**面向对象作业**

1. Q: Java的对象如何算相同 – 举出一个场景，你必须改写现有类库的 equals方法?

**A：默认情况下也就是从超类Object继承而来的equals方法与‘==’是完全等价的，比较的都是对象的内存地址，但我们可以重写equals方法，使其按照我们的需求的方式进行比较，如String类重写了equals方法，使其比较的是字符的序列，而不再是内存地址。在第三次作业答案中已经说明对象相等两种情况，一种两个对象的内存地址是否相等，另一种是两个对象的内容是否相等。因此可以通过重写equals方法来实现我们的需求：**

**首先新建一个类Car,简单比较equals和==的输出结果：**

1. **package com.test;**
2. **public class Car {**
3. **private int batch;**
4. **public Car(int batch) {**
5. **this.batch = batch;**
6. **}**
7. **public static void main(String[] args) {**
8. **Car c1 = new Car(1);**
9. **Car c2 = new Car(1);**
10. **System.out.println(c1.equals(c2));**
11. **System.out.println(c1 == c2);**
12. **}**
13. **}**

**运行结果：false，false。因此重写Car类的equals方法**

1. @Override
2. public boolean equals(Object obj) {
3. if (obj instanceof Car) {
4. Car c = (Car) obj;
5. return batch == c.batch;
6. }
7. return false;
8. }

**再次运行：true，false。使用instanceof来判断引用obj所指向的对象的类型，如果obj是Car类对象，就可以将其强制转为Car对象，然后比较两辆Car的batch，相等返回true不相等返回false。当然如果obj不是 Car对象，自然也得返回false。**

1. Q: 总结JavaScript语言的面向对象特征，你认为JavaScript(是/否)归属于面向对象语言的理由是什么？

**A:** **JavaScript面向对象拥有三大特征，分别为封装、继承、多态，但实际上在javaScript脚本语言中是不存在多态的，但是可以用JavaScript的方式实现多态中的两种效果重载、重写。**

**封装：把抽象出的属性和对属性的方法封装在一起对外实现接口开放。代码举例：**

1. class Package {
2. constructor (animal){
3. this.animal = animal
4. }
5. zoo (animal) {
6. console.log('this is'+ this.animal)
7. }
8. static private () {
9. console.log('我是私有方法')
10. }
11. }
12. let newPackage = new Package('大象')
13. newPackage.zoo()
14. newPackage.private()

**通过animal参数传递来达到我们想要的结果，但是class类里面的静态方法是不会对外开放的所以会找不到这个函数。**

**继承:** **继承可以使得子类具有父类别的各种属性和方法，而不需要再次编写相同的代码。**

1. class PackageSon extends Package{
2. constructor(animal, food){
3. super(animal);
4. this.food = food
5. }
6. zoo () {
7. console.log('这是'+ this.animal)
8. }
9. eat () {
10. console.log(this.animal+this.food)
11. }
12. }
13. let newPackageSon = new PackageSon('大象', '吃草')
14. newPackageSon.zoo()
15. newPackageSon.eat()

**例子使用上面**Package**的super方法，其实它是实现了在这里指向了父类中的this对象，然后子类的构造函数再对其进行修改，然后zoo函数实现了对父类的重写，eat函数是子类新增的函数**

**多态：多态可以表现出我们的代码的多种状态，同一操作作用于不同的对象，可以有不同的解释，产生不同的执行结果。**

**重载举例：**

1. class overload {
2. init (callback) {
3. if (callback === 'go') {
4. console.log('我是go')
5. }else if(callback === 'eat') {
6. console.log('我是eat')
7. }else {
8. console.log('我是sprot')
9. }
10. }
11. }
13. var newOverload = new overload()
14. newOverload.init('go')
15. newOverload.init('eat')
16. newOverload.init('sprot')

**重写举例：**

1. class rewrite {
2. go () {
3. console.log('我在走路')
4. }
5. sport () {
6. console.log('我在运动')
7. }
8. }
9. class rewriteSon extends rewrite{
10. go () {
11. console.log('我回家了')
12. }
13. }
14. var newRewriteSon = new rewriteSon()
15. newRewriteSon.go()
16. newRewriteSon.sport()

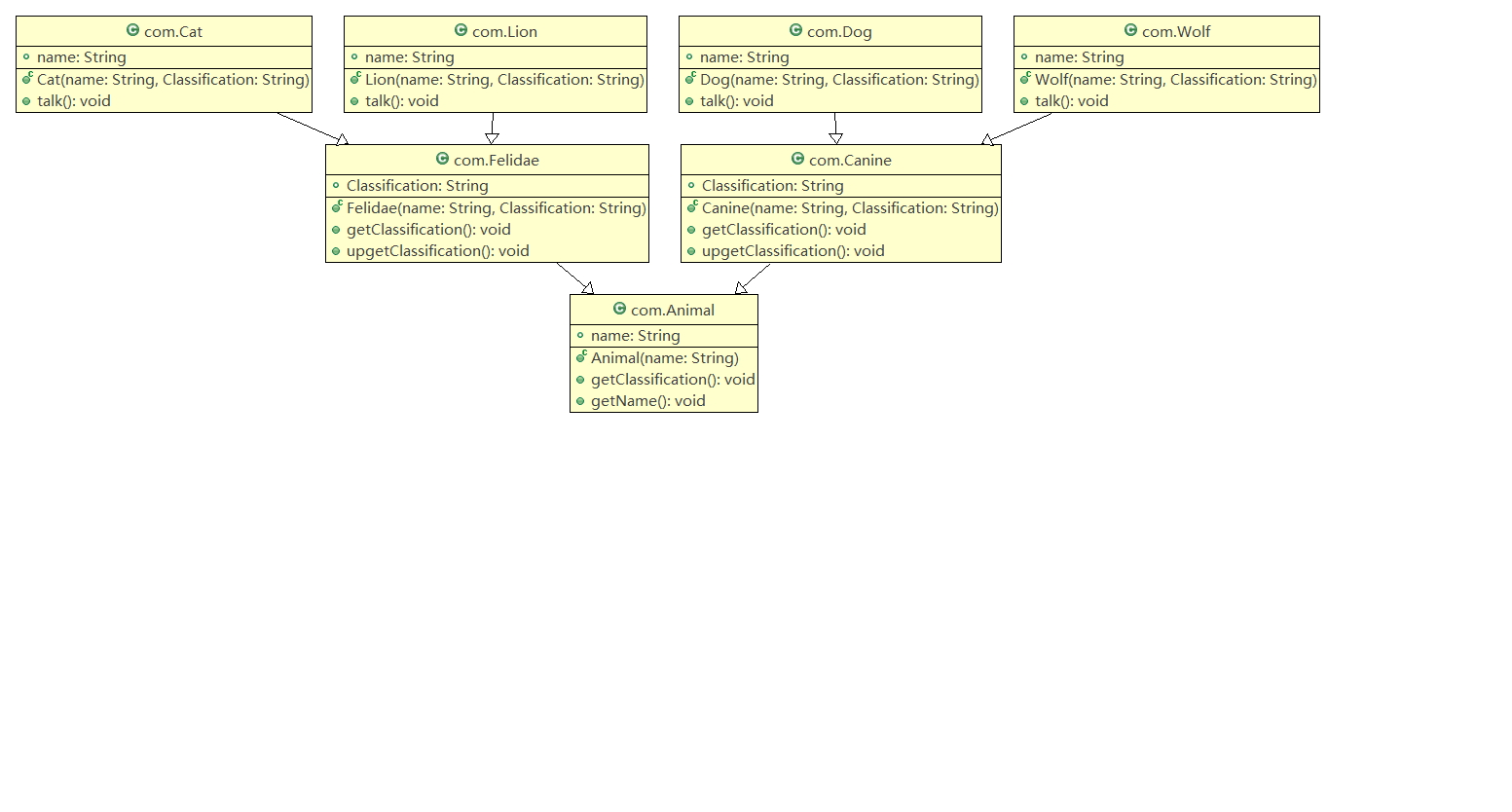
**因此综上所述，JavaScript是归属于面向对象语言，它能够按照自己的规则实现面向对象的三个特点：封装、继承、多态。**

1. Q：Java 类的作用域—class TalkingClock是一个类，class TimePrinter是一个类，为什么TimePrinter可以使用 TalkingClock的私有变量，请分析这么使用的潜在安全风险。

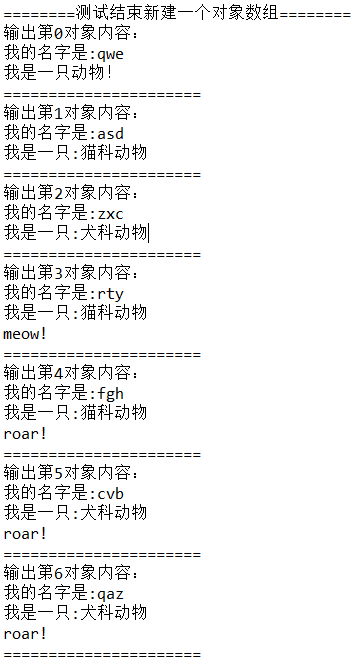
**A：内部类TimePrinter既可以访问自身的数据域，也可以访问创建它的外围类TalkingClock对象的数据域。当使用了内部类的时候，编译器做了这样一件事：它在外围类添加了一个静态方法 static boolean access$0(外部类)。内部类方法将调用这个函数这个是有风险的，因为任何人都可以通过access$0方法很容易的读取到外围类的私有域，黑客可以使用十六进制编辑器轻松创建一个用虚拟机指令调用这个函数的类文件。总而言之如果内部类访问了私有数据域，就有可能通过附加在外围类所在的包中的其他类访问它们。**

1. Q：多态作业

A：1.类图如下所示，首先我实现了Animal这个父类，Animal这个类里面带有getName()和getClassification()两个方法，然后两个子类Felidae和Canine继承Animal,重写了getClassification()这个方法，并且用upgetClassification()调用父类被覆盖的方法，最后Cat和Lion两个子类继承Felidae，Dog和Wolf继承Canine，在四个新类里面新定义一个talk()方法。



2.程序运行结果：



3.程序源码：

**package** com;

**class** Animal{

**public** String name;

**public** Animal(String name) {

**this**.name = name;

}

**public** **void** getClassification() {

System.***out***.println("我是一只动物！");

}

**public** **void** getName() {

System.***out***.println("我的名字是:" + name);

}

}

**class** Felidae **extends** Animal{

**public** String Classification;

**public** Felidae(String name,String Classification) {

**super**(name);

**this**.Classification = Classification;

}

**public** **void** getClassification() {

System.***out***.println("我是一只:" + Classification);

}

**public** **void** upgetClassification() {

**super**.getClassification();

}

}

**class** Canine **extends** Animal{

**public** String Classification;

**public** Canine(String name,String Classification) {

**super**(name);

**this**.Classification = Classification;

}

**public** **void** getClassification() {

System.***out***.println("我是一只:" + Classification);

}

**public** **void** upgetClassification() {

**super**.getClassification();

}

}

**class** Cat **extends** Felidae{

**public** String name;

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

**super**(name,Classification);

}

**public** **void** talk() {

System.***out***.println("meow!");

}

}

**class** Lion **extends** Felidae{

**public** String name;

**public** Lion(String name,String Classification) {

**super**(name,Classification);

}

**public** **void** talk() {

System.***out***.println("roar!");

}

}

**class** Dog **extends** Canine{

**public** String name;

**public** Dog(String name,String Classification) {

**super**(name,Classification);

}

**public** **void** talk() {

System.***out***.println("roar!");

}

}

**class** Wolf **extends** Canine{

**public** String name;

**public** Wolf(String name,String Classification) {

**super**(name,Classification);

}

**public** **void** talk() {

System.***out***.println("roar!");

}

}

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

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

// **TODO** Auto-generated method stub

Animal test = **new** Animal("lubenwei");

test.getName();

test.getClassification();

System.***out***.println("================");

Felidae F = **new** Felidae("uzi","猫科动物");

F.getClassification();

F.getName();

System.***out***.println("================");

Cat Q = **new** Cat("doinb","猫科动物");

Q.getClassification();

Q.getName();

Q.talk();

System.***out***.println("========测试结束新建一个对象数组========");

//为了运用多态的特性我们新建一个Animal[]类的数组然后对每个对象数组

//赋值不同对象的构造器，之后要调用talk()运用类型转换即可

Animal[] obj;

obj = **new** Animal[7];

Animal test0 = **new** Animal("qwe");

Animal test1 = **new** Felidae("asd","猫科动物");

Animal test2 = **new** Canine("zxc","犬科动物");

Animal test3 = **new** Cat("rty","猫科动物");

Animal test4 = **new** Lion("fgh","猫科动物");

Animal test5 = **new** Dog("cvb","犬科动物");

Animal test6 = **new** Wolf("qaz","犬科动物");

obj[0] = test0;

obj[1] = test1;

obj[2] = test2;

obj[3] = test3;

obj[4] = test4;

obj[5] = test5;

obj[6] = test6;

**for**(**int** i = 0;i <7; i++) {

System.***out***.println("输出第" + i + "对象内容：");

obj[i].getName();

obj[i].getClassification();

//假如是如下cat、lion、dog、wolf类的话输出talk()内容

**if**(obj[i] **instanceof** Cat) {

Cat object = (Cat) obj[i];

object.talk();

}**else** **if**(obj[i] **instanceof** Lion) {

Lion object = (Lion) obj[i];

object.talk();

}**else** **if**(obj[i] **instanceof** Dog) {

Dog object = (Dog) obj[i];

object.talk();

}**else** **if**(obj[i] **instanceof** Wolf) {

Wolf object = (Wolf) obj[i];

object.talk();

}

System.***out***.println("======================");

}

}

}

1. Q:Python语言作业题C3算法

A：图一的MRO列表：A->B->D->C->E

图二的MRO列表：A->B->C->D