**枚举**

请解释enum与Enum区别：

enum是⼀个关键字，使用enum定义的枚举类本质上就相对于⼀个类继承了Enum这个抽象类。

enum并不是⼀种新的结构，只是对⼀种类型的包装,继承java.lang.Enumf父类。

Enum类：

protected Enum(String name, int ordinal);//构造方法
public final String name();//枚举名字
public final int ordinal();//枚举序号
values()返回的是⼀个枚举的对象数组。

枚举本身支持switch判断

枚举还可以实现接口，这样枚举中的每⼀个对象都变成了接口对象

**注解：**

**@Override：覆写，发生在继承关系中**

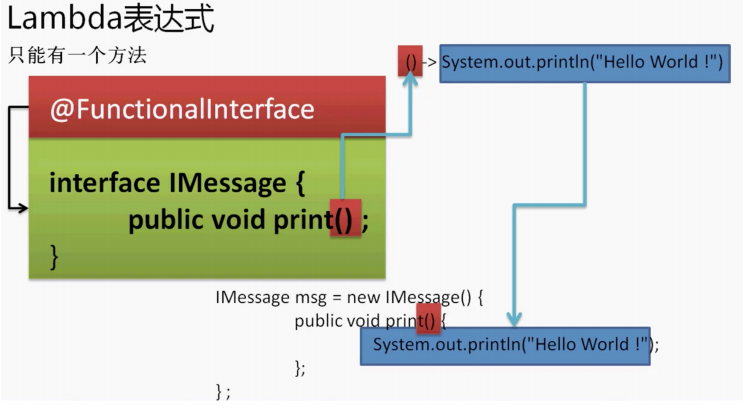
**@**[**Deprecated**](mailto:Deprecated@SuppressWarnings)**：过期申明，表示方法已经不建议使用，但使用也不会出错**

[**@SuppressWarnings**](mailto:Deprecated@SuppressWarnings)**：当调用了某些操作可能产生问题的时候就会出现警告进行，不想总提示警告，这个时候可以使用压制警告。**

**Lamda表达式**

函数式编程：接口必须只有⼀个抽象方法，@FunctionalInterface 注释表示此接口为函数式编程接口，只有一个抽象方法

单行语句：(参数) -> 单⾏语句 ;
多行语句：(参数) -> {} ;



@FunctionalInterface // 是⼀个函数式编程接⼝，只允许有⼀个抽象⽅法
interface IMessage {
public void print() ; // 这是⼀个接⼝，接⼝中的抽象⽅法必须由⼦类覆写。
}
public class TestDemo {
public static void main(String[] args) {
// 函数式编程的使⽤，⽬的还是输出⼀句话
IMessage message = () -> System.out.println("Hello World");
message.print();
}
}

**引用方法：JDK1.8开始，追加了方法引用**

1. 引用静态方法：类名称::static 方法名称 ；

2. 引用某个对象的方法：实例化对象 :: 普通方法 ；

3. 引用某个特定类的方法： 类名称 :: 普通方法 ；

4. 引用构造方法： 类名称 :: new 。

**函数式接口：**

1. 功能型函数式接口：输入⼀个数据，将数据处理后进行输出。

public interface Function<T, R> R apply(T t);
//如果现在确定操作的数据是int(T)，则可以使⽤IntFunction接⼝
例如：
Function<Integer, String> fun = String :: valueOf ;
System.out.println(fun.apply(1000));

2. 供给型函数式接口：

public interface Supplier T get();
例如：
Supplier<String> sup = "hello"::toUpperCase ;
System.out.println(sup.get());

3. 消费型函数式接口：

public interface Consumer void accept(T t);
例如：
Consumer<String> cons = System.out :: println ;
cons.accept("嘿嘿嘿");

4. 断言型接口：

public interface Predicate boolean test(T t);
例如：
Predicate<String> pre = "##123shdbs" :: startsWith ;
System.out.println(pre.test("##"));