Kotlin 泛型 Kotlin 对象表达式和对象声明

Kotlin 枚举类

枚举类最基本的用法是实现一个类型安全的枚举。

枚举常量用逗号分隔,每个枚举常量都是一个对象。

```
enum class Color{
    RED,BLACK,BLUE,GREEN,WHITE
}
```

枚举初始化

每一个枚举都是枚举类的实例，它们可以被初始化：

```
enum class Color(val rgb: Int) {
    RED(0xFF0000),
    GREEN(0x00FF00),
    BLUE(0x0000FF)
}
```

默认名称为枚举字符名，值从0开始。若需要指定值，则可以使用其构造函数：

```
enum class Shape(value:Int){
    ovel(100),
    rectangle(200)
}
```

枚举还支持以声明自己的匿名类及相应的方法、以及覆盖基类的方法。如：

```
enum class ProtocolState {
    WAITING {
        override fun signal() = TALKING
    },

    TALKING {
        override fun signal() = WAITING
    };

    abstract fun signal(): ProtocolState
}
```

如果枚举类定义任何成员，要使用分号将成员定义中的枚举常量定义分隔开

使用枚举常量

Kotlin 中的枚举类具有合成方法，允许遍历定义的枚举常量，并通过其名称获取枚举常数。

```
EnumClass.valueOf(value: String): EnumClass  // 转换指定 name 为枚举值，若未匹配成功，会抛出IllegalArgument
Exception
EnumClass.values(): Array<EnumClass>         // 以数组的形式，返回枚举值
```

获取枚举相关信息：

```
val name: String //获取枚举名称
val ordinal: Int //获取枚举值在所有枚举数组中定义的顺序
```

实例

```
enum class Color{
    RED,BLACK,BLUE,GREEN,WHITE
}

fun main(args: Array<String>) {
    var color:Color=Color.BLUE

    println(Color.values())
    println(Color.valueOf("RED"))
    println(color.name)
    println(color.ordinal)

}
```

自 Kotlin 1.1 起，可以使用 `enumValues<T>()` 和 `enumValueOf<T>()` 函数以泛型的方式访问枚举类中的常量 ：

```
enum class RGB { RED, GREEN, BLUE }

inline fun <reified T : Enum<T>> printAllValues() {
    print(enumValues<T>().joinToString { it.name })
}



fun main(args: Array<String>) {
    printAllValues<RGB>() // 输出 RED, GREEN, BLUE
}
```

Kotlin 泛型 Kotlin 对象表达式和对象声明

点我分享笔记