**0703-Java编程语言基础--面向对象(下1）**

1. **继承**

**继承的特点：**

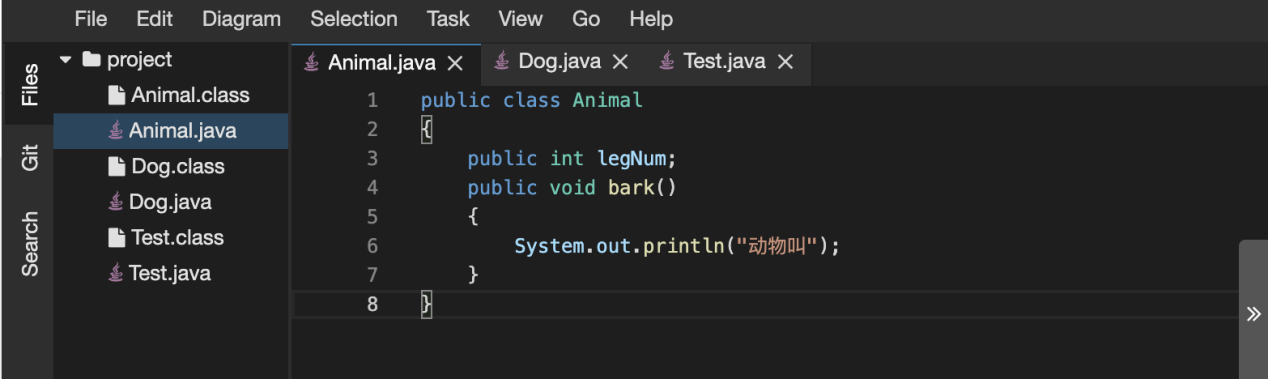
**子类拥有父类除 private 以外的所有属性和方法**

**子类可以拥有自己的属性和方法**

**子类可以重写实现父类的方法**

**Java 中的继承是单继承，一个类只有一个父类**

**注： Java 实现多继承的一个办法是 implements（实现）接口**

****

public class Animal

{

public int legNum;

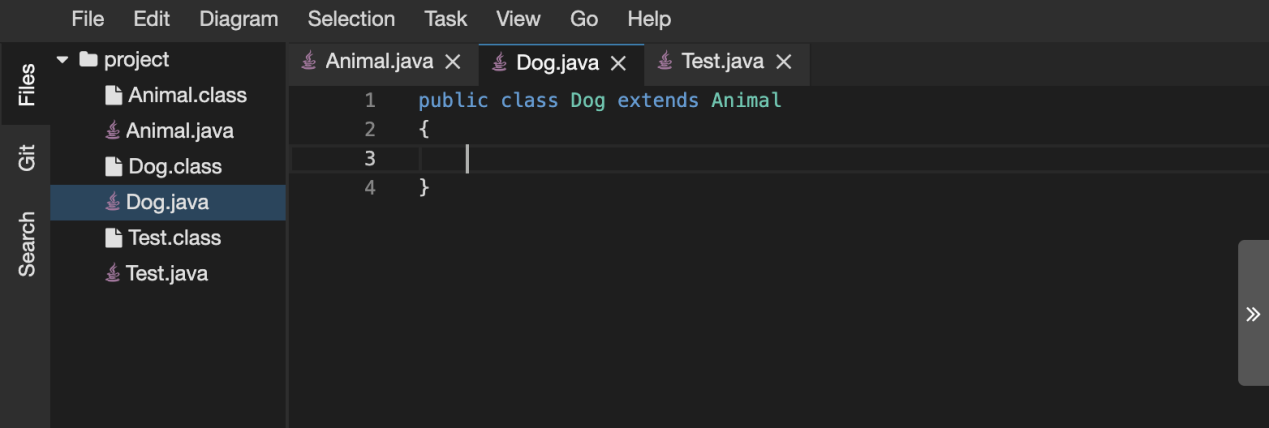
public void bark()

{

System.out.println("动物叫");

}

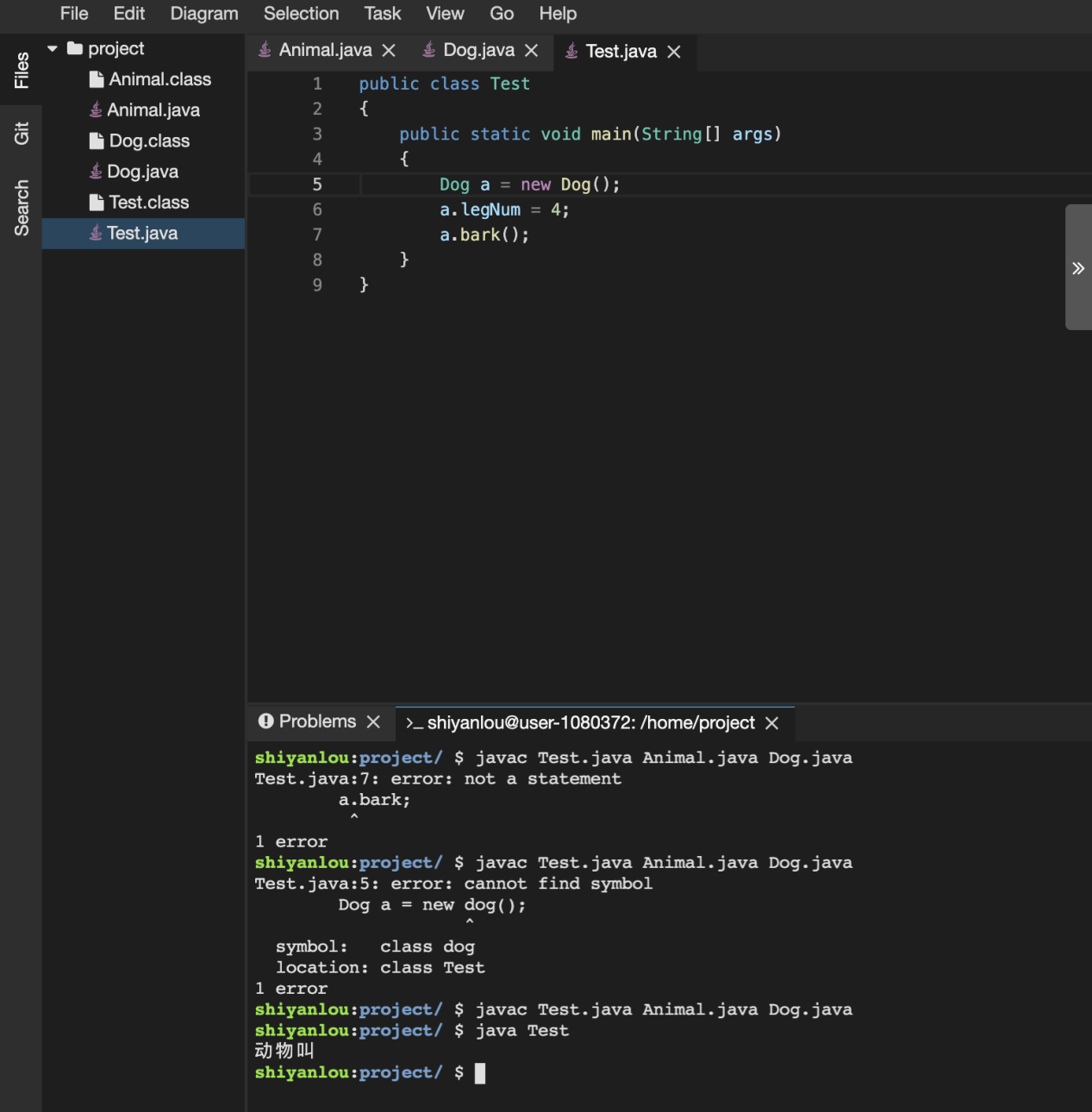
}

****

public class Dog extends Animal

{

}

****

public class Test

{

public static void main(String[] args)

{

Dog a = new Dog();

a.legNum = 4;

a.bark();

}

}

**11.super**

**super关键字在子类内部使用，代表父类对象。**

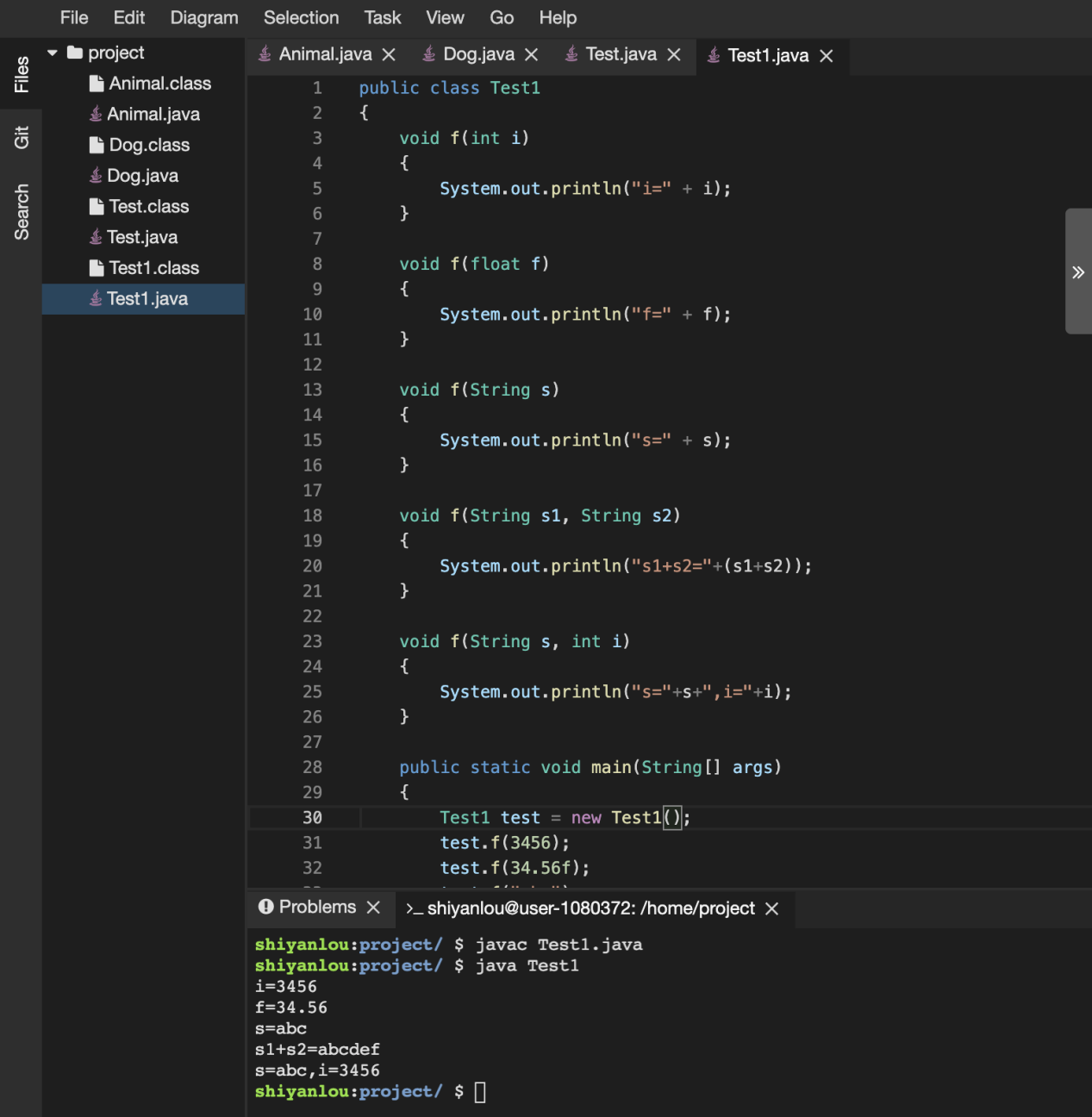
**访问父类的属性 super.属性名**

**访问父类的方法 super.bark()**

**子类构造方法需要调用父类的构造方法时，在子类的构造方法体里最前面的位置：super()**

1. **方法重载与重写**

**方法重载是指在一个类中定义多个同名的方法，但要求每个方法具有不同的参数的类型或参数的个数。方法重载一般用于创建一组任务相似但是参数不同的方法。**

****

public class Test1

{

void f(int i)

{

System.out.println("i=" + i);

}

void f(float f)

{

System.out.println("f=" + f);

}

void f(String s)

{

System.out.println("s=" + s);

}

void f(String s1, String s2)

{

System.out.println("s1+s2="+(s1+s2));

}

void f(String s, int i)

{

System.out.println("s="+s+",i="+i);

}

public static void main(String[] args)

{

Test1 test = new Test1();

test.f(3456);

test.f(34.56f);

test.f("abc");

test.f("abc","def");

test.f("abc",3456);

}

}

**方法重载有以下几种规则：**

**方法中的参数列表必须不同。比如：参数个数不同或者参数类型不同。**

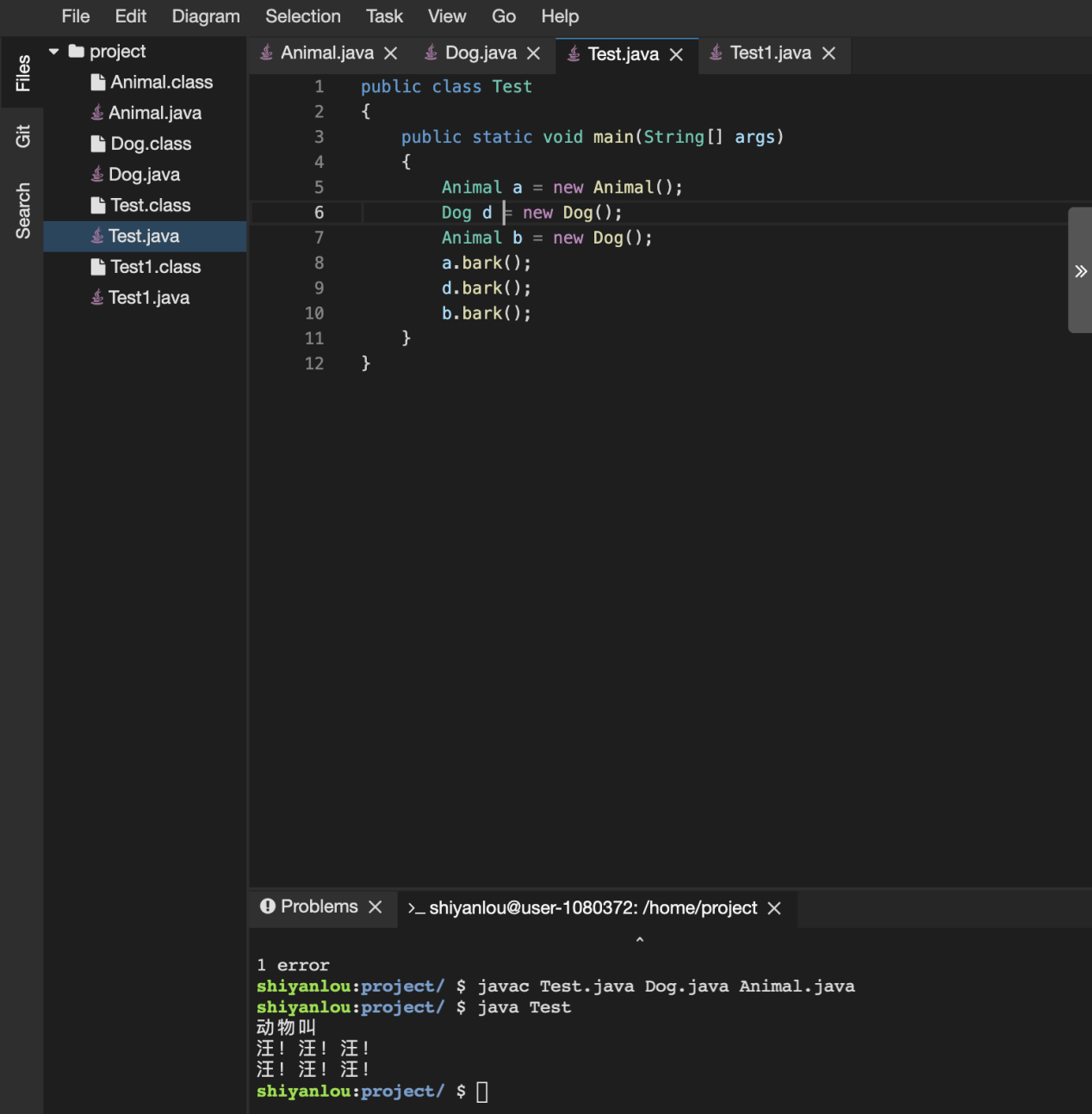
**重载的方法中允许抛出不同的异常**

**可以有不同的返回值类型，但是参数列表必须不同**

**可以有不同的访问修饰符**

**方法重写的子类可以继承父类的方法，但如果子类对父类的方法不满意，想在里面加入适合自己的一些操作时，就需要将方法进行重写。并且子类在调用方法中，优先调用子类的方法。**

**重写的方法一定要与原父类的方法语法保持一致，比如返回值类型，参数类型及个数，和方法名都必须一致。**

****

public class Test

{

public static void main(String[] args)

{

Animal a = new Animal();

Dog d = new Dog();

Animal b = new Dog();

a.bark();

d.bark();

b.bark();

}

}

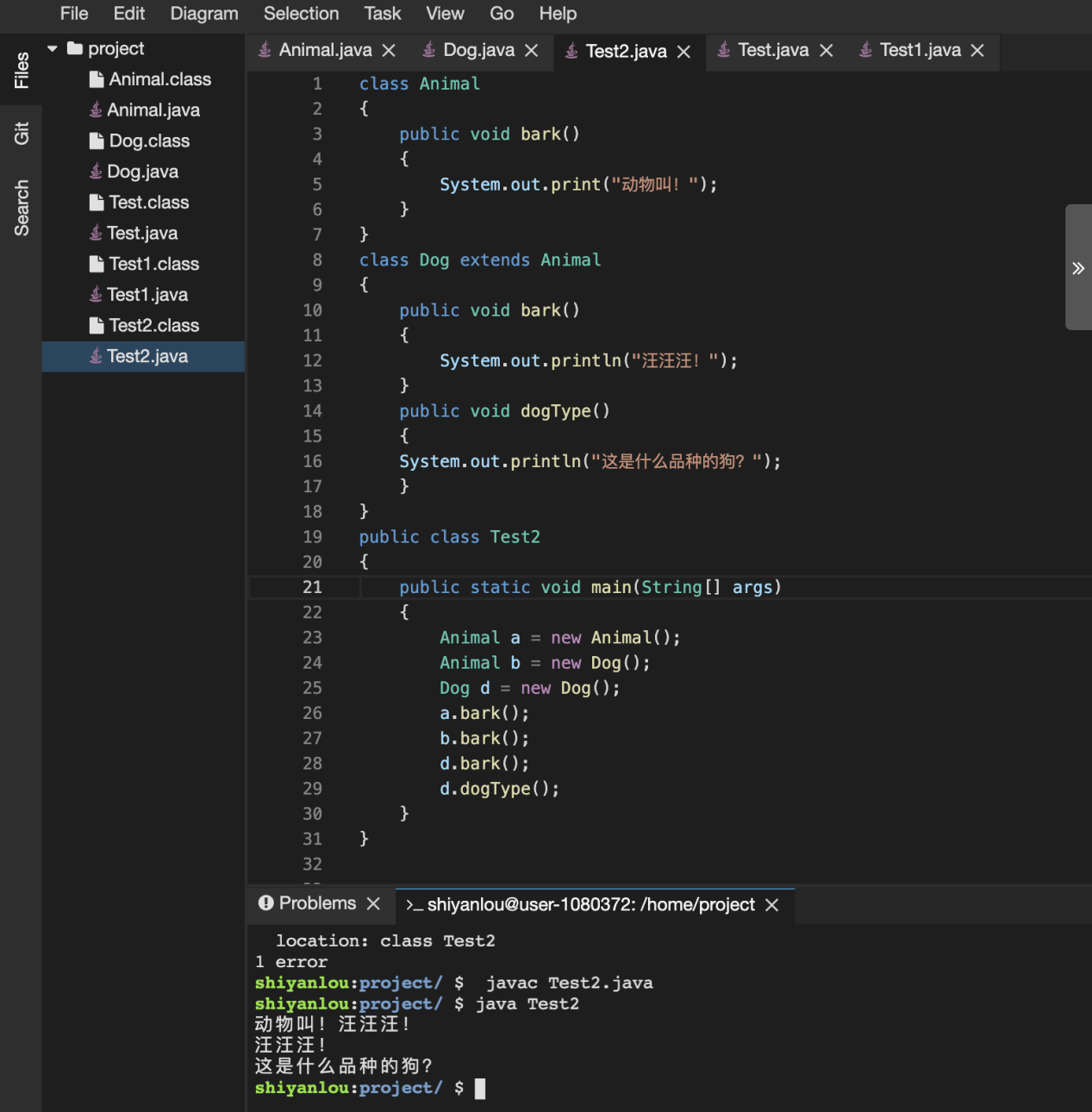
**13.多态**

**多态是指允许不同类的对象对同一消息做出响应。即同一消息可以根据发送对象的不同而采用多种不同的行为方式。多态也称作动态绑定（dynamic binding），是指在执行期间判断所引用对象的实际类型，根据其实际的类型调用其相应的方法。**

**通俗地讲，只通过父类就能够引用不同的子类，这就是多态，我们只有在运行的时候才会知道引用变量所指向的具体实例对象。**

1. **向上转型**

**如果定义了一个指向子类对象的父类引用类型，那么它除了能够引用父类中定义的所有属性和方法外，还可以使用子类强大的功能。但是对于只存在于子类的方法和属性就不能获取。**

****

class Animal

{

public void bark()

{

System.out.print("动物叫！");

}

}

class Dog extends Animal

{

public void bark()

{

System.out.println("汪汪汪！");

}

public void dogType()

{

System.out.println("这是什么品种的狗？");

}

}

public class Test2

{

public static void main(String[] args)

{

Animal a = new Animal();

Animal b = new Dog();

Dog d = new Dog();

a.bark();

b.bark();

d.bark();

d.dogType();

}

}

**另：b.dogType()编译不通过，因为 b 是父类的引用，指向子类的对象，因此不能获取子类的方法（dogType()方法）,同时当调用 bark()方法时，由于子类重写了父类的 bark()方法,所以调用子类中的 bark()方法。**

**向上转型，在运行时，会遗忘子类对象中与父类对象中不同的方法，也会覆盖与父类中相同的方法——重写。（方法名，参数都相同）**

1. **多态的实现条件**

**Java 实现多态有三个必要条件：继承、重写和向上转型（即父类引用指向子类对象）。**

**只有满足上述三个条件，才能够在同一个继承结构中使用统一的逻辑实现代码处理不同的对象，从而达到执行不同的行为。**

1. **多态的实现方式**

**Java 中多态的实现方式：继承父类进行方法重写，抽象类和抽象方法，接口实现。**