2、具体内容

引用传递是整个Java开发与设计过程指之中最为重要的技术组成，对于引用传递也与实际生活密切相关。

■引用应用分析一

假设数现在生活比较好，于是有的人可以有一辆汽车，当然，有些人屌丝是没有汽车的，只有11卢车可以使用，现在要求可以通过面向独享的设计来解决实现以上的这种关系转换。

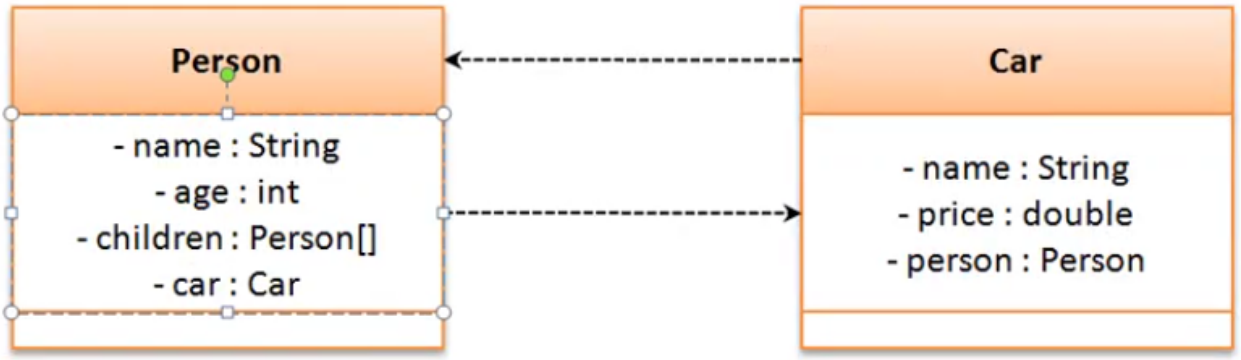

```
class Car
{
	private String name;
	private double price;
	private Person person;//车应该属于一个人
	public Car(String name,double price){
		this.name = name;
		this.price = price;
	}
	public Person getPerson() {
		return person;
	}
	public void setPerson(Person person) {
```

```
        this.person = person;
    }
    public String getInfo(){
        return "汽车名字："+this.name+"汽车价格："+this.price;
    }
}
class Person
{
    private String name;
    private int age;
    private Car car;//一个人有一辆车
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public Car getCar() {
        return car;
    }
    public void setCar(Car car) {
        this.car = car;
    }
    public String getInfo() {
        return "person [name=" + name + ", age=" + age + "]";
    }
    //setter、getter略
}
public class JavaDemo {
    public static void main(String[] args) {
        //第一步：声明对象并且设置彼此的关系
        Person person = new Person("qianglin",29);
        Car car = new Car("bingli",80000000.0);
        person.setCar(car);//一个人有一辆车
        car.setPerson(person);//一辆车属于一个人
        //第二部：根据关系获取数据
        //Car c = person.getCar();
        //c.getInfo();
        System.out.println(person.getCar().getInfo());
        System.out.println(car.getPerson().getInfo());
    }
}
```

本次所操作的两个类型：Person、Car都是自定义类型，但是Person和Car都可以明确的描述出某一类群体，现在是针对于群体的关系作出的一种设置。

■引用应用分析二

现在已经确定好了人鱼车的关系，但是现在也可以进一步进行该操作的完善，例如：一个人肯定会有孩子，孩子也会成年也会有车。




```
class Car
{
	private String name;
	private double price;
	private Person person;//车应该属于一个人
	public Car(String name,double price){
		this.name = name;
		this.price = price;
	}
	public Person getPerson() {
		return person;
	}
	public void setPerson(Person person) {
		this.person = person;
	}
	public String getInfo(){
		return "汽车名字："+this.name+"汽车价格："+this.price;
	}
}
class Person
{
	private String name;
	private int age;
	private Car car;//一个人有一辆车
	private Person children[];//一个人有多个孩子
	public Person[] getChildren() {
		return children;
	}
	public void setChildren(Person[] children) {
		this.children = children;
	}
	public Person(String name, int age) {
		this.name = name;
		this.age = age;
	}
	public Car getCar() {
		return car;
	}
	public void setCar(Car car) {
		this.car = car;
```

```
    }
    public String getInfo() {
        return "person [name=" + name + ", age=" + age + "]";
    }
    //setter、getter略
}
public class JavaDemo {
    public static void main(String[] args) {
        //第一步：声明对象并且设置彼此的关系
        Person person = new Person("qianglin",29);
        Person childA = new Person("qianglin",18);
        Person childB = new Person("renyi",19);
        childA.setCar(new Car("BMW X1",300000.0));//匿名对象
        childB.setCar(new Car("falali",1500000.0));
        person.setChildren(new Person[] {childA,childB});//一个人有多个孩子
        Car car = new Car("bingli",80000000.0);
        person.setCar(car);//一个人有一辆车
        car.setPerson(person);//一辆车属于一个人
        //第二部：根据关系获取数据
        //Car c = person.getCar();
        //c.getInfo();
        System.out.println(person.getCar().getInfo());
        System.out.println(car.getPerson().getInfo());
        //根据人找到所有的孩子以及孩子对应的汽车
        for(int x=0;x<person.getChildren().length;x++) {
            System.out.println("\t|- "+person.getChildren()[x].getInfo());
            System.out.println("\t\t|-"+person.getChildren()[x].getInfo());
        }
    }
}
```

这些关系的匹配都是通过引用数据类型的关联来完成的。

■引用应用分析三

假设说现在要求你定义出一种可以描述电脑组成的类，那么在这样的状态下就必须进行拆分，电脑分为两部分：显示器、主机，而主机上需要设置有一些列的硬件。

```
class 电脑{
    private 显示器 对象数组[] ;
    private 主机 对象;
}
class 显示器{}
class 主机{
    private 主板 对象;
    private 鼠标 对象;
    private 键盘 对象;
}
class 主板{
    private 内存 对象数组[];
    private cpu 对象数组[];
```

```
        private 显卡 对象;
        private 硬盘 对象数组[];
}
class 键盘{}
class 鼠标{}
class 内存{}
class CPU{}
class 显卡{}
class 硬盘{}
```

任何的人类的产品都是可以拆分的，而后进行重新组合，所以这样的设计在java之中被称为合成设计模式。