**Kotlin**

### 一、基础篇

1. 常见数据的类型：

Byte、Short、Int、Long、Float、Double、Boolean、Char、String

1. kotlin可以进行类型推断

var 可变变量

val 不可变变量，不是常量

可空类型不能直接赋值给非空类型，需要强制转换。反过来则可以

|  |
| --- |
| **var** age: Int = 12 **var** age2: Int? = 13age = age2!! |

没有返回值默认为Unit，该关键字可以不写

1. kotlin中字符串的比较“==”比较的是字符串的内容。

str.equals(string, true)：表示忽略字母大小写进行字符串内容比较。

kotlin中的when类似于java中的switch

|  |
| --- |
| **fun** grade(score: Int): String {  **var** result = **when** (score) {  10 -> **"$**score **is full marks"  else** -> **"still need work"** }  **return** result } |

1. 区间

|  |
| --- |
| **var** nums = 1 .. 16 *// [1, 16]* **var** nums2 = 1 *until* 100 *// [1, 100) // step步长，必须是整数* **for** (num **in** nums *step* 2) {  *print*(**"$**num **,"**) } *// 反转* **var** reversed = nums2.*reversed*() *// 区间总数* **var** count = nums2.*count*() |
| *// ---------list----------- // 泛型可进行类型推断* **var** lists = *listOf*<String>(**"1"**, **"2"**) **for** ((i, e) **in** lists.*withIndex*()) {  *println*(**"index:$**i **value:$**e**"**) }  *// ---------map-------------* **var** map = HashMap<String, String>() map[**"good"**] = **"好"** map[**"study"**] = **"学习" for** ((key, value) **in** map.**entries**) {  *println*(**"key:$**key **value:$**value**"**) } |

1. 函数表达式

|  |
| --- |
| **fun** main(args: Array<String>) {  **var** add2 = **{**x: Int , y: Int **->** x + y**}  var** add3: (Int, Int) -> Int = **{**x, y **->** x + y**}** *println*(*add*(y = 2))  *println*(add2(3,2)) } **fun** add(x: Int = 3, y: Int): Int {  **return** x + y } **fun** add1(x: Int, y: Int): Int = x + y |

1. 字符串转数字：”12”.toInt()

从键盘输入数据（返回都是字符串）：readLine()

1. 尾递归优化

|  |
| --- |
| */\*\*  \* 处理 java.lang.StackOverflowError  \* 尾递归：函数的返回值必须为函数的本身。  \*/* **tailrec fun** add5(x: Int, result: Int): Int {  **if** (x == 1) {  **return** 1 + result  } **else** {  **return** *add5*(x -1, result + x)  } } |

1. open关键字修饰的类允许被继承，修饰的方法允许子类重写。

当一个类或方法被abstract修饰时，就不再需要open修饰。

is关键字相当于instanceof

|  |
| --- |
| */\*\*  \* 静态代理  \*/* **interface** IWashBowl {  **fun** washing() } */\*\*  \* 单例  \*/* **object** BigHeadSon: IWashBowl {  **override fun** washing() {  *println*(**"son washing is paid 1 dollar once"**);  } }  **class** SmallFather: IWashBowl **by** BigHeadSon {  **override fun** washing() {  *println*(**"father"**)  BigHeadSon.washing()  *println*(**"son helping father to wash"**)  } }  **fun** main(args: Array<String>) {  **var** father = SmallFather(); } |

1. 枚举 enum class

印章类 sealed class：子类类型有限的class。印章类不能直接实例化该类，需要实例化其子类。

匿名函数

|  |
| --- |
| names.*forEach* **{** name **->** *println*(name) **}** names.*forEach* **{** *println*(**it**) **}** |

1. 图像处理

|  |
| --- |
| **fun** main(args: Array<String>) {  **var** image = BufferedImage(100, 100, BufferedImage.*TYPE\_INT\_RGB*)  **var** width = 0 .. 99  **var** height = 0 ..99  *// 方法一： // for (w in width) { // for (h in height) { // image.setRGB(w, h, 0xff0000) // } // }   // 方法二 // image.apply { // for (w in width) { // for (h in height) { // setRGB(w, h, 0xff0000) // } // } // }   // 方法三 // for (w in width) { // height.forEach { image.setRGB(w, it, 0xffff00) } // }   // 方法四* width.*forEach* **{  val** w = **it** height.*forEach* **{** image.setRGB(w, **it**, 0x00ffff) **}  }** ImageIO.write(image, **"bmp"**, File(**"c.bmp"**)) } |

1. 高级函数：

iterable.maxBy( (T) -> (R) ) 得到容器里某一项最大的对象

minBy( (T) -> (R) )

filter( (T) -> Boolean )

map( (T) -> (R) ): List<R>

any( (T) -> Boolean ): Boolean 是否有

count( (T) -> Boolean ): Int

find( (T) -> Boolean ): T? 查找并返回第一个满足条件的对象

groupBy( (T) -> K ): Map<K, List<T>> 按照条件分组

1. DSL（领域特定语言）：扩展函数、中缀表达式

中缀表达式可使代码中点的调用变成空格。（infix）

1. kotlin编译

kotlin编译后会在原文件名后加上Kt，形成class文件。

编译：kotlinc Hello.kt

运行：kotlin HelloKt

反编译：javap HelloKt.class

|  |
| --- |
| // kotlin反编译  public final class HelloKt {  public static final void main(java.lang.String[]);  } |
| // java反编译  public class Hello {  public Hello();  public static void main(java.lang.String[]);  } |

全部进行反编译：javap -c HelloKt.class

将kotlin文件打包为一个jar：kotlinc Hello.kt -include-runtime -d HelloKt.jar

解压：jar -xvf Hello.jar

1. 与java的一些差别
2. java中的包名必须和物理磁盘上的文件夹名相同，koltin不是必须的，是一个逻辑上的概念。
3. kotlin允许块注释的嵌套，java不允许。
4. kotlin包导入时允许使用别名。
5. **val** max: Int = **if** (x > y) x **else** y
6. idea快捷键ctrl + Shife + 回车：补全末尾括号

### 二、gradle