有时变量的取值只在一个有限的集合内，例如服装的尺寸只有小、中、大和超大四种。针对这种情况，可以自定义枚举类型，将所有可能的取值列举出来。例如：

```
enum Size {SMALL, MEDIUM, LARGE, EXTRA_LARGE};
Size s = Size.MEDIUM;
```

枚举类型的变量只能存储这个类型声明中给定的某个枚举值，或者特殊值`null`。

枚举类型实际上是一个类，如果需要，可以为枚举类添加构造器、方法和字段。例如：

```
public enum Size
{
    SMALL("S"), MEDIUM("M"), LARGE("L"), EXTRA_LARGE("XL");

    private String abbreviation;

    private Size(String abbreviation) {this.abbreviation = abbreviation;}
    public String getAbbreviation() {return abbreviation;}
}
```

枚举值实际上是枚举类的实例，默认调用构造器，所以需要传入参数。枚举类的构造器必须是私有的，保证不会在别处声明新的枚举值。因此，在比较两个枚举类的值时，直接使用`==`即可。

所有的枚举类都是`Enum`类的子类，继承了这个类的方法。`toString`方法返回枚举常量名，签名和用法如下：

```
/* 签名 */
String toString()
/* 用法 */
System.out.println(Size.SMALL.toString()); // 输出字符串 "SMALL"
```

`toString`方法的逆方法是静态方法`valueOf`，它返回给定类中指定名字的枚举常量。它的签名和用法如下：

```
/* 签名 */
static Enum valueOf(Class enumClass, String name)
/* 用法 */
Size s = Enum.valueOf(Size.class, "SMALL"); // 将 s 设置成 Size.SMALL
```

每个枚举类都有一个静态的`values`方法，它返回一个包含全部枚举值的数组。例如：

```
Size[] values = Size.values();
```

`ordinal`方法返回枚举常量在`enum`声明中的位置，位置从0开始计数。它的签名和用法如下：

```
/* 签名 */
int ordinal()
/* 用法 */
System.out.println(Size.MEDIUM.ordinal()); // 输出 1
```