

PROGRAMMER AVEC KOTLIN

PROGRAMMER AVEC KOTLIN

Josh Skeen
David Greenhalgh

Traduit de l'américain par Dominique Maniez

DUNOD

Traduction autorisée depuis l'édition en anglais de l'ouvrage Kotlin Programming : The Big Nerd Ranch Guide (1re édition) par Josh Skeen et David Greenhalgh, publié par Pearson Education, Inc en 2019 dans la collection des Big Nerd Ranch Guides.

Copyright © 2018 Big Nerd Ranch LLC.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without Permission from Pearson Education, Inc.


Image de couverture : © batman6794-adobestock

La photo de la couverture représente l'une des fortifications de l'île de Kotlin.
Cette île est située en mer Baltique, à l'entrée de la baie de la Néva, à une vingtaine de kilomètres de Saint-Petersbourg.
Son extrémité orientale abrite la ville et le port fortifiés de Kronstadt.
L'île de Kotlin a donné son nom au langage Kotlin qui a été créé par une équipe de développeurs de JetBrains à Saint-Petersbourg.

Le pictogramme qui figure ci-contre mérite une explication. Son objet est d'alerter le lecteur sur la menace que représente pour l'avenir de l'écrit, particulièrement dans le domaine de l'édition technique et universitaire, le développement massif du photocopillage.

Le Code de la propriété intellectuelle du 1^{er} juillet 1992 interdit en effet expressément la photocopie à usage collectif sans autorisation des ayants droit. Or, cette pratique s'est généralisée dans les établissements d'enseignement supérieur, provoquant une baisse brutale des achats de livres et de revues, au point que la possibilité même pour les auteurs de créer des œuvres nouvelles et de les faire éditer correctement est aujourd'hui menacée.

Nous rappelons donc que toute reproduction, partielle ou totale, de la présente publication est interdite sans autorisation de l'auteur, de son éditeur ou du Centre français d'exploitation du droit de copie (CFC, 20, rue des Grands-Augustins, 75006 Paris).



© Dunod, 2020

11, Rue Paul Bert, 92240 Malakoff

www.dunod.com

ISBN 978-2-10-081099-4

Le Code de la propriété intellectuelle n'autorisant, aux termes de l'article L. 122-5, 2° et 3° a), d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite » (art. L. 122-4).

Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

Table des matières

Avant-propos	XV
1 Présentation de Kotlin	XV
1.1 <i>Les raisons du succès de Kotlin</i>	XV
1.2 <i>À qui est destiné ce livre ?</i>	XVI
1.3 <i>Comment utiliser ce livre</i>	XVI
2 Remerciements	XVIII
Chapitre 1 – Votre première application Kotlin	1
1.1 Installation d’IntelliJ IDEA	1
1.2 Votre premier projet Kotlin	2
1.2.1 <i>Création de votre premier fichier Kotlin</i>	6
1.2.2 <i>Exécution de votre fichier Kotlin</i>	8
1.3 Kotlin REPL	10
1.4 Pour les plus curieux : pourquoi utiliser IntelliJ ?	12
1.5 Pour les plus curieux : choix du support de la JVM	12
1.6 Défi : test des opérateurs arithmétiques dans REPL	13
Chapitre 2 – Variables, constantes, et types	15
2.1 Types	15
2.2 Déclaration d’une variable	16
2.3 Types prédéfinis de Kotlin	18
2.4 Variables en lecture seule	18

2.5	Inférence de type	22
2.6	Constantes de compilation	23
2.7	Inspection du bytecode Kotlin	25
2.8	Pour les plus curieux : types primitifs Java en Kotlin	27
2.9	Défi : acquisition d'une monture	28
2.10	Défi : la corne de la licorne	29
2.11	Défi : le miroir magique	29
Chapitre 3 – Conditions		31
3.1	Instruction If/else	31
3.1.1	<i>Ajout de plusieurs conditions</i>	<i>35</i>
3.1.2	<i>Instructions if/else imbriquées</i>	<i>36</i>
3.1.3	<i>Conditions plus élégantes</i>	<i>37</i>
3.2	Plages	43
3.3	Expressions when	44
3.4	Modèles de chaînes	47
3.5	Défi : test des plages	48
3.6	Défi : amélioration de l'Aura	49
3.7	Défi : format de l'état configurable	49
Chapitre 4 – Fonctions		51
4.1	Extraction de code dans des fonctions	51
4.2	Anatomie d'une fonction	54
4.2.1	<i>En-tête de fonction</i>	<i>54</i>
4.2.2	<i>Corps de la fonction</i>	<i>57</i>
4.2.3	<i>Portée de la fonction</i>	<i>57</i>
4.3	Appel d'une fonction	58
4.4	Refactoring des fonctions	59
4.5	Écriture de vos propres fonctions	61
4.6	Arguments par défaut	63
4.7	Fonctions à expression unique	64

4.8	Fonctions Unit	65
4.9	Arguments de fonction nommés	66
4.10	Pour les plus curieux : le type Nothing	67
4.11	Pour les plus curieux : fonctions de niveau fichier en Java	68
4.12	Pour les plus curieux : surcharge de fonctions	69
4.13	Pour les plus curieux : noms de fonction entre guillemets obliques	70
4.14	Défi : fonctions à expression unique	72
4.15	Défi : niveau d'ébriété au Fireball	72
4.16	Défi : état d'ébriété	72
Chapitre 5 – Fonctions anonymes et type de fonction		73
5.1	Fonctions anonymes	73
5.1.1	Type de fonction	75
5.1.2	Retours implicites	77
5.1.3	Arguments de la fonction	77
5.1.4	Mot-clé <i>it</i>	78
5.1.5	Accepter plusieurs arguments	79
5.2	Prise en charge de l'inférence de type	79
5.3	Définition d'une fonction qui accepte une fonction	80
5.3.1	Syntaxe abrégée	81
5.4	Inlining de fonction	82
5.5	Références des fonctions	84
5.6	Type de fonction en tant que type de retour	85
5.7	Pour les plus curieux : en Kotlin les lambdas sont des closures	87
5.8	Pour les plus curieux : lambdas versus classes internes anonymes	87
Chapitre 6 – Sécurité des références à null et exceptions		89
6.1	Nullabilité	89
6.2	Type null explicite de Kotlin	91
6.3	Compilation versus exécution	92
6.4	Sécurité de null	93
6.4.1	Option n°1 : l'opérateur d'appel sécurisé	94

6.4.2	Option n°2 : l'opérateur double-bang	96
6.4.3	Option n° 3 : vérification de la nullité d'une valeur avec if	97
6.5	Exceptions	99
6.5.1	Lever une exception	101
6.5.2	Exceptions personnalisées	102
6.5.3	Gestion des exceptions	103
6.6	Préconditions	105
6.7	Null : quelle est son utilité ?	107
6.8	Pour les plus curieux : exceptions vérifiées versus exceptions non vérifiées ...	108
6.9	Pour les plus curieux : comment la nullabilité est-elle appliquée ?	108
Chapitre 7 – Chaînes de caractères		111
7.1	Extraction de sous-chaînes	111
7.1.1	<i>substring</i>	111
7.1.2	<i>split</i>	114
7.2	Manipulation de chaînes	115
7.2.1	<i>Les chaînes sont immutables</i>	118
7.3	Comparaison de chaînes	118
7.4	Pour les plus curieux : Unicode	119
7.5	Pour les plus curieux : parcourir les caractères d'une chaîne	120
7.6	Défi : améliorer DragonSpeak	121
Chapitre 8 – Nombres		123
8.1	Types numériques	123
8.2	Entiers	124
8.3	Nombres décimaux	126
8.4	Conversion d'une chaîne en un type numérique	127
8.5	Conversion d'un Int en un Double	128
8.6	Mise en forme du type Double	129
8.7	Conversion d'un type Double en un type Int	130
8.8	Pour les plus curieux : manipulation des bits	132
8.9	Défi : calcul du nombre de pintes restantes	133

8.10	Défi : gestion du solde négatif	133
8.11	Défi : Dragoncoin	133
Chapitre 9 – Fonctions standard		135
9.1	apply	135
9.2	let	136
9.3	run	137
9.4	with	139
9.5	also	139
9.6	takeIf	140
9.6.1	takeUnless	141
9.7	Utilisation des fonctions standard de la bibliothèque	141
Chapitre 10 – Listes et sets		143
10.1	Listes	143
10.1.1	Accès aux éléments d'une liste	145
10.1.2	Modifier le contenu d'une liste	147
10.2	Itération	150
10.3	Lecture d'un fichier pour stocker son contenu dans une liste	155
10.4	Déstructuration	157
10.5	Ensembles (sets)	157
10.5.1	Création d'un set	158
10.5.2	Ajout d'éléments à un set	159
10.6	Boucles while	161
10.7	L'expression break	163
10.8	Conversion de collection	164
10.9	Pour les plus curieux : types Array	164
10.10	Pour les plus curieux : lecture seule versus immutabilité	165
10.11	Défi : menu de la taverne mis en forme	167
10.12	Défi : amélioration de la mise en forme du menu de la taverne	167

Chapitre 11 – Maps	169
11.1 Création d'une map	169
11.2 Accès aux valeurs d'une map	171
11.3 Ajout d'entrées dans une map	172
11.4 Modification des valeurs d'une map	173
11.5 Défi : le videur de la taverne	177
Chapitre 12 – Définition de classes	179
12.1 Définition d'une classe	179
12.2 Création des instances	180
12.3 Fonctions de classe	181
12.4 Visibilité et encapsulation	182
12.5 Propriétés des classes	183
12.5.1 <i>Getters et setters des propriétés</i>	185
12.5.2 <i>Visibilité des propriétés</i>	188
12.5.3 <i>Propriétés calculées</i>	189
12.6 Refactoring de NyetHack	190
12.7 Utilisation des packages	197
12.8 Pour les plus curieux : les propriétés var et val en détail	198
12.9 Pour les plus curieux : protection contre les conditions de course	201
12.10 Pour les plus curieux : package privé	203
Chapitre 13 – Initialisation	205
13.1 Constructeurs	206
13.1.1 <i>Constructeurs principaux</i>	206
13.1.2 <i>Définition des propriétés dans un constructeur principal</i>	207
13.1.3 <i>Constructeurs secondaires</i>	208
13.1.4 <i>Arguments par défaut</i>	210
13.1.5 <i>Arguments nommés</i>	210
13.2 Bloc initialisateur	211
13.3 Initialisation des propriétés	212
13.4 Ordre d'initialisation	215

13.5	Retardement de l'initialisation	216
13.5.1	<i>Initialisation late</i>	217
13.5.2	<i>Initialisation lazy</i>	218
13.6	Pour les plus curieux : déjouer les pièges de l'initialisation	219
13.7	Défi : l'énigme d'Excalibur	222
Chapitre 14 – Héritage		225
14.1	Définition de la classe Room	225
14.2	Création d'une sous-classe	226
14.3	Contrôle de type	233
14.4	Hiérarchie des types Kotlin	234
14.4.1	<i>Casting de type</i>	236
14.4.2	<i>Casting intelligent</i>	237
14.5	Pour les plus curieux : Any	237
Chapitre 15 – Objets		239
15.1	Mot-clé object	239
15.1.1	<i>Déclarations d'objets</i>	240
15.1.2	<i>Expressions d'objets</i>	245
15.1.3	<i>Objets compagnons</i>	246
15.2	Classes imbriquées	247
15.3	Classes de données	249
15.3.1	<i>toString</i>	251
15.3.2	<i>equals</i>	251
15.3.3	<i>copy</i>	252
15.3.4	<i>Déstructuration des déclarations</i>	252
15.4	Classes énumérées	253
15.5	Surcharge des opérateurs	255
15.6	Explorer le monde de NyetHack	257
15.7	Pour les plus curieux : définition de la comparaison structurelle	260
15.8	Pour les plus curieux : types de données algébriques	262
15.9	Défi : commande "Quitter"	265

15.10 Défi : implémenter une carte du monde	265
15.11 Défi : faire sonner la cloche	265
Chapitre 16 – Interfaces et classes abstraites	267
16.1 Définition d'une interface	267
16.2 Implémentation d'une interface	268
16.3 Implémentations par défaut	271
16.4 Classes abstraites	271
16.5 Combat dans NyetHack	274
Chapitre 17 – Génériques	279
17.1 Définition des types génériques	279
17.2 Fonctions génériques	281
17.3 Paramètres de type générique multiples	282
17.4 Contraintes génériques	284
17.5 vararg et get	285
17.6 in et out	287
17.7 Pour les plus curieux : mot-clé reified	292
Chapitre 18 – Extensions	295
18.1 Définition des fonctions d'extension	295
18.1.1 Définition d'une extension sur une superclasse	296
18.2 Fonctions d'extension génériques	297
18.3 Propriétés d'extension	299
18.4 Extensions sur des types nullable	301
18.5 Sous le capot des extensions	301
18.6 Emploi des extensions dans NyetHack	302
18.7 Définition d'un fichier d'extensions	304
18.8 Renommer une extension	306
18.9 Extensions dans la bibliothèque standard Kotlin	306
18.10 Pour les plus curieux : les littéraux de fonction avec des récepteurs	308

18.11 Défi : extension toDragonSpeak	309
18.12 Défi : extension du cadre	309
Chapitre 19 – Bases de la programmation fonctionnelle	311
19.1 Catégories de fonctions	311
19.1.1 Transformations	312
19.1.2 Filtres	314
19.1.3 Combinaisons	315
19.2 Pourquoi la programmation fonctionnelle ?	316
19.3 Séquences	318
19.4 Pour les plus curieux : profilage	320
19.5 Pour les plus curieux : Arrow.kt	320
19.6 Défi : inverser les valeurs d'une Map	321
19.7 Défi : appliquer la programmation fonctionnelle à Tavern.kt	322
19.8 Défi : fenêtre glissante	323
Chapitre 20 – Interopérabilité Java	325
20.1 Interopérabilité avec une classe Java	326
20.2 Interopérabilité et nullité	327
20.3 Mappage de types	329
20.4 Getters, setters et interopérabilité	331
20.5 Au-delà des classes	333
20.6 Exceptions et interopérabilité	341
20.7 Types de fonction en Java	344
Chapitre 21 – Créer votre première application Android avec Kotlin	347
21.1 Android Studio	347
21.1.1 Configuration de Gradle	350
21.1.2 Organisation du projet	353
21.2 Définition d'une interface utilisateur	354
21.3 Exécution de l'application sur un émulateur	356
21.4 Génération d'un personnage	357

21.5	La classe Activity	359
21.6	Liaison avec les vues	360
21.7	Propriétés synthétiques des extensions Android de Kotlin.....	362
21.8	Définition d'un écouteur de clics	364
21.9	État de l'instance enregistrée	365
21.9.1	<i>Lecture à partir de l'état de l'instance enregistrée</i>	<i>367</i>
21.10	Refactoring d'une extension	368
21.11	Pour les plus curieux : bibliothèques Android KTX et Anko.....	370
Chapitre 22 – Introduction aux coroutines		373
22.1	Analyse des données des personnages	373
22.2	Récupération des données depuis le Web	375
22.3	Le thread principal Android	378
22.4	Activation des coroutines.....	379
22.5	Spécification d'une coroutine avec GlobalScope.async.....	379
22.6	launch vs GlobalScope.async/await	381
22.7	Suspension des fonctions	381
22.8	Défi : données en direct.....	382
22.9	Défi : force minimale	382
Conclusion.....		383
Conseils.....		383
Marketing direct		383
Merci		384
Encore plus de défis.....		385
Un nouveau projet pour améliorer vos compétences : Exercism.....		385
Glossaire.....		391
Index.....		399

Avant-propos

Pour Baker, mon petit bug préféré.

— J.S.

À Rebecca, une femme motivée, patiente, belle, qui est à l'origine de ce livre.

Pour maman et papa, qui placent l'éducation au-dessus de tout.

— D.G.

1 PRÉSENTATION DE KOTLIN

En 2011, JetBrains a annoncé le développement du langage de programmation Kotlin, qui constituait une alternative à l'écriture de code dans des langages comme Java ou Scala s'exécutant sur la machine virtuelle Java. Six ans plus tard, Google a annoncé que Kotlin devenait un outil de développement officiellement pris en charge par le système d'exploitation Android.

Kotlin est rapidement passé du statut de langage promis à un brillant avenir à celui de langage faisant fonctionner des applications du premier système d'exploitation mobile. Aujourd'hui, de grandes entreprises comme Google, Uber, Netflix, Capital One, Amazon, et bien d'autres encore ont adopté Kotlin à cause de ses nombreux avantages, notamment sa syntaxe concise, ses fonctionnalités modernes et son interopérabilité transparente avec le code Java hérité.

1.1 Les raisons du succès de Kotlin

Pour comprendre l'attrait de Kotlin, vous devez d'abord comprendre le rôle de Java dans le paysage du développement logiciel moderne. Les deux langages sont étroitement liés, parce que le code Kotlin est le plus souvent écrit pour la machine virtuelle Java.

Java est un langage robuste et éprouvé qui a été l'un des langages les plus utilisés dans les bases de code de production pendant des années. Cependant, depuis la sortie de Java en 1995, les critères d'un bon langage de programmation ont évolué. Il manque

à Java de nombreuses améliorations que les développeurs travaillant avec des langages plus modernes apprécient.

Kotlin bénéficie de l'expérience acquise car certaines décisions de conception prises lors de la création de Java (et d'autres langages, comme Scala) ont mal vieilli. Kotlin a énormément évolué par rapport aux langages plus anciens et a corrigé leurs principaux défauts. Vous en apprendrez davantage dans les prochains chapitres sur la façon dont Kotlin améliore Java et offre une expérience de développement plus fiable.

Kotlin n'est pas seulement un meilleur langage pour écrire du code pour fonctionner sur la machine virtuelle Java : il est aussi un langage multiplateforme généraliste ; Kotlin peut être utilisé pour écrire des applications MacOs et Windows natives, des applications JavaScript, et, bien sûr, des applications Android. Cette indépendance par rapport aux plateformes signifie que Kotlin a une grande variété d'utilisations.

1.2 À qui est destiné ce livre ?

Nous avons écrit ce livre pour les développeurs de toutes sortes : développeurs Android expérimentés qui veulent des fonctionnalités modernes qui vont au-delà de ce que Java peut offrir, développeurs côté serveur intéressés par les fonctionnalités de Kotlin, et nouveaux développeurs qui cherchent à s'aventurer dans un langage compilé hautement performant.

La prise en charge d'Android est peut-être la raison pour laquelle vous avez ce livre entre vos mains, mais cet ouvrage ne se limite pas à la programmation Kotlin pour Android. En fait, sauf dans les deux derniers chapitres, tout le code Kotlin de ce livre est agnostique et n'est pas lié à l'infrastructure Android. Cela dit, si vous êtes intéressé par l'emploi de Kotlin pour le développement d'applications Android, ce livre montre quelques modèles courants qui font de l'écriture d'applications Android un jeu d'enfant en Kotlin.

Bien que Kotlin ait été influencé par un certain nombre d'autres langages, vous n'avez pas besoin de connaître les tenants et les aboutissants d'un autre langage pour apprendre Kotlin. De temps en temps, nous faisons le parallèle entre le code Java et le code Kotlin que vous avez écrit. Si vous avez déjà programmé en Java, cela vous aidera à comprendre la relation entre les deux langages. Si vous ne connaissez pas Java, voir comment un autre langage s'attaque aux mêmes problèmes vous aidera à saisir les principes qui ont façonné le développement de Kotlin.

1.3 Comment utiliser ce livre

Ce livre n'est pas un guide de référence. Notre objectif est de vous guider à travers les parties les plus importantes du langage de programmation Kotlin. Vous allez travailler à travers des exemples, et approfondir vos connaissances au fur et à mesure de votre progression. Pour tirer le meilleur parti de ce livre, nous vous recommandons de saisir le code des exemples. En réalisant les projets que nous proposons, vous renforcerez la mémorisation du langage et vous pourrez exploiter vos travaux d'un chapitre à l'autre.

En effet, chaque chapitre s'appuie sur les sujets évoqués dans le chapitre précédent, si bien que nous vous recommandons de lire les chapitres dans l'ordre. Même si vous sentez que vous êtes familier avec un sujet dans d'autres langages, nous vous suggérons néanmoins de lire l'intégralité de ce livre, car Kotlin gère de nombreux problèmes de manière unique. Vous commencerez par des sujets élémentaires, comme les variables et les listes, puis vous attaquerez les techniques de programmation orientée objet et fonctionnelle, ce qui vous permettra de comprendre pourquoi Kotlin est un langage si puissant. À la fin du livre, vous serez passé du statut de programmeur Kotlin débutant à celui de développeur plus avancé.

Cela dit, prenez votre temps et élargissez votre horizon : consultez la référence du langage Kotlin qui est disponible à <https://kotlinlang.org/docs/reference/> afin de compléter vos connaissances sur les sujets qui ont piqué votre curiosité, et faites des expériences.

Pour les plus curieux

La plupart des chapitres de ce livre ont une ou deux sections intitulées « Pour les plus curieux ». Beaucoup de ces sections éclairent les mécanismes sous-jacents du langage Kotlin. Les exemples décrits dans les chapitres ne dépendent pas des informations contenues dans ces sections dont nous vous conseillons néanmoins la lecture.

Défis

La plupart des chapitres se terminent par un ou plusieurs défis. Ce sont des problèmes supplémentaires à résoudre qui sont conçus pour améliorer votre compréhension de Kotlin. Nous vous encourageons à les relever afin d'améliorer votre maîtrise de Kotlin.

Conventions typographiques

Au fur et à mesure que vous créez les projets de ce livre, nous vous guiderons en vous présentant un sujet, puis en montrant comment appliquer vos nouvelles connaissances. Pour plus de clarté, nous nous en tenons aux conventions typographiques suivantes.

Les variables, les valeurs et les types s'affichent dans une police à espacement fixe. Les noms des classes, des fonctions et des interfaces sont imprimés en gras.

Tous les listings de code sont affichés dans une police à espacement fixe. Si vous devez saisir un code figurant dans un listing, ce code est imprimé en gras. Si vous devez supprimer du code dans un listing, ce code est barré. Dans l'exemple suivant, on vous demande de supprimer la ligne définissant la variable `y` et d'ajouter une variable appelée `z` :

```
var x = "Python"  
var y = "Java"  
var z = "Kotlin"
```

Prenez votre temps avec les exemples de ce livre. Une fois que vous serez familier avec la syntaxe de Kotlin, nous pensons que vous trouverez le processus

de développement en Kotlin clair, pragmatique et fluide. D'ici là, gardez à l'esprit que l'apprentissage d'un nouveau langage peut être très gratifiant.

Kotlin est un langage relativement jeune, si bien que les conventions de codage ne sont pas encore fixées. Au fil du temps, vous développerez probablement votre propre style, mais nous avons tendance à adhérer aux guides de style Kotlin de JetBrains et de Google :

- Conventions de codage de JetBrains :
<https://kotlinlang.org/docs/reference/coding-conventions.html>
- Guide de style de Google, y compris les conventions pour le code Android et l'interopérabilité :
<https://developer.android.com/kotlin/style-guide>

2 REMERCIEMENTS

Nous avons bénéficié d'un grand soutien pour écrire ce livre. Sans cette aide, ce livre ne serait pas ce qu'il est, et il n'aurait peut-être même jamais vu le jour, si bien que nous devons exprimer notre gratitude envers de nombreuses personnes.

Tout d'abord, nous devons dire merci à nos collègues de Big Nerd Ranch. Merci à Stacy Henry et Aaron Hillegass de nous avoir donné le temps et l'espace pour écrire ce livre. Il a été extrêmement gratifiant d'apprendre et d'enseigner Kotlin. Nous espérons que ce livre sera à la hauteur de la confiance et du soutien que nous avons reçus.

Des remerciements particuliers sont également dus à nos autres collègues de Big Nerd Ranch. Votre enseignement attentif a révélé de nombreux bugs dans le texte, et vos recommandations éclairées ont conduit à de nombreuses améliorations dans notre approche. C'est vraiment merveilleux d'avoir des collègues comme vous. Merci Kristin Marsicano, Bolot Kerimbaev, Brian Gardner, Chris Stewart, Paul Turner, Chris Hare, Mark Allison, Andrew Lunsford, Rafael Moreno Cesar, Eric Maxwell, Andrew Bailey, Jeremy Sherman, Christian Keur, Mikey Ward, Steve Sparks, Mark Dalrymple, CBQ, et tous les autres du Ranch qui nous ont aidés dans l'élaboration de ce travail.

Nos collègues du service marketing et des ventes ont également été déterminants. Les cours ne seraient littéralement jamais programmés sans leur travail. Merci Heather Sharpe, Mat Jackson, Rodrigo « Ram Rod » Perez-Velasco, Nicholas Stolte, Justin Williams, Dan Barker, Israel Machovec, Emily Herman, Patrick Freeman, Ian Eze et Nikki Porter. Nous ne pouvons pas réaliser ce que nous faisons sans ce que vous faites.

Des remerciements spéciaux et un peu de karma supplémentaire sont également dus à nos étudiants fantastiques qui ont été assez aventureux pour se joindre à nous pour la première version de ce cours et qui ont eu la gentillesse de nous aider à identifier nos erreurs. Sans vos commentaires et vos idées sur la façon d'améliorer le cours, ce texte ne serait pas aussi abouti. Ces valeureux étudiants sont : Santosh Katta, Abdul Hannan, Chandra Mohan, Benjamin Di Gregorio, Peng Wan, Kapil Bhalla, Girish Hanchinal, Hashan Godakanda, Mithun Mahadevan, Brittany Berlanga, Natalie Ryan, Balarka Velidi, Pranay Airan, Jacob Rogers, Jean-Luc Delpech, Dennis Lin, Kristina Thai,

Reid Baker, Setareh Lotfi, Harish Ravichandran, Matthew Knapp, Nathan Klee, Brian Lee, Heidi Muth, Martin Davidsson, Misha Burshteyn, Kyle Summers, Cameron Hill, Vidhi Shah, Fabrice Di Meglio, Jared Burrows, Riley Brewer, Michael Krause, Tyler Holland, Gajendra Singh, Pedro Sanchez, Joe Cyboski, Zach Waldowski, Noe Arzate, Allan Caine, Zack Simon, Josh Meyers, Rick Meyers, Stephanie Guevara, Abdulrahman Alshmrani, Robert Edwards, Maribel Montejano et Mohammad Yusuf.

Nous tenons à adresser des remerciements spéciaux à nos collègues et membres de la communauté Android qui nous ont aidés à tester l'exactitude du livre, sa clarté et sa lisibilité. Sans votre regard extérieur, nous aurions eu du mal à achever ce livre. Merci Jon Reeve, Bill Phillips, Matthew Compton, Vishnu Rajeevan, Scott Stanlick, Alex Lumans, Shauvik Choudhary et Jason Atwood.

Nous devons également remercier les nombreuses personnes talentueuses qui ont travaillé sur le livre avec nous. Elizabeth Holaday, notre éditrice, a contribué à améliorer le manuscrit et à diminuer ses faiblesses. Anna Bentley, notre relectrice, a trouvé et corrigé les erreurs et, au bout du compte, elle nous a fait paraître plus intelligents que nous ne le sommes.

Enfin, merci à tous nos étudiants. Être votre professeur nous offre aussi la possibilité d'être étudiant, à bien des égards, et nous vous en sommes infiniment reconnaissants. L'enseignement fait partie des plus grandes choses que nous faisons, et cela a été un plaisir de travailler avec vous. Nous espérons que la qualité de ce livre correspond à votre enthousiasme et à votre détermination.

1

Votre première application Kotlin

Dans ce chapitre, vous allez écrire votre premier programme Kotlin à l'aide d'IntelliJ IDEA. Tout en accomplissant ce rituel qui est un passage obligé dans les ouvrages de programmation, vous vous familiariserez avec votre environnement de développement en créant un nouveau projet Kotlin, en écrivant et en exécutant du code Kotlin, puis en inspectant le résultat. Le projet que vous allez créer dans ce chapitre servira de bac à sable pour tester en douceur les nouveaux concepts que vous aborderez tout au long de ce livre.

1.1 INSTALLATION D'INTELLIJ IDEA

IntelliJ IDEA est un environnement de développement intégré (EDI) pour Kotlin créé par JetBrains (qui a également inventé le langage Kotlin). Pour commencer, téléchargez IntelliJ IDEA Community Edition à partir du site Web de JetBrains à l'adresse <https://www.jetbrains.com/fr-fr/idea/download> (figure 1.1).

Une fois que le fichier d'installation est téléchargé, suivez les instructions relatives à votre plateforme qui sont disponibles à :

<https://www.jetbrains.com/help/idea/installation-guide.html>

IntelliJ IDEA, que l'on appelle en abrégé IntelliJ, vous aide à écrire du code Kotlin bien formé. Cette application simplifie également le processus de développement avec des outils intégrés pour l'exécution, le débogage, l'inspection et le refactoring de votre code. Vous pouvez en apprendre plus sur les raisons pour lesquelles nous recommandons IntelliJ pour l'écriture du code Kotlin dans la section intitulée « Pour les plus curieux : pourquoi utiliser IntelliJ ? » qui se trouve à la fin de ce chapitre.

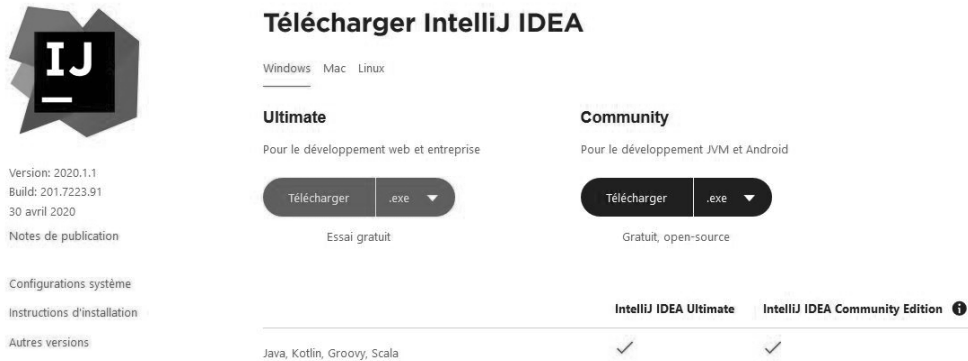


Figure 1.1 — Téléchargement de la version Community d’IntelliJ IDEA

1.2 VOTRE PREMIER PROJET KOTLIN

Félicitations, vous disposez à présent du langage de programmation Kotlin et d’un environnement de développement puissant. Maintenant, il ne vous reste plus qu’une chose à faire : apprendre à parler Kotlin couramment. Votre première mission consiste à créer un projet Kotlin.

Ouvrez IntelliJ et la boîte de dialogue de bienvenue à IntelliJ IDEA s’affichera (figure 1.2)



Figure 1.2 — Boîte de dialogue de bienvenue

Si ce n'est pas la première fois que vous ouvrez IntelliJ depuis que vous l'avez installé, vous pouvez être dirigé directement vers le dernier projet que vous avez ouvert. Pour revenir à la boîte de dialogue de bienvenue, fermez le projet à l'aide de la commande **File** → **Close Project**.

Cliquez sur **Create New Project**. IntelliJ affiche la boîte de dialogue **New Project**. Dans la boîte de dialogue du nouveau projet, sélectionnez **Kotlin** dans le volet gauche, puis **JVM|IDEA** dans le volet droit (figure 1.3)

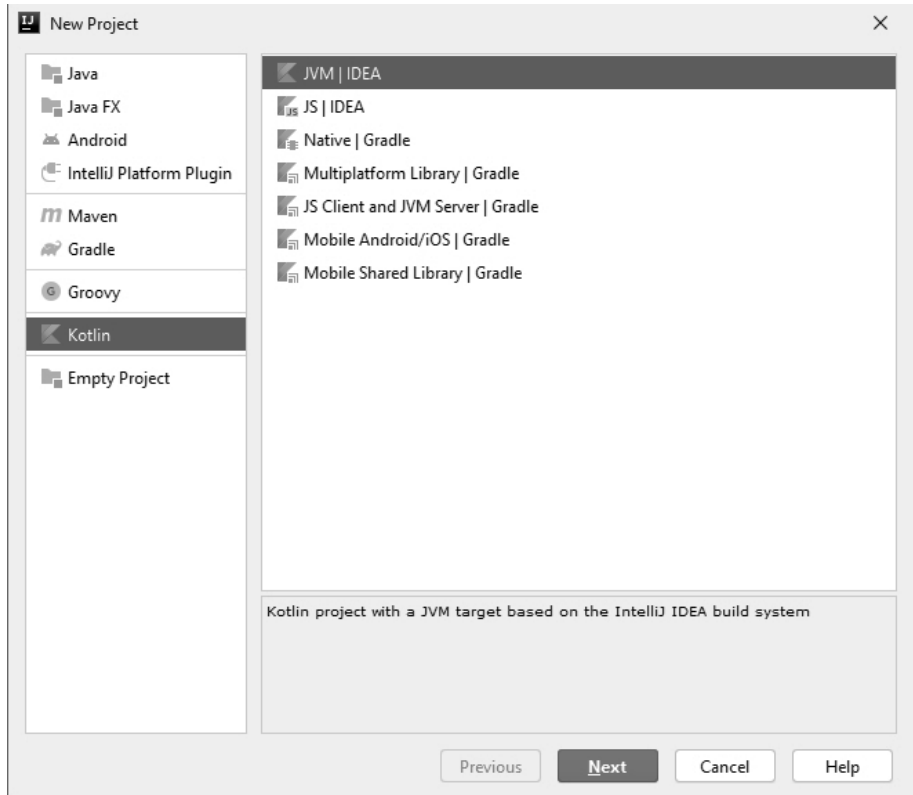


Figure 1.3 – Création d'un projet JVM|IDEA

Vous pouvez utiliser IntelliJ pour écrire du code dans d'autres langages que Kotlin, notamment Java, Python, Scala et Groovy. En sélectionnant **JVM|IDEA**, vous dites à IntelliJ que vous avez l'intention d'utiliser Kotlin. Plus précisément, **JVM|IDEA** dit à IntelliJ que vous avez l'intention d'écrire du code Kotlin qui *cible*, ou s'exécute sur la machine virtuelle Java. Un des avantages de Kotlin est qu'il dispose d'une série d'outils qui permettent d'écrire du code Kotlin pouvant fonctionner sur différents systèmes d'exploitation et plateformes.

À partir de maintenant, nous nous référerons à la machine virtuelle Java sous l'appellation abrégée de « JVM », puisque c'est sous ce nom qu'elle est habituellement désignée dans la communauté des développeurs Java. Vous pourrez en apprendre