向上转型：

```
class Animal {
    public void makeSound() {
        System.out.println("动物发出声音");
    }
}

class Dog extends Animal {
    @Override
    public void makeSound() {
        System.out.println("汪汪汪");
    }

    public void bark() {
        System.out.println("狗叫");
    }
}

public class Main {
    public static void main(String[] args) {
        Dog dog = new Dog(); // 创建一个 Dog 对象，dog 指向堆内存中的 Dog 对象实例
        Animal animal = dog; // 向上转型：animal 指向的仍然是堆内存中 *同一个* Dog 对象实例

        animal.makeSound(); // 输出：汪汪汪 （调用的是子类重写的方法）
        // animal.bark(); // 编译错误：Animal 类没有 bark() 方法
        dog.bark();//dog可以调用子类的方法

        System.out.println("dog的地址："+dog);
        System.out.println("animal的地址："+animal);
    }
}
```

输出结果：

```
汪汪汪
狗叫
dog的地址：Dog@76ed5528
animal的地址：Dog@76ed5528
```

向下转型：

```
public class Main {
    public static void main(String[] args) {
        Dog dog = new Dog(); // 创建Dog对象，dog指向堆内存中的Dog对象
        Animal animal1 = dog; // 向上转型：animal1 指向堆内存中的 *同一个* Dog 对象

        Dog dog1 = (Dog) animal1; // 向下转型：dog1 指向的仍然是堆内存中 *同一个* Dog 对象
        dog1.bark(); // 可以调用 Dog 类特有的方法

        Animal animal2 = new Animal(); // 创建Animal对象，animal2指向堆内存中的Animal对象
        //Dog dog2 = (Dog) animal2; // 向下转型，不安全，运行时会抛出 ClassCastException，因为animal2指向的是Animal对象而不是Dog对象
        //dog2.bark();

            if(animal2 instanceof Dog){
            Dog dog2 = (Dog) animal2;
            dog2.bark();
        }else{
            System.out.println("animal2不是Dog类型，无法转换");
        }

        System.out.println("dog的地址："+dog);
        System.out.println("animal1的地址："+animal1);
        System.out.println("dog1的地址："+dog1);
        System.out.println("animal2的地址："+animal2);

    }
}
```

输出结果：

```
狗叫
animal2不是Dog类型，无法转换
dog的地址：Dog@76ed5528
animal1的地址：Dog@76ed5528
dog1的地址：Dog@76ed5528
animal2的地址：Animal@29453f44
```

要在这个例子中实现安全的向下转型，你需要确保父类引用实际上指向的是子类对象。简单来说，你必须先进行向上转型，然后再将这个父类引用向下转型回原来的子类类型。

```
class Animal {
    public void makeSound() {
        System.out.println("动物发出声音");
    }
}

class Dog extends Animal {
    @Override
    public void makeSound() {
        System.out.println("汪汪汪");
    }

    public void bark() {
        System.out.println("狗叫");
    }
}

public class Main {
    public static void main(String[] args) {
        // 1. 创建 Dog 对象
        Dog myDog = new Dog();

        // 2. 向上转型：将 Dog 对象赋值给 Animal 类型的引用
        Animal myAnimal = myDog; // myAnimal 指向的是 myDog 所指向的同一个 Dog 对象

        // 3. 向下转型：将 Animal 类型的引用强制转换为 Dog 类型
        if (myAnimal instanceof Dog) { // 重要的类型检查！
            Dog anotherDog = (Dog) myAnimal; // 现在 anotherDog 指向的也是同一个 Dog 对象
            anotherDog.bark(); // 安全地调用 Dog 类特有的方法
            anotherDog.makeSound();//调用重写后的方法
        } else {
            System.out.println("无法转换为 Dog 类型");
        }

        // 另一种方式，更简洁：
        Animal anotherAnimal = new Dog();
        if(anotherAnimal instanceof Dog){
            ((Dog) anotherAnimal).bark();
            ((Dog)anotherAnimal).makeSound();
        }else{
            System.out.println("无法转换为 Dog 类型");
        }
    }
}
```