**每日作业卷**

**就业班JavaSE 第二天【方法重写、this、super、抽象类】**

# 关卡1

## 训练案例1

### 训练考核知识点

重写

### 训练描述

请阐述你对重写的理解

### 操作步骤描述

1. 请问什么是重写 ?

2. 请问什么时候需要重写,并举例说明 ?

3. 请问重写有哪些注意事项 ?

1. 子类重写父类的方法

2. 父类的功能不满足子类的需求，比如Person类有一个eat()方法描述人吃饭，子类Student类重写eat()方法，描述学生吃食堂。

3. 参数列表必须完全与被重写的方法相同  
返回的类型必须一直与被重写的方法的[返回类型](https://www.baidu.com/s?wd=%E8%BF%94%E5%9B%9E%E7%B1%BB%E5%9E%8B&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)相同   
访问修饰符的限制一定要大于被重写方法的访问修饰符

## 训练案例2

### 训练考核知识点

子类的成员与父类的成员变量重名

### 训练描述

请阐述当子类的成员与父类的成员变量重名时,如何访问成员变量 ?

### 操作步骤描述

1. 请说出当子类的成员与父类的成员变量重名时,在子类中,如何访问子类的成员变量?
2. 请说出当子类的成员与父类的成员变量重名时,在子类中,如何访问父类的成员变量?
   * + 1. this.变量名
       2. super.变量名

## 训练案例3

### 训练考核知识点

this与super关键字

### 训练描述

请阐述this与super关键字的作用和注意事项.

### 操作步骤描述

1. 请说如何使用this关键字访问本类的成员变量?
2. 请说如何使用this关键字访问本类的成员方法?
3. 请说如何使用this关键字在构造方法中,调用本类的其他构造方法?
4. 请说如何使用super关键字访问父类的成员变量?
5. 请说如何使用super关键字访问父类的成员方法?
6. 请说如何使用super关键字在构造方法中,调用父类的构造方法?
7. 请说如果在构造方法中我们没有this调用父类的构造方法,也没有super调用父类的构造方法,会怎么样?
8. 请问在构造方法中可以同时使用this调用本类构造方法,super调用父类构造方法吗?

this.成员变量

this.成员方法

this();或this(参数);

super.成员变量

super.成员方法

super();或super(参数);

默认调用父类的无参构造

可以

## 训练案例4

### 训练考核知识点

抽象方法

抽象类

### 训练描述

请阐述你对抽象方法和抽象类的理解

### 操作步骤描述

1. 请说出什么抽象方法 ?
2. 请说出什么抽象类 ?
3. 请说出抽象方法的特点是什么 ?
4. 请问什么时候需要使用抽象方法 ?
5. 请抽象类可以直接创建对象吗? 为什么?
6. 请问什么是方法实现 ?
7. 请问抽象类可以有构造方法吗,构造方法在抽象类有意义吗?
8. 请问抽象类一定是父类吗?
9. 请问抽象类中可以没有抽象方法吗 ?
10. 请问是先有抽象类还是先有子类?
11. 抽象方法指只有功能声明，没有功能主体实现的方法
12. 抽象类用来描述一种类型应该具备的基本特征与功能，具体如何去完成这些行为由子类通过方法重写来完成
13. 没有功能主体实现的方法
14. 不明确实现细节但又需要声明的方法可以使用抽象方法的方式完成
15. 不可以，抽象类不象具体类那样描述一类具体事物
16. 实现方法的功能
17. 可以，有意义
18. 不一定
19. 可以
20. 抽象类

## 训练案例5

### 训练考核知识点

继承的格式

子类访问父类的成员变量

子类访问父类的方法

在子类中使用super关键调用父类构造方法

### 训练描述

请用代码描述:

一只黄色的加菲猫和一条白色名称为洛洛的哈士奇狗

要求: 把猫和狗共性抽取到动物类中

### 操作步骤描述

1. 定义动物类(Animal)
   1. 成员变量 名称(name), 颜色(color)
   2. 要求:成员变量私有,提供空参和有参的构造方法
2. 定义猫类(Cat),继承Animal类
   1. 提供空参,有参构造方法,在有参构造中调用父类的构造方法
3. 定义狗类(Dog)
   1. 成员变量: 品种(breed)
   2. 提供空参,有参构造方法,在有参构造中调用父类的构造方法
4. 创建测试类Test
   1. 提供main方法
   2. 在main方法中
      1. 使用Cat类的带参构造创建猫对象 c
      2. 使用 Dog 类的带参构造创建狗对象 d

**package** com.java.test01;

**public** **class** Animal {

**private** String name;

**private** String color;

**public** Animal() {

**super**();

}

**public** Animal(String name, String color) {

**super**();

**this**.name = name;

**this**.color = color;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getColor() {

**return** color;

}

**public** **void** setColor(String color) {

**this**.color = color;

}

}

**package** com.java.test01;

**public** **class** Cat **extends** Animal {

**public** Cat() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Cat(String name, String color) {

**super**(name, color);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test01;

**public** **class** Dog **extends** Animal {

**private** String breed;

**public** Dog() {

**super**();

}

**public** Dog(String name, String color, String breed) {

**super**(name, color);

**this**.breed = breed;

// **TODO** Auto-generated constructor stub

}

**public** String getBreed() {

**return** breed;

}

**public** **void** setBreed(String breed) {

**this**.breed = breed;

}

}

**package** com.java.test01;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Cat c = **new** Cat("加菲猫", "黄色");

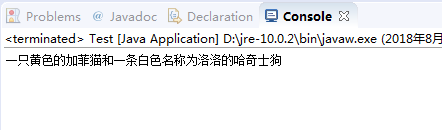
Dog d = **new** Dog("洛洛", "白色", "哈奇士");

System.***out***.println("一只" + c.getColor() + "的" + c.getName() + "和一条" + d.getColor() + "名称为" + d.getName() + "的"

+ d.getBreed() + "狗");

}

}



## 训练案例6

### 训练考核知识点

继承后对象的内存图

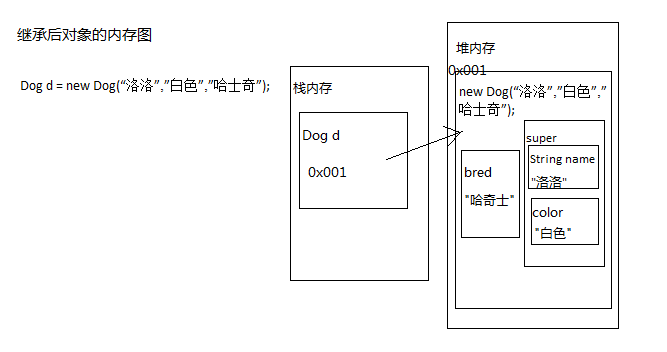
### 训练描述

请绘制Dog d = new Dog(“洛洛”,”白色”,”哈士奇”); 这句代码的内存图

说明: 这里Dog类是第5题中的Dog类

### 操作步骤描述

1. 打开画图软件
2. 在画布的左上角写上: 继承后对象的内存图
3. 在左边写上: Dog d = new Dog(“洛洛”,”白色”,”哈士奇”);
4. 在这句代码右边,绘制大长方形方块,在左上角标注 栈内存
5. 在这个长方形方块的右侧,在绘制一个更大的方块标注为: 堆内存
6. 在栈内存的方块内部,绘制一个小方块, 在左上角标注为Dog d; 表示为局部变量d开辟的存储空间
7. 在堆内存的方法内部,绘制一个大一点方块,用于描述创建对象分配的存储空间,在这个方块,上面写 new Dog(“洛洛”,”白色”,”哈士奇”); 表示创建对象分配的存储空间
8. 在这个对象的方块内,右边绘制一个比较大方块,这个方法左上角标注为super表为父类开辟的存储空间.
9. 在父类的存储空间,绘制一个小方块,左上角标注为String name; 表示父类的name的成员变量; 小方块内部写上 “洛洛”,表示该成员变量的值为 “洛洛”
10. 再在父类的方块中绘制一个小方块,左上角标注为color,表示为成员变量color分配的存储空间,方块内部写上”白色”,表示该成员变量的值为”白色”
11. 在当前对象的方块内部,父类存储空间外部,绘制一个小方块,左上角标注为breed,表示为子类成员变量breed分配的存储空间,方块内部写上”哈士奇”,表示成员变量breed的值为哈士奇
12. 在对象方块的左上角,标注0x001, 表示该对象的内存地址
13. 在栈内存中,代表局部变量d的小方块中写上0x001,表示局部变量d中存储是对象的地址;
14. 绘制一条从栈内存局部变量d中0x001到堆内存对象的0x001带有箭头的线,表示栈内存的变量,指向堆内存的对象



# 关卡2

## 训练案例1

### 训练考核知识点

继承格式

方法重写

子类调用父类的构造方法

### 训练描述

请使用代码描述:

工资为8000元的30岁的王小平老师,会吃饭(吃工作餐)和讲课.

成绩为90分的15岁的李小乐学生,会吃饭(吃学生餐)和学习.

提示: 把老师和学生的共性抽取人类中,人类不使用抽象类

### 操作步骤描述

1. 定义Person类
   * 1. 成员变量(私有): 名称(name),年龄(age)
     2. 成员方法: 吃饭(void eat())
        1. 输出格式: 30岁的王小平在吃饭
     3. 提供空参和带参构造方法
     4. 提供setXxx和getXxx方法
2. 定义老师类(Teacher),继承Person类
   * 1. 成员变量: salary(工资)
     2. 成员方法:
        1. 重写父类的 eat()方法
           1. 输出格式:: 工资为8000元的30岁的王小平老师在吃工作餐
        2. 特有方法: lecture() 讲课方法
           1. 输出格式:: 工资为8000元的30岁的王小平老师在讲课
     3. 提供空参和带参构造方法
     4. 提供setXxx和getXxx方法
3. 定义学生类(Student),继承Person类
   * 1. 成员变量: score(成绩)
     2. 成员方法:
        1. 重写父类的 eat()方法
           1. 输出格式:: 成绩为90分的15岁的李小乐学生在吃学生餐
        2. 特有方法: study() 学习方法
           1. 输出格式:: 成绩为90分的15岁的李小乐学生在学习
     3. 提供空参和带参构造方法
     4. 提供setXxx和getXxx方法
4. 定义Test类
   1. 提供main方法
   2. 在main方法中
      1. 创建老师对象t,并把名称赋值为”王小平”,年龄赋值为30,工资赋值为8000
      2. 调用老师对象t的吃饭方法
      3. 调用老师对象t的讲解方法
      4. 创建学生对象 s,并把名称赋值为”李小乐”,年龄赋值为14,成绩赋值为90分.
      5. 调用学生对象 s 的吃饭方法
      6. 调用学生对象 s 的学习方法

**package** com.java.test02;

**public** **class** Person {

**private** String name;

**private** **int** age;

**public** **void** eat() {

System.***out***.println(**this**.age + "岁的" + **this**.name + "在吃饭");

}

**public** Person() {

}

**public** Person(String name, **int** age) {

**this**.name = name;

**this**.age = 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;

}

}

**package** com.java.test02;

**public** **class** Teacher **extends** Person {

**private** **double** salary;

@Override

**public** **void** eat() {

System.***out***.println("工资为" + **this**.salary + "元的" + **this**.getAge() + "岁的" + **this**.getName() + "老师在吃工作餐");

}

**public** **void** lecture() {

System.***out***.println("工资为" + **this**.salary + "元的" + **this**.getAge() + "岁的" + **this**.getName() + "老师在讲课");

}

**public** Teacher() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Teacher(String name, **int** age, **double** salary) {

**super**(name, age);

**this**.salary = salary;

// **TODO** Auto-generated constructor stub

}

**public** **double** getSalary() {

**return** salary;

}

**public** **void** setSalary(**double** salary) {

**this**.salary = salary;

}

}

**package** com.java.test02;

**public** **class** Student **extends** Person {

**private** **int** score;

@Override

**public** **void** eat() {

System.***out***.println("成绩为" + **this**.score + "分的" + **this**.getAge() + "岁的" + **this**.getName() + "学生在吃学生餐");

}

**public** **void** study() {

System.***out***.println("成绩为" + **this**.score + "分的" + **this**.getAge() + "岁的" + **this**.getName() + "学生在学习");

}

**public** Student() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Student(String name, **int** age, **int** score) {

**super**(name, age);

**this**.score = score;

// **TODO** Auto-generated constructor stub

}

**public** **int** getScore() {

**return** score;

}

**public** **void** setScore(**int** score) {

**this**.score = score;

}

}

**package** com.java.test02;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Teacher teacher = **new** Teacher("王小平", 30, 8000);

teacher.eat();

teacher.lecture();

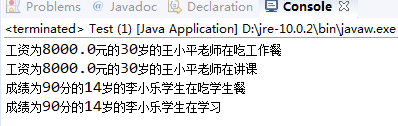
Student student = **new** Student("李小乐", 14, 90);

student.eat();

student.study();

}

}



## 训练案例2

### 训练考核知识点

继承格式

方法重写

子类调用父类的构造方法

### 训练描述

请使用代码描述:

2岁的红色的公鸡会吃饭(啄米)和打鸣

1岁的黑色的鸭子会吃饭(吃鱼)和游泳.

提示: 把公鸡和鸭子的共性抽取家禽类中,家禽类不使用抽象类

### 操作步骤描述

1. 定义家禽类(Poultry)
   * 1. 成员变量(私有): 颜色(color),年龄(age)
     2. 成员方法: 吃饭(void eat())
        1. 输出格式: 2岁的红色家禽在吃饭
     3. 提供空参和带参构造方法
     4. 提供setXxx和getXxx方法
2. 定义公鸡类(Cock),继承Poultry类
   * 1. 成员方法:
        1. 重写父类的 eat()方法
           1. 输出格式:: 2岁的红色公鸡在啄米
        2. 特有方法: 打鸣(crow)
           1. 输出格式:: 2岁的红色公鸡在打鸣
     2. 提供空参和带参构造方法
3. 定义鸭子类(Duck),继承Poultry类
   * 1. 成员方法:
        1. 重写父类的 eat()方法
           1. 输出格式: 1岁的黑色的鸭子在吃鱼
        2. 特有方法: swim() 游泳方法
           1. 输出格式: 1岁的黑色的鸭子在游泳
     2. 提供空参和带参构造方法
4. 定义Test类
   1. 提供main方法
   2. 在main方法中
      1. 创建公鸡对象c,并把颜色赋值为”红色”,年龄赋值为2
      2. 调用公鸡对象c的吃饭方法
      3. 调用公鸡对象c的打鸣方法
      4. 创建鸭子对象 d,并把颜色赋值为”黑色”,年龄赋值为1.
      5. 调用鸭子对象 d 的吃饭方法
      6. 调用鸭子对象 d 的游泳方法

**package** com.java.test03;

**public** **class** Poultry {

**private** String color;

**private** **int** age;

**public** **void** eat() {

System.***out***.println(**this**.age + "岁的" + **this**.color + "家禽在吃饭");

}

**public** Poultry() {

}

**public** Poultry(String color, **int** age) {

**this**.color = color;

**this**.age = age;

}

**public** String getColor() {

**return** color;

}

**public** **void** setColor(String color) {

**this**.color = color;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

}

**package** com.java.test03;

**public** **class** Cock **extends** Poultry {

@Override

**public** **void** eat() {

System.***out***.println(**this**.getAge() + "岁的" + **this**.getColor() + "公鸡在啄米");

}

**public** **void** crow() {

System.***out***.println(**this**.getAge() + "岁的" + **this**.getColor() + "公鸡在打鸣");

}

**public** Cock() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Cock(String color, **int** age) {

**super**(color, age);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test03;

**public** **class** Duck **extends** Poultry {

@Override

**public** **void** eat() {

System.***out***.println(**this**.getAge() + "岁的" + **this**.getColor() + "鸭子在吃鱼");

}

**public** **void** swim() {

System.***out***.println(**this**.getAge() + "岁的" + **this**.getColor() + "鸭子在游泳");

}

**public** Duck() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Duck(String color, **int** age) {

**super**(color, age);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test03;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Cock cock = **new** Cock("红色", 2);

cock.eat();

cock.crow();

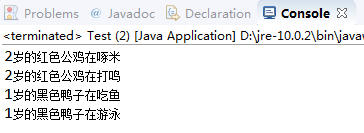
Duck duck = **new** Duck("黑色", 1);

duck.eat();

duck.swim();

}

}



## 训练案例3

### 训练考核知识点

继承格式

方法重写

子类调用父类的构造方法

抽象方法

抽象类

### 训练描述有

请使用代码描述:

18岁的演员景甜会吃饭(吃小龙虾)和跳舞

30岁的歌手薛之谦会吃饭(吃大闸蟹)和唱歌.

要求: 把演员和歌手的共性抽取人类中,使用抽象类和抽象方法

### 操作步骤描述

1. 定义抽象类 人类(Person)
   * 1. 成员变量(私有): 姓名(name),年龄(age)
     2. 抽象方法: 吃饭(void eat())
     3. 提供空参和带参构造方法
     4. 提供setXxx和getXxx方法
2. 定义演员类(Actor),继承Person类
   * 1. 成员方法:
        1. 重写父类的 eat()方法
           1. 输出格式: 18岁的演员景甜在吃小龙虾
        2. 特有方法: 跳舞(dance)
           1. 输出格式: 18岁的演员景甜在跳白天鹅.
     2. 提供空参和带参构造方法
3. 定义歌手类(Singer),继承Person类
   * 1. 成员方法:
        1. 重写父类的 eat()方法
           1. 输出格式: 30岁的歌手薛之谦在吃大闸蟹
        2. 特有方法: sing() 唱歌
           1. 输出格式: 30岁的歌手薛之谦在演唱丑八怪
     2. 提供空参和带参构造方法
4. 定义Test类
   1. 提供main方法
   2. 在main方法中
      1. 创建演员对象a,并把名称赋值为”景甜”,年龄赋值为18
      2. 调用演员对象a的吃饭方法
      3. 调用演员对象a的跳舞方法
      4. 创建歌手对象 s,并把姓名赋值为”薛之谦”,年龄赋值为30.
      5. 调用歌手对象 s 的吃饭方法
      6. 调用歌手对象 s 的唱歌方法

**package** com.java.test04;

**public** **abstract** **class** Person {

**private** String name;

**private** **int** age;

**public** **abstract** **void** eat();

**public** Person() {

}

**public** Person(String name, **int** age) {

**this**.name = name;

**this**.age = 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;

}

}

**package** com.java.test04;

**public** **class** Actor **extends** Person {

@Override

**public** **void** eat() {

System.***out***.println(**this**.getAge() + "的演员" + **this**.getName() + "在吃小龙虾");

}

**public** **void** dance() {

System.***out***.println(**this**.getAge() + "的演员" + **this**.getName() + "在跳白天鹅");

}

**public** Actor() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Actor(String name, **int** age) {

**super**(name, age);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test04;

**public** **class** Singer **extends** Person {

@Override

**public** **void** eat() {

System.***out***.println(**this**.getAge() + "的歌手" + **this**.getName() + "在吃大闸蟹");

}

**public** **void** sing() {

System.***out***.println(**this**.getAge() + "的歌手" + **this**.getName() + "在演唱丑八怪");

}

**public** Singer() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Singer(String name, **int** age) {

**super**(name, age);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test04;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Actor actor = **new** Actor("景甜", 18);

actor.eat();

actor.dance();

Singer singer = **new** Singer("薛之谦", 30);

singer.eat();

singer.sing();

}

}



## 训练案例4

### 训练考核知识点

继承格式

方法重写

子类调用父类的构造方法

抽象方法

抽象类

### 训练描述有

请使用代码描述:

四条腿黑色的狗会吃饭(吃骨头)和看家

两条腿的绿色鹦鹉吃饭(吃小米)和说话.

要求: 把狗和鹦鹉的共性抽取动物类中,使用抽象类和抽象方法

### 操作步骤描述

1. 定义抽象类 动物类(Animal)
   * 1. 成员变量(私有): 颜色(color),腿的个数(numOfLegs)
     2. 抽象方法: 吃饭(void eat())
     3. 提供空参和带参构造方法
     4. 提供setXxx和getXxx方法
2. 定义狗类(Dog),继承Animal类
   * 1. 成员方法:
        1. 重写父类的 eat()方法
           1. 输出格式:: 4条腿黑色的狗在啃骨头
        2. 特有方法: 看家(lookHome)
           1. 输出格式:: 4条腿黑色的狗在看家.
     2. 提供空参和带参构造方法
3. 定义鹦鹉类(Parrot),继承Animal类
   * 1. 成员方法:
        1. 重写父类的 eat()方法
           1. 输出格式:: 2条腿的绿色鹦鹉在吃小米
        2. 特有方法: say() 说明
           1. 输出格式:: 2条腿的绿色鹦鹉在说你好,丑八怪
     2. 提供空参和带参构造方法
4. 定义Test类
   1. 提供main方法
   2. 在main方法中
      1. 创建Dog对象d,并把颜色赋值为”黑色”,腿的个数赋值为4
      2. 调用Dog对象d的吃饭方法
      3. 调用Dog对象d的看家方法
      4. 创建Porrot对象 p,并把颜色赋值为”绿色”,腿的个数赋值为2.
      5. 调用Porrot对象 p 的吃饭方法
      6. 调用Porrot对象 p 的说话方法

**package** com.java.test05;

**public** **abstract** **class** Animal {

**private** String color;

**private** **int** numOfLegs;

**public** **abstract** **void** eat();

**public** Animal() {

}

**public** Animal(String color, **int** numOfLegs) {

**this**.color = color;

**this**.numOfLegs = numOfLegs;

}

**public** String getColor() {

**return** color;

}

**public** **void** setColor(String color) {

**this**.color = color;

}

**public** **int** getNumOfLegs() {

**return** numOfLegs;

}

**public** **void** setNumOfLegs(**int** numOfLegs) {

**this**.numOfLegs = numOfLegs;

}

}

**package** com.java.test05;

**public** **class** Dog **extends** Animal {

@Override

**public** **void** eat() {

System.***out***.println(**this**.getNumOfLegs() + "条腿" + **this**.getColor() + "的狗在啃骨头");

}

**public** **void** lookHome() {

System.***out***.println(**this**.getNumOfLegs() + "条腿" + **this**.getColor() + "的狗在看家");

}

**public** Dog() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Dog(String color, **int** numOfLegs) {

**super**(color, numOfLegs);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test05;

**public** **class** Parrot **extends** Animal {

@Override

**public** **void** eat() {

System.***out***.println(**this**.getNumOfLegs() + "条腿的" + **this**.getColor() + "鹦鹉在吃小米");

}

**public** **void** say() {

System.***out***.println(**this**.getNumOfLegs() + "条腿的" + **this**.getColor() + "鹦鹉在说你好,丑八怪");

}

**public** Parrot() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Parrot(String color, **int** numOfLegs) {

**super**(color, numOfLegs);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test05;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Dog dog = **new** Dog("黑色", 4);

dog.eat();

dog.lookHome();

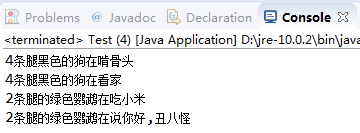
Parrot parrot = **new** Parrot("绿色", 2);

parrot.eat();

parrot.say();

}

}



## 训练案例5

### 训练考核知识点

继承格式

方法重写

子类调用父类的构造方法

抽象类

抽象方法

### 训练描述

请使用代码描述

在动物园中有多种动物,每种动物都年龄和腿的个数,按吃的不同分为食肉动物和食草动物

食肉动物: 有老虎,啄木鸟

食草动物: 有长颈鹿,兔子

吃的东西:

老虎: 年龄为 2的4条腿的老虎在吃羊

啄木鸟: 年龄为 1岁的2条腿的啄木鸟的在吃昆虫

长颈鹿: 年龄为 3 岁的4条腿的长颈鹿在吃树叶

兔子: 年龄为 1岁的4条腿的兔子在吃狗尾草

### 操作步骤描述

一. 分析继承关系:

动物类(Animal)

食草类动物(Herbivore)

食肉动物(Carnivore)

老虎(Tiger)

啄木鸟(Pecker)

长颈鹿(Giraffe)

兔子(Rabbit)

最顶层父类: 动物类(Animal)

食肉类动物(Carnivore)和食草类动物(Herbivore)继承自动物类(Animal)

老虎(Tiger)和啄木鸟(Pecker)继承食肉类动物(Carnivore)

长颈鹿(Giraffe)和兔子(Rabbit)继承食草动物(Herbivore)

1. 代码实现步骤
2. 定义抽象类动物类(Animal)
   1. 成员变量:年龄(age),腿的个数(numOfLegs)
   2. 抽象方法: void eat()
   3. 提供空参和有参构造方法和setXxx和getXxx方法
3. 定义抽象类食肉类动物(Carnivore)继承自动物类(Animal)
   1. 提供空参和有参构造方法
4. 定义抽象类食草类动物(Herbivore)继承自动物类(Animal)
   1. 提供空参和有参构造方法
5. 定义老虎类(Tiger)继承食肉类动物(Carnivore)
   1. 重写eat()方法
      1. 输出格式: 年龄为 2的4条腿的老虎在吃羊
   2. 提供空参和有参构造方法
6. 定义啄木鸟(Pecker)继承食肉类动物(Carnivore)
   1. 重写eat()方法
      1. 输出格式: 年龄为 1岁的2条腿的啄木鸟的在吃昆虫
   2. 提供空参和有参构造方法
7. 定义长颈鹿(Giraffe)继承食草动物(Herbivore)
   1. 重写eat()方法
      1. 输出格式: 年龄为 3 岁的4条腿的长颈鹿在吃树叶
   2. 提供空参和有参构造方法
8. 定义兔子(Rabbit)继承食草动物(Herbivore)
   1. 重写eat()方法
      1. 输出格式: 年龄为 1岁的4条腿的兔子在吃狗尾草
   2. 提供空参和有参构造方法
9. 定义测试类Test
   1. 提供main方法
   2. 在main方法中
      1. 创建老虎对象 t,年龄赋值为2,腿的个数赋值为4;调用老虎对象t的eat()方法
      2. 创建啄木鸟对象 p,年龄赋值为1,腿的个数赋值为2;调用啄木鸟对象p的eat()方法
      3. 创建长颈鹿对象 g,年龄赋值为3,腿的个数赋值为4;调用长颈鹿对象g的eat()方法
      4. 创建兔子对象 r,年龄赋值为1,,腿的个数赋值为4;调用兔子对象r的eat()方法

**package** com.java.test06;

**public** **abstract** **class** Animal {

**private** **int** age;

**private** **int** numOfLegs;

**public** **abstract** **void** eat();

**public** Animal() {

**super**();

}

**public** Animal(**int** age, **int** numOfLegs) {

**super**();

**this**.age = age;

**this**.numOfLegs = numOfLegs;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

**public** **int** getNumOfLegs() {

**return** numOfLegs;

}

**public** **void** setNumOfLegs(**int** numOfLegs) {

**this**.numOfLegs = numOfLegs;

}

}

**package** com.java.test06;

**public** **abstract** **class** Carnivore **extends** Animal {

**public** Carnivore() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Carnivore(**int** age, **int** numOfLegs) {

**super**(age, numOfLegs);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test06;

**public** **abstract** **class** Herbivore **extends** Animal {

**public** Herbivore() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Herbivore(**int** age, **int** numOfLegs) {

**super**(age, numOfLegs);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test06;

**public** **class** Giraffe **extends** Herbivore {

@Override

**public** **void** eat() {

System.***out***.println("年龄为" + **this**.getAge() + "岁的" + **this**.getNumOfLegs() + "条腿的长颈鹿在吃树叶");

}

**public** Giraffe() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Giraffe(**int** age, **int** numOfLegs) {

**super**(age, numOfLegs);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test06;

**public** **class** Pecker **extends** Carnivore {

@Override

**public** **void** eat() {

System.***out***.println("年龄为" + **this**.getAge() + "岁的" + **this**.getNumOfLegs() + "条腿的啄木鸟的在吃昆虫");

}

**public** Pecker() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Pecker(**int** age, **int** numOfLegs) {

**super**(age, numOfLegs);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test06;

**public** **class** Rabbit **extends** Herbivore {

@Override

**public** **void** eat() {

System.***out***.println("年龄为" + **this**.getAge() + "岁的" + **this**.getNumOfLegs() + "条腿的兔子在吃狗尾草");

}

**public** Rabbit() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Rabbit(**int** age, **int** numOfLegs) {

**super**(age, numOfLegs);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test06;

**public** **class** Tiger **extends** Carnivore {

@Override

**public** **void** eat() {

System.***out***.println("年龄为" + **this**.getAge() + "的" + **this**.getNumOfLegs() + "条腿的老虎在吃羊");

}

**public** Tiger() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Tiger(**int** age, **int** numOfLegs) {

**super**(age, numOfLegs);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test06;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Tiger tiger = **new** Tiger(2, 4);

tiger.eat();

Pecker pecker = **new** Pecker(1, 2);

pecker.eat();

Giraffe giraffe = **new** Giraffe(3, 4);

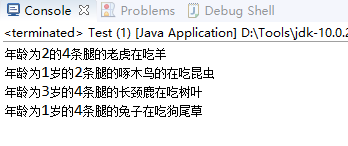
giraffe.eat();

Rabbit rabbit = **new** Rabbit(1, 4);

rabbit.eat();

}

}



## 训练案例6

### 训练考核知识点

继承格式

方法重写

子类调用父类的构造方法

抽象方法

抽象类

类名做参数类型

组合关系

### 训练描述

请用代码描述：

20岁1.75米的男孩慕容紫英有一个18岁1.6米瓜子脸女朋友欧阳青青

欧阳青青在洗一件白色的李宁牌子的衣服

慕容紫英去散步(和欧阳青青)

慕容紫英用带有阳刚之气的声音对欧阳青青说:我会守护你一生一世

欧阳青青微笑着用于甜美的声音对慕容紫英说:有你在我就安心.

要求: 男孩和女孩的共有成员提取到人类中,使用抽象类和抽象方法

### 操作步骤描述

1. 定义衣服类(Clothes)
   1. 成员变量:
      1. 颜色(color)
      2. 品牌(brand)
   2. 提供 无参,带参构造和setXxx和getXxx方法
2. 抽象类(Person)
   1. 成员变量
      1. 姓名 name
      2. 年龄 age
      3. 身高 height
   2. 成员方法
      1. 抽象方法 void say(String content)
   3. 要求
      1. 提供无参和带参构造
      2. 提供setXxx和getXxx方法

3. 定义Girl类,继承Person类

* 1. 成员变量
     1. 脸型(String face)
  2. 成员方法:
     1. 重写父类的 void say(String content) 方法
        1. 输出格式: 欧阳青青微笑着用于甜美的声音对慕容紫英说:有你在我就安心.
     2. 特有方法: 洗衣服 wash(Clothes c)

输出格式: 欧阳青青在洗一件白色的李宁牌子的衣服

* 1. 要求:
     1. 提供无参和带参构造
     2. 提供setXxx和getXxx方法

1. 定义Boy类,继承Person类
   1. 成员变量
      1. 女朋友 (girlFriend)
   2. 成员方法:
      1. 重写父类的 void say(String content) 方法

输出格式:容紫英用带有阳刚之气的声音对欧阳青青说:我会守护你一生一世

* + 1. 特有方法: 散步 walking()

输出格式: 慕容紫英和欧阳青青在河边的林荫小道上手牵着手散步

* 1. 要求:
     1. 提供无参和带参构造
     2. 提供setXxx和getXxx方法

5. 定义测试类Test

* 1. 提供main方法
  2. 在main方法中
     1. 创建衣服对象 c,品牌初始化为李宁,颜色初始化为白色
     2. 创建女孩对象 girl,姓名赋值为欧阳青青,年龄赋值为18,身高赋值为1.6,脸型赋值为瓜子脸
     3. 创建男孩对象 boy,姓名赋值为慕容紫英,年龄赋值以为20,身高赋值1.75,女朋友赋值为girl
     4. 调用对象girl的洗衣服方法
     5. 调用对象boy的散步方法
     6. 调用对象boy的说话方法,传入:”我会守护你一生一世”
     7. 调用对象 girl 的说话方法,传入: “有你在我就安心.”

**package** com.java.test07;

**public** **class** Clothes {

**private** String color;

**private** String brand;

**public** Clothes() {

**super**();

}

**public** Clothes(String color, String brand) {

**super**();

**this**.color = color;

**this**.brand = brand;

}

**public** String getColor() {

**return** color;

}

**public** **void** setColor(String color) {

**this**.color = color;

}

**public** String getBrand() {

**return** brand;

}

**public** **void** setBrand(String brand) {

**this**.brand = brand;

}

}

**package** com.java.test07;

**public** **class** Girl **extends** Person {

**private** String face;

@Override

**public** **void** say(String content) {

System.***out***.println(**this**.getName() + "微笑着用于甜美的声音对慕容紫英说:" + content);

}

**public** **void** wash(Clothes c) {

System.***out***.println(**this**.getName() + "在洗一件" + c.getColor() + "的" + c.getBrand() + "牌子的衣服");

}

**public** Girl() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Girl(String name, **int** age, **double** height, String face) {

**super**(name, age, height);

**this**.face = face;

// **TODO** Auto-generated constructor stub

}

**public** String getFace() {

**return** face;

}

**public** **void** setFace(String face) {

**this**.face = face;

}

}

**package** com.java.test07;

**public** **class** Boy **extends** Person {

**private** String girlFriend;

@Override

**public** **void** say(String content) {

System.***out***.println(**this**.getName() + "用带有阳刚之气的声音对" + **this**.girlFriend + "说:我会守护你一生一世");

}

**public** **void** walking() {

System.***out***.println(**this**.getName() + "和" + **this**.girlFriend + "在河边的林荫小道上手牵着手散步");

}

**public** Boy() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Boy(String name, **int** age, **double** height, String girlFriend) {

**super**(name, age, height);

**this**.girlFriend = girlFriend;

// **TODO** Auto-generated constructor stub

}

**public** String getGirlFriend() {

**return** girlFriend;

}

**public** **void** setGirlFriend(String girlFriend) {

**this**.girlFriend = girlFriend;

}

}

**package** com.java.test07;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Clothes clothes = **new** Clothes("白色", "李宁");

Girl girl = **new** Girl("欧阳青青", 18, 1.6, "瓜子脸");

Boy boy = **new** Boy("慕容紫英", 20, 1.75, "gir");

girl.wash(clothes);

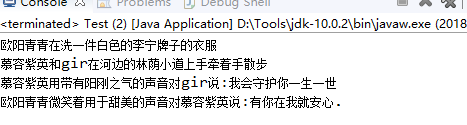
boy.walking();

boy.say("我会守护你一生一世");

girl.say("有你在我就安心.");

}

}



# 关卡3

## 训练案例1

### 训练考核知识点

继承格式

方法重写

子类调用父类的构造方法

### 训练描述

请用代码描述:

白色四条腿的北极熊(Bear)会吃(吃蜂蜜)和抓鱼(catchFish)

黑色四条腿的大熊猫(Panda)会吃(吃竹子)和爬树(climbTree)

要求: 把北极熊和大熊猫的共性提取动物类(Animal)中,不使用抽象类

**package** com.java.test08;

**public** **class** Animal {

**private** String color;

**private** String legNumber;

**public** **void** eat() {

}

**public** Animal() {

**super**();

}

**public** Animal(String color, String legNumber) {

**super**();

**this**.color = color;

**this**.legNumber = legNumber;

}

**public** String getColor() {

**return** color;

}

**public** **void** setColor(String color) {

**this**.color = color;

}

**public** String getLegNumber() {

**return** legNumber;

}

**public** **void** setLegNumber(String legNumber) {

**this**.legNumber = legNumber;

}

}

**package** com.java.test08;

**public** **class** Bear **extends** Animal {

@Override

**public** **void** eat() {

System.***out***.println(**this**.getColor() + **this**.getLegNumber() + "腿的北极熊会吃蜂蜜");

}

**public** **void** catchFish() {

System.***out***.println(**this**.getColor() + **this**.getLegNumber() + "腿的北极熊会抓鱼");

}

**public** Bear() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Bear(String color, String legNumber) {

**super**(color, legNumber);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test08;

**public** **class** Panda **extends** Animal {

@Override

**public** **void** eat() {

System.***out***.println(**this**.getColor() + **this**.getLegNumber() + "腿的大熊猫会吃竹子");

}

**public** **void** climbTree() {

System.***out***.println(**this**.getColor() + **this**.getLegNumber() + "腿的大熊猫会爬树");

}

**public** Panda() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Panda(String color, String legNumber) {

**super**(color, legNumber);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test08;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Bear bear = **new** Bear("白色", "四条");

bear.eat();

bear.catchFish();

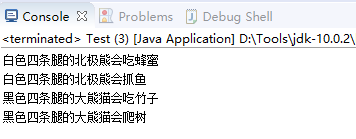
Panda panda = **new** Panda("黑色", "四条");

panda.eat();

panda.climbTree();

}

}



## 训练案例2

### 训练考核知识点

继承格式

方法重写

子类调用父类的构造方法

抽象类

抽象方法

### 训练描述

请使用代码描述:

4个轮子的白色的汽车可以跑(开着跑)和加油

2个轮子的红色电动自行车可以跑(骑着跑)和充电

要求: 把汽车和电动自行车的共性抽取到交通工具类中,使用抽象类和抽象方法

**package** com.java.test09;

**public** **abstract** **class** Vehicle {

**private** String color;

**private** **int** numner;

**public** **abstract** **void** run();

**public** Vehicle() {

**super**();

}

**public** Vehicle(String color, **int** numner) {

**super**();

**this**.color = color;

**this**.numner = numner;

}

**public** String getColor() {

**return** color;

}

**public** **void** setColor(String color) {

**this**.color = color;

}

**public** **int** getNumner() {

**return** numner;

}

**public** **void** setNumner(**int** numner) {

**this**.numner = numner;

}

}

**package** com.java.test09;

**public** **class** Car **extends** Vehicle {

@Override

**public** **void** run() {

System.***out***.println(**this**.getNumner() + "个轮子的" + **this**.getColor() + "的汽车可以开着跑");

}

**public** **void** jiaYou() {

System.***out***.println(**this**.getNumner() + "个轮子的" + **this**.getColor() + "的汽车可以加油");

}

**public** Car() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Car(String color, **int** numner) {

**super**(color, numner);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test09;

**public** **class** Bicycle **extends** Vehicle {

@Override

**public** **void** run() {

System.***out***.println(**this**.getNumner() + "个轮子的" + **this**.getColor() + "电动自行车可以骑着跑");

}

**public** **void** chongDian() {

System.***out***.println(**this**.getNumner() + "个轮子的" + **this**.getColor() + "电动自行车可以充电");

}

**public** Bicycle() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Bicycle(String color, **int** numner) {

**super**(color, numner);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test09;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Car car = **new** Car("白色", 4);

car.run();

car.jiaYou();

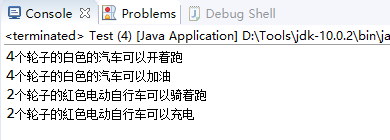
Bicycle bicycle = **new** Bicycle("紅色", 2);

bicycle.run();

bicycle.chongDian();

}

}



## 训练案例3

### 训练考核知识点

继承格式

方法重写

子类调用父类的构造方法

抽象类

抽象方法

### 训练描述

请使用代码描述

在极客营有很多员工(Employee),按照工作内容不同分教研部员工(Teacher)和行政部员工(AdminStaff)

教研部根据教学的方式不同又分为讲师(Lecturer)和助教(Tutor)

行政部根据负责事项不同,有分为维护专员(Maintainer),采购专员(Buyer)

公司的每一个员工都编号,姓名和其负责的工作

工作内容:

讲师: 工号为 666 的讲师 傅红雪 在讲课

助教: 工号为 668的助教 顾棋 在帮助学生解决问题

维护专员: 工号为 686 的维护专员 庖丁 在解决不能共享屏幕问题

采购专员: 工号为 888 的采购专员 景甜 在采购音响设备

提示: 参考今天的综合案例

**package** com.java.test10;

**public** **abstract** **class** Employee {

**private** **int** id;

**private** String name;

**public** **abstract** **void** work();

**public** Employee() {

**super**();

}

**public** Employee(**int** id, String name) {

**super**();

**this**.id = id;

**this**.name = name;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

}

**package** com.java.test10;

**public** **abstract** **class** AdminStaff **extends** Employee {

**public** AdminStaff() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** AdminStaff(**int** id, String name) {

**super**(id, name);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test10;

**public** **abstract** **class** Teacher **extends** Employee {

**public** Teacher() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Teacher(**int** id, String name) {

**super**(id, name);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test10;

**public** **class** JiangShi **extends** Teacher {

@Override

**public** **void** work() {

System.***out***.println("工号为" + **this**.getId() + "的讲师" + **this**.getName() + "在讲课");

}

**public** JiangShi() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** JiangShi(**int** id, String name) {

**super**(id, name);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test10;

**public** **class** ZhuJiao **extends** Teacher {

@Override

**public** **void** work() {

System.***out***.println("工号为" + **this**.getId() + "的助教" + **this**.getName() + "在帮助学生解决问题");

}

**public** ZhuJiao() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** ZhuJiao(**int** id, String name) {

**super**(id, name);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test10;

**public** **class** CaiGouZhuanYuan **extends** AdminStaff {

@Override

**public** **void** work() {

System.***out***.println("工号为" + **this**.getId() + "的采购专员" + **this**.getName() + "在采购音响设备");

}

**public** CaiGouZhuanYuan() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** CaiGouZhuanYuan(**int** id, String name) {

**super**(id, name);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test10;

**public** **class** WeiHuZhuanYuan **extends** AdminStaff {

@Override

**public** **void** work() {

System.***out***.println("工号为" + **this**.getId() + "的维护专员" + **this**.getName() + "在解决不能共享屏幕问题");

}

**public** WeiHuZhuanYuan() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** WeiHuZhuanYuan(**int** id, String name) {

**super**(id, name);

// **TODO** Auto-generated constructor stub

}

}

**package** com.java.test10;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

JiangShi jiangShi = **new** JiangShi(666, "傅红雪");

jiangShi.work();

ZhuJiao zhuJiao = **new** ZhuJiao(668, "顾棋");

zhuJiao.work();

WeiHuZhuanYuan weiHuZhuanYuan = **new** WeiHuZhuanYuan(686, "庖丁");

weiHuZhuanYuan.work();

CaiGouZhuanYuan caiGouZhuanYuan = **new** CaiGouZhuanYuan(888, "景甜");

caiGouZhuanYuan.work();

}

}

