《Java编程思想》的作者Bruce Eckel认为，Kotlin或将取代Java

阿里巴巴资深程序员呕心沥血之作，揭秘Kotlin编程的精华

全面涵盖Kotlin基础语法、进阶实战技巧和项目案例开发等实用内容

Kotlin

从入门到进阶实战

陈光剑　编著

清华大学出版社

Kotlin
从入门到进阶实战

陈光剑 编著

清华大学出版社
北 京

内 容 简 介

本书从 Kotlin 语言的基础语法讲起，逐步深入到 Kotlin 进阶实战，并在最后配合项目实战案例，重点介绍了使用 Kotlin+Spring Boot 进行服务端开发和使用 Kotlin 进行 Android 应用程序开发的内容，让读者不但可以系统地学习 Kotlin 编程的相关知识，而且还能对 Kotlin 应用开发有更为深入的理解。

本书分为 14 章，涵盖的主要内容有 Kotlin 简介，Kotlin 语法基础，类型系统与可空类型，类与面向对象编程，函数与函数式编程，扩展函数与属性，集合类，泛型，文件 I/O 操作、正则表达式与多线程，使用 Kotlin 创建 DSL，运算符重载与约定，元编程、注解与反射，Kotlin 集成 Spring Boot 服务端开发，使用 Kotlin 进行 Android 开发。

本书内容通俗易懂，案例丰富，实用性强，特别适合 Kotlin 语言的入门读者和进阶读者阅读，也适合 Android 程序员、Java 程序员等其他编程爱好者阅读，还适合作为相关培训机构的教材。

图书在版编目（CIP）数据

Kotlin 从入门到进阶实战 / 陈光剑编著. —北京：清华大学出版社，2018
ISBN 978-7-302-50872-4

Ⅰ. ①K…　Ⅱ. ①陈…　Ⅲ. ①JAVA 语言－程序设计　Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字（2018）第 181509 号

责任编辑：杨如林
封面设计：欧振旭
责任校对：徐俊伟
责任印制：李红英

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn, http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：三河市金元印装有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　印　张：17　　字　　数：428 千字
版　　次：2018 年 9 月第 1 版　　印　　次：2018 年 9 月第 1 次印刷
定　　价：69.80 元

产品编号：080566-01

前　言

当下，互联网、大数据和云计算迅猛发展，数以百万计的应用程序在服务器和移动端运行。这些应用程序的开发语言有很大一部分是用软件界已经流行了 20 年之久的主力编程语言 Java 编写的。

毫无疑问，Java 语言历史悠久，影响力巨大。历经 20 多年的发展，它已经成为一门非常成熟的编程语言，性能强大而稳定。Java 虚拟机 JVM 的生态也繁荣昌盛，经久不衰。但 Java 也背负着历史的包袱，如它有空指针、语法啰嗦和不支持一等函数等缺点。如果用一辆汽车来比喻编程语言，Java 拥有一个高效而可靠的发动机，但其防抱死刹车系统和动力转向系统却不是那么可控。Java 语言在使用时需要小心检查可能出现的空指针，还要处理异常、重复生成冗长而单调的样板代码行等问题。

对于开发人员而言，编程语言的防危性（safety）和安全性（security）是至关重要的。要是有一门语言既能继承 Java 的所有优点及其强大而完备的生态库，又能更加简单、安全和可控，那真是再好不过了。我们很高兴地看到，Kotlin 就是一门这样的语言。

目前，图书市场上 Kotlin 相关图书还很少，尤其是实用性强的书更是凤毛麟角。为了帮助广大的编程人员系统地学习这门开发语言，笔者编写了本书。本书从 Kotlin 语言的基础语法讲起，逐步介绍了 Kotlin 的扩展函数、一等函数支持、Lambda 表达式、强大的 DSL 支持、运算符重载与约定、无编程、注解与反射等特性，并配合项目实战案例，详细介绍了使用 Kotlin+Spring Boot 进行服务端开发和使用 Kotlin 进行 Android 应用程序开发的内容。通过阅读本书，读者不但可以系统地学习 Kotlin 编程的相关知识，而且还能对 Kotlin 应用开发有更为深入的理解。

本书特色

1. 内容全面，讲解由浅入深，符合学习规律

本书内容涵盖了 Kotlin 语言的基础语法和大部分最常用的核心知识点和开发技巧，还详细介绍了两个实用性很强的项目开发案例。讲解遵循由浅入深、循序渐进的原则，让读者的学习曲线更加平滑。这样的内容梯度安排和讲解，符合读者的编程语言学习规律，可以取得较好的学习效果。

2. 图文并茂，讲解生动有趣，阅读起来不枯燥

技术学习，有时一图胜千言。本书在介绍知识点时尽量给出简单易懂的图示帮助读者理解，这使得整个学习过程变得简单、有趣。

3. 用代码示例引导学习，可以大大提高动手编程能力

本书非常注重内容的实用性和可操作性，书中重点介绍的知识点都给出了大量代码示例，并且对代码做了详细的注释和讲解，这样可以大大提高读者实际动手编程的能力。

4. 偏重于实战讲解，不涉及不常用的知识

相比笔者的另外一本书《Kotlin 极简教程》，本书内容更加偏重于 Kotlin 编程实战讲解。书中对于 Kotlin 基础知识和语言特性的讲解更加精简，重点突出；而对于编程实战中不常用的一些内容不做过多介绍，比如没有介绍目前不常用的 Kotlin Native 和实验阶段的协程（Coroutine）两个专题；但增加了在编程实践中较为常用的元编程、注解与反射，运算符重载与约定两章的内容。

5. 项目案例实用性强，可以提高项目开发水平

本书最后两章配合项目实战案例，详细介绍了使用 Kotlin+Spring Boot 进行服务端开发和使用 Kotlin 开发 Android 应用程序的相关内容。这两个项目案例可以带领读者体验实际的 Kotlin 应用开发，可以大幅度提高读者的项目实战开发水平。

本书内容

第 1 章主要介绍了 Kotlin 编程语言的基本特性、编程哲学、学习工具，以及为什么要学 Kotlin 和 JVM 语言生态等内容。

第 2 章主要介绍了 Kotlin 语法基础，主要内容包括变量和标识符、关键字与修饰符、流程控制语句、操作符与重载、包声明等内容。

第 3 章主要介绍了 Kotlin 的类型系统、可空类型、安全操作符、特殊类型、类型检测与类型转换等内容。

第 4 章主要介绍了 Kotlin 的类与面向对象编程，包括声明类、抽象类与接口、object 对象、数据类、注解、枚举和内部类等内容。

第 5 章主要介绍了 Kotlin 函数式编程，包括声明函数、Lambda 表达式、高阶函数及 Kotlin 中的特殊函数等内容。

第 6 章主要介绍了 Kotlin 扩展函数与属性，以及扩展函数的实现原理和扩展中的 this 关键字。

第 7 章主要介绍了 Kotlin 集合类，包括常用的 3 种集合类、不可变集合类、创建集合类、遍历集合中的元素、映射函数、过滤函数、排序函数和元素去重等内容。

第 8 章主要介绍了 Kotlin 的泛型，包括为何引入泛型、泛型接口、泛型类、泛型函数、类型上界、协变与逆变、out T 与 in T、类型擦除等内容。

第 9 章主要介绍了 Kotlin 语言的文件 I/O 操作、网络 I/O 操作、执行 Shell 命令、正则表达式和多线程编程等相关内容。

第 10 章主要介绍了怎样使用 Kotlin 语言创建 DSL，包括什么是 DSL、Kotlin 的 DSL 特性支持，同时实现了一个集合类的流式 Kotlin DSL 实例及一个 SQL 风格的集合类 DSL 实例。

第 11 章主要介绍了 Kotlin 的运算符重载与约定，包括什么是运算符重载、重载二元算术运算符、重载自增自减一元运算符、重载比较运算符及重载计算赋值运算符等内容。

第 12 章主要介绍了 Kotlin 元编程、注解与反射的相关内容，包括元编程简介、声明注解、使用注解、处理注解、反射、类引用、函数引用、属性引用、绑定函数、使用反射获取泛型信息等内容。

第 13 章介绍 Kotlin 集成 Spring Boot 服务端开发，首先用 Spring Boot 快速开发一个 Restful Hello World 示例，然后给出了一个完整的图片爬虫 Web 应用项目案例。

第 14 章介绍如何使用 Kotlin 进行 Android 开发，首先给出了一个简单的 Kotlin 版本的 Hello World Android 示例程序，然后详细介绍了用 Kotlin 开发一个电影指南 Android 应用程序综合项目案例。

本书读者对象

- Kotlin 入门人员；
- Kotlin 进阶开发人员；
- Android 程序员；
- Java 程序员；
- 其他编程爱好者；
- 相关培训机构的学员。

本书源程序获取方式

本书涉及的源代码需要读者自行下载。请登录清华大学出版社网站 www.tup.com.cn，搜索到本书页面，在页面上找到“资源下载”栏目，然后单击“课件下载”或者“网络资源”按钮即可下载。

作者与致谢

笔者现就职于阿里巴巴集团，曾经参与了多种平台工具的开发，主要使用 Java、Android、Scala、Groovy 和 Kotlin 等语言或工具进行领域建模、架构设计和工具开发等，积累了大量经验。

感谢在本书写作过程中提供过帮助的各位朋友！也感谢在本书出版过程中提供过帮助的各位编辑，没有你们的付出，本书就不会顺利和读者见面！最后感谢各位读者选择了本书，祝你们学习愉快！

虽然笔者对书中所述内容都尽量核实，并多次进行文字校对，但因时间所限，加之水平所限，书中可能还存在疏漏和错误，敬请广大读者批评指正。联系 E-mail：bookservice2008@163.com。

陈光剑
于杭州

目　录

第1章　Kotlin是什么

Kotlin是一种非研究性并且非常务实的工业级编程语言，它的使命就是帮助程序员解决实际工程实践中的问题。使用Kotlin语言让Java程序员的工作变得更轻松，Java语言中的那些空指针错误、浪费时间的冗长的样板代码、啰嗦的语法限制等，在Kotlin语言中统统消失。Kotlin语言简单、务实，语法简洁而强大，安全且表达力强，极富生产力。

本章首先简单介绍Kotlin语言的发展历史和语言特性，然后简述为什么要学习Kotlin语言，最后简要介绍JVM语言家族。

1.1　初识Kotlin

Kotlin是一种基于JVM的静态类型编程语言。Kotlin从开始推出至今已经有7年，2016年官方正式发布了首个稳定版本。Kotlin发展简史如下：

- 2011年7月，JetBrains推出Kotlin项目。
- 2012年2月，JetBrains以Apache 2许可证开源此项目。
- 2016年2月15日，Kotlin v1.0（第1个官方稳定版本）发布。
- 2017 Google I/O大会上，Kotlin“转正”。

Kotlin具备类型推断、多范式支持、可空性表达、扩展函数、模式匹配等诸多下一代编程语言特性。

Kotlin的编译器kompiler可以被独立出来并嵌入到Maven、Ant或Gradle工具链中。这使得在IDE中开发的代码能够利用已有的机制来构建，可以在新环境中自由使用。

让我们从Hello World开始。与C、C++、Java语言一样，Kotlin程序的入口点是一个名为main()的函数，它传递一个包含任何命令行参数的数组。代码示例如下：

```
package com.easy.kotlin                    //(1)
fun main(args: Array<String>) {            //(2)
    val name = "World"
    println("Hello,$name!")                //(3)
}
```

上面的代码简单说明如下。

（1）：Kotlin中包package的使用与Java基本相同。有一点不同的是Kotlin的package命名可以与包路径不同。

（2）：Kotlin变量声明args:Array类似于Pascal，先写变量名args，冒号隔开，再在后面写变量的类型Array。与Scala和Groovy一样，代码行末尾的分号是可选的，在大多

数情况下，编译器根据换行符就能够推断语句已经结束。Kotlin 中使用 fun 关键字声明函数（方法），充满乐趣的 fun。

（3）：Kotlin 中的打印函数是 println()（虽然背后封装的仍然是 Java 的 System.out.println() 方法）。Kotlin 中支持字符串模板$name，如果是表达式，则使用${expression}语法。

1.2 语 言 特 性

人们为什么喜欢 Kotlin？Kotlin 为什么值得我们去学习？下面是一个不完全的清单列表。

- 与 Java 及 JVM 的完全互操作性；
- 多平台：适合 Android、浏览器（JavaScript）和本地系统编程（native）；
- 语法简洁不啰嗦（便于学习）；
- 富于表现力和高效的生产力；
- 支持类型推断。例如，我们可以只写 val number=23，编译器会推断这是一个 Int；
- 可以使用数据类（data class）以极简的方式创建 POJO；
- 运算符重载相当简单；
- 快速、方便地扩展内置类、自定义类的函数与属性；
- 区分可空类型和不可空类型。直接在编译期语法层面检查可空类型，提供空安全保障；
- Kotlin 含有功能丰富的集合类 Stream API；
- 集成扩展了简单实用的文件 I/O、正则匹配、线程等工具类；
- 提供了实用强大的函数式编程支持：一等函数支持，Lambda 表达式、高阶函数等；
- 能够轻松、方便地创建 DSL；
- 使用更加轻量级的协程进行并发编程；
- IntelliJIDEA 开发工具的一等支持；
- Android 开发有 Android Studio 3 内置原生支持；
- 提供的 Anko 库（https://github.com/Kotlin/anko）使得 Android 开发速度更快，充满更多的乐趣等。

Kotlin 的优势是既有 Java 的完整生态（Kotlin 完全无缝使用各类 Java API 框架库），又有现代语言的高级特性（语法糖）。

Kotlin 语言的设计初衷之一是为了 JetBrains 团队内部使用，旨在帮助公司降低成本。用过 IntelliJ IDEA 的程序员都知道 JetBrains 团队的出品皆是良品。毫无疑问，Kotlin 的设计是务实的。发展和促进 Kotlin 的好处大于其成本，在这个过程中，Kotlin 已经演变成了一个 JetBrains 的效率工具，其显著的务实特性吸引了一大批 Java 程序员，并成为了 JetBrains 工具生态系统中重要的一员。

在未来几年内，Kotlin 有望成为主要的非 Java 的 JVM 语言，甚至有一天成为下一个“Java”语言。可以预测的是，Kotlin 将大大提升整个 Java 互联网开发者的效率和质量。

Kotlin 语言的特性可以简单概括为以下几方面。

1. 实用主义（Pragmatic）

务实、注重工程实践性。我们经常会听到人们说编程是数学，或者是工程，是艺术，是科学，这些说法都是很有道理的。Kotlin 是一门偏重工程实践、编程上有极简风格的语言。

2. 极简主义（Minimalist）

Kotlin 语法简洁优雅不啰嗦，类型系统中一切皆是引用（reference）。

3. 空安全（Null Safety）

Kotlin 中有一个简单完备的类型系统来支持空安全。

4. 多范式（multi-paradigm）

Kotlin 同时支持 OOP 与 FP 编程范式。各种编程风格的组合可以让我们更加直接地表达算法思想和解决问题的方案，可以赋予我们在思考上有更大的自由度和灵活性。

5. 可扩展

Kotlin 可直接扩展类的函数与属性（extension functions & properties）。这与我们在 Java 中经常写的 util 类是完全不一样的体验！Kotlin 是一种非常注重用户体验的语言。

6. 高阶函数与闭包（higher-order functions & closures）

Kotlin 的类型中，函数类型（function type）也是一等类型（first class type）。在 Kotlin 中可以把函数当成值进行传递，这直接赋予了 Kotlin 函数式编程的特性，使用 Kotlin 可以写出一些非常“优雅”的代码。

7. 支持快速实现DSL

有了扩展函数、闭包等特性的支持，使用 Kotlin 实现一个 DSL 将会相当简单、方便。

1.2.1　Kotlin 与 Java 完全互操作

Kotlin 是基于 JVM 平台的静态编程语言，同时在设计之初就把与 Java 的互操作性当作重要目标。正如官方网站所宣传的那样：100% interoperable with Java and Android。下面我们举个简单例子来展示 Kotlin 中使用 Java 的 ArrayList 类与使用 JUnit 测试框架进行单元测试。代码示例如下：

```
fun getArrayList(): List<String> {      //(1)函数声明
    val arrayList = ArrayList<String>() //(2)Kotlin 中直接调用 Java 的 API
    arrayList.add("A")
    arrayList.add("B")
```

```
    arrayList.add("C")
    return arrayList
}
```

代码说明如下。

（1）：声明了一个返回 List<String>的函数，我们看到在 Kotlin 中使用 fun 关键字来声明函数。

（2）：创建了一个 ArrayList<String>对象，我们可以看到，在 Kotlin 中创建对象不再使用 new 关键字了，尖括号里面的 String 是泛型信息。该语法与 Java 语言基本类似。关于集合类与泛型的相关内容，将在第 7 章和第 8 章中具体介绍。

下面使用 JUnit 框架进行单元测试。代码如下：

```
package com.easy.kotlin                          //(3)包声明

import org.junit.Assert
import org.junit.Test
import org.junit.runner.RunWith
import org.junit.runners.JUnit4                  //(4)导入 JUnit 的类

@RunWith(JUnit4::class)     //(5) 直接使用 Java 生态库 JUnit 中的注解@RunWith
class FullJavaInteroperabilityTest {
    @Test                                        //(6)标记这是一个测试方法
    fun test() {
        val list = getArrayList()                //(7)调用被测试函数
        Assert.assertTrue(list.size == 3)        //(8)断言
    }
}
```

代码说明如下。

（3）：是包声明，使用 package 关键字。

（4）：使用 import 导入 JUnit4 类。

（5）：Kotlin 中使用@RunWith 注解，方式与 Java 语法类似。注解中的参数是 JUnit4::class，是 JUnit4 类的引用。我们将在第 12 章中介绍注解与反射。

（6）：使用 JUnit 的@Test 注解来标注这是一个测试方法。

（7）：调用被测试函数 getArrayList()。

（8）：使用 JUnit 的 Assert 类的 API 进行断言操作。

1.2.2 扩展函数与扩展属性

扩展函数与扩展属性的“好玩”之处在于，可以在不修改原来类的条件下自定义函数和属性，使它们表现得就像是属于这个类一样。例如，我们给 String 类型扩展一个返回字符串首字母的 firstChar()函数，代码如下：

```
fun String.firstChar(): String {    //给 String 类扩展一个 firstChar()函数
    if (this.length == 0) {         //这里的 this 代表调用者对象
        return ""
    }
    return this[0].toString()       //返回下标为 0 的字符并转成 String 类型
}
```

然后就可以在代码中直接这样调用该函数：

```
"abc".firstChar()                    //调用我们自定义的扩展函数
```

代码显得相当简洁。

1.2.3　不可空类型与空安全

使用 Kotlin 编程比 Java 更加安全，至少在空指针问题上写起代码来会更加“开心”。Kotlin 中引入了不可空类型与可空类型来明确声明一个变量是否可能为 null，同时在编译期通过类型是否匹配来检查空指针异常，大大降低了空指针异常出现的概率。同时，Kotlin 还提供了 Elvis 操作符、安全调用符等极简的语法格式，使开发者从 Java 的 null 防御式编程中被解放出来。例如下面的这段代码：

```
>>> var a = "abc"    //声明一个字符串,编译器会默认推断变量 a 的类型为不可空的 String
>>> a = null         //不可空类型不能赋值为 null
error: null can not be a value of a non-null type String
a = null
    ^
```

当声明了不可空类型 String 的变量 a 后，在后面使用变量 a 的代码中就不能给 a 赋值为 null。如果给 a 赋值 null，编译器会直接报错。而这个时候，如果我们想声明一个可空的 String 类型，可以这样写：

```
var b:String? = "abc"   //声明一个可空的 String?类型
```

但是这个时候，如果想调用变量 b 的方法，就不能直接像下面这样写了：

```
>>> b.length          //可空类型不能直接调用方法，需要使用安全调用符?.或者断言调用!!.
error: only safe (?.) or non-null asserted (!!.) calls are allowed on a
nullable receiver of type String?
b.length
 ^
```

我们可以看到，上面的代码运行报错：

```
可空类型 String? 只有使用安全调用符(?.)和非空断言调用符(!!)才允许调用其方法。
```

如果在 IDE 中会直接提示报错，编译不通过。上面的代码可以使用安全调用符进行如下改写：

```
>>> b?.length   //使用安全调用符
3
>>> b=null
>>> b?.length   //null 对象使用安全调用符访问 length 属性，直接返回 null
null
```

这个问号确实非常简洁易懂，同时能够时刻提醒我们：这个调用者有可能是 null 的。这个语法明显比 Java 8 中引入的 Optional<String>更加简单、直接。