6.面向对象与案例

1.Address案例分析

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，并且可以返回完整的信息

```
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： " + this.country + "、省份：" + this.province + "、城市 ： " + this.city + " 街道：" + this.street + "邮编 ： " + this.zipcode ;
    }
    public void setCountry(String country) {
        this.country = country ;
    }
    public void setProvince(String province) {
        this.province = province ;
    }
    public void setCity(String city) {
        this.city = city ;
    }
    public void setStreet(String street) {
        this.street = street ;
    }
    public void setZipcode(String zipcode) {
        this.zipcode = zipcode;
    }
    public String getCountry() {
        return this.country ;
    }
    public String getProvince() {
        return this.province ;
    }
    public String getCity() {
        return this.city ;
    }
    public String getStreet() {
```

```
        return this.street ;
    }
    public String getZipcode() {
        return this.zipcode ;
    }
}
public class JavaStudy {
    public static void main(String args[]) {
        System.out.println(new Address("中国","北京","北京","天安门街道","0001").getInfo()) ;
    }
}
```

2.Employee案例分析

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，薪水基本增长率，及计算增长后的工资总额操作方法。

这个程序的功能超过了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，但是应该是从简单java类开始。

```
class Employee {
    private long empno ;
    private String ename ;
    private double sakary ;
    private double rate ;
    public Employee() {}
    public Employee(long empno,String ename,double sakary,double rate) {
        this.empno = empno ;
        this.ename = ename ;
        this.sakary = sakary ;
        this.rate = rate ;
    }
    public double salaryIncValue() {    //所得到的薪水增长额度
        return this.sakary * this.rate ;
    }
    public double salaryIncResult() {   //计算工资总额
        this.sakary = this.sakary * (1 + this.rate) ;
        return this.sakary ;
    }
    public String getInfo() {
        return "编号：" + this.empno + "姓名 :" + this.ename + "工资: " + this.sakary + "增长率：" +
this.rate ;
    }
}
public class JavaStudy {
    public static void main(String args[]) {
        Employee emp = new Employee(2019L,"MLDN",10,0.1) ;
        System.out.println(emp.getInfo()) ;
        System.out.println("工资调整额度：" + emp.salaryIncValue()) ;
        System.out.println("上调后计算工资：" + emp.salaryIncResult()) ;
    }
}
```

虽然他不是一个由简单java类构成的，但是使用这种方法可能会比较方便

3.Dog案例分析
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这些属性，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

```
 class Dog {
    private String name ;
    private String color ;
    private int age ;
    public Dog() {}
    public Dog(String name,String color,int age) {
        this.name = name ;
        this.color = color ;
        this.age = age ;
    }
    public String getInfo() {
        return "DOG名字：" + this.name + "DOG颜色" + this.color + "年龄：" + this.age ;
    }
}
public class JavaStudy {
    public static void main(String args[]) {
        Dog dog = new Dog("Ying","白",2) ;
        System.out.println(dog.getInfo()) ;
    }
}
```

4.Account案例分析
构造一个银行账户类，类的构成包裹一下内容：
数据成员用户的账户名称，用户账户余额（private 数据类型）
方法包括开户（设置账户名称及余额），利用构造方法完成
查询余额

```
class Account {
    private String name ;
    private double balance ;
    private Account() {}
    private Account(String name) {
        this(name,0.0) ; //调用双参数构造
    }
    public Account(String name,double balance) {
        this.name = name ;
        this.balance = balance ;
    }
    public double getBalance() {
        return this.balance ;
    }
    public String getInfo() {
        return "账户：" + this.name + "余额：" + this.balance ;
    }
}
public class JavaStudy {
    public static void main(String []args) {
        Account account = new Account("JSCSD",19) ;
        System.out.println(account.getBalance()) ;
```

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

5.User案例分析

设计一个表示用户的User类，其中变量有用户名、口令和记录用户个数的变量，定义了是哪个构造，方法（无参，为用户名赋值，为用户和口令赋值）、获取和设置口令的方法和返回类的相关信息的方法。

在简单java类的定义里面和最佳有static统计操作即可；

```
class User {
    private String uid ;
    private String password ;
    private static int count = 0 ;
    public User() {
        this("NO","mldn") ;
    }
    public User(String uid) {
        this(uid,"mldnjava") ;
    }
    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 ;
    }
    public String getInfo() {
        return "用户名：" + this.uid + " 密码：" + this.password ;
    }
}
public class JavaStudy {
    public static void main(String args[]) {
        User userA = new User() ;
        User userB = new User("小强") ;
        User userC = new User("处处","我太菜") ;
        System.out.println(userA.getInfo()) ;
        System.out.println(userB.getInfo()) ;
        System.out.println(userC.getInfo()) ;
        System.out.println("用户个数" + User.getCount()) ;
    }
}
```

6.Book 案例分析

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）书架，并拥有静态数据成员册数，记录图书的总量。

在构造方法中利用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 ; //编号
    private String title ; //书名
    private double price ; //价格
    private static int count = 0 ;
    public Book(String title,double price) {
```

```
        this.bid = count + 1 ; //先赋值在进行count的自增
        this.title = title ;
        this.price = price ;
        count ++ ;
    }
    public String getInfo() {
        return "图书编号： " + this.bid + "  图书名称： " + this.title + "图书价格" + this.price ;
    }
    public static int getCount() {
        return count ;
    }
}
public class JavaStudy {
    public static void main(String args[]) {
        Book b1 = new Book("Java",3000) ;
        Book b2 = new Book("C++",800000) ;
        System.out.println(b1.getInfo()) ;
        System.out.println(b2.getInfo()) ;
        System.out.println("图书总册数" + Book.getCount()) ;
    }
}
```