Java枚举详解


静默虚空 (/u/e336bc023b9d) + 关注
2016.11.24 16:34* 字数 1661 阅读 2425 评论 1 喜欢 17
(/u/e336bc023b9d)

概念

`enum` 的全称为 enumeration， 是 JDK 1.5 中引入的新特性。

在Java中，被 `enum` 关键字修饰的类型就是枚举类型。形式如下：

```
enum Color { RED, GREEN, BLUE }
```

如果枚举不添加任何方法，**枚举值默认为从0开始的有序数值**。以 Color 枚举类型举例，它的枚举常量依次为RED：0，GREEN：1，BLUE：2

枚举的好处：可以将常量组织起来，统一进行管理。

枚举的典型应用场景：错误码、状态机等。

枚举类型的本质

尽管 `enum` 看起来像是一种新的数据类型，事实上，**enum是一种受限制的类，并且具有自己的方法**。

创建enum时，编译器会为你生成一个相关的类，这个类继承自 `java.lang.Enum` 。

`java.lang.Enum` 类声明

```
public abstract class Enum<E extends Enum<E>>
        implements Comparable<E>, Serializable { ... }
```

枚举的方法

在enum中，提供了一些基本方法：

`values()`： 返回enum实例的数组，而且该数组中的元素严格保持在enum中声明时的顺序。

`name()`： 返回实例名。

`ordinal()`： 返回实例声明时的次序，从0开始。

`getDeclaringClass()`： 返回实例所属的enum类型。

`equals()` ： 判断是否为同一个对象。

可以使用 `==` 来比较 `enum` 实例。

此外， `java.lang.Enum` 实现了 `Comparable` 和 `Serializable` 接口，所以也提供 `compareTo()` 方法。

例：展示enum的基本方法

```
public class EnumMethodDemo {
    enum Color {RED, GREEN, BLUE;}
    enum Size {BIG, MIDDLE, SMALL;}
    public static void main(String args[]) {
        System.out.println("============ Print all Color ============");
        for (Color c : Color.values()) {
            System.out.println(c + " ordinal: " + c.ordinal());
        }
        System.out.println("============ Print all Size ============");
        for (Size s : Size.values()) {
            System.out.println(s + " ordinal: " + s.ordinal());
        }

        Color green = Color.GREEN;
        System.out.println("green name(): " + green.name());
        System.out.println("green getDeclaringClass(): " + green.getDeclari
ngClass());
        System.out.println("green hashCode(): " + green.hashCode());
        System.out.println("green compareTo Color.GREEN: " + green.compareT
o(Color.GREEN));
        System.out.println("green equals Color.GREEN: " + green.equals(Colo
r.GREEN));
        System.out.println("green equals Size.MIDDLE: " + green.equals(Size
.MIDDLE));
        System.out.println("green equals 1: " + green.equals(1));
        System.out.format("green == Color.BLUE: %b\n", green == Color.BLUE)
;
    }
}
```

输出

```
=========== Print all Color ===========
RED ordinal: 0
GREEN ordinal: 1
BLUE ordinal: 2
=========== Print all Size ===========
BIG ordinal: 0
MIDDLE ordinal: 1
SMALL ordinal: 2
green name(): GREEN
green getDeclaringClass(): class org.zp.javase.enumeration.EnumDemo$Color
green hashCode(): 460141958
green compareTo Color.GREEN: 0
green equals Color.GREEN: true
green equals Size.MIDDLE: false
green equals 1: false
green == Color.BLUE: false
```

枚举的特性

枚举的特性，归结起来就是一句话：

> **除了不能继承，基本上可以将 `enum` 看做一个常规的类。**

但是这句话需要拆分去理解，让我们细细道来。

枚举可以添加方法

在概念章节提到了，**枚举值默认为从0开始的有序数值** 。那么问题来了：如何为枚举显示的赋值。

Java 不允许使用 = 为枚举常量赋值

如果你接触过C/C++，你肯定会很自然的想到赋值符号 `=` 。在C/C++语言中的enum，可以用赋值符号 = 显示的为枚举常量赋值；但是，很遗憾，**Java 语法中却不允许使用赋值符号 `=` 为枚举常量赋值**。

例：C/C++ 语言中的枚举声明

```
typedef enum{
    ONE = 1,
    TWO,
    THREE = 3,
    TEN = 10
} Number;
```

enum 可以添加普通方法、静态方法、抽象方法、构造方法

Java虽然不能直接为实例赋值，但是它有更优秀的解决方案：**为 enum 添加方法来间接实现显示赋值。**

创建 `enum` 时，可以为其添加多种方法，甚至可以为其添加构造方法。

注意一个细节：如果要为enum定义方法，那么必须在enum的最后一个实例尾部添加一个分号。此外，在enum中，必须先定义实例，不能将字段或方法定义在实例前面。否则，编译器会报错。

例：全面展示如何在枚举中定义普通方法、静态方法、抽象方法、构造方法

```
public enum ErrorCode {
    OK(0) {
        public String getDescription() {
            return "成功";
        }
    },
    ERROR_A(100) {
        public String getDescription() {
            return "错误A";
        }
    },
    ERROR_B(200) {
        public String getDescription() {
            return "错误B";
        }
    };

    private int code;

    // 构造方法：enum的构造方法只能被声明为private权限或不声明权限
    private ErrorCode(int number) { // 构造方法
        this.code = number;
    }
    public int getCode() { // 普通方法
        return code;
    } // 普通方法
    public abstract String getDescription(); // 抽象方法
    public static void main(String args[]) { // 静态方法
        for (ErrorCode s : ErrorCode.values()) {
            System.out.println("code: " + s.getCode() + ", description: " +
 s.getDescription());
        }
    }
}
```

注：上面的例子并不可取，仅仅是为了展示枚举支持定义各种方法。下面是一个简化的例子

例：一个错误码枚举类型的定义

本例和上例的执行结果完全相同。

```
public enum ErrorCodeEn {
    OK(0, "成功"),
    ERROR_A(100, "错误A"),
    ERROR_B(200, "错误B");

    ErrorCodeEn(int number, String description) {
        this.code = number;
        this.description = description;
    }
    private int code;
    private String description;
    public int getCode() {
        return code;
    }
    public String getDescription() {
        return description;
    }
    public static void main(String args[]) { // 静态方法
        for (ErrorCodeEn s : ErrorCodeEn.values()) {
            System.out.println("code: " + s.getCode() + ", description: " +
 s.getDescription());
        }
    }
}
```

枚举可以实现接口

`enum` 可以像一般类一样实现接口。

同样是实现上一节中的错误码枚举类，通过实现接口，可以约束它的方法。

```
public interface INumberEnum {
    int getCode();
    String getDescription();
}

public enum ErrorCodeEn2 implements INumberEnum {
    OK(0, "成功"),
    ERROR_A(100, "错误A"),
    ERROR_B(200, "错误B");

    ErrorCodeEn2(int number, String description) {
        this.code = number;
        this.description = description;
    }

    private int code;
    private String description;

    @Override
    public int getCode() {
        return code;
    }

    @Override
    public String getDescription() {
        return description;
    }
}
```

枚举不可以继承

enum 不可以继承另外一个类，当然，也不能继承另一个 enum 。

因为 `enum` 实际上都继承自 `java.lang.Enum` 类，而 Java 不支持多重继承，所以enum不能再继承其他类，当然也不能继承另一个 `enum` 。

枚举的应用场景

组织常量

在JDK1.5 之前，在Java中定义常量都是 `public static final TYPE a;` 这样的形式。有了枚举，你可以将有关联关系的常量组织起来，使代码更加易读、安全，并且还可以使用枚举提供的方法。

枚举声明的格式

注：如果枚举中没有定义方法，也可以在最后一个实例后面加逗号、分号或什么都不加。

下面三种声明方式是等价的：

```
enum Color { RED, GREEN, BLUE }
enum Color { RED, GREEN, BLUE, }
enum Color { RED, GREEN, BLUE; }
```

switch 状态机

我们经常使用switch语句来写状态机。JDK7以后，switch已经支持 `int`、`char`、`String`、`enum` 类型的参数。这几种类型的参数比较起来，使用枚举的switch代码更具有可读性。

```
enum Signal {RED, YELLOW, GREEN}

public static String getTrafficInstruct(Signal signal) {
    String instruct = "信号灯故障";
    switch (signal) {
        case RED:
            instruct = "红灯停";
            break;
        case YELLOW:
            instruct = "黄灯请注意";
            break;
        case GREEN:
            instruct = "绿灯行";
            break;
        default:
            break;
    }
    return instruct;
}
```

组织枚举

可以将类型相近的枚举通过接口或类组织起来。

但是一般用接口方式进行组织。

原因是：Java接口在编译时会自动为enum类型加上 `public static` 修饰符；Java类在编译时会自动为 `enum` 类型加上static修饰符。看出差异了吗？没错，就是说，在类中组织 `enum`，如果你不给它修饰为 `public`，那么只能在本包中进行访问。

例：在接口中组织 enum

```
public interface Plant {
    enum Vegetable implements INumberEnum {
        POTATO(0, "土豆"),
        TOMATO(0, "西红柿");

        Vegetable(int number, String description) {
            this.code = number;
            this.description = description;
        }

        private int code;
        private String description;

        @Override
        public int getCode() {
            return 0;
        }

        @Override
        public String getDescription() {
            return null;
        }
    }

    enum Fruit implements INumberEnum {
        APPLE(0, "苹果"),
        ORANGE(0, "桔子"),
        BANANA(0, "香蕉");

        Fruit(int number, String description) {
            this.code = number;
            this.description = description;
        }

        private int code;
        private String description;

        @Override
        public int getCode() {
            return 0;
        }

        @Override
        public String getDescription() {
            return null;
        }
    }
}
```

例：在类中组织 enum

本例和上例效果相同。

```
public class Plant2 {
    public enum Vegetable implements INumberEnum {...}  // 省略代码
    public enum Fruit implements INumberEnum {...} // 省略代码
}
```

策略枚举

EffectiveJava中展示了一种策略枚举。这种枚举通过枚举嵌套枚举的方式，将枚举常量分类处理。

这种做法虽然没有switch语句简洁，但是更加安全、灵活。

例：EffectvieJava中的策略枚举范例

```
enum PayrollDay {
    MONDAY(PayType.WEEKDAY), TUESDAY(PayType.WEEKDAY), WEDNESDAY(
            PayType.WEEKDAY), THURSDAY(PayType.WEEKDAY), FRIDAY(PayType.WEE
KDAY), SATURDAY(
            PayType.WEEKEND), SUNDAY(PayType.WEEKEND);

    private final PayType payType;

    PayrollDay(PayType payType) {
        this.payType = payType;
    }

    double pay(double hoursWorked, double payRate) {
        return payType.pay(hoursWorked, payRate);
    }

    // 策略枚举
    private enum PayType {
        WEEKDAY {
            double overtimePay(double hours, double payRate) {
                return hours <= HOURS_PER_SHIFT ? 0 : (hours - HOURS_PER_SH
IFT)
                        * payRate / 2;
            }
        },
        WEEKEND {
            double overtimePay(double hours, double payRate) {
                return hours * payRate / 2;
            }
        };
        private static final int HOURS_PER_SHIFT = 8;

        abstract double overtimePay(double hrs, double payRate);

        double pay(double hoursWorked, double payRate) {
            double basePay = hoursWorked * payRate;
            return basePay + overtimePay(hoursWorked, payRate);
        }
    }
}
```

(/apps/redir
utm_source

测试 banner-clicl

```
System.out.println("时薪100的人在周五工作8小时的收入：" + PayrollDay.FRIDAY.pay(
8.0, 100));
System.out.println("时薪100的人在周六工作8小时的收入：" + PayrollDay.SATURDAY.pa
y(8.0, 100));
```

EnumSet和EnumMap

Java中提供了两个方便操作enum的工具类——EnumSet和EnumMap。

`EnumSet` 是枚举类型的高性能 `Set` 实现。它要求放入它的枚举常量必须属于同一枚举类型。
`EnumMap` 是专门为枚举类型量身定做的 `Map` 实现。虽然使用其它的Map实现（如HashMap）也能完成枚举类型实例到值得映射，但是使用EnumMap会更加高效：它只能接收同一枚举类型的实例作为键值，并且由于枚举类型实例的数量相对固定并且有限，所以EnumMap使用数组来存放与枚举类型对应的值。这使得EnumMap的效率非常高。

```
// EnumSet的使用
System.out.println("EnumSet展示");
EnumSet<ErrorCodeEn> errSet = EnumSet.allOf(ErrorCodeEn.class);
for (ErrorCodeEn e : errSet) {
    System.out.println(e.name() + " : " + e.ordinal());
}

// EnumMap的使用
System.out.println("EnumMap展示");
EnumMap<StateMachine.Signal, String> errMap = new EnumMap(StateMachine.Sign
al.class);
errMap.put(StateMachine.Signal.RED, "红灯");
errMap.put(StateMachine.Signal.YELLOW, "黄灯");
errMap.put(StateMachine.Signal.GREEN, "绿灯");
for (Iterator<Map.Entry<StateMachine.Signal, String>> iter = errMap.entrySe
t().iterator(); iter.hasNext();) {
    Map.Entry<StateMachine.Signal, String> entry = iter.next();
    System.out.println(entry.getKey().name() + " : " + entry.getValue());
}
```


您的鼓励是我持续更新的动力。

赞赏支持

静默虚空 (/u/e336bc023b9d) ♂
写了 122641 字，被 347 人关注，获得了 491 个喜欢
(/u/e336bc023b9d)
+关注

一线搬砖工。 前端+后端，两手都想抓，两手都不硬。 我的信条：Talk is cheap, show me the code.

喜欢 17

更多分享

登录 (/sign后发表评论source=desktop&utm_medium=not-signed-in-com

1条评论 只看作者 按时间倒序 按时间正序

leilifengxingmw (/u/3c9c20cce3d0)
2楼 · 2018.06.03 22:22
(/u/3c9c20cce3d0)
写的很好😄

赞 回复

被以下专题收入，发现更多相似内容

 代码改变世界 (/c/0f5e015fc36c?utm_source=desktop&utm_medium=notes-included-collection)

 今日看点 (/c/3sT4qY?utm_source=desktop&utm_medium=notes-included-collection)

 程序员 (/c/NEt52a?utm_source=desktop&utm_medium=notes-included-collection)

 java (/c/507fc303443e?utm_source=desktop&utm_medium=notes-included-collection)

 程序员 (/c/5dba97da56e3?utm_source=desktop&utm_medium=notes-included-collection)

推荐阅读 更多精彩内容 > (/)

Java初级面试题 (/p/08153f5678de?utm_campaign=maleskine&utm_c...

1. Java基础部分 基础部分的顺序：基本语法，类相关的语法，内部类的语法，继承相关的语法，异常的语法，线程的语法，集合的语法，io的语法，虚拟机方面的语法。 1、一个".java"源文件中是否可以包括多个

 会飞的鱼69 (/u/0ac615e458d5?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

百战程序员V1.2——尚学堂旗下高端培训_ Java1573题 (/p/49ad52bd54...

百战程序员_ Java1573题 QQ群：561832648489034603 掌握80%年薪20万掌握50%年薪10万 全程项目穿插, 从易到难，含17个项目视频和资料持续更新，请关注www.itbaizhan.com 国内最牛七星级团队马士

 Albert陈凯 (/u/185a3c553fc6?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

Effective Java一书笔记 (/p/e125acdaf45d?utm_campaign=maleskine...

对象的创建与销毁 Item 1: 使用static工厂方法，而不是构造函数创建对象：仅仅是创建对象的方法，并非Factory Pattern优点命名、接口理解更高效，通过工厂方法的函数名，而不是参数列表来表达其语义

 孙小磊 (/u/11b0c761d5c7?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

(/p/207855f0b00a?

utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendatio

《花样旅行》想让明星前一秒美滋滋别墅派对，后一刻惨兮兮泥地打滚？...

播出平台：韩国，SBS 播出时段：深夜时段23:20 节目时长：55min 播出格式：试播，共两期 节目简介
一档观众可直接参与的新概念旅行节目。 每期节目6位嘉宾分成两组进行三天两夜的旅行，并通过观众5次

小型想象媒体实验室 (/u/010583d61d50?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

---

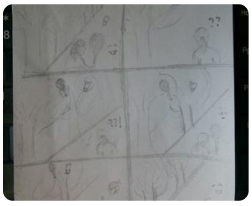
致那些没有存在感得人 (/p/5dd7d76fd414?utm_campaign=maleskine&...

一般来说，具备独立精神的人，大多不爱抱怨，也不爱迁怒他人。他们往往与自己死磕，注重内心的感
受，在每一次困境之时都自省内心，有时甚至甘之如饴。他人的温暖，适量即可，多了便是负担，没有也

小猪猪笨笨笨 (/u/8e8d69be938f?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

---

(/p/b572a2a0bf10?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendatio

午间打球小插曲 (/p/b572a2a0bf10?utm_campaign=maleskine&utm_co...

阳光大好，肚子饱饱， 拿起球拍，出去“叫嚣”， 拍起球落，满是烟硝， 汗水挥洒，羽毛乱飘。 ------ 午
间打球小记 P.S: 有氧(仰)运动，对低头族最好

尘间小妖 (/u/8a21eae6f248?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

---

想多或是想少 (/p/ca173cacaf32?utm_campaign=maleskine&utm_cont...

由于天性或者说是规律，大家总是觉得女生（人）总是想太多了。事业，爱情，家庭，友谊等等，当说不
清或没有理由得时候就开始以这个为理由（是不是好无辜……）当说客。这其中得关系也确有缘由，想多有

周周周啊啊啊z (/u/b89c500ba502?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio