1. enum 的全称为 enumeration， 是 JDK 1.5 中引入的新特性，存放在 java.lang 包中。

枚举可以理解为列举所有，穷举的意思。

比如：一年有 12 个月，把 12 个月全部列举出来；一年有四季，把四季全部列举出来；一周有 7 天，把七天全部列举出来；等等...

2. 枚举和普通类的区别：

A.使用的关键字不同，枚举使用关键字是 enum，普通类使用的关键字是 class;

B.普通类可以使用 extends 去继承其他类，而 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）。

C.枚举通常具有枚举实例，实例之间以逗号分隔，以分号结尾！

枚举和普通类的联系：

A.可以把 enum 看成是一个普通的 class;

B.枚举也可以有自己的属性和方法，以及构造函数等；

C.枚举也可以像普通类一样去实现接口；

3. 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。

定义枚举：

在 JDK1.5 之前，我们定义常量都是： public static final.... 。现在有了枚举，可以把相关的常量分组到一个枚举类型里，而且枚举提供了比常量更多的方法。

```
public enum EnumTest {
  MON, TUE, WED, THU, FRI, SAT, SUN;
}
```

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); ...
```

4. 遍历、switch 等常用操作

枚举遍历：

```
for (EnumTest e : EnumTest.values()) {
        System.out.println(e.toString());
```

```
        }
```

JDK1.6 之前的 switch 语句只支持 int,char,enum 类型，使用枚举，能让我们的代码可读性更强。

```
        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;
        }
```

5. 给 enum 自定义属性和方法

如果打算自定义自己的方法，那么必须在 enum 实例序列的最后添加一个分号。

而且 Java 要求必须先定义 enum 实例。

```
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;
        }
    }; //枚举实例，注意是逗号，分隔，分号;结尾!

    private int value; //成员变量

    private EnumTest(int value) { //构造方法
        this.value = value;
    }

  //普通方法
```

```
    public int getValue() {
        return value;
    }
    public boolean isRest() {
        return false;
    }
}
```

6. 实现接口

所有的枚举都继承自 java.lang.Enum 类。由于 Java 不支持多继承，所以枚举对象不能再继承其他类。

```
1. public interface Behaviour {
2.     void print();
3.     String getInfo();
4.  }
5.
6.  public enum Color implements Behaviour {
7.     RED("红色", 1), GREEN("绿色", 2), BLANK("白色", 3), YELLO("黄色", 4);
8.     // 成员变量
9.     private String name;
10.    private int index;
11.
12.    // 构造方法
13.    private Color(String name, int index) {
14.        this.name = name;
15.        this.index = index;
16.    }
```

```
17.
18.     // 接口方法
19.     @Override
20.     public String getInfo() {
21.         return this.name;
22.     }
23.
24.     // 接口方法
25.     @Override
26.     public void print() {
27.         System.out.println(this.index + ":" + this.name);
28.     }
29. }
```

7. 使用接口组织枚举

```
1. public interface Food {
2.     enum Coffee implements Food {
3.         BLACK_COFFEE, DECAF_COFFEE, LATTE, CAPPUCCINO
4.     }
5.
6.     enum Dessert implements Food {
7.         FRUIT, CAKE, GELATO
8.     }
```

9. }

8. 关于枚举集合的使用

java.util.EnumSet 和 java.util.EnumMap 是两个枚举集合。

EnumSet 保证集合中的元素不重复;EnumMap 中的 key 是 enum 类型，而 value 则可以是任意类型。

```
// EnumSet 的使用
EnumSet<EnumTest> weekSet = EnumSet.allOf(EnumTest.class);
for (EnumTest day : weekSet) {
    System.out.println(day);
}
// EnumMap 的使用
EnumMap<WeekEnum2, String> weekMap = new EnumMap(WeekEnum2.class);
weekMap.put(WeekEnum2.MONDAY, "一");
weekMap.put(WeekEnum2.TUESDAY, "二");
weekMap.put(WeekEnum2.WEDNESDAY, "三");
weekMap.put(WeekEnum2.THURDAY, "四");
weekMap.put(WeekEnum2.FRIDAY, "五");
weekMap.put(WeekEnum2.SATURDAY, "六");
weekMap.put(WeekEnum2.SUNDAY, "日");
for (Entry<WeekEnum2, String> entry : weekMap.entrySet()) {
System.out.println(entry.getKey().getIndex() + "--" + entry.getKey().getDesc()
+ "--" + entry.getValue());
}
```

9. Enum 类实现了 Comparable 接口， int compareTo(E other) 如果枚举常量在 other 之前，则返回一个负值； 如果 this==other，则返回 0；否则，返回正值。 枚举常量的出现次序在 enum 声明中给出。（所以不能直接用<,>符号比较两个枚举值）

```
// 枚举实例的比较
WeekEnum2 enum2 = WeekEnum2.MONDAY;
int compareTo = enum2.compareTo(WeekEnum2.SATURDAY);
System.out.println("compareTo : " +compareTo); //值为-5
//值并不是标准的-1、0、1，用 switch..case 会有 bug，这一点要特别注意！
if (compareTo < 0) {
System.out.println(enum2.getDesc() + " 在前面");
} else if (compareTo > 0) {
System.out.println(enum2.getDesc() + " 在后面");
} else {
System.out.println(enum2.getDesc() + " 位置一样！");
}
```

其他知识：

A.在 java 代码编辑器中 **alt+shift+s** 快速打开源码编辑的右键菜单，

例如：重写方法！

B.javap 命令：http://blog.csdn.net/hantiannan/article/details/7659904

执行反编译 javap WeekEnum 或者 javap -c WeekEnum

```
D:\GitHub\JavaBase\JavaCore\bin\com\ddb\javacore\enumdemo>javap WeekEnum.class
//以下内容为输出结果：
Compiled from "WeekEnum.java"
public final class com.ddb.javacore.enumdemo.WeekEnum extends
java.lang.Enum<com
.ddb.javacore.enumdemo.WeekEnum> {
public static final com.ddb.javacore.enumdemo.WeekEnum MONDAY;
public static final com.ddb.javacore.enumdemo.WeekEnum TUESDAY;
public static final com.ddb.javacore.enumdemo.WeekEnum WEDNESDAY;
public static final com.ddb.javacore.enumdemo.WeekEnum THURSDAY;
public static final com.ddb.javacore.enumdemo.WeekEnum FRIDAY;
public static final com.ddb.javacore.enumdemo.WeekEnum SATURDAY;
public static final com.ddb.javacore.enumdemo.WeekEnum SUNDAY;
static {};
public static com.ddb.javacore.enumdemo.WeekEnum[] values();
public static com.ddb.javacore.enumdemo.WeekEnum valueOf(java.lang.String);
}
//反编译的输出追加到指定的文件中
D:\GitHub\JavaBase\JavaCore\bin\com\ddb\javacore\enumdemo>javap
WeekEnum.class >>text.txt
D:\GitHub\JavaBase\JavaCore\bin\com\ddb\javacore\enumdemo>javap
WeekEnum2.class >>text.txt
```

参考链接：

A.java enum(枚举)使用详解 + 总结 - ido - 博客园
http://www.cnblogs.com/hyl8218/p/5088287.html
B.java enum 的用法详解 - 博客频道 - CSDN.NET
http://blog.csdn.net/yechaodechuntian/article/details/23561935?utm_source=tuicool&utm_medium=referral
C.全面解读 Java 中的枚举类型 enum 的使用_java_脚本之家
http://www.jb51.net/article/83104.htm

案例代码：

```
package com.xxx.web.errorcode;
public interface IBaseErrorEnum {
```

```
String getReason();
int getErrorCode();
String name();
}

package com.xxx.web.errorcode;
import com.xxx.web.define.ModuleTypeDefine;
import lombok.Getter;
@Getter
public enum CommDatabaseEnum implements IBaseErrorEnum {
UNKNOWN(0, "数据库操作异常"),
CONNECT_FAILED(1, "连接数据库失败"),

INSERT_ERROR(100, "插入数据失败"),
INSERT_EXIST(101, "插入数据失败,数据已经存在"),
INSERT_ITEM_REPEAT(102,"插入数据失败,插入列表中存在重复的项目:"),
CLONE_EXIST(103, "克隆数据失败,数据已经存在"),
NAME_TOO_LONG(104, "配置名称过长"),
DESC_TOO_LONG(105, "配置描述过长"),

DELETE_ERROR(200, "删除数据失败"),
DELETE_NON_EXISTENT(201, "删除数据失败, 删除对象不存在."),
DELETE_LOCKED(202, "删除数据失败,数据被锁定."),
DELETE_BIND(203, "删除数据失败,部分数据被绑定."),
DELETE_SELECTED_BIND(205, "删除数据失败,您选中的数据已经被绑定"),
DELETE_BIND_ALL(204, "删除数据失败,数据被绑定."),

UPDATE_ERROR(300, "更新数据失败"),
UPDATE_NON_EXISTENT(301, "更新数据失败, 更新对象不存在"),
UPDATE_EXISTENT(302, "更新数据失败, 更新对象已存在"),
UPDATE_ITEM_REPEAT(303,"更新数据失败,更新列表中存在重复的项目:"),

SELECT_ERROR(400, "查找数据失败"),
SELECT_NON_EXISTENT(401, "查找数据失败,查找对象不存在"),
//Used for middle table
ASSOCIATE_ERROR(600,"创建数据关联失败"),
DISASSOCIATE_ERROR(601,"解除数据关联失败");

private int errorCode;
private String reason;
~~private static final int MODULE_TYPE = ModuleTypeDefine.DATABASE;~~
~~private static int offset = ErrorCodeUtil.register(MODULE_TYPE);~~

private CommDatabaseEnum(int errorCode, String reason) {
```

```
this.errorCode = errorCode;
this.reason = reason;
}
public int getErrorCode() {
return errorCode + offset;
}
}
```