* **枚举类介绍**
* JDK1.5之前需要自定义枚举类。
* JDK1.5新增的enum关键字用于定义枚举类。
* 若枚举类只有一个成员，则可以作为一种单例模式的实现方式。
* **枚举类的属性**
* **枚举类对象的属性不应允许被改动**,所以应该**使用private final修饰**。
* 枚举类的使用private final修饰的属性应该**在构造器中为其赋值**。
* 若枚举类显式的定义了带参数的构造器,则在列出枚举值时也必须对应的传入参数。

/\*\*  
 \* 自定义枚举类  
 \*/  
class Season {  
 /\*\*  
 \* 1）提供类的属性，声明为private final。  
 \*/  
 private final String seasonName;  
 private final String seasonDesc;  
  
 /\*\*  
 \* 2）声明为final的属性，在构造器中初始化。  
 \*/  
 private Season(String seasonName, String seasonDesc) {  
 this.seasonName = seasonName;  
 this.seasonDesc = seasonDesc;  
 }  
  
 //3）通过公共的方法来调用属性。  
  
 public String getSeasonName() {  
 return seasonName;  
 }  
  
 public String getSeasonDesc() {  
 return seasonDesc;  
 }  
  
 /\*\*  
 \* 4）创建枚举类的对象：将类的对象声明为public static final  
 \*/  
 public static final Season ***SPRING*** = new Season("spring","穿暖花开");  
 public static final Season ***SUMMER*** = new Season("summer","夏日炎炎");  
 public static final Season ***AUTUMN*** = new Season("autumn","秋高气爽");  
 public static final Season ***WINTER*** = new Season("winter","白雪皑皑");  
  
 @Override  
 public String toString() {  
 return "Season{" +  
 "seasonName='" + seasonName + '\'' +  
 ", seasonDesc='" + seasonDesc + '\'' +  
 '}';  
 }  
  
 public void show() {  
 System.***out***.println("这是一个季节");  
 }  
}

* **Enum枚举类**
* **必须在枚举类的第一行声明枚举类对象**。
* 枚举类和普通类的区别：
* 使用enum定义的枚举类默认继承了java.lang.Enum类。
* 枚举类的构造器只能使用private访问控制符。
* 枚举类的所有实例必须在枚举类中显式列出(,分隔;结尾).列出的实例系统会自动添加public static final修饰。
* JDK 1.5中**可以在switch表达式中使用Enum定义的枚举类的对象作为表达式**，case子句可以直接使用枚举值的名字,无需添加枚举类作为限定。
* 主要方法：
* **values()方法**：返回枚举类型的对象数组。该方法可以很方便地遍历所有的枚举值。

//1）values()  
Season1[] seasons = Season1.values();  
for (int i = 0; i < seasons.length; i++) {  
 System.***out***.println(seasons[i]);  
}

* **valueOf(String str)**：可以把一个字符串转为对应的枚举类对象。要求字符串必须是枚举类对象的“名字”。如不是，会有运行时异常。

//2）valuesOf(String name)：要求传入的形参name是枚举类对象的名字。  
// 否则报错  
String str = "SPRING";  
Season1 season1 = Season1.valueOf(str);  
System.***out***.println(season1);

* 实现接口的枚举类
* 和普通Java类一样，枚举类可以实现一个或多个接口。
* **若需要每个枚举值在调用实现的接口方法呈现出不同的行为方式,则可以让每个枚举值分别来实现该方法**。

interface info {  
 void show();  
}  
  
/\*\*  
 \* 枚举类：  
 \* 使用enum关键字定义枚举类  
 \* >常用的方法  
 \* >让枚举类实现接口  
 \*/  
enum Season1 implements info {  
 SPRING("spring", "穿暖花开") {  
 @Override  
 public void show() {  
 System.out.println("春天在哪里？");  
 }  
 },  
 SUMMER("summer", "夏日炎炎") {  
 @Override  
 public void show() {  
 System.out.println("生如夏花");  
 }  
 },  
 AUTUMN("autumn", "秋高气爽") {  
 @Override  
 public void show() {  
 System.out.println("秋天是用来分手的季节");  
 }  
 },  
 WINTER("winter", "白雪皑皑") {  
 @Override  
 public void show() {  
 System.out.println("冬天里的一把火");  
 }  
 };  
  
 private final String seasonName;  
 private final String seasonDesc;  
  
 private Season1(String seasonName, String seasonDesc) {  
 this.seasonName = seasonName;  
 this.seasonDesc = seasonDesc;  
 }  
  
 public String getSeasonName() {  
 return seasonName;  
 }  
  
 public String getSeasonDesc() {  
 return seasonDesc;  
 }  
  
 @Override  
 public String toString() {  
 return "Season{" +  
 "seasonName='" + seasonName + '\'' +  
 ", seasonDesc='" + seasonDesc + '\'' +  
 '}';  
 }  
}

* **使用Enum定义的Season**

enum Season1 {  
 SPRING("spring", "穿暖花开"),  
 SUMMER("summer", "夏日炎炎"),  
 AUTUMN("autumn", "秋高气爽"),  
 WINTER("winter", "白雪皑皑");  
  
 private final String seasonName;  
 private final String seasonDesc;  
  
 private Season1(String seasonName, String seasonDesc) {  
 this.seasonName = seasonName;  
 this.seasonDesc = seasonDesc;  
 }  
  
 public String getSeasonName() {  
 return seasonName;  
 }  
  
 public String getSeasonDesc() {  
 return seasonDesc;  
 }  
}