2、具体内容

初期的最可靠的也是最简单的分析依据：简单java类

■案例分析一

编写并测试一个代表地址的Address类，地址信息由国家、省份、城市、街道、邮编组成，并可以返回完整的地址信息。

```
class Address{
	private String country;
	private String province;
	private String city;
	private String street;
	private String zipcode;
	public Address() {}
	public Address(String country, String province, String city, String street, String zipcode)
{
		this.country = country;
		this.province = province;
		this.city = city;
		this.street = street;
		this.zipcode = zipcode;
	}
	public String getInfo() {
		return " [国家=" + country + ", 省份=" + province + ", 城市=" + city + ", 街道=" +
street
				+ ", 邮编=" + zipcode + "]";
	}
	public String getCountry() {
		return country;
	}
	public void setCountry(String country) {
		this.country = country;
	}
```

```
	public String getProvince() {
		return province;
	}
	public void setProvince(String province) {
		this.province = province;
	}
	public String getCity() {
		return city;
	}
	public void setCity(String city) {
		this.city = city;
	}
	public String getStreet() {
		return street;
	}
	public void setStreet(String street) {
		this.street = street;
	}
	public String getZipcode() {
		return zipcode;
	}
	public void setZipcode(String zipcode) {
		this.zipcode = zipcode;
	}

}
public class JavaDemo {

	public static void main(String[] args) {
		System.out.println(new Address("中华人民共和
国","beijing","beijing","zhaoyanglu","10000").getInfo());
	}

}
```

■案例分析二

定义并测试一个代表员工的Employee类。员工属性包括"编号"、"姓名"、"基本薪水"、"薪水增长率"，还包括计算薪水增长额及计算增长后的工资总额的操作方法。

这个程序的功能已经超过了简单Java类的定义范畴，因为简单Java类里面不需要涉及到复杂的计算逻辑，但是设计的思考应该是从简单java类开始。

```
class Employee{
	private long empno;
	private String ename;
	private double salary;
	private double rate;
```

```
	public Employee() {
	}

	public Employee(long empno, String ename, double salary, double rate) {
		super();
		this.empno = empno;
		this.ename = ename;
		this.salary = salary;
		this.rate = rate;
	}
	//setter、getter略
	public String getInfo() {
		return "Employee [empno=" + this.empno + ", ename=" + this.ename + ",
salary=" + this.salary + ", rate=" + this.rate + "]";
	}
	public double salaryIncValue() {//得到薪水增长额度
		return this.salary*this.rate;
	}
	public double salaryIncResult() {
		this.salary= this.salary*(1+this.rate);
		return this.salary;
	}
}
public class JavaDemo {

	public static void main(String[] args) {
		Employee emp = new Employee(7369L,"SMITH",3000.0,0.3);
		System.out.println(emp.getInfo());
		System.out.println("工资调整额度："+emp.salaryIncValue());
		System.out.println("工资增长后："+emp.salaryIncResult());
		System.out.println(emp.getInfo());
	}
}
```

■案例分析三

设计一个dog类，有名字、颜色、年龄等属性，定义构造方法来初始化类的这些属性，定义方法输出dog信息，编写应用程序使用dog类。

```
class Dog{
	private String name;
	private String color;
	private int age;
	public Dog(String name, String color, int age) {
		super();
		this.name = name;
		this.color = color;
		this.age = age;
```

```
    }
    public Dog() {
        super();
    }
    public String getInfo() {
        return "Dog [name=" + name + ", color=" + color + ", age=" + age + "]";
    }
    //setter、getter略
}
public class JavaDemo {

    public static void main(String[] args) {
        Dog dog = new Dog("gaogao","black",1);
        System.out.println(dog.getInfo());
    }
}
```

■案例分析四

构造一个银行账户类，类的构成包括如下内容：

（1）、数据成员用户的账户名称、用户的账户余额（private数据类型）

（2）、方法包括开户（设置账户名称及余额），利用构造方法完成

（3）、查询余额

```
class Account{
    private String name;
    private double balance;
    public Account(String name, double balance) {
        super();
        this.name = name;
        this.balance = balance;
    }
    public Account(String name) {
        this(name,0.0);//调用双参构造
    }
    public String getInfo() {
        return "Account [name=" + name + ", balance=" + balance + "]";
    }
    //setter、getter略
    public double getBalance() {
        return balance;
    }
}
public class JavaDemo {

    public static void main(String[] args) {
        Account account = new Account("haha",9000000.0);
```

```
        System.out.println(account.getInfo());
        System.out.println(account.getBalance());
    }
}
```

■案例分析五

设计一个表示用户的User类，类中的变量有用户名、口令和记录用户个数的变量，定义类的3个构造方法（无参、为用户名赋值、为用户名和口令赋值）、获取和设置口令的方法和返回类信息的方法。

在简单java类的定义里面追加有static统计操作即可。

```
class User{
    private String uid;
    private String password;
    private static int count = 0;
    public User() {
        this("NOUID","password");
    }
    public User(String uid) {
        this(uid,"password");
        this.uid = uid;
    }
    public User(String uid, String password) {
        this.uid = uid;
        this.password = password;
        count++;//个数追加
    }
    public static int getCount() {//获取用户个数
        return count;
    }
    //getter、setter略
    public String getInfo() {
        return "User [uid=" + uid + ", password=" + password + "]";
    }
}
public class JavaDemo {

    public static void main(String[] args) {
        User userA = new User();
        User userB = new User("xg");
        User userC = new User("xg","no");
        System.out.println(userA.getInfo());
        System.out.println(userB.getInfo());
        System.out.println(userC.getInfo());
        System.out.println("user number is:"+User.getCount());
    }
```

```
}
```

■案例分析六

声明一个图书类，其数据成员为书名、编号（利用静态变量实现自动编号）、书价，并拥有静态数据成员册数、记录图书的总册数，在构造方法中利用此静态变量为对象的编号赋值，在主方法中定义多个对象，并求出总册数。

```
class Book{
	private int bid;//ID
	private String title;//book name
	private double price;//book price
	private static int count = 0;

	public Book() {
	}

	public Book(String title, double price) {
		this();
		this.bid = count + 1;		//先赋值在进行count自增
		this.title = title;
		this.price = price;
		count++;
	}
	//setter、getter略
	public String getInfo() {
		return "Book [bid=" + bid + ", title=" + title + ", price=" + price + "]";
	}
	public static int getCount() {
		return count;
	}
}
public class JavaDemo {

	public static void main(String[] args) {
		Book b1 = new Book("java",89.2);
		Book b2 = new Book("oracle",79.2);
		System.out.println(b1.getInfo());
		System.out.println(b2.getInfo());
		System.out.println("book number are:"+Book.getCount());

	}
}
```

在面向对象最基础的开发里面，简单java类时解决先期设计最好的方案。