**枚举**

## 一、原始的接口定义常量

public interface IConstants {

String MON = "Mon";

String TUE = "Tue";

String WED = "Wed";

String THU = "Thu";

String FRI = "Fri";

String SAT = "Sat";

String SUN = "Sun";

}

## 二、语法（定义）

**1 创建枚举类型要使用 enum 关键字**，隐含了所创建的类型都是 java.lang.Enum 类的子类（java.lang.Enum 是一个抽象类）。枚举类型符合通用模式 Class Enum<E extends Enum<E>>，而 E 表示枚举类型的名称。枚举类型的每一个值都将映射到 protected Enum(String name, int ordinal) 构造函数中，在这里，每个值的名称都被转换成一个字符串，并且序数设置表示了此设置被创建的顺序。

package com.hmw.test;

/\*\*

\* 枚举测试类

\* @author <a href="mailto:hemingwang0902@126.com">何明旺</a>

\*/

public enum EnumTest {

MON, TUE, WED, THU, FRI, SAT, SUN;

}

**2 这段代码实际上调用了7次 Enum(String name, int ordinal)：**

new Enum<EnumTest>("MON",0);

new Enum<EnumTest>("TUE",1);

new Enum<EnumTest>("WED",2);

... ...

## 三、遍历、switch 等常用操作

**1 对enum进行遍历和switch的操作示例代码：**

public class Test {

public static void main(String[] args) {

for (EnumTest e : EnumTest.values()) {

System.out.println(e.toString());

}

System.out.println("----------------我是分隔线------------------");

EnumTest test = EnumTest.TUE;

switch (test) {

case MON:

System.out.println("今天是星期一");

break;

case TUE:

System.out.println("今天是星期二");

break;

// ... ...

default:

System.out.println(test);

break;

}

}

}

输出结果：

MON

TUE

WED

THU

FRI

SAT

SUN

----------------我是分隔线------------------

今天是星期二

## 四、enum 对象的常用方法介绍

int compareTo(E o) 比较此枚举与指定对象的顺序。

Class<E> getDeclaringClass() 返回与此枚举常量的枚举类型相对应的 Class 对象。

String name() 返回此枚举常量的名称，在其枚举声明中对其进行声明。

int ordinal() 返回枚举常量的序数（它在枚举声明中的位置，其中初始常量序数为零）。

String toString()返回枚举常量的名称，它包含在声明中。

static <T extends Enum<T>> T valueOf(Class<T> enumType, String name) 返回带指定名称的指定枚举类型的枚举常量。

public class Test {

public static void main(String[] args) {

EnumTest test = EnumTest.TUE;

//compareTo(E o)

switch (test.compareTo(EnumTest.MON)) {

case -1:

System.out.println("TUE 在 MON 之前");

break;

case 1:

System.out.println("TUE 在 MON 之后");

break;

default:

System.out.println("TUE 与 MON 在同一位置");

break;

}

//getDeclaringClass()

System.out.println("getDeclaringClass(): " + test.getDeclaringClass().getName());

//name() 和 toString()

System.out.println("name(): " + test.name());

System.out.println("toString(): " + test.toString());

//ordinal()， 返回值是从 0 开始

System.out.println("ordinal(): " + test.ordinal());

}

}

输出结果：

TUE 在 MON 之后

getDeclaringClass(): com.hmw.test.EnumTest

name(): TUE

toString(): TUE

ordinal(): 1

## 五、给 enum 自定义属性和方法

给 enum 对象加一下 value 的属性和 getValue() 的方法：

package com.hmw.test;

public enum EnumTest {

MON(1), TUE(2), WED(3), THU(4), FRI(5), SAT(6) {

@Override

public boolean isRest() {

return true;

}

},

SUN(0) {

@Override

public boolean isRest() {

return true;

}

};

private int value;

private EnumTest(int value) {

this.value = value;

}

public int getValue() {

return value;

}

public boolean isRest() {

return false;

}

}

public class Test {

public static void main(String[] args) {

System.out.println("EnumTest.FRI 的 value = " + EnumTest.FRI.getValue());

}

}

**输出结果：**

EnumTest.FRI 的 value = 5

## 六、EnumSet，EnumMap 的应用

public class Test {

public static void main(String[] args) {

// EnumSet的使用

EnumSet<EnumTest> weekSet = EnumSet.allOf(EnumTest.class);

for (EnumTest day : weekSet) {

System.out.println(day);

}

// EnumMap的使用

EnumMap<EnumTest, String> weekMap = new EnumMap(EnumTest.class);

weekMap.put(EnumTest.MON, "星期一");

weekMap.put(EnumTest.TUE, "星期二");

// ... ...

for (Iterator<Entry<EnumTest, String>> iter = weekMap.entrySet().iterator(); iter.hasNext();) {

Entry<EnumTest, String> entry = iter.next();

System.out.println(entry.getKey().name() + ":" + entry.getValue());

}

}

}

## 七、原理分析

enum 的语法结构尽管和 class 的语法不一样，但是经过编译器编译之后产生的是一个class文件。该class文件经过反编译可以看到实际上是生成了一个类，

该类继承了java.lang.Enum<E>。EnumTest 经过反编译(javap com.hmw.test.EnumTest 命令)之后得到的内容如下：

public class com.hmw.test.EnumTest extends java.lang.Enum{

public static final com.hmw.test.EnumTest MON;

public static final com.hmw.test.EnumTest TUE;

public static final com.hmw.test.EnumTest WED;

public static final com.hmw.test.EnumTest THU;

public static final com.hmw.test.EnumTest FRI;

public static final com.hmw.test.EnumTest SAT;

public static final com.hmw.test.EnumTest SUN;

static {};

public int getValue();

public boolean isRest();

public static com.hmw.test.EnumTest[] values();

public static com.hmw.test.EnumTest valueOf(java.lang.String);

com.hmw.test.EnumTest(java.lang.String, int, int, com.hmw.test.EnumTest);

}

**所以，实际上 enum 就是一个 class，只不过 java 编译器帮我们做了语法的解析和编译而已。**

**package** com.hiya.se.enun;

**public** **enum** HiyaEnunParams

{

***NORMAL***("Normal", "正常"),

***SHOP\_NOT\_EXIST***("ShopNotExist", "商户不存在"),

***ORDER\_MATERIAL\_CREATION\_FLOW\_NOT\_EXIST***("OrderMaterialCreationFlowNotExist", "订单没有启动物料制作服务相关流程"),

***SYNC\_MATERIAL\_STATUS\_FAILED***("SyncMaterialStatusFailed", "物料制作状态同步失败"),

***SHOP\_AGENT\_NOT\_EXIST***("ShopAgentNotExist", "商户对应的服务商不存在");

**private** String name;

**private** String displayName;

HiyaEnunParams()

{

}

HiyaEnunParams(String name)

{

**this**.name = name;

}

HiyaEnunParams(String name, String displayName)

{

**this**.name = name;

**this**.displayName = displayName;

}

**public** String getName()

{

**return** name;

}

**public** **void** setName(String name)

{

**this**.name = name;

}

**public** String getDisplayName()

{

**return** displayName;

}

**public** **void** setDisplayName(String displayName)

{

**this**.displayName = displayName;

}

**public** **static** String getByName(String name)

{

**for** (HiyaEnunParams statusEnum : *values*())

{

**if** (statusEnum.getName().equals(name))

{

**return** statusEnum.getDisplayName();

}

}

**return** "";

}

}

package com.hiya.se.enun;

import java.util.EnumMap;

import java.util.EnumSet;

public class EnumClient

{

public static void main(String[] args)

{

for (HiyaEnun e : HiyaEnun.values())

{

System.out.println(e.toString());

}

HiyaEnun valTue = HiyaEnun.TUE;

switch (valTue)

{

case MON:

System.out.println("今天是星期一");

break;

case TUE:

System.out.println("今天是星期二");

break;

// ... ...

default:

System.out.println(valTue);

break;

}

// compareTo(E o)

switch (valTue.compareTo(HiyaEnun.MON))

{

case -1:

System.out.println("TUE 在 MON 之前");

break;

case 1:

System.out.println("TUE 在 MON 之后");

break;

default:

System.out.println("TUE 与 MON 在同一位置");

break;

}

// getDeclaringClass()

System.out.println("getDeclaringClass(): " + valTue.getDeclaringClass().getName());

// name() 和 toString()

System.out.println("name(): " + valTue.name());

System.out.println("toString(): " + valTue.toString());

// ordinal()， 返回值是从 0 开始

System.out.println("ordinal(): " + valTue.ordinal());

System.out.println("EnumTest.FRI 的 value = " + HiyaEnunParams.getByName("fgetrert"));

HiyaEnunParams.NORMAL.getName();

HiyaEnunParams.NORMAL.getDisplayName();

EnumSet<HiyaEnunParams> stateSet = EnumSet.allOf(HiyaEnunParams.class);

for (HiyaEnunParams s : stateSet)

{

System.out.println(s);

}

EnumMap<HiyaEnunParams, String> stateMap = new EnumMap<HiyaEnunParams, String>(HiyaEnunParams.class);

stateMap.put(HiyaEnunParams.NORMAL, "is On");

stateMap.put(HiyaEnunParams.SHOP\_AGENT\_NOT\_EXIST, "is off");

for (HiyaEnunParams s : HiyaEnunParams.values())

{

System.out.println(s.name() + ":" + stateMap.get(s));

}

System.out.println(HiyaSignleEnun.INSTANCE.nameList);

System.out.println(HiyaSignleEnun.INSTANCE2.nameList);

System.out.println(HiyaSignleEnun.INSTANCE2.getNameList());

}

}

package com.hiya.se.enun;

import java.util.ArrayList;

import java.util.List;

public enum HiyaSignleEnun

{

INSTANCE,INSTANCE2;

public List<String> nameList;

HiyaSignleEnun()

{

nameList = new ArrayList<>();

nameList.add("WADE");

}

public List<String> getNameList()

{

return nameList;

}

public void setNameList(List<String> nameList)

{

this.nameList = nameList;

}

}

## 八、总结

**可以把 enum 看成是一个普通的 class，它们都可以定义一些属性和方法，不同之处是：enum 不能使用 extends 关键字继承其他类，因为 enum 已经继承了 java.lang.Enum（java是单一继承）。**