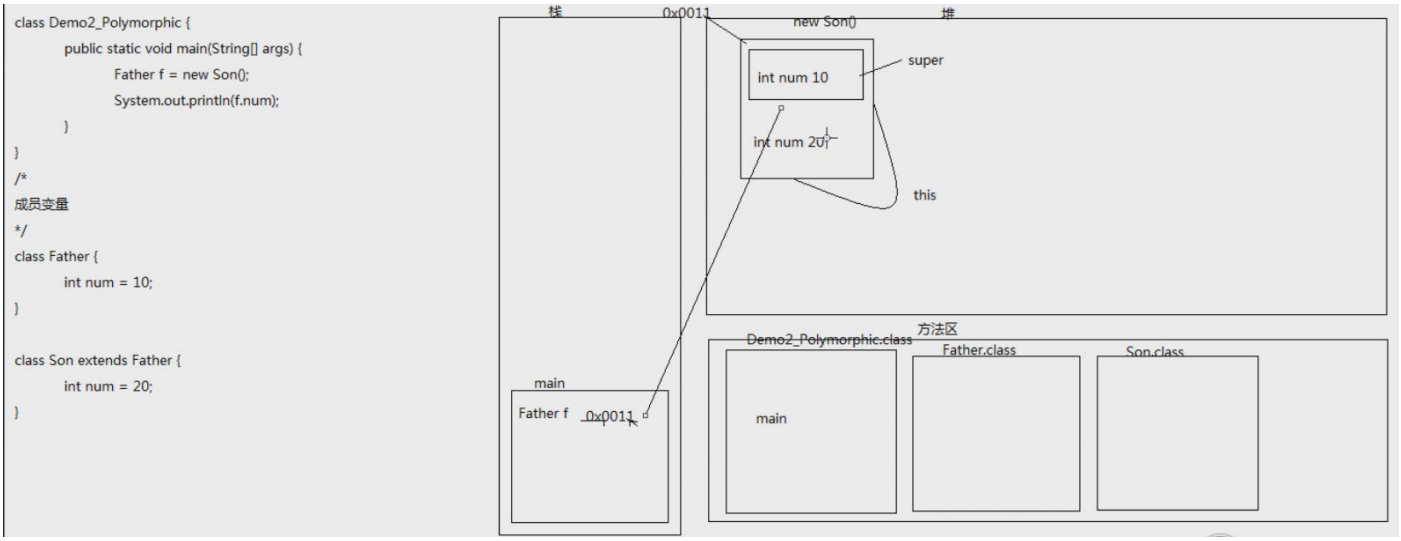
●09.01_面向对象(多态的概述及其代码体现)

- A:多态(polymorphic)概述
    - 事物存在的多种形态
- B:多态前提
    - a:要有继承关系。
    - b:要有方法重写。
    - c:要有父类引用指向子类对象。
- C:案例演示
    - 代码体现多态

```
public class Demo9 {
    public static void main(String[] args) {
        Animal c = new Cat();//父类引用指向子类的对象叫多态
      c.eat();//Cannot resolve method 'eat' in 'Animal'注释掉父类中的eat 方法之后,发现编译通不过,
        /*
        结论:在多态中编译的时候会检查父类中是否有被重写的eat 方法
          编译看右边(父类)
             运行看左边(子类)
         */
    }
}
class Animal{
    String color="黑色";
    public void eat(){
        System.out.println("动物在吃东西,"+color);
    }
}
//继承
class Cat extends Animal{
    String color = "白色";
    public void eat(){
        System.out.println("猫吃鱼,"+color);
    }
}
```

●09.02_面向对象(多态中的成员访问特点之成员变量)

```java
class Demo2_Polymorphic {
    public static void main(String[] args) {
        Father f = new Son();                    //父类引用指向子类对象
        System.out.println(f.num);
    }
}
/*
成员变量
*/
class Father {
    int num = 10;
}

class Son extends Father {
    int num = 20;
}
```



- 成员变量
  - (编译看父类，运行看父类)(和普通的继承一样)

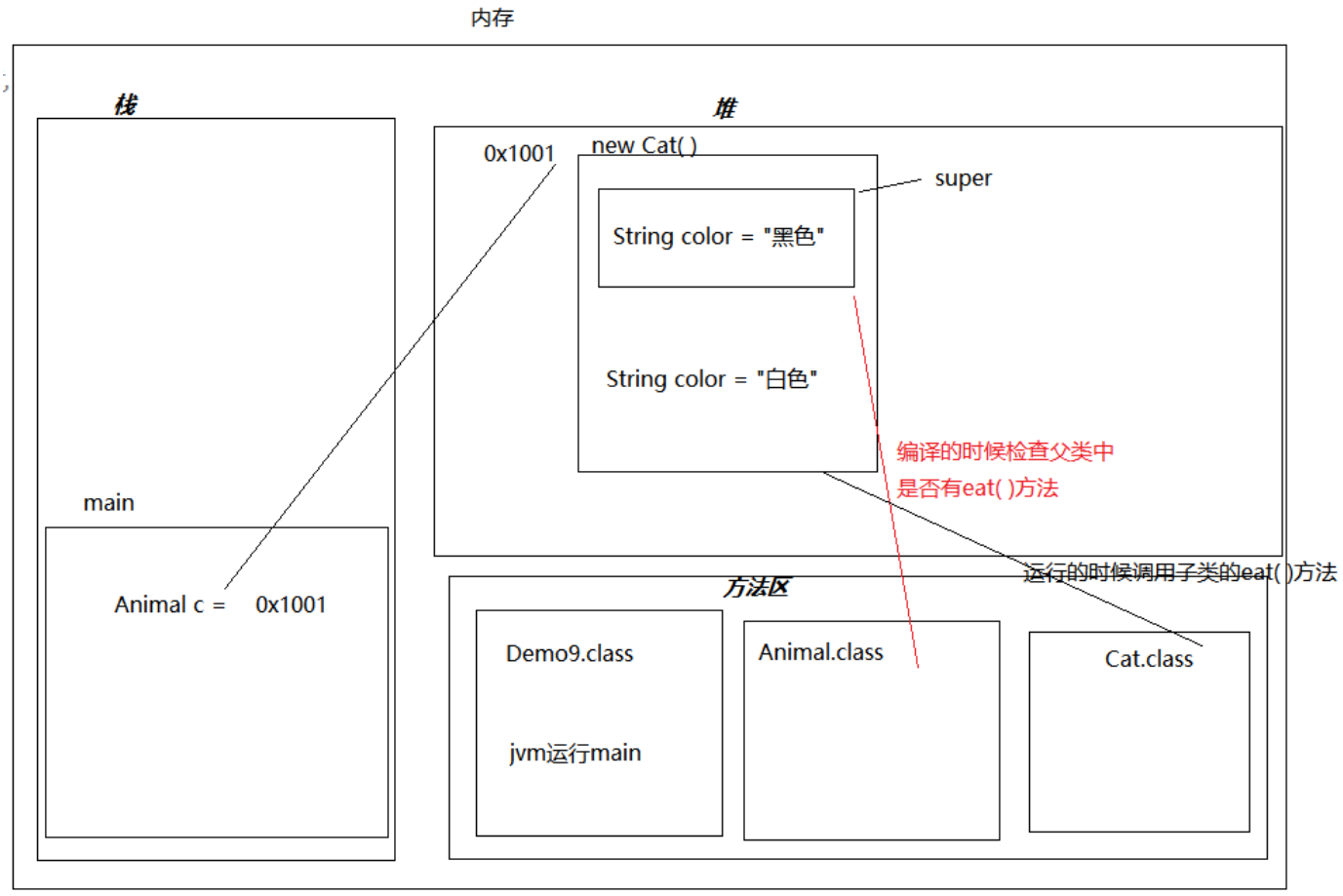
●09.03_面向对象(多态中的成员访问特点之成员方法)

```
成员方法
编译看左边(父类)，运行看右边(子类)。动态绑定
*/
class Father {
    int num = 10;
    public void print() {
        System.out.println("father");
    }
}

class Son extends Father {
    int num = 20;

    public void print() {
        System.out.println("son");
    }
}
```

```
Father f = new Son();
//f.print();
```

- 成员方法
    - 编译看左边(父类)，运行看右边(子类)。

出现动态绑定



多态有动态绑定的现象

●09.04_面向对象(多态中的成员访问特点之静态成员方法)

```
class Father {
    int num = 10;
    public void print() {
        System.out.println("father");
    }

    public static void method() {
        System.out.println("father static method");
    }
}

class Son extends Father {
    int num = 20;

    public void print() {
        System.out.println("son");
    }

    public static void method() {
        System.out.println("father static method");
    }
}
```

```
Father f = new Son();
//f.print();
f.method();
```

执行父类的

相当于 father 的 method 方法

静态方法

- 编译看左边(父类)，运行看左边(父类)。
- (静态和类相关，算不上重写，所以，访问还是左边的)，当普通类里的情况来理解
- 只有非静态的成员方法,编译看左边,运行看右边

●09.05_面向对象(钢铁侠的故事)

- A:案例分析
  - 多态的现象

```
public class Demo10 {
    public static void main(String[] args) {
        Person p = new IronMan();
        p.business();
        p.saveWorld();
```

```
//        p.paoNiu();
        //我们发现了一个问题,在多态中,父类的引用不能调到子类自己特有的方法,怎么解决这个问题?多态中的短板
    }
}
class Person{
    String name = "托尼";
    public void business(){
        System.out.println(name+"卖军火");
    }
    public void saveWorld(){
        System.out.println("有一颗正义的心");
    }
}
class IronMan extends Person{
    String name = "钢铁侠";
    //子类自己特有的方法
    public void fly(){
        System.out.println("可以飞");
    }
    //子类自己特有的方法
    public void fire(){
        System.out.println("发射火箭炮");
    }
    //重写
    public void saveWorld(){
        System.out.println(name+"拯救受灾难的人群");
        System.out.println("攻击坏人");
        fly();
        fire();
    }
    //子类自己特有的方法
    public void paoNiu(){
        System.out.println("经常晚上抱着自己的秘书飞,带她展示 IronMan 盔甲的功能");
    }
}
```

●09.06_面向对象(多态中向上转型和向下转型)

- A:案例演示
    - 详细讲解多态中向上转型和向下转型

```
public class Demo10 {
    public static void main(String[] args) {
        Person p = new IronMan();  //子类的数据类型被提升为父类的数据类型
```

```
        p.business();
          p.saveWorld();

//          p.paoNiu();
          //我们发现了一个问题,在多态中,父类的引用不能调到子类自己特有的方法,怎么解决这个问题?多态中的短板
        System.out.println("---------------");

          //向下强制转型
        IronMan ir = (IronMan) p;
          //向下强制转型之后就可以直接调用子类特有的方法
        ir.paoNiu();
          ir.fire();
          ir.fly();
    }
}
class Person{
    String name = "托尼";
    public void business(){
        System.out.println(name+"卖军火");
    }
    public void saveWorld(){
        System.out.println("有一颗正义的心");
    }
}
class IronMan extends Person{
    String name = "钢铁侠";
    //子类自己特有的方法
   public void fly(){
        System.out.println("可以飞");
    }
    //子类自己特有的方法
   public void fire(){
        System.out.println("发射火箭炮");
    }
    //重写
   public void saveWorld(){
        System.out.println(name+"拯救受灾难的人群");
        System.out.println("攻击坏人");
        fly();
        fire();
    }
    //子类自己特有的方法
   public void paoNiu(){
        System.out.println("经常晚上抱着自己的秘书飞,带她展示 IronMan 盔甲的功能");
    }
}
```

多态的应用场景之一：

```
public class Demo1 {
    public static void main(String[] args) {
        Oringe o = new Oringe();
        o.name = "橘子";
        o.juicer();

        Banana b = new Banana();
        b.name = "香蕉";
        b.juicer();

        Apple a = new Apple();
        a.name = "苹果";
        a.juicer();
    }
}
class Fruit{
    String name;

    public void juicer(){
        System.out.println("榨汁成"+name+"汁");
    }
}
class Oringe extends Fruit{

}
class Banana extends Fruit{

}
class Apple extends Fruit{

}
```

```
public class Demo1 {
    public static void main(String[] args) {
        Fruit f= new Fruit();
        Oringe o = new Oringe();
        o.name = "橘子";
        f.juicer(o); // Fruit f =new Oringe();

        Banana b = new Banana();
        b.name= "香蕉";
        f.juicer(b); //Fruit f = new Banana();

        Apple a = new Apple();
        a.name ="苹果";
        f.juicer(a);
```

```
    }
}
class Fruit{
    String name;

    public void juicer(Fruit f){    //简单工厂设计模式
      System.out.println("榨汁成"+f.name+"汁");
    }
}
class Oringe extends Fruit{

}
class Banana extends Fruit{

}
class Apple extends Fruit{

}
```

●09.07_面向对象(多态的好处和弊端)

- A:多态的好处
    - a:提高了代码的维护性(继承保证)
    - b:提高了代码的扩展性(由多态保证)
- B:案例演示
    - 多态的好处
    - 可以当作形式参数,可以接收任意子类对象
- C:多态的弊端
    - 不能使用子类的特有属性和行为（强转后才可以使用子类特有的属性和方法）。
- D:案例演示

●09.08_面向对象(多态中的题目分析题)

- A:看下面程序是否有问题，如果没有，说出结果

```
class Fu {
```

```
    public void show() {
        System.out.println("fu--show");
    }
}
class Zi extends Fu {
    public void show() {
        System.out.println("zi__show");
    }
    public void method() {
            System.out.println("zi__method");
        }
    }
public class Test1Demo {
    public static void main(String[] args) {
        Fu f = new Zi();
        f.method();//会报错,父类引用不能调用子类特有的方法
        f.show();   //输出 zi____show
    }
}
```

- B:看下面程序是否有问题，如果没有，说出结果

```
class A {
    public void show() {
        show2();
    }
    public void show2() {
        System.out.println("我");
    }
}
class B extends A {
    public void show2() {
        System.out.println("爱");
    }
}
class C extends B {
    public void show() {
       super.show();
    }
    public void show2() {
        System.out.println("你");
    }
}
public class Test2DuoTai {
    public static void main(String[] args) {
        A a = new B();
        a.show();
```

```
        B b = new C();
        b.show();
    }
}
```

```
爱
你
```

课堂练习：

写一个汽车类
属性：品牌
用多态的方式
汽车输品牌，及价格
家用车
出租车
卡车