2、具体内容

很多的编程语言都会提供有枚举的概念，但是Java一直到JDK1.5之后才提出了所谓的枚举的概念。在实际的开发之中枚举的主要作用是用于定义有限个数对象的一种结构（多例设计），枚举就属于多例设计，并且其结构要比多例设计更加的简单。

■枚举的基本定义

从JDK1.5之后在程序之中提供有enum的关键字，利用此关键字可以实现枚举的定义。

范例：定义一个枚举

```
enum Color {  // 枚举类
    RED,GREEN,BLUE ; // 实例化对象
}
public class JavaDemo {
    public static void main(String args[]) {
        Color c = Color.RED ; // 获取实例化对象
        System.out.println(c) ;
    }
}
```

```
enum Color {  // 枚举类
    红色,绿色,蓝色 ;      // 实例化对象
}
public class JavaDemo {
    public static void main(String args[]) {
        Color c = Color.红色 ; // 获取实例化对象
        System.out.println(c) ;
    }
}
```

如果此时采用了多例设计模式来进行设计，那么需要编写很多的程序代码，这样对于开发的复杂度是比较高的，因为里面毕竟牵扯到了构造方法的私有化以及静态方法。多例设计与枚举设计虽然可以实现相同的功能，但是使用枚举可以在程序编译的时候就判断所使用的的实例化对象是否存在。

在进行枚举处理的时候还可以利用value()方法获取所有的枚举对象进行输出。

范例：获取所有的枚举对象

```
enum Color {  // 枚举类
    RED,GREEN,BLUE ;  // 实例化对象
}
public class JavaDemo {
```

```
    public static void main(String args[]) {
        for (Color c : Color.values()) {
            System.out.println(c) ;
        }
    }
}
```

如果此时同样的功能需要通过多例设计来解决的话，那么就需要使用对象数组了。

从JDK1.5追加了枚举结构之后，就可以在switch之中进行枚举项的判断。

范例：观察枚举与switch处理

```
enum Color {  // 枚举类
    RED,GREEN,BLUE ;  // 实例化对象
}
public class JavaDemo {
    public static void main(String args[]) {
        Color c = Color.RED ;
        switch(c) {       // 直接支持枚举
            case RED :
                System.out.println("红色") ;
                break ;
            case GREEN :
                System.out.println("绿色") ;
                break ;
            case BLUE :
                System.out.println("蓝色") ;
                break ;
        }
    }
}
```

多例上是无法实现这种与switch直接连接的，多例要想实现它就需要编写大量的if判断。

■Enum类

严格意义上来讲枚举并不属于一种新的结构，它的本质相当于是一个类，但是这个类默认会继承Enum类，首先观察一下Enum类的基本定义：

```
public abstract class Enum<E extends Enum<E>>
extends Object
implements Comparable<E>, Serializable
```

现在定义的枚举类的类型就是Enum中所使用的E类型。下面来观察一下Enum类中定义的方法：

No	方法名称	类型	描述
01	protected Enum(String name, int ordinal)	构造	传入名字和序号
02	public final String name()	普通	获得对象名字

03	public final int ordinal()	普通	获得对象序号

范例：观察Enum类的存在

```
enum Color {  // 枚举类
	RED,GREEN,BLUE ;  // 实例化对象
}
public class JavaDemo {
	public static void main(String args[]) {
		for (Color c : Color.values()) {
			System.out.println(c.ordinal() + " - " + c.name()) ;
		}
	}
}
```

在枚举之中每一个对象的序号都是根据枚举对象的定义顺序来决定的。

面试题：请解释enum与Enum的区别？

- enum：是从JDK1.5之后提供的一个关键字，用于定义枚举类；
- Enum：是一个抽象类，所以使用enum关键字定义的类就默认继承了此类。

■定义枚举结构

一直在强调枚举本身就属于一种多例设计模式，那么既然是多例设计模式，那么在一个类之中可以定义的结构是非常有多的。

例如：构造方法、普通方法、属性等，那么这些内容在枚举类中依然可以直接定义，但是需要注意的是：枚举类中定义的构造方法不能够采用非私有化定义（public无法使用）。

范例：在枚举类中定义其它的结构

```
enum Color {  // 枚举类
	RED("红色"),GREEN("绿色"),BLUE("蓝色") ;	// 枚举对象要写在首行
	private String title ; // 定义属性
	private Color(String title) {
		this.title = title ;
	}
	public String toString() {
		return this.title ;
	}
}
public class JavaDemo {
	public static void main(String args[]) {
		for (Color c : Color.values()) {
			System.out.println(c.ordinal() + " - " + c.name() + " - " + c) ;
		}
	}
}
```

本程序在简化程度上一定要远远高于多例设计模式 。除了这种基本的结构之外，在枚举类中也可以实现接口的继承。

范例：让枚举实现接口

```
interface IMessage {
	public String getMessage() ;
}
enum Color implements IMessage {	// 枚举类
	RED("红色"),GREEN("绿色"),BLUE("蓝色") ;	// 枚举对象要写在首行
	private String title ; // 定义属性
	private Color(String title) {
		this.title = title ;
	}
	public String toString() {
		return this.title ;
	}
	public String getMessage() {
		return this.title ;
	}
}
public class JavaDemo {
	public static void main(String args[]) {
		IMessage msg = Color.RED ;
		System.out.println(msg.getMessage()) ;
	}
}
```

在枚举类里面最有意思的是它可以直接定义抽象方法，并且要求每一个枚举对象都要独立覆写此方法。

范例：观察枚举中定义抽象方法

```
enum Color {	// 枚举类
	RED("红色") {
		public String getMessage() {
			return this.toString() ;
		}
	},GREEN("绿色") {
		public String getMessage() {
			return this.toString() ;
		}
	},BLUE("蓝色") {
		public String getMessage() {
			return this.toString() ;
		}
	} ;	// 枚举对象要写在首行
	private String title ; // 定义属性
	private Color(String title) {
		this.title = title ;
	}
	public String toString() {
		return this.title ;
	}
	public abstract String getMessage() ;
}
```

```
public class JavaDemo {
	public static void main(String args[]) {
		System.out.println(Color.RED.getMessage()) ;
	}
}
```

发现枚举的定义是非常灵活的，但是在实际的使用之中，枚举更多情况下还是建议使用它的正确用法，就是定义一个实例即可。

■枚举的实际应用

下面为了更好的巩固枚举的使用，编写一个程序来观察枚举的应用，例如：现在定义一个Person，里面一定有性别，性别肯定不希望用户随意输入，所以使用枚举最合适。

枚举：使用枚举

```
enum Sex {
	MALE("男"),FEMALE("女") ;
	private String title ;
	private Sex(String title) {
		this.title = title ;
	}
	public String toString() {
		return this.title ;
	}
}
class Person {
	private String name ;
	private int age ;
	private Sex sex ;
	public Person(String name,int age,Sex sex) {
		this.name = name ;
		this.age = age ;
		this.sex = sex ;
	}
	public String toString() {
		return "姓名：" + this.name + "、年龄：" + this.age + "、性别：" + this.sex ;
	}
}
public class JavaDemo {
	public static void main(String args[]) {
		System.out.println(new Person("张三",20,Sex.MALE)) ;
	}
}
```

这个程序实际上不使用枚举也可以正常实现，追加几个判断即可，所以对于枚举，愿意用就用，不愿意用就不用。