# 实现枚举类

两种方式实现枚举类,可以了解枚举的本质,重点运用是第二类了.

## 27.13\_JDK5新特性(自己实现枚举类)

A:枚举概述

**是指将变量的值一一列出来,变量的值只限于列举出来的值的范围内**。举例：一周只有7天，一年只有12个月等。

B:**回想单例设计模式：单例类是一个类只有一个实例**

**那么多例类就是一个类有多个实例，但不是无限个数的实例**，而是有限个数的实例。这才能是枚举类。

C:案例演示

自己实现枚举类

1,自动拆装箱

2,泛型

3,可变参数

4,静态导入

5,增强for循环

6,互斥锁

7,枚举

**共有三种实现方式:**

-----------------------------------------------------------------------------------------------------------------------------------------------------------------

public class Demo1\_Enum {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

//demo1();

//demo2();

Week3 mon = Week3.MON;

mon.show();

}

public static void demo2() {

Week2 mon = Week2.MON;

System.out.println(mon.getName());

}

public static void demo1() {

Week mon = Week.MON;

Week tue = Week.TUE;

Week wed = Week.WED;

System.out.println(mon);

}

}

-----------------------------------------------------------------------------------------------------------------------------------------------------------------

**空参**

public class Week {

public static final Week MON = new Week(); //单例类的一种实现方式

public static final Week TUE = new Week();

public static final Week WED = new Week();

private Week(){} //私有构造,不让其他类创建本类对象,enum枚举也是这个实现原理

}

-----------------------------------------------------------------------------------------------------------------------------------------------------------------

**有参**

public class Week2 {

public static final Week2 MON = new Week2("星期一");

public static final Week2 TUE = new Week2("星期二");

public static final Week2 WED = new Week2("星期三");

private String name;

private Week2(String name){

this.name = name;

} //私有构造,不让其他类创建本类对象

public String getName() {

return name;

}

}

-----------------------------------------------------------------------------------------------------------------------------------------------------------------

**抽象类,匿名子类对象**

public abstract class Week3 {

public static final Week3 MON = new Week3("星期一") {

public void show() {

System.out.println("星期一");

}

};

public static final Week3 TUE = new Week3("星期二"){

public void show() {

System.out.println("星期二");

}

};

public static final Week3 WED = new Week3("星期三"){

public void show() {

System.out.println("星期三");

}

};

private String name;

private Week3(String name){

this.name = name;

} //私有构造,不让其他类创建本类对象

public String getName() {

return name;

}

public abstract void show();

}

-----------------------------------------------------------------------------------------------------------------------------------------------------------------

## ■■■■27.14\_JDK5新特性(通过enum实现枚举类)

A:案例演示

通过enum实现枚举类

-----------------------------------------------------------------------------------------------------------------------------------------------------------------

public class Demo1\_Enum {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

//demo1();

//demo2();

//demo3();

Week3 mon = Week3.TUE;

switch (mon) {

case MON:

System.out.println("星期一");

break;

case TUE:

System.out.println("星期二");

break;

}

}

//抽象方法,对应自定义的enmu

public static void demo3() {

Week3 mon = Week3.MON;

mon.show();

}

//有参,拿到参数

public static void demo2() {

Week2 mon = Week2.MON;

System.out.println(mon.getName());

}

//无参,打印的是类名

public static void demo1() {

Week mon = Week.MON;

System.out.println(mon);

}

}

-----------------------------------------------------------------------------------------------------------------------------------------------------------------

public enum Week {

MON,TUE,WED;

}

-----------------------------------------------------------------------------------------------------------------------------------------------------------------

public enum Week2 {

MON("星期一"),TUE("星期二"),WED("星期三");

private String name;

private Week2(String name) {

this.name = name;

}

public String getName() {

return name;

}

public String toString() {

return name;

}

}

-----------------------------------------------------------------------------------------------------------------------------------------------------------------

public enum Week3 {

MON("星期一"){

public void show() {

System.out.println("星期一");

}

},TUE("星期二"){

public void show() {

System.out.println("星期二");

}

},WED("星期三"){

public void show() {

System.out.println("星期三");

}

};

private String name;

private Week3(String name) {

this.name = name;

}

public String getName() {

return name;

}

public abstract void show();

}

-----------------------------------------------------------------------------------------------------------------------------------------------------------------

# 枚举注意事项

## ■■■■27.15\_JDK5新特性(枚举的注意事项)

A:案例演示

1,定义枚举类要用关键字enum,他的地位和class,interface一样

2,所有枚举类都是Enum的子类

3,枚举类的第一行上必须是枚举项，最后一个枚举项后的分号是可以省略的，但是如果枚举类有其他的东西，这个分号就不能省略。建议不要省略,不需使用new 关键字，不需显式调用构造器。自动添加public static final修饰

4,枚举类可以有构造器，但必须是private的，它默认的也是private的。

5,枚举类也可以有抽象方法(默认在enum前边添加了abstract),他可以实现一个或者多个接口，枚举项必须重写该方法

6,使用enum定义、非抽象的枚举类默认使用final修饰，不可以被继承

枚举在switch语句中的使用

# 常见方法

Valus() 返回值数组

## 27.16\_JDK5新特性(枚举类的常见方法)

A:枚举类的常见方法

int ordinal()

int compareTo(E o)

String name()

String toString()

<T> T valueOf(Class<T> type,String name)

values()

此方法虽然在JDK文档中查找不到，但每个枚举类都具有该方法，它遍历枚举类的所有枚举值非常方便

B:案例演示

枚举类的常见方法

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

public class Demo2\_Enum {

/\*\*

\* int ordinal()

\* int compareTo(E o)

\* String name()

\* String toString()

\* <T> T valueOf(Class<T> type,String name)

\* values()

\* 此方法虽然在JDK文档中查找不到，但每个枚举类都具有该方法，它遍历枚举类的所有枚举值非常方便

\*/

public static void main(String[] args) {

//demo1();

// Week2 mon = Week2.valueOf(Week2.class, "MON"); //通过字节码对象获取枚举项

// System.out.println(mon);

**Week2[] arr = Week2.values();**

for (Week2 week2 : arr) {

System.out.println(week2);

}

}

public static void demo1() {

Week2 mon = Week2.MON;

Week2 tue = Week2.TUE;

Week2 wed = Week2.WED;

/\*System.out.println(mon.ordinal()); //枚举项都是有编号的

System.out.println(tue.ordinal());

System.out.println(wed.ordinal());

System.out.println(mon.compareTo(tue)); //比较的是编号

System.out.println(mon.compareTo(wed));\*/

System.out.println(mon.name()); //获取实例名称

System.out.println(mon.toString()); //调用重写之后的toString方法

}

}

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------