**中国矿业大学计算机学院**

**2019 级本科生课程报告**

课程名称 Java语言与网络编程

学生姓名 李 春 阳

学 号 10193657

班 级 信息安全2019-1班

任课教师 张 爱 娟

报告时间 2021.12.5

目录

[1. 问题一 3](#_Toc89590656)

[1.1 问题描述 3](#_Toc89590657)

[1.2 问题分析 3](#_Toc89590658)

[1.3 代码展示 4](#_Toc89590659)

[1.4 运行结果 5](#_Toc89590660)

[2. 问题二 5](#_Toc89590661)

[2.1 问题描述 5](#_Toc89590662)

[2.2 问题分析 6](#_Toc89590663)

[2.3 代码展示 6](#_Toc89590664)

[2.4 运行结果 6](#_Toc89590665)

[3. 问题三 7](#_Toc89590666)

[3.1 问题描述 7](#_Toc89590667)

[3.2 问题分析 7](#_Toc89590668)

[3.3 代码展示 8](#_Toc89590669)

[3.4 运行结果 12](#_Toc89590670)

## 1. 问题一

### 1.1 问题描述

读程序，写结果：

|  |
| --- |
| 1. class A { 2. public String Show(D obj) { return ("A and D"); } 3. public String Show(A obj) { return ("A and A"); } 4. } 5. class B extends A { 6. public String Show(B obj) { return ("B and B"); } 7. public String Show(A obj) { return ("B and A"); } 8. } 9. class C extends B { 10. public String Show(C obj) { return ("C and C"); } 11. public String Show(B obj) { return ("C and B"); } 12. } 13. class D extends B { 14. public String Show(D obj) { return ("D and D"); } 15. public String Show(B obj) { return ("D and B"); } 16. } 17. public class mainTest { 18. public static void main(String args[]){ 19. A a1 = new A(); 20. A a2 = new B(); 21. B b = new B(); 22. C c = new C(); 23. D d = new D(); 24. System.out.println(a1.Show(b)); 25. System.out.println(a1.Show(c)); 26. System.out.println(a1.Show(d)); 27. System.out.println(a2.Show(b)); 28. System.out.println(a2.Show(c)); 29. System.out.println(a2.Show(d)); 30. System.out.println(b.Show(b)); 31. System.out.println(b.Show(c)); 32. System.out.println(b.Show(d)); 33. } 34. } |

### 1.2 问题分析

#### 动态链编规则

程序运行并且采用动态绑定调用方法时，虚拟机必须调用与ⅹ所引用对象的实际类型对应的那个方法。假设 x的实际类型是D，它是C类的子类。如果D类定义了方法 f(String),就会调用这个方法；否则，将在D类的超类中寻找f(String)，以此类推。

调用方法时先在本身的类寻找方法，找不到则在父类中寻找。

#### 寻找方法的顺序

根据编译时类型:首先在本类中寻找方法,若未找到则到超类中寻找,若还是未找到,则根据向上转型,现在本类中寻找,未找到则在超类中寻找.

根据运行时类型:根据编译时确定的方法,在运行时类型的类中寻找匹配的方法,此时若该方法被继承则结果与执行父类的该方法一致,若该方法被重写则执行子类中重写的方法.

### 1.3 代码展示

|  |
| --- |
| 1. class A { 2. public String Show(D obj) { return ("A and D"); } 3. public String Show(A obj) { return ("A and A"); } 4. } 5. class B extends A { 6. public String Show(B obj) { return ("B and B"); } 7. public String Show(A obj) { return ("B and A"); } 8. } 9. class C extends B { 10. public String Show(C obj) { return ("C and C"); } 11. public String Show(B obj) { return ("C and B"); } 12. } 13. class D extends B { 14. public String Show(D obj) { return ("D and D"); } 15. public String Show(B obj) { return ("D and B"); } 16. } 17. public class mainTest { 18. public static void main(String args[]){ 19. A a1 = new A(); 20. A a2 = new B(); 21. B b = new B(); 22. C c = new C(); 23. D d = new D(); 24. System.out.println(a1.Show(b)); 25. System.out.println(a1.Show(c)); 26. System.out.println(a1.Show(d)); 27. System.out.println(a2.Show(b)); 28. System.out.println(a2.Show(c)); 29. System.out.println(a2.Show(d)); 30. System.out.println(b.Show(b)); 31. System.out.println(b.Show(c)); 32. System.out.println(b.Show(d)); 33. } 34. } |

### 1.4 运行结果



## 2. 问题二

### 2.1 问题描述

读程序，写结果：

|  |
| --- |
| 1. class Base { 2. private String name = "base"; 3. public Base() { 4. tellName();} 5. public void tellName() { 6. System.out.println("Base tell name: " + name); } 7. } 8. public class Dervied extends Base { 9. private String name = "dervied"; 10. public Dervied() { 11. tellName();} 12. public void tellName() { 13. System.out.println("Dervied tell name: " + name);} 14. public static void main(String[] args){ 15. new Dervied(); 16. } 17. } |

### 2.2 问题分析

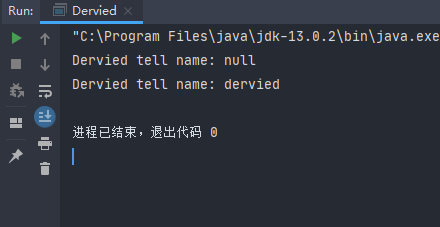
静态方法和静态变量、成员变量、构造函数 , 这里的成员变量和构造函数是指同一个类中的顺序 , 比如此题中先初始化父类的成员变量然后是父类构造函数 , 此时子类的成员变量还未初始化。

对于静态变量、静态初始化块、变量、初始化块、构造器，它们的初始化顺序依次是（静态变量、静态初始化块）>（变量、初始化块）>构造器。

### 2.3 代码展示

|  |
| --- |
| 1. class Base { 2. private String name = "base"; 3. public Base() { 4. tellName();} 5. public void tellName() { 6. System.out.println("Base tell name: " + name); } 7. } 8. public class Dervied extends Base { 9. private String name = "dervied"; 10. public Dervied() { 11. tellName();} 12. public void tellName() { 13. System.out.println("Dervied tell name: " + name);} 14. public static void main(String[] args){ 15. new Dervied(); 16. } 17. } |

### 2.4 运行结果



## 3. 问题三

### 3.1 问题描述

生成动物

1. 循环通过标准输入端输入需要生成的动物，当遇到结束标志，则结束程序运行。

2. 每次生成动物，通过标准输出端显示动物的信息。

3. 动物的信息包括：目前所有动物的总数，当前这一类动物的总数。

4. 整个程序结构用工厂模式设计，保证将来动物园有新的动物加入时，程序可扩展。

### 3.2 问题分析

#### 工厂模式

工厂模式（Factory Pattern）是 Java 中最常用的设计模式之一。这种类型的设计模式属于创建型模式，它提供了一种创建对象的最佳方式。

在工厂模式中，我们在创建对象时不会对客户端暴露创建逻辑，并且是通过使用一个共同的接口来指向新创建的对象。

#### 介绍

意图：定义一个创建对象的接口，让其子类自己决定实例化哪一个工厂类，工厂模式使其创建过程延迟到子类进行。

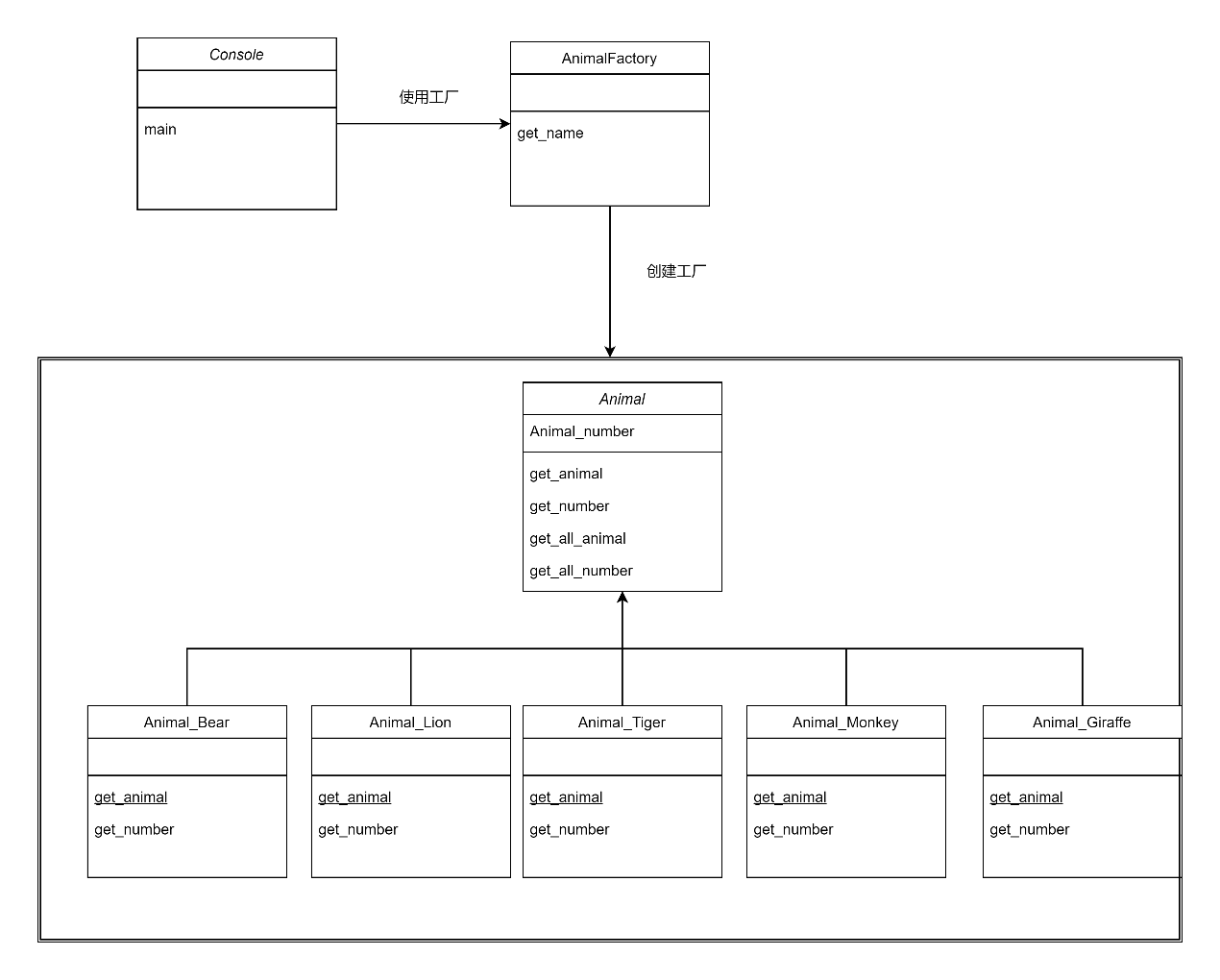
主要解决：主要解决接口选择的问题。

何时使用：我们明确地计划不同条件下创建不同实例时。

如何解决：让其子类实现工厂接口，返回的也是一个抽象的产品。

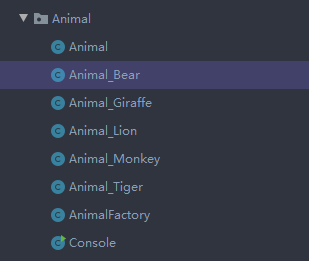
关键代码：创建过程在其子类执行。

下面是设计类图：



### 3.3 代码展示

#### Animal 包文件：



#### Console

|  |
| --- |
| 1. package Animal; 2. import java.util.Scanner; 3. public class Console { 4. public static void main(String[] args) { 5. Scanner scan = new Scanner(System.in); 6. System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*欢迎来到lowly-li的动物园!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); 7. while (true) { 8. System.out.println("请选择你的操作：\t[0]退出\t[1]添加动物\t[2]查看全部动物"); 9. int option = scan.nextInt(); 10. if(option == 0){ 11. System.out.println("欢迎下次光临动物园"); 12. break; 13. } 14. else if(option == 1){ 15. *//循环添加：* 16. while (true){ 17. System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); 18. System.out.println("请选择添加的动物：\t[0]结束添加\t[1]熊\t[2]狮子\t[3]老虎\t[4]猴子\t[5]长颈鹿"); 19. int animal\_index = scan.nextInt(); 20. if (animal\_index == 0){ 21. break; 22. } 23. Animal animal = AnimalFactory.get\_name(animal\_index); 24. if(animal != null){ 25. System.out.println(animal.get\_animal()); 26. System.out.println(animal.get\_number()); 27. } 28. else{ 29. System.out.println("其他动物还未引进，请重新输入"); 30. } 31. } 32. } 33. else if(option == 2){ 34. Animal animal = new Animal(); 35. System.out.println("动物园的动物有："); 36. System.out.println(animal.get\_all\_animal()); 37. animal.get\_all\_number(); 38. } 39. else { 40. System.out.println("输入错误，请重新输入"); 41. } 42. System.out.println(); 43. System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); 44. } 45. scan.close(); 46. } 47. } |

#### AnimalFactory

|  |
| --- |
| 1. package Animal; 2. public class AnimalFactory { 3. public static Animal get\_name(int index){ 4. Animal animal = null; 5. if(index == 1){ 6. animal = new Animal\_Bear(); 7. } 8. else if(index == 2){ 9. animal = new Animal\_Lion(); 10. } 11. else if(index == 3){ 12. animal = new Animal\_Tiger(); 13. } 14. else if(index == 4){ 15. animal = new Animal\_Monkey(); 16. } 17. else if(index == 5){ 18. animal = new Animal\_Giraffe(); 19. } 20. return animal; 21. } 22. } |

#### Animal

|  |
| --- |
| 1. package Animal; 2. public class Animal { 3. static int bear=0,lion=0,tiger=0,monkey=0,giraffe=0; 4. String get\_animal(){ 5. return null; 6. } 7. String get\_number(){ 8. return null; 9. } 10. String get\_all\_animal(){ 11. String names = ""; 12. if(bear > 0){ 13. names = names + "熊,"; 14. } 15. if(lion > 0){ 16. names = names + "狮子,"; 17. } 18. if(tiger > 0){ 19. names = names + "老虎,"; 20. } 21. if(monkey > 0){ 22. names = names + "猴子,"; 23. } 24. if(giraffe > 0){ 25. names = names + "长颈鹿,"; 26. } 27. return names; 28. } 29. void get\_all\_number(){ 30. System.out.println("动物园的动物及其数量有："); 31. System.out.println("熊的数量:"+ bear); 32. System.out.println("狮子的数量:"+ lion); 33. System.out.println("老虎的数量:"+ tiger); 34. System.out.println("猴子的数量:"+ monkey); 35. System.out.println("长颈鹿的数量:"+ giraffe); 36. } 37. } |

#### Animal\_Bear

|  |
| --- |
| 1. package Animal; 2. public class Animal\_Bear extends Animal{ 3. public String get\_animal(){ 4. return "这是一只熊"; 5. } 6. public String get\_number(){ 7. bear++; 8. return "熊的数量:"+ bear; 9. } 10. } |

#### Animal\_Lion

|  |
| --- |
| 1. package Animal; 2. public class Animal\_Lion extends Animal{ 3. public String get\_animal(){ 4. return "这是一只狮子"; 5. } 6. public String get\_number(){ 7. lion++; 8. return "狮子的数量:"+ lion; 9. } 10. } |

#### Animal\_Tiger

|  |
| --- |
| 1. package Animal; 2. public class Animal\_Tiger extends Animal { 3. public String get\_animal(){ 4. return "这是一只老虎"; 5. } 6. public String get\_number(){ 7. tiger++; 8. return "老虎的数量:"+ tiger; 9. } 10. } |

#### Animal\_Monkey

|  |
| --- |
| 1. package Animal; 2. public class Animal\_Monkey extends Animal { 3. public String get\_animal(){ 4. return "这是一只猴子"; 5. } 6. public String get\_number(){ 7. monkey++; 8. return "猴子的数量:"+ monkey; 9. } 10. } |

#### Animal\_Giraffe

|  |
| --- |
| 1. package Animal; 2. public class Animal\_Giraffe extends Animal { 3. public String get\_animal(){ 4. return "这是一只长颈鹿"; 5. } 6. public String get\_number(){ 7. giraffe++; 8. return "长颈鹿的数量:"+ giraffe; 9. } 10. } |

### 3.4 运行结果

|  |
| --- |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*欢迎来到lowly-li的动物园!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  请选择你的操作： [0]退出 [1]添加动物 [2]查看全部动物  1  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  请选择添加的动物： [0]结束添加 [1]熊 [2]狮子 [3]老虎 [4]猴子 [5]长颈鹿  1  这是一只熊  熊的数量:1  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  请选择添加的动物： [0]结束添加 [1]熊 [2]狮子 [3]老虎 [4]猴子 [5]长颈鹿  2  这是一只狮子  狮子的数量:1  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  请选择添加的动物： [0]结束添加 [1]熊 [2]狮子 [3]老虎 [4]猴子 [5]长颈鹿  3  这是一只老虎  老虎的数量:1  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  请选择添加的动物： [0]结束添加 [1]熊 [2]狮子 [3]老虎 [4]猴子 [5]长颈鹿  4  这是一只猴子  猴子的数量:1  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  请选择添加的动物： [0]结束添加 [1]熊 [2]狮子 [3]老虎 [4]猴子 [5]长颈鹿  5  这是一只长颈鹿  长颈鹿的数量:1  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  请选择添加的动物： [0]结束添加 [1]熊 [2]狮子 [3]老虎 [4]猴子 [5]长颈鹿  6  其他动物还未引进，请重新输入  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  请选择添加的动物： [0]结束添加 [1]熊 [2]狮子 [3]老虎 [4]猴子 [5]长颈鹿  4  这是一只猴子  猴子的数量:2  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  请选择添加的动物： [0]结束添加 [1]熊 [2]狮子 [3]老虎 [4]猴子 [5]长颈鹿  2  这是一只狮子  狮子的数量:2  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  请选择添加的动物： [0]结束添加 [1]熊 [2]狮子 [3]老虎 [4]猴子 [5]长颈鹿  2  这是一只狮子  狮子的数量:3  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  请选择添加的动物： [0]结束添加 [1]熊 [2]狮子 [3]老虎 [4]猴子 [5]长颈鹿  0  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  请选择你的操作： [0]退出 [1]添加动物 [2]查看全部动物  2  动物园的动物有：  熊,狮子,老虎,猴子,长颈鹿,  动物园的动物及其数量有：  熊的数量:1  狮子的数量:3  老虎的数量:1  猴子的数量:2  长颈鹿的数量:1  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  请选择你的操作： [0]退出 [1]添加动物 [2]查看全部动物  0  欢迎下次光临动物园  进程已结束，退出代码 0 |

