

狗类：Dog

属性：品种 体重 毛发颜色 昵称 龄

方法：叫 showInfo()

猫类：Cat

属性：品种 体重 毛发颜色 昵称 龄

方法：叫 showInfo()

猪类：Pig

属性：品种 体重 昵称 龄

方法：叫 showInfo()

添加测试类：对象创建。分别叫 showInfo()

参考代码

```
1
```

课程目标

1 继承

2 super和super()

3 抽象 ==== 难点

课程实施

1 继承

1-1 作用

减少多个类存在的冗余代码

1-2 概念

继承：java提供一种子类沿用父类非私有属性和方法的编码形式。

1-3 语法

```
1 public class 子类 extends 父类{
2
3 }
```

课堂案例：西游记

```
1 唐僧：和尚
2 孙悟空：行者 念经
3 猪八戒：使者
4 沙和尚：罗汉
5
6 OOP创建三个类：
7 行者类：
8 属性：nickName age sex
```

```
方法：打斗   念经

使者类：
属性：nickName   age    sex
方法：吃   念经

罗汉类：
属性：nickName   age    sex
方法：念经

继承解决多个类存在的重复的属性和方法，
```

1-4 继承实施步骤

```
1. 抽取多个类中重复的属性和方法形成一个独立的类，一般称为父类

2.分别创建自己的类，自己的类需要使用父类中的属性和方法，使用extends继承即可
```

课堂案例参考代码

- 父类：HeShang

```java
package cn.kgc.xyj;

/**
 * @Author: lc
 * @Date: 2022/3/23
 * @Description: 和尚类，作为父类，属性和方法都是子类共同存在的
 * @Version: 1.0
 */
public class HeShang {
    //属性
    public String nickName;
    public int age;
    public char sex;

    //方法
    public void showInfo(){
        System.out.println(nickName+","+age+","+sex);
    }

    public void nianJing(){
        System.out.println("是和尚都要念经");
    }
}
```

- 子类：XingZhe

```java
package cn.kgc.xyj;

/**
 * @Author: lc
 * @Date: 2022/3/23
 * @Description: 行者类
 * @Version: 1.0
```

```
 */
//子类  extends 父类  子类  is a 父类的一种,满足伦理关系
public class XingZhe extends HeShang {//XingZhe是子  HeShang是父
    //提供子类特有的属性和方法
    public void daDou(){
        System.out.println("打斗....");
    }
}
```

- 子类：ShiZhe

```
package cn.kgc.xyj;

/**
 * @Author: lc
 * @Date: 2022/3/23
 * @Description: 使者类
 * @Version: 1.0
 */
public class ShiZhi extends HeShang{
    public void eat(){
        System.out.println("吃吃吃...");
    }
}
```

- 子类：LuoHan

```
package cn.kgc.xyj;

/**
 * @Author: lc
 * @Date: 2022/3/23
 * @Description: 罗汉类
 * @Version: 1.0
 */
public class LuoHan extends HeShang{

}
```

- 测试类：主要是通过对象能否使用父类中定义的属性和方法

```
package cn.kgc.xyj;

/**
 * @Author: lc
 * @Date: 2022/3/23
 * @Description: 测试类
 * @Version: 1.0
 */
public class Tester {
    public static void main(String[] args) {
        XingZhe xz = new XingZhe();
        xz.nickName="孙悟空";
        xz.showInfo();
        xz.daDou();

```

```
        ShiZhi sz = new ShiZhi();
        sz.eat();

        LuoHan lh=new LuoHan();
        lh.showInfo();
    }
}
```

1-5 继承的三大特征

```
1.单根性
  一个子类有且只能有一个父类
2.相对性
  描述父子类的关系时，必须基于相对关系进行描述
LuoHan相对于HeShang是子类  HeShang相对于LuoHan是父类
3.传递性
  类与类之间发生继承关系，祖先辈的属性通过父类传递孙子类
```

1-6 继承的使用细节

问题一：子类会不会出现与父类同名的属性

问题二：子类能不能继承父类的构造方法？？？不能继承！

子类默认在自己的构造方法里面调用父类的无参构造方法super(),如果子类在使用super()找父类无参构造方法，父类没有提供无参构造方法，子类自动报错！！！

解决方案：

1.父类添加一个无参构造方法

2.父类没有提供无参构造方法，子类构造方法需要程序员super(实参列表)显示指定调用父类哪个构造方法

子类希望super()显式指定父类的构造方法调用的情况，建议：子类的构造方法形参列表设计调用的父类的构造方法形参一样。

问题三：子类继承父类，同名方法怎么办？

子类里面：super.父类的方法 this.子类的方法

测试类里面：创建的是哪个类的对象，就用的是自己的方法。子类调用了一个方法，子类没有提供该方法，自动找父类的。

子类调用方法顺序：先找自己的，自己没有，就找父类，一直找不到，IDEA直接会提示错误！！！

2 super和super()

2-1 super概念

super：父类的对象

this：当前类的对象

2-2 super的作用

this用在当前类，区分同名的全局变量和局部变量 this.当前类的属性或方法

super用在子类中，区分子类和父类同名的属性。super.父类的属性或方法

2-3 super()语句概念

this(): 当前类的构造方法

super(): 父类的构造方法

2-4 super()语句的作用

在子类中表示父类构造方法。具体是哪个父类构造方法，看super()中传入的实参。

2-5 super()使用的注意事项

super()必须放在子类构造方法第一行。super()和this()不能不同使用！！！

3 子类和父类构造方法的调用关系

创建子类对象时，先调用父类构造方法，再调用子类构造方法，如果存在多级继承，则一层一层向上找父类，直到找到最顶层的父类开始执行。

课堂案例

- Father

```
package cn.kgc.demo;

/**
 * @Author: lc
 * @Date: 2022/3/23
 * @Description: cn.kgc.demo
 * @Version: 1.0
 */
public class Father {
    public Father(){
        System.out.println("Father的无参构造方法");
    }
    public Father(int a){
        System.out.println("Father的有参构造方法");
    }
}

```

- Son

```java
package cn.kgc.demo;

/**
 * @Author: lc
 * @Date: 2022/3/23
 * @Description: cn.kgc.demo
 * @Version: 1.0
 */
public class Son extends Father{
    public Son(){
        System.out.println("Son的无参构造方法");
    }
}
```

- 测试类

```java
package cn.kgc.demo;

/**
 * @Author: lc
 * @Date: 2022/3/23
 * @Description: cn.kgc.demo
 * @Version: 1.0
 */
public class TestFather {
    public static void main(String[] args) {
        /**
         * 小结：创建子类对象时，不仅调用子类的构造方法，首先调用父类的构造方法，再调用子类的构造方法
         */
        Son son=new Son();
    }
}
```

课堂案例2：分析super()调用的意义

- Father类

```java
package cn.kgc.demo;

/**
 * @Author: lc
 * @Date: 2022/3/23
 * @Description: cn.kgc.demo
 * @Version: 1.0
 */
public class Father {
    public String name;
    public int age;

    public Father(){
        System.out.println("Father的无参构造方法");
    }
    public Father(String name,int age){
```

```
        this.name=name;
        this.age=age;
        System.out.println("Father的有参构造方法");
    }
}
```

- Son类

```
package cn.kgc.demo;

/**
 * @Author: lc
 * @Date: 2022/3/23
 * @Description: cn.kgc.demo
 * @Version: 1.0
 */
public class Son extends Father{
    public String phone;//手机号码

    public Son(){
        System.out.println("Son的无参构造方法");
    }

    public Son(String name, int age) {
        super(name, age);
    }

    public Son(String name, int age, String phone) {
        //this(phone); //this()和super()不能同时出现
        super(name, age);
        this.phone = phone;
    }

    public Son(String phone){
        this.phone=phone;
    }
}
```

- 测试类，分析代码创建对象的执行过程

```
package cn.kgc.demo;

/**
 * @Author: lc
 * @Date: 2022/3/23
 * @Description: cn.kgc.demo
 * @Version: 1.0
 */
public class TestFather {
    public static void main(String[] args) {
        /**
         * 小结：创建子类对象时，不仅调用子类的构造方法，首先调用父类的构造方法，再调用子类的构造方法
         */
        //SonSon ss=new SonSon();
        Son son=new Son("jack",21,"189723456789");
```

```
    }
}
```

4 重写

重载：同一个类中，同名，形参不同

重写：发生在父子类之间，子类出现与父类同名、同参、同返回值的方法

4-1 作用！！

子类可以在父类实现的基础上，做优化！！！

子类提供同一个行为，与父类不同的实现方式，不同子类对于同一个功能具有不同实现方式

4-2 重写时机

子类与父类同一个方法，想有不同的实现代码

4-3 课堂案例

- 父类：HeShang类

```java
package cn.kgc.xyj;

/**
 * @Author: lc
 * @Date: 2022/3/23
 * @Description: 和尚类，作为父类，属性和方法都是子类共同存在的
 * @Version: 1.0
 */
public class HeShang {
    //属性
    public String nickName;
    public int age;
    public char sex;

    public int num=100;//父类中属性

    //方法
    public void showInfo(){
        System.out.println(nickName+","+age+","+sex);
    }

    public void nianJing(){
        System.out.println("是和尚都要念经");
    }
}
```

- 第一个子类：XingZhe

```java
package cn.kgc.xyj;

/**
 * @Author: lc
 * @Date: 2022/3/23
```

```
 * @Description: 行者类
 * 念经：有什么好念的，不念经！！
 * @Version: 1.0
 */
//子类  extends 父类  子类  is a 父类的一种,满足伦理关系
public class XingZhe extends HeShang {//XingZhe是子  HeShang是父
    //提供子类特有的属性和方法
    public void daDou(){
        System.out.println("打斗....");
    }

    @Override  //重写的方法特有的标记
    public void nianJing(){
        System.out.println("有什么好念的，不念经！！");
    }
}
```

- 第二个子类：ShiZhe

```
package cn.kgc.xyj;

/**
 * @Author: lc
 * @Date: 2022/3/23
 * @Description: 使者类
 * 猪八戒念经：师傅盯着我，我就好好念，师傅不在，我就不念了
 * @Version: 1.0
 */
public class ShiZhi extends HeShang{
    public void eat(){
        System.out.println("吃吃吃...");
    }

    @Override
    public void nianJing() {
        System.out.println("师傅盯着我，我就好好念，师傅不在，我就不念了");
    }
}
```

- 第三个子类：LuoHan

```
package cn.kgc.xyj;

/**
 * @Author: lc
 * @Date: 2022/3/23
 * @Description: 罗汉类
 * 念经：好好听师傅的话，认真念经
 * @Version: 1.0
 */
public class LuoHan extends HeShang{
    //只需要关注子类自己特有的属性和方法定义编写
    public int num=-100;//子类中属性

    public void showNum(){
        System.out.println("父类的num="+super.num);
```

```
        //父类和子类出现同名的属性，子类优先使用自己的
        System.out.println("子类的num="+this.num);
    }
    //alt+insert 输入方法名敲回车
    @Override
    public void nianJing() {
        System.out.println("好好听师傅的话，认真念经");
    }
}
```

- 测试类：创建三个不同的对象，看看方法执行的结果如何？

```
package cn.kgc.xyj;

/**
 * @Author: lc
 * @Date: 2022/3/23
 * @Description: 测试类
 * @Version: 1.0
 */
public class Tester {
    public static void main(String[] args) {
        XingZhe xz = new XingZhe();
        xz.nianJing();//XingZhe没有念经，使用的是HeShang

        ShiZhi sz = new ShiZhi();
        sz.nianJing();

        LuoHan lh=new LuoHan();
        lh.nianJing();
    }
}
//输出结果如下所示：
有什么好念的，不念经！！
师傅盯着我，我就好好念，师傅不在，我就不念了
好好听师傅的话，认真念经
```

大整合：将封装+继承+构造方法融合一起的使用案例

- 父类：动物类Animal

```
package cn.kgc.animal2;

/**
 * @Author: lc
 * @Date: 2022/3/23
 * @Description: cn.kgc.animal2
 * @Version: 1.0
 */
public class Animal {
    //属性 品种    昵称
    private String brand;//继承：非私有的属性和方法
    private String nickName;

    public String getBrand() {
        return brand;
```

```
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public String getNickName() {
        return nickName;
    }

    public void setNickName(String nickName) {
        this.nickName = nickName;
    }
    //构造方法
    public Animal(String brand, String nickName) {
        this.brand = brand;
        this.nickName = nickName;
    }


    //方法  叫    showInfo()
    public void shout(){
        //动物怎么叫？？
        //System.out.println("动物会叫");
    }

    public String showInfo(){
        return nickName+","+brand;
    }
}
```

- 子类：狗类Dog

```
package cn.kgc.animal2;

/**
 * @Author: lc
 * @Date: 2022/3/23
 * @Description: 描述狗的基本信息和行为
 * @Version: 2.0
 */
public class Dog extends Animal{
    //Dog有没有个性化的属性
    private String color;

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }
    //构造方法

    public Dog(String brand, String nickName, String color) {
        super(brand, nickName);
        this.color = color;
```

```
    }

    //showInfo()输出属性个数与父类方法内容一样？
    @Override
    public String showInfo() {
        return getNickName()+","+getBrand()+","+color;
    }

    @Override
    public void shout() {
        System.out.println("汪汪汪叫...");
    }
}
```

- 测试类：Tester

```
package cn.kgc.animal2;

/**
 * @Author: lc
 * @Date: 2022/3/23
 * @Description: cn.kgc.animal2
 * @Version: 1.0
 */
public class Tester {
    public static void main(String[] args) {
        Dog dog = new Dog("金毛", "大黄", "黄色");
        System.out.println(dog.showInfo());
        dog.shout();
    }
}
//输出结果如下所示
大黄,金毛,黄色
汪汪汪叫...
```

课程总结

1.继承实现方式、实现步骤

1-1 继承三大特性、继承概念

2.继承之后。两个类同名属性、同名方法、构造方法

3.重写

4.super super()

预习安排

Object类型 static关键字 final关键字