

พัฒนา Mobile App บนระบบ Android ด้วย Kotlin



การเขียนโปรแกรมด้วย
Kotlin สำหรับ Android
จากพื้นฐานจนถึงการใช้งานจริง



การใช้เครื่องมือต่างๆ ของ
Android Studio ทั้งการออกแบบ
เขียนโค้ด และทดสอบ



พัฒนาแอปบนระบบ Android
พร้อมตัวอย่างและเวิร์กช็อป
ที่หลากหลายรูปแบบ



ดาวน์โหลดโค้ดหนังสือเล่มนี้ได้ที่:
<http://www.developerthai.com>



บัญชา ปะสีละเตสัง

ผู้เขียนหนังสือขายดีระดับ Best Seller
ด้าน Programming

พัฒนา Mobile App บนระบบ Android ด้วย Kotlin

โดย บัญชา ปะสีละเตลัง

สงวนลิขสิทธิ์ตามกฎหมาย โดย บัญชา ปะสีละเตลัง © พ.ศ. 2563

ห้ามคัดลอก ลอกเลียน ดัดแปลง ทำซ้ำ จัดพิมพ์ หรือกระทำการอื่นใด โดยวิธีการใดๆ ในรูปแบบใดๆ
ไม่ว่าส่วนหนึ่งส่วนใดของหนังสือเล่มนี้ เพื่อเผยแพร่ในสื่อทุกประเภท หรือเพื่อวัตถุประสงค์ใดๆ
นอกจากจะได้รับอนุญาต

ข้อมูลทางบรรณานุกรมของหอสมุดแห่งชาติ

บัญชา ปะสีละเตลัง.

พัฒนา Mobile App บนระบบ Android ด้วย Kotlin.--กรุงเทพฯ : ซีเอ็ดยูเคชั่น, 2563.
524 หน้า.

1. แอนดรอยด์ (ระบบปฏิบัติการคอมพิวเตอร์).
2. โปรแกรมประยุกต์บนอุปกรณ์เคลื่อนที่.
- I. ชื่อเรื่อง.

005.428

Barcode (e-book) : 9786160841660

ผลิตและจัดจำหน่ายโดย



บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน)
SE-EDUCATION PUBLIC COMPANY LIMITED

เลขที่ 1858/87-90 ถนนเทพรัตน แขวงบางนาใต้ เขตบางนา กรุงเทพฯ 10260
โทรศัพท์ 0-2826-8000

[หากมีคำแนะนำหรือติชม สามารถติดต่อได้ที่ comment@se-ed.com]

คำนำ

Kotlin เป็นภาษาคอมพิวเตอร์ที่ถูกสร้างขึ้นเพื่อแก้ไขข้อบกพร่องบางส่วนที่มีใน Java ซึ่งปัจจุบันนี้มีผู้สนใจศึกษา Kotlin เพิ่มมากขึ้นเรื่อยๆ เนื่องจากมีรูปแบบการเขียนโค้ดที่ง่ายกว่า สามารถเรียนรู้ได้อย่างรวดเร็ว และสิ่งสำคัญคือ Kotlin ถูกเลือกให้เป็นหนึ่งในภาษาที่ใช้พัฒนาแอปบนระบบ Android ได้เช่นเดียวกับ Java แต่จะเขียนโค้ดสั้นกว่า รวมทั้งมีเทคนิคต่างๆ ที่ช่วยลดความยุ่งยากในหลายรูปแบบ อย่างไรก็ตาม Kotlin ยังต้องอาศัยการทำงานโดยใช้ตัวแปลภาษาของ Java ดังนั้น เราควรศึกษา Kotlin ควบคู่กับ Java มากกว่าที่จะเลือกเพียงอย่างเดียวอย่างหนึ่ง

เพื่อให้เราสามารถเรียนรู้ทั้งการเขียนโปรแกรมด้วย Kotlin และการพัฒนาแอปบน Android ไปพร้อมๆ กันได้ ผู้เขียนจึงได้รวบรวมเนื้อหาที่เราจำเป็นต้องรู้ทั้งหมด ไว้ในหนังสือเล่มนี้อย่างครบถ้วน ซึ่งก็หวังว่า จะช่วยให้ผู้อ่านสามารถนำไปพัฒนาแอปบนระบบ Android ด้วย Kotlin ได้ตามต้องการ

บัญชา ปะสีละเตลัง

banchar_pa@yahoo.com

facebook.com/developperthai



Kotlin



สารบัญ

บทที่ 1 Kotlin Basic & Data Type	19
เกี่ยวกับ Kotlin	19
การติดตั้ง JDK	21
การติดตั้ง IntelliJ	22
การสร้างและจัดการโปรเจกต์บน IntelliJ	23
การกำหนดพอนต์สำหรับ IntelliJ	27
องค์ประกอบพื้นฐานของ Kotlin	28
• ฟังก์ชัน main()	28
• บล็อก { }	28
• การรันโปรแกรม	29
• การเขียนคำอธิบายภายในโค้ด	29
• จุดสิ้นสุดคำสั่งและเครื่องหมาย Semicolon (;)	30
• คีย์เวิร์ดของ Kotlin	31
• การแสดงผลด้วยฟังก์ชัน print() และ println()	32
ชนิดข้อมูลพื้นฐานใน Kotlin	33
การกำหนดตัวแปร	34
• การประกาศตัวแปร	35
• การกำหนดค่าให้กับตัวแปร	35
การกำหนดค่าคงที่	38

Type Inference	39
การแปลงชนิดข้อมูล.....	39
ลักษณะเพิ่มเติมเกี่ยวกับสตริง.....	40
• การเชื่อมต่อสตริง	40
• การเขียนสตริงหลายบรรทัด	41
• การแทรกค่าของตัวแปรในสตริง.....	42
• การเขียนอักขระพิเศษบางตัวในสตริง.....	42
การรับข้อมูลทางคีย์บอร์ด.....	43

บทที่ 2 Operator & Control Flow 45

โอเปอเรเตอร์ชนิดต่างๆ.....	45
• โอเปอเรเตอร์สำหรับการกำหนดค่า.....	45
• โอเปอเรเตอร์สำหรับการคำนวณ.....	46
• โอเปอเรเตอร์สำหรับการคำนวณและกำหนดค่า	46
• โอเปอเรเตอร์การเพิ่มและลดค่า.....	47
• โอเปอเรเตอร์การเปรียบเทียบ	47
• โอเปอเรเตอร์ทางตรรกะ.....	48
• โอเปอเรเตอร์การกำหนดช่วงข้อมูล	48
• โอเปอเรเตอร์ in และ !in.....	50
การตรวจสอบเงื่อนไขด้วย if-else.....	51
• การใช้คำสั่ง if.....	51
• การใช้คำสั่ง else	52
• การใช้คำสั่ง else if	53
• การใช้คำสั่ง if ซ้อนกัน.....	54
• การตรวจสอบหลายเงื่อนไขพร้อมกัน	55
• การใช้คำสั่ง if แบบ Conditional Expression.....	56
การใช้คำสั่ง when	57
การวนรูปแบบ for-in.....	59
การวนลูปด้วยฟังก์ชัน repeat	60
การใช้รูปแบบ while.....	61
การใช้รูปแบบ do-while	62

การวางลูปซ้อนกัน	63
คำสั่ง break	64
คำสั่ง continue	64
บล็อกและขอบเขตของตัวแปร	65

บทที่ 3 Array, String & Number 67

ชุดข้อมูลแบบ Pair และ Triple	67
• การเก็บข้อมูลในแบบ Pair	67
• การเก็บข้อมูลในแบบ Triple	68
การจัดเก็บข้อมูลแบบ Array	69
• การสร้างอาร์เรย์แบบไม่ระบุชนิดข้อมูล	69
• การสร้างอาร์เรย์ของชนิดข้อมูลพื้นฐาน	69
• การเข้าถึงสมาชิกของอาร์เรย์ด้วยลูป for-in	70
• เมธอดที่น่าสนใจของอาร์เรย์	71
ข้อมูลชนิด String	72
• การหาความยาวของสตริง	73
• การเปลี่ยนรูปแบบตัวพิมพ์	73
• การตัดช่องว่างออกจากสตริง	74
• การตรวจสอบสตริงว่าง	74
• การตรวจสอบจุดเริ่มต้นและสิ้นสุดสตริง	75
• การอ่านส่วนของสตริง	76
• การค้นหาตำแหน่ง	77
• การแทนที่สตริง	78
• การแยกและรวมสตริง	78
ฟังก์ชันสำหรับการคำนวณ	79
การสร้างเลขสุ่ม	80
การจัดรูปแบบตัวเลข	81
• คลาส NumberFormat	81
• คลาส DecimalFormat	83

บทที่ 4 Function & Lambda..... 85

พื้นฐานการสร้างและเรียกใช้ฟังก์ชัน	85
• การสร้างฟังก์ชันในเบื้องต้น	85
• หลักการกำหนดพารามิเตอร์ขั้นพื้นฐาน.....	86
• ตำแหน่งการเขียนและการเรียกใช้ฟังก์ชัน	87
การส่งผลลัพธ์กลับจากฟังก์ชัน	88
การสร้างฟังก์ชันแบบ Single Expression	91
การออกจากฟังก์ชัน.....	92
ขอบเขตของตัวแปรที่อยู่ในและนอกฟังก์ชัน	93
ลักษณะเพิ่มเติมเกี่ยวกับพารามิเตอร์	93
• การระบุชื่ออาร์กิวเมนต์	93
• พารามิเตอร์แบบ Default Value.....	94
• พารามิเตอร์แบบ Variable Argument	95
• พารามิเตอร์แบบอาร์เรย์	96
การส่งหลายๆ ค่ากลับจากฟังก์ชัน.....	97
• การส่งค่ากลับแบบอาร์เรย์.....	97
• การส่งค่ากลับแบบ Pair หรือ Triple.....	98
ฟังก์ชันแบบโอเวอร์โหลด	98
Lambda Expression.....	99
• รูปแบบทั่วไปของ Lambda	100
• พารามิเตอร์ it.....	101
• Type Inference.....	102
• Function Type.....	102
• Higher Order Function.....	103
การใช้ Lambda กับเมธอดของอาร์เรย์	105

บทที่ 5 Null Safety & Exception..... 107

ลักษณะของค่า null.....	107
การใช้งานตัวแปรที่เป็น Nullable	109
• การตรวจสอบด้วย if.....	109
• การใช้ Elvis Operator	110
• การใช้ Safe Call Operator	111

● การใช้ฟังก์ชัน ?.let { }	112
● การใช้วิธี Not Null Assertion Operator	113
● ฟังก์ชันที่ส่งค่ากลับแบบ Nullable	114
การตรวจสอบข้อผิดพลาดด้วย try-catch	115
การใช้ try-catch ในแบบ Expression.....	118
การใช้คำสั่ง finally.....	118
การใช้คำสั่ง throw.....	120

บทที่ 6 Class & Object 123

การสร้างคลาสและอินสแตนซ์.....	123
เมธอดของคลาส.....	125
พรีอ็อปเพอเรเตอร์ของคลาส	126
● การสร้างและจัดเก็บข้อมูลในพรีอ็อปเพอเรเตอร์.....	126
● บล็อก get/set และ Backing Field.....	127
คอนสตรัคเตอร์ของคลาส.....	129
● Primary Constructor แบบพรีอ็อปเพอเรเตอร์.....	129
● Primary Constructor แบบพารามิเตอร์.....	130
● Secondary Constructor	130
● Init Block	133
โมดิไฟเออร์แบบ private	133
พรีอ็อปเพอเรเตอร์แบบ lateinit.....	135
Companion Object.....	136
Enum Class.....	138
การใช้ฟังก์ชัน apply กับสมาชิกของคลาส	140

บทที่ 7 Inheritance & Interface 143

Open Class และการสืบทอด.....	143
คอนสตรัคเตอร์กับการสืบทอด.....	146
● กรณีที่ Superclass ไม่มีคอนสตรัคเตอร์	146
● กรณีที่ Superclass มีเฉพาะ Primary Constructor.....	146
● กรณีที่ Superclass มีเฉพาะ Secondary Constructor	147
● กรณีที่ Superclass มีทั้ง Primary และ Secondary Constructor.....	148

• การสร้างคอนสตรักเตอร์เพิ่มเติมให้แก่ Subclass.....	149
การโอเวอร์ไรด์สมาชิกของ Subclass.....	150
• การโอเวอร์ไรด์พรีอ็อปเพอเรเตอร์.....	150
• การโอเวอร์ไรด์เมธอด.....	151
• การอ้างถึงสมาชิกของ Superclass.....	151
โมดิไฟเออร์ private และ protected.....	152
Abstract Class.....	154
การสร้างและใช้งานอินเทอร์เฟซ.....	155
• การสร้างอินเทอร์เฟซ.....	155
• การสืบทอดและใช้งานอินเทอร์เฟซ.....	156
Object Expression.....	157
การตรวจสอบชนิดข้อมูลด้วย is และ !is.....	159
การแปลงชนิดข้อมูลด้วย as และ as?.....	163
• การใช้ as (Unsafe Cast Operator).....	163
• การใช้ as? (Safe Cast Operator).....	164
• Smart Cast.....	165

บทที่ 8 Generic & Collection..... 167

คลาสแบบ Generic.....	167
ฟังก์ชันแบบ Generic.....	170
ขอบเขตของ Generic Type.....	171
Generic Type ในแบบ in และ out.....	172
อาร์เรย์แบบ Generic.....	173
คอลเล็กชันประเภท List.....	174
• Mutable List.....	174
• Immutable List.....	176
• การใช้ Iterator ร่วมกับ List.....	176
คอลเล็กชันประเภท Set.....	177
คอลเล็กชันประเภท Map.....	178
การใช้แลมบ์ดากับคอลเล็กชัน.....	180

บทที่ 9 Android Studio	183
เกี่ยวกับระบบ Android	183
การติดตั้ง Android Studio	184
• การติดตั้งบน Windows	184
• การติดตั้งบน MacOS	186
การติดตั้ง SDK เพิ่มเติม	186
การสร้างโปรเจกต์	189
องค์ประกอบพื้นฐานของ Android Studio	192
การปรับแต่ง Code Editor	194
การติดตั้ง Android Virtual Device	195
การทดสอบกับเครื่องจริง	200
• ขั้นตอนบนเครื่องจริง	200
• ขั้นตอนบน Android Studio	201
 บทที่ 10 Basic App Development.....	203
พื้นฐานของการสร้าง UI	203
• Activity และ Layout	204
• การจัดวางวิวในมุมมอง Design	207
• การกำหนด Constraint ในเบื้องต้น	208
• การใช้เครื่องมือเพิ่มเติมสำหรับมุมมอง Design	209
แอตทริบิวต์ของวิว	211
• การกำหนดแอตทริบิวต์ในมุมมอง Design	211
• การกำหนดแอตทริบิวต์ในมุมมอง Text	212
• แอตทริบิวต์พื้นฐานของวิว	214
การกำหนด Resource	215
• ชนิดของ Resource	216
• Resource ชนิด values	216
• การสร้างไฟล์เพื่อจัดเก็บข้อมูลชนิด values	218
• อักขระพิเศษใน XML Resource	218
• การจัดเก็บภาพใน drawable	219
• การเข้าถึง Resource จาก Attribute Window และ XML	220

• การเข้าถึง Resource จากคลาส R	221
• การใช้แอ็พเจ็ท resources.....	222
การกำหนดข้อความบนทูลบาร์ของ Activity.....	223
แนวทางการสร้างไอคอนของแอป.....	224
• ลักษณะของ Image Asset	225
• การกำหนด Foreground.....	226
• การกำหนด Background.....	227
• Preview และการบันทึกไฟล์.....	228
• การตั้งค่าให้เป็นไอคอนของแอป.....	229

บทที่ 11 Views & UI..... 231

การอ้างถึงวิวใน Kotlin	231
อินสแตนซ์และตัวแปรที่ต้องใช้ร่วมกัน.....	234
การจัดการอีเวนต์ของวิว.....	236
• การจัดการอีเวนต์ด้วยวิธี Object Expression	237
• การจัดการอีเวนต์ด้วยวิธี Lambda Expression	239
การแบ่งกลุ่มของวิว.....	242
พรีอเพอร์ตีพื้นฐานของวิว.....	242
การใช้ TextView	243
การใช้ Button.....	245
การใช้วิวในกลุ่ม Text	246
การใช้ CheckBox และ RadioButton	248
• การใช้ CheckBox.....	248
• การใช้ RadioButton และ RadioGroup.....	249
การใช้ Switch และ ToggleButton	251
การใช้ Spinner.....	252
การใช้ ImageView	254
การใช้ ImageButton.....	256
การใช้ TextInputLayout	257
การสร้างปุ่ม FloatingActionButton.....	259

บทที่ 12 Toast, SnackBar & Layout.....	263
การแสดงความด้วย Toast.....	263
การแสดงความด้วย SnackBar	265
• การนำสแน็คบาร์มาใช้งาน.....	266
• การแสดงสแน็คบาร์อย่างง่าย	267
• การกำหนดปุ่มคำสั่งบนสแน็คบาร์.....	268
ConstraintLayout	270
• องค์ประกอบพื้นฐานของ ConstraintLayout.....	270
• Margin Constraint.....	271
• Bias Constraint.....	273
• ขนาดของวิว.....	275
• Baseline Constraint.....	277
• Align Constraint	277
• Chain Constraint.....	280
• ตัวอย่างการใช้ Constraint หลายแบบร่วมกัน.....	282
บทที่ 13 Activity & Explicit Intent	297
วงจรสถานะของ Activity.....	297
Callback Method ของ Activity	298
การโอเวอร์ไรด์ Callback Method ของ Activity	300
การสร้าง Activity เพิ่มเติม.....	301
การตั้งค่า Launcher Activity	302
การส่งข้อมูลระหว่าง Activity ด้วย Intent.....	303
การควบคุมปุ่ม Back	312
การออกจาก App	313
การล็อกทิศทางการหมุนหน้าจอ.....	314
การจัดเก็บสถานะของ Activity เมื่อหมุนหน้าจอ.....	315
บทที่ 14 Fragment, ViewPager & Tab	319
ลักษณะเกี่ยวกับ Fragment.....	319
• Lifecycle ของ Fragment.....	320

● องค์ประกอบของ Fragment.....	321
การสร้าง Fragment ในเบื้องต้น.....	322
● การสร้างไฟล์ของ Fragment.....	322
● การกำหนด Fragment ให้กับ Activity.....	324
Fragment Transaction.....	327
การสร้างและจัดการ UI ของ Fragment.....	330
การแสดง Fragment ด้วย ViewPager	334
● ส่วนของ Fragment.....	335
● ส่วนของ ViewPager.....	335
● ส่วนของ PagerAdapter	336
● ส่วนโค้ด Kotlin ของ Activity.....	337
การใช้ไลบรารี ViewPagerDotsIndicator.....	338
การใช้ TabLayout.....	345
● การสร้างส่วน TabLayout.....	346
● ส่วนของ ViewPager	346
● ส่วนของ Fragment.....	347
● ส่วนของ PagerAdapter	347
● ส่วนโค้ด Kotlin ของ Activity.....	348
● การแสดงชื่อแท็บด้วยภาพ Icon	348

บทที่ 15 Menu & Navigation..... 353

การสร้าง Options Menu.....	353
● การสร้างรายการเมนูใน Resource.....	355
● การจัดการเมนูในส่วนของ Kotlin.....	357
การสร้าง Popup Menu	360
การสร้าง Context Menu.....	363
การสร้าง Bottom Navigation View	366
● หลักการของ Bottom Navigation View	366
● การสร้าง Bottom Navigation Activity.....	367
● การใช้ Bottom Navigation ร่วมกับ Fragment.....	369
การสร้าง Navigation Drawer.....	372

● องค์ประกอบของ Navigation Drawer ในส่วน XML.....	374
● ข้อกำหนดในส่วนโค้ด Kotlin	375
การใช้ Navigation Drawer ร่วมกับ Fragment.....	377

บทที่ 16 SQLite Database 379

การสร้างคลาสสำหรับ SQLiteOpenHelper.....	379
การสร้างตารางฐานข้อมูล.....	381
● คำสั่ง SQL สำหรับสร้างตาราง	381
● การอัปเดตเวอร์ชันฐานข้อมูล.....	384
จัดการฐานข้อมูลด้วยกระบวนการ CRUD.....	386
● การเพิ่มข้อมูล	386
● การอ่านข้อมูล	387
● การปรับปรุงข้อมูล.....	389
● การลบข้อมูล.....	389
การเพิ่มและแก้ไขข้อมูลโดยใช้ ContentValues.....	396
● การกำหนดข้อมูลให้แก่ ContentValues.....	396
● การใช้ ContentValues ร่วมกับการเพิ่มข้อมูล.....	396
● การใช้ ContentValues ร่วมกับการแก้ไขข้อมูล.....	397

บทที่ 17 RecyclerView & CardView 401

การสร้าง RecyclerView ในเบื้องต้น.....	401
● องค์ประกอบพื้นฐานของ RecyclerView	402
● ขั้นตอนการสร้าง RecyclerView	402
การกำหนดอีเวนต์ที่เกิดกับรายการ	412
การสร้าง Shape Drawable ประกอบรายการ	417
● การสร้างรูปทรง Shape Drawable.....	418
● การนำ Shape Drawable มาใช้กับรายการ	418
● การกำหนดอักษระและสีพื้นหลังของ Drawable.....	419
การใช้ CardView.....	423
สร้างรายการ RecyclerView จากฐานข้อมูล SQLite.....	429
การสร้าง RecyclerView ไว้บน Fragment	433

บทที่ 18 Permission & Implicit Intent 439

การแบ่งระดับ Permission ในแอนดรอยด์.....	439
การกำหนด Permission ในไฟล์ Manifest.....	441
การกำหนด Permission ในแบบ Manual	442
การตรวจสอบ Permission ในขณะรัน	443
การใช้ Implicit Intent ที่น่าสนใจ.....	450
● เปิดเว็บเพจและค้นหาข้อมูล.....	451
● การส่งอีเมล.....	451
● การตรวจสอบอุปกรณ์ว่ารองรับการโทรหรือไม่.....	452
● การโทรออกแบบโทรทันที	453
● การโทรออกแบบแสดงปุ่มตัวเลข.....	454
● การส่ง SMS.....	455
● การแสดงแผนที่ Google Maps	455
● การตั้งค่าของระบบ.....	458
ตรวจสอบเป้าหมายการส่ง Intent จาก PackageManager.....	459
การกำหนด Intent Filter.....	459

บทที่ 19 Multimedia & Camera 463

การจัดเก็บไฟล์มัลติมีเดียไว้ใน Resources.....	463
องค์ประกอบของ MediaPlayer.....	464
การเล่นไฟล์เสียง	465
การเล่นไฟล์วิดีโอ	468
การเลือกไฟล์ Multimedia จาก External Storage.....	471
● การเข้าถึงไฟล์ใน External Storage.....	471
● การนำไฟล์จาก External Storage มาใช้ในแอป	472
การถ่ายภาพนิ่งและวิดีโอ	473
● การถ่ายภาพนิ่ง.....	473
● การถ่ายวิดีโอ	473
การนำภาพที่ถ่ายกลับมาใช้ในแอป	475
การสแกน QR Code และ Barcode	483
การเปิดและปิด Flashlight.....	486

บทที่ 20 Alert & Custom Dialog.....	489
ลักษณะทั่วไปของ Dialog.....	489
การแสดง Message Dialog.....	491
การแสดง Item Dialog.....	493
การแสดง Single Choice Dialog.....	494
การแสดง MultiChoice Dialog.....	496
การแสดง DatePicker Dialog.....	497
การแสดง TimePicker Dialog.....	499
การสร้าง Custom Dialog.....	500
 บทที่ 21 Workshop: To-Do List.....	 503
ลักษณะของแอป To-Do List.....	503
สร้างฐานข้อมูลและคลาส SQLiteOpenHelper.....	504
การสร้าง RecyclerView.....	505
การลบรายการ.....	508
การอ่านข้อมูลที่ MainActivity.....	510
เพิ่มรายการสิ่งที่ต้องทำ.....	511
สร้างไอคอนของแอป.....	514
 บทที่ 22 Workshop: Torchlight	 517
ขั้นตอนการออกแบบ.....	517
ขั้นตอนการเขียนโค้ด.....	518
สร้างไอคอนของแอป.....	522





Kotlin



Kotlin Basic & Data Type



Kotlin เป็นภาษาคอมพิวเตอร์ที่ถูกพัฒนาขึ้นเพื่อแก้ไขความยุ่งยาก และข้อบกพร่องบางส่วนที่มีในภาษา Java แต่อย่างไรก็ตาม Kotlin ยังต้องอาศัยการทำงานร่วมกับตัวแปลภาษาของ Java ซึ่งในปัจจุบัน มีผู้สนใจศึกษา Kotlin เพิ่มมากขึ้นเรื่อยๆ เนื่องจากมีรูปแบบการเขียนโค้ดที่ง่ายและเรียนรู้ได้เร็วกว่า Java และโดยเฉพาะอย่างยิ่งการพัฒนาแอปบนระบบ Android ที่สามารถเขียนด้วย Kotlin ได้เช่นเดียว Java แต่โค้ดที่ได้จะสั้นและกระชับกว่ามาก ซึ่งก่อนที่จะเราจะเข้าสู่ขั้นตอนการพัฒนาแอปบน Android ก็จำเป็นต้องเรียนรู้หลักการเขียนโปรแกรมด้วย Kotlin เสียก่อน โดยในบทนี้จะกล่าวถึงสิ่งที่เราควรรู้จักตั้งแต่เริ่มแรก เช่น การติดตั้งเครื่องมือ และการเซตค่าที่จำเป็นต่างๆ รวมไปถึงองค์ประกอบของภาษาที่สำคัญในเบื้องต้น เพื่อเป็นพื้นฐานสำหรับการนำไปใช้งานในบทต่อไป

เกี่ยวกับ Kotlin

Kotlin เป็นภาษาคอมพิวเตอร์ที่พัฒนาขึ้นมาโดย JetBrains ซึ่งเป็นบริษัทผู้สร้างโปรแกรมประเภท IDE หลายตัวที่ได้รับความนิยมทั่วโลก เช่น IntelliJ IDEA, PyCharm, PhpStorm, WebStorm รวมถึง Android Studio ก็สร้างขึ้นจากแพลตฟอร์มของ JetBrains เช่นเดียวกัน โดยวัตถุประสงค์เริ่มแรกนั้น ทาง JetBrains ต้องการสร้าง Kotlin เพื่อใช้แก้ไขข้อบกพร่องที่มีในภาษา Java สำหรับการพัฒนาซอฟต์แวร์ของบริษัทเองเท่านั้น แล้วต่อมา Kotlin ก็ได้รับการพัฒนาจนสามารถทำงานได้อย่างสมบูรณ์ และครอบคลุมการเขียนโค้ดจากภาษา Java โดยตรง จึงได้นำออกเผยแพร่ต่อสาธารณชน จนกระทั่งทาง Google ได้นำ Kotlin มาเป็นภาษาทางเลือกในการพัฒนาแอปบนระบบ Android

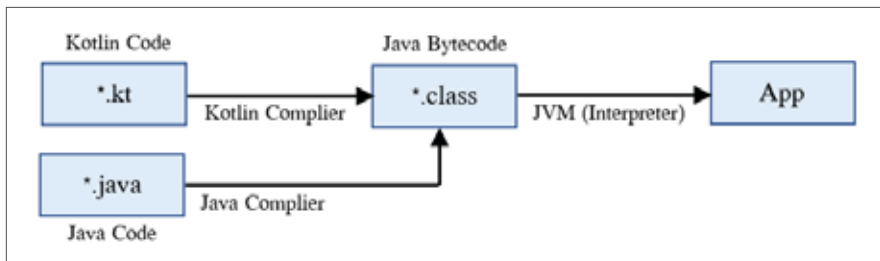


นอกเหนือจากเดิมที่เคยใช้ Java เป็นหลัก ทำให้ Kotlin เริ่มเป็นที่รู้จักและใช้กันแพร่หลายมากขึ้น สำหรับคำว่า Kotlin นั้น มาจากชื่อเกาะแห่งหนึ่งในทะเล Baltic ใกล้กับประเทศรัสเซีย ซึ่งคาดว่าน่าจะได้นำแนวทางมาจากคำว่า Java ที่เป็นชื่อเกาะในประเทศอินโดนีเซีย

ข้อดีที่น่าสนใจของภาษา Kotlin มีดังต่อไปนี้

- เขียนโค้ดได้สั้นและกระชับกว่าการเขียนด้วยภาษา Java โดยตรง
- แม้ Kotlin จะเป็นภาษาที่อยู่บนพื้นฐานของ OOP แต่ก็สามารถเขียนโปรแกรมแบบ Functional ได้
- แก้ไขข้อบกพร่องหลายอย่างของ Java รวมถึงการเพิ่มฟีเจอร์ใหม่ๆ ที่ไม่มีมาก่อนใน Java
- ทำงานอยู่บน Java Virtual Machine (JVM) จึงสามารถใช้ข้ามแพลตฟอร์มได้
- สามารถใช้ทดแทน Java ในการพัฒนาแอปบนระบบ Android ได้อย่างสมบูรณ์

สำหรับขั้นตอนการประมวลผลของ Kotlin เมื่อเทียบกับ Java มีหลักการโดยสังเขปดังนี้



จากภาพ สามารถอธิบายเพิ่มเติมได้ดังนี้

- โค้ดของภาษา Kotlin ที่เราเขียน จะกำหนดส่วนขยายของไฟล์เป็น .kt
- เมื่อโค้ดถูกคอมไพล์ (Compile) จะได้ไฟล์ที่มีส่วนขยาย **.class** เพิ่มขึ้นมา ซึ่งเรียกว่า Java Bytecode อันเป็นโค้ดมาตรฐานของภาษา Java หรือกล่าวได้ว่า เมื่อถึงขั้นตอนนี้ จะเปลี่ยนจาก Kotlin ให้เป็น Java นั่นเอง
- เราก็สามารถนำไฟล์ **.class** ไปประมวลผลบนระบบใดก็ได้ที่มี JVM ติดตั้งอยู่ ซึ่งการประมวลผลไฟล์ **.class** บน JVM จะเรียกว่าการแปลคำสั่ง หรือ Interpret เพื่อทำงานตามที่กำหนด (สามารถดูรายละเอียดเพิ่มเติมได้จากหนังสือ Java ของผู้เขียน)

อาจสรุปได้ว่า Kotlin เป็นภาษาที่ช่วยลดความยุ่งยากจากการเขียนโค้ดด้วยภาษา Java โดยตรง เพราะนอกจากจะเขียนได้สั้นและกระชับกว่าแล้ว ยังเพิ่มฟีเจอร์ใหม่ๆ ที่น่าสนใจอีกหลายประการ และที่สำคัญคือ เป็นอีกภาษาหนึ่งที่ใช้พัฒนาแอปบนระบบ Android ได้

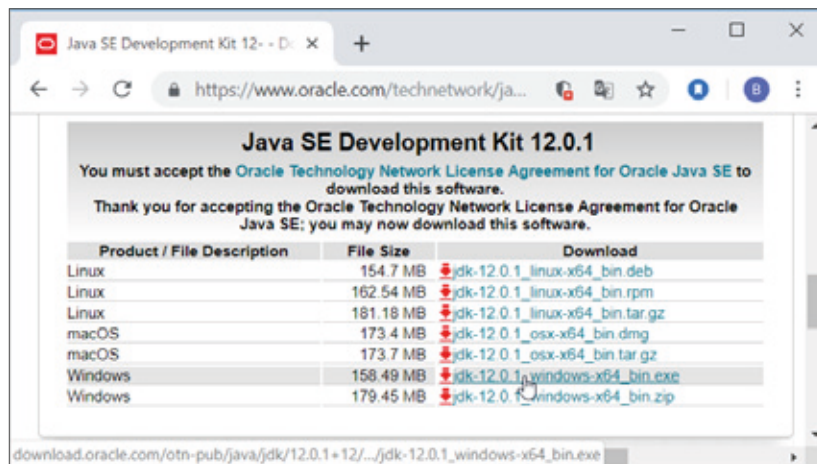
เนื่องจาก Kotlin ยังคงทำงานอยู่บนพื้นฐานของ Java และองค์ประกอบบางอย่างยังต้องใช้ร่วมกับ Java ดังนั้น Kotlin จึงไม่ใช่ภาษาที่จะใช้ทดแทน Java แต่เป็นภาษาที่เราควรเรียนรู้ควบคู่ไปกับ Java

การติดตั้ง JDK

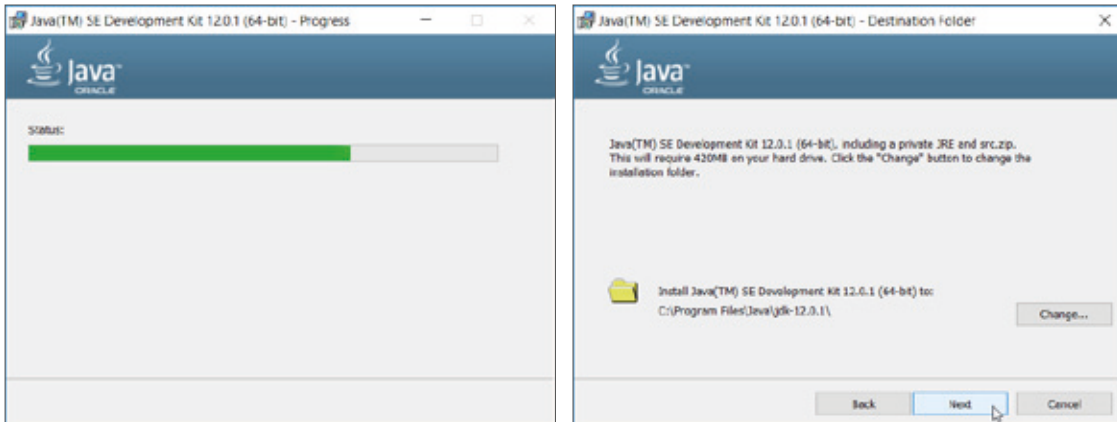
จากที่ได้กล่าวไปแล้วว่า Kotlin ทำงานอยู่บนพื้นฐานของ Java โดยชุดเครื่องมือสำหรับการพัฒนาภาษานี้เรียกว่า Java Development Kit (JDK) นี่คือนี่จำเป็นลำดับแรกที่เราต้องติดตั้ง ดังขั้นตอนต่อไปนี้

1. เราสามารถดาวน์โหลด JDK ได้ที่เว็บไซต์ <https://www.oracle.com/technetwork/java/javase/downloads/index.html>

โดยให้เลือกรุ่น Java SE และเลือกชนิดไฟล์ติดตั้งตามระบบปฏิบัติการที่เราใช้งานอยู่



2. เมื่อดาวน์โหลดเสร็จแล้วให้ดับเบิลคลิกที่ไฟล์ติดตั้ง หลังจากนั้นก็คลิกปุ่ม Next ไปเรื่อยๆ จนกว่ากระบวนการจะเสร็จสิ้น สำหรับตำแหน่งการติดตั้งบนระบบ Windows ตามปกติจะอยู่ที่
C:\Program Files\Java\

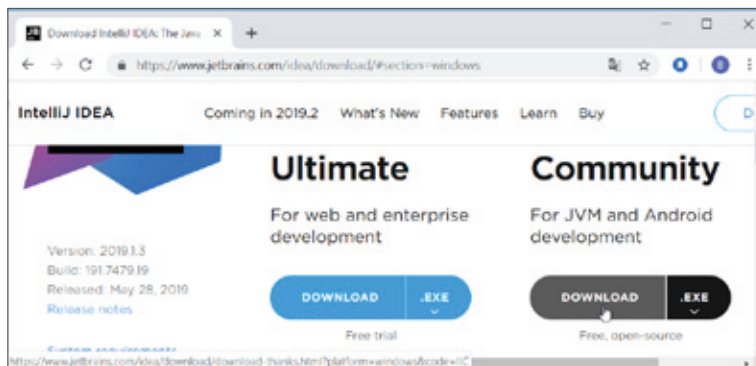


หลังการติดตั้ง หากจะใช้ตามแนวทางของหนังสือเล่มนี้ ก็ไม่จำเป็นต้องเซตค่าใดๆ เพราะเราไม่ใช้งานผ่าน JDK โดยตรง แต่จะใช้ผ่านโปรแกรมประเภท IDE แทน

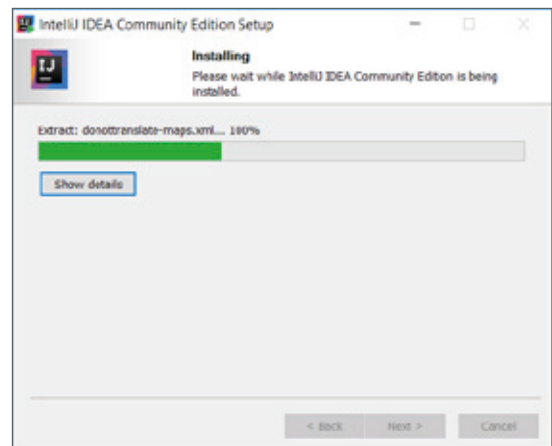
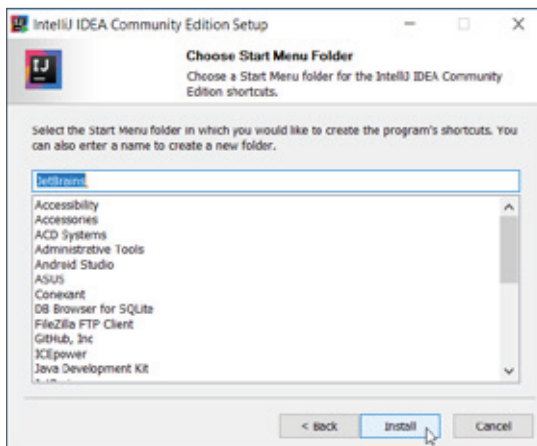
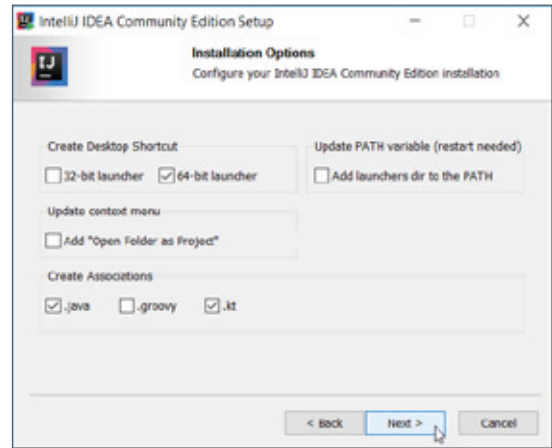
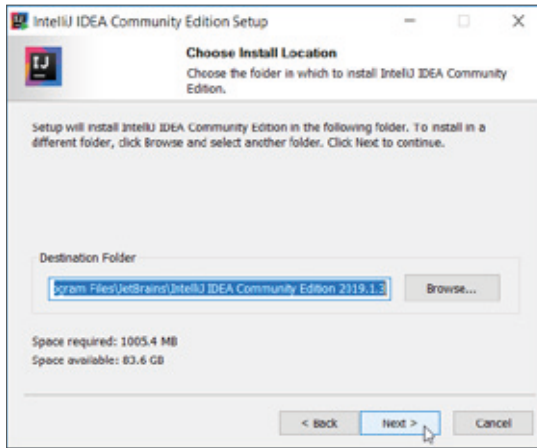
การติดตั้ง IntelliJ

IntelliJ เป็นโปรแกรมประเภท IDE (Integrated Development Environment) ของบริษัท JetBrains ผู้สร้างภาษา Kotlin นั่นเอง จุดเด่นของ IntelliJ คือรูปร่างหน้าตาที่ดูเรียบง่าย สวยงาม นอกจากนี้ก็ยังมีเครื่องมือมาช่วยทั้งในการเขียนโค้ดและการออกแบบต่างๆ อย่างครบถ้วน แล้วยังมีรุ่นที่สามารถนำมาใช้งานได้ฟรีอีกด้วย และที่สำคัญคือ IntelliJ คือพื้นฐานของ **Android Studio** ที่เราจะได้ศึกษาต่อจาก Kotlin ดังนั้น หากเราคุ่นเคยกับ IntelliJ มาก่อนแล้ว เมื่อไปพัฒนาแอปบน Android Studio ก็ไม่ต้องเสียเวลาไปเริ่มต้นใหม่หรือปรับตัอะไรมาก เพราะสามารถเรียนรู้เพิ่มเติมเฉพาะส่วนที่เกี่ยวกับแอนดรอยด์ได้เลย โดยขั้นตอนการติดตั้ง IntelliJ มีดังต่อไปนี้

1. เราสามารถดาวน์โหลด IntelliJ ได้จากเว็บไซต์ <https://www.jetbrains.com/idea/download> โดยแนะนำให้เลือกรุ่น **Community Edition** ซึ่งสามารถนำมาใช้ได้ฟรี

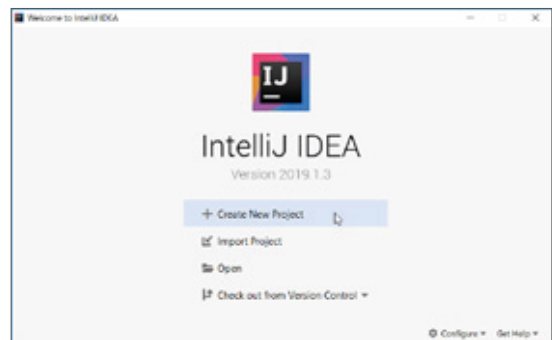


2. หลังจากนั้นก็ติดตั้งลงไปเหมือนกับโปรแกรมทั่วไป โดยขั้นตอนที่น่าสนใจเป็นดังภาพ

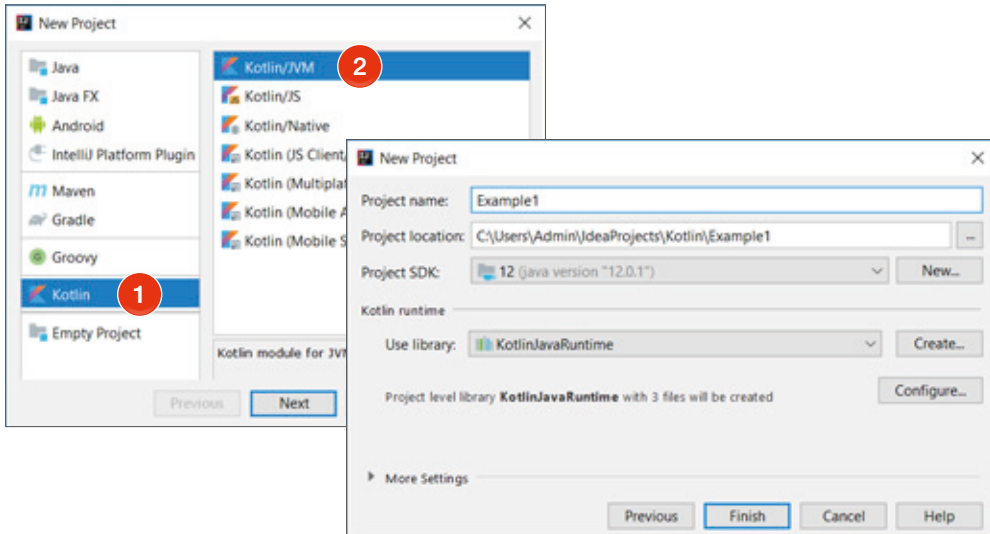


การสร้างและจัดการโปรเจกต์บน IntelliJ

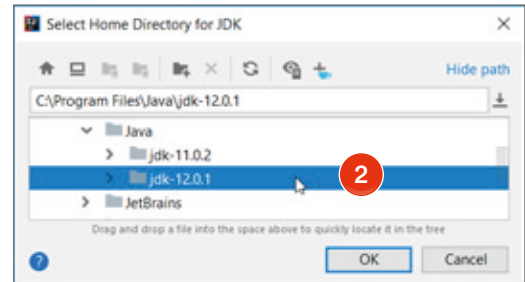
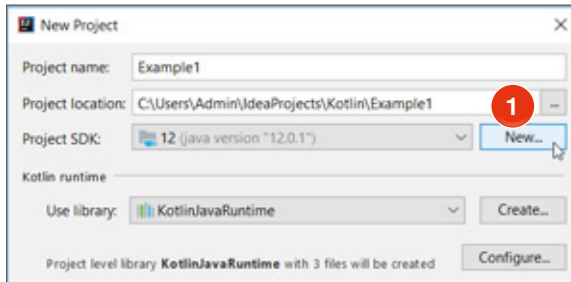
การเขียนโปรแกรมด้วย IntelliJ จะต้องเริ่มที่การสร้างโปรเจกต์ (Project) ใหม่เสมอ โดยเปิดเข้าสู่ IntelliJ แล้วทำตามขั้นตอนดังต่อไปนี้



1. เมื่อปรากฏหน้าจอ Welcome to IntelliJ IDEA เลือกที่เมนู **Create New Project**
2. เลือก **Kotlin > Kotlin/JVM**

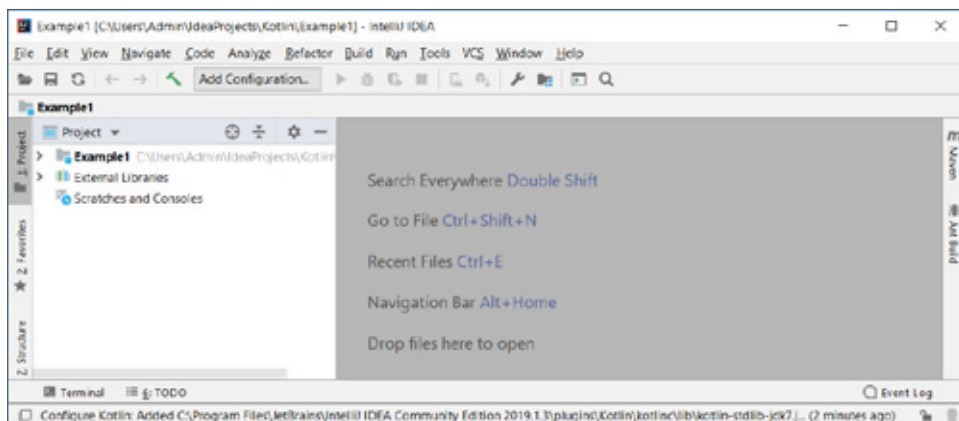


3. ต่อไปเป็นการตั้งชื่อโปรเจกต์ ก็ให้กำหนดชื่อตามต้องการ ส่วนตำแหน่งการจัดเก็บโปรเจกต์นั้น ตามปกติ IntelliJ จะแนะนำค่าดีฟอลต์เป็น **C:\Users\ชื่อผู้ใช้\IdeaProjects** ซึ่งจะใช้ร่วมกันทั้ง Java และ Kotlin แต่ผู้เขียนแนะนำว่าควรแยกของ Kotlin ไว้ต่างหาก เช่น สมมติว่าจะสร้างโปรเจกต์ชื่อ Example1 ก็ทำดังนี้
 - กำหนดค่าในช่อง **Project name** เป็น Example1
 - แล้วกำหนดตำแหน่งในช่อง Project location เป็น
C:\Users\ชื่อผู้ใช้\IdeaProjects\Kotlin\Example1
(หากโพลเดอร์ Kotlin ยังไม่มีอยู่ก่อน IntelliJ จะสร้างขึ้นมาให้เอง)
4. ถ้าเราเพิ่งจะเริ่มใช้ IntelliJ เป็นครั้งแรก และยังไม่ได้กำหนดตำแหน่งที่ติดตั้ง JDK ซึ่งต้องใช้สำหรับ Kotlin ก็ให้ทำดังนี้
 - 1) คลิกปุ่ม New...
 - 2) เลือกตำแหน่งที่เราติดตั้ง JDK เอาไว้

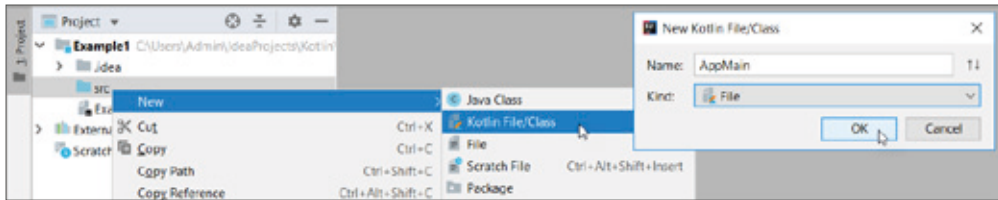


การเซตตำแหน่ง JDK นี้ จะทำเฉพาะเมื่อสร้างโปรเจกต์บน IntelliJ เป็นครั้งแรกเท่านั้น แล้วครั้งต่อไป เราไม่ต้องเซตค่านี้อีก เพราะโปรแกรมจะจดจำตำแหน่งนี้ไว้ให้เราโดยอัตโนมัติ ยกเว้นแต่เราเปลี่ยนตำแหน่งการติดตั้ง JDK ใหม่ หรืออัปเดตเป็นเวอร์ชันใหม่

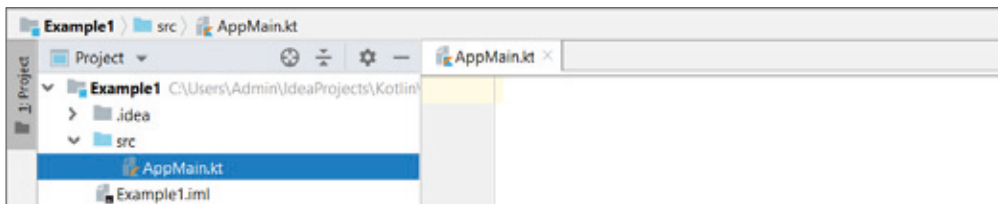
5. เมื่อเข้าสู่โปรแกรม IntelliJ ก็จะมีลักษณะดังนี้



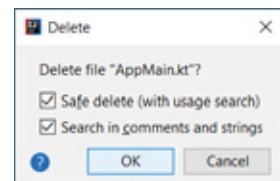
- เนื่องจาก IntelliJ ไม่ได้สร้างไฟล์มาให้เราล่วงหน้า ดังนั้น สิ่งที่ต้องทำเป็นอันดับแรกคือ การเพิ่มไฟล์ใหม่เข้ามาในโปรเจกต์ โดยที่วินโดว์ Project (หากไม่ปรากฏ ให้คลิกตรงแท็บด้านข้างทางซ้ายมือ หรือเลือกจากเมนู View > Tool Windows > Project) คลิกขยายโหนดชื่อโปรเจกต์ (กรณีของผู้เขียนตั้งชื่อเป็น Example1) จากนั้นคลิกขวาที่โฟลเดอร์ **src > New > Kotlin File/Class** แล้วกำหนดชื่อไฟล์ลงไป และในช่อง Kind ต้องเลือกชนิด File ด้วย



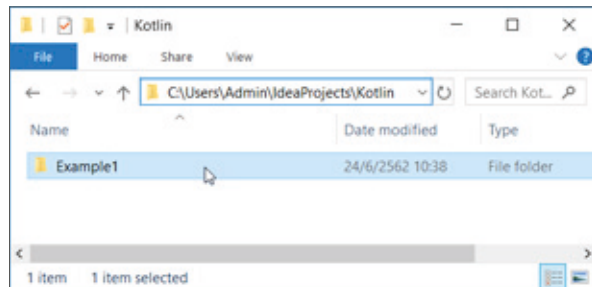
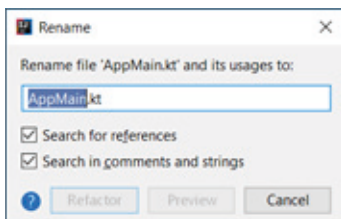
- ไฟล์สำหรับการเขียนโค้ดจะถูกสร้างขึ้นในโฟลเดอร์ src ซึ่งเราจะใช้งานส่วนนี้ในลำดับต่อไป



- การลบไฟล์ ให้คลิกขวาที่ชื่อไฟล์ตรงโฟลเดอร์ src แล้วเลือก Delete เมื่อมีไดอะล็อกขึ้นมาให้เลือก 옵션 ก็แนะนำให้เลือกทุกอันเพื่อให้ไฟล์ถูกลบอย่างสมบูรณ์
- การเปลี่ยนชื่อไฟล์ ให้คลิกขวาที่ชื่อไฟล์ตรงโฟลเดอร์ src แล้วเลือก



Refactor > Rename

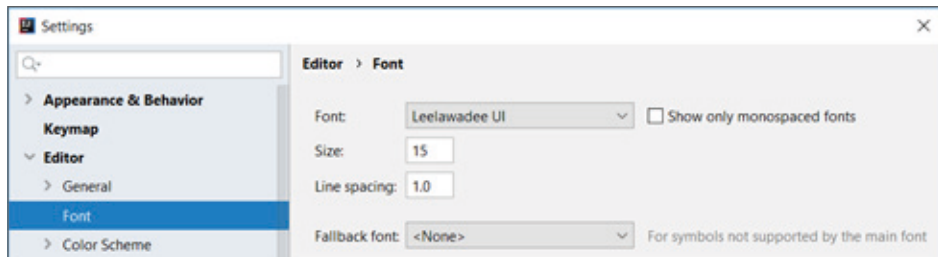


- ถ้าต้องการลบโปรเจกต์ ไม่สามารถลบผ่าน IntelliJ ได้ ต้องเปิดเข้าไปยังตำแหน่งที่เราจัดเก็บโปรเจกต์เอาไว้คือ **C:\Users\ชื่อผู้ใช้\IdeaProjects\Kotlin** แล้วลบโฟลเดอร์ของโปรเจกต์นั้นเหมือนกับโฟลเดอร์ทั่วไป

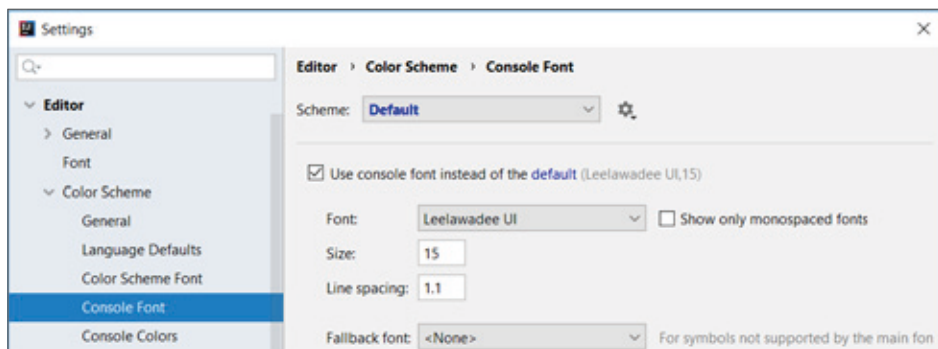
การกำหนดฟอนต์สำหรับ IntelliJ

ฟอนต์ของโค้ดและส่วนผลลัพธ์ที่ IntelliJ กำหนดมาให้ล่วงหน้า อาจเกิดปัญหาในการแสดงภาษาไทย หรือมีรูปแบบที่ไม่ค่อยถูกใจเรานัก ซึ่งก็สามารถเปลี่ยนแปลงใหม่ได้ดังนี้

1. เลือกที่เมนู **File > Settings...**
2. ถ้าต้องการแก้ไขฟอนต์ของโค้ด ก็มีขั้นตอนคือ
 - เลือกที่ **Editor > Font**
 - เอาเครื่องหมาย **Show only monospaced fonts** ออก
 - เลือกชนิดฟอนต์และกำหนดขนาดตามต้องการ โดยในกรณีของผู้เขียน ได้เลือกชนิด Leelawadee ขนาด 15



3. ถ้าต้องการกำหนดฟอนต์ของส่วน Console (แสดงผลลัพธ์) ก็มีขั้นตอนคือ
 - เลือก **Editor > Color Scheme > Console Font**
 - เอาเครื่องหมาย **Show only monospaced fonts** ออก
 - เลือกชนิดฟอนต์และกำหนดขนาดตามต้องการ โดยในกรณีของผู้เขียนเลือกชนิด Leelawadee ขนาด 15



องค์ประกอบพื้นฐานของ Kotlin

การเขียนโปรแกรมทุกระดับ ตั้งแต่ขั้นพื้นฐานจนถึงระดับสูง เราจำเป็นต้องกำหนดองค์ประกอบบางอย่าง คล้ายๆ กัน สำหรับใน Kotlin ก็มีสิ่งที่เราควรรู้จักในเบื้องต้นดังต่อไปนี้

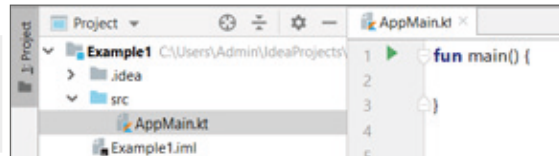
ฟังก์ชัน main()

ภาษา Kotlin รองรับการเขียนโปรแกรมทั้งแบบ Functional และแบบ OOP แต่ในเบื้องต้นนี้ เราจะศึกษา ในแบบฟังก์ชันกันไปก่อน โดยอย่างน้อยที่สุด เราต้องสร้างฟังก์ชันชื่อ main เพื่อกำหนดการทำงานเริ่มแรก ซึ่งมี รูปแบบพื้นฐานคือ

```
fun main() {  
}
```

- **fun** มาจากคำว่า function เพื่อบ่งชี้ว่าเป็นการสร้างฟังก์ชัน
- **main** เป็นชื่อที่บ่งบอกว่า เป็นฟังก์ชันหลักของแอปพลิเคชัน ซึ่งต้องใช้ชื่อนี้และเขียนด้วยตัวพิมพ์เล็กทั้งหมดเท่านั้น

```
fun main() {  
}
```



บล็อก { }

ใน Kotlin จะใช้วงเล็บ { } ในการกำหนดจุดเริ่มต้นและจุดสิ้นสุดของบล็อกการทำงานในแต่ละส่วน เช่น ฟังก์ชัน เงื่อนไข ลูป คลาส และอื่นๆ อีกหลายกรณี ซึ่งภายในบล็อก { } อาจมีบล็อกย่อยๆ ซ้อนลงไปอีกตาม ความซับซ้อนของโปรแกรม เช่น (ให้ดูผ่านๆ ไปก่อน)

```
fun main() {  
    for (...) {  
        if (...) {  
            ...  
        } else {  
            ...  
        }  
    }  
}
```

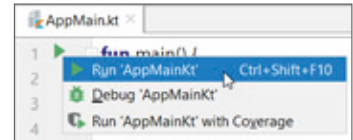
การรันโปรแกรม

หากเป็นการรันโปรแกรมครั้งแรกสุด (ไม่เคยรันโปรเจกต์นี้มาก่อน) อาจทำดังนี้

วิธีที่ 1 เลือกจากเมนู **Run > Run 'ชื่อไฟล์'**

วิธีที่ 2 คลิกไอคอน ▶ ตรงฟังก์ชัน main แล้วเลือก **Run 'ชื่อไฟล์'**

วิธีที่ 3 คลิกขวาบน Code Editor แล้วเลือก **Run 'ชื่อไฟล์'**



วิธีที่ 4 กด <Alt + Shift + F10>

แต่ในการรันครั้งต่อไป สามารถคลิกที่ไอคอน ▶ บนทูลบาร์ได้เลย (ถ้ายังไม่เคยรันมาก่อน ไอคอนนี้จะอยู่ในสถานะ Disable คือคลิกไม่ได้) หรือจะใช้วิธีการเดียวกันกับการรันครั้งแรกก็ได้

การเขียนคำอธิบายภายในโค้ด

การเขียนคำอธิบาย (Comment) ใน Kotlin ก็ใช้รูปแบบเดียวกันกับ Java ซึ่งอาจเลือกวิธีใดวิธีหนึ่งดังต่อไปนี้

Line Comment (//)

Line Comment (หรือ Single-Line Comment) มักใช้กับการเขียนคำอธิบายสั้นๆ ให้จบภายในหนึ่งบรรทัด โดยใช้เครื่องหมาย Slash จำนวน 2 อัน (//) เช่น

```
fun main() { //ฟังก์ชันหลัก
    //เริ่มต้น ให้รับข้อมูลจากผู้ใช้ แล้วตรวจสอบความถูกต้อง
    //...
    //print("Hello") บรรทัดนี้ไม่ถูกประมวลผล แม้จะเป็นโค้ดของ Kotlin

    println("Hi") //แต่บรรทัดนี้ถูกประมวลผล เพราะเขียนไว้ก่อน Comment
}
```

Block Comment (`/* */`)

Block Comment (หรือ Multi-Line Comment) มักใช้กับกรณีการเขียนคำอธิบายที่มีความยาวหลายบรรทัด กำหนดจุดเริ่มต้นด้วยเครื่องหมาย `/*` และสิ้นสุดด้วย `*/` เช่น

```
fun main() {    /* เมธอดหลัก */
    /* บรรทัดต่อไปนี้ไม่ถูกประมวลผล แม้จะเป็นคำสั่งโค้ดของ Kotlin
    print("Hello ")
    print("World")
    */

    println("Hi")    /* แต่บรรทัดนี้ถูกประมวลผล เพราะไม่อยู่ใน Comment */

    /* หรือเขียน Comment แทรกไว้ระหว่างโค้ดก็ได้ เพื่อให้โค้ดส่วนนั้นไม่ถูกประมวลผล */
    if (a == 0 /* && b == 1 */) {
        //...
    }
}
```

จุดสิ้นสุดคำสั่งและเครื่องหมาย Semicolon (;)

สำหรับ Kotlin ถือว่าโค้ดที่อยู่ในบรรทัดเดียวกัน คือคำสั่งเดียวกัน และเมื่อขึ้นบรรทัดใหม่ก็ถือว่าเป็นคำสั่งใหม่ ดังนั้น จึงไม่จำเป็นต้องใช้เครื่องหมาย Semicolon (;) เป็นตัวแสดงจุดสิ้นสุดคำสั่งเหมือนกับ Java เช่น โค้ดต่อไปนี้

```
var x = 10.5
val y = 0.5
print(x + y)
```

อย่างไรก็ตาม หากเราใส่เครื่องหมาย ; เพื่อแสดงจุดสิ้นสุดคำสั่งก็สามารถทำได้ เนื่องจากโปรแกรมเมอร์บางคนอาจจะคุ้นเคยกับการใช้งานในรูปแบบดังกล่าวจากภาษาอื่นๆ หรือนอกจากนี้ หากเราใส่ ; เพื่อปิดท้ายคำสั่งก็สามารถนำคำสั่งถัดไป มาเขียนต่อท้ายในบรรทัดนั้นได้เลย เช่น

```
var x = 10.5; let y = 0.5;
print(x + y);
```

คีย์เวิร์ดของ Kotlin

คีย์เวิร์ด (Keyword) หรือคำสงวน (Reserved Words) หมายถึง คำที่ใช้เป็นคำสั่งหรือควบคุมการทำงานของโปรแกรม เราต้องไม่นำคำเหล่านี้ไปตั้งเป็นชื่อตัวแปร ฟังก์ชัน คลาส หรือชื่อขององค์ประกอบอื่นๆ โดยคีย์เวิร์ดใน Kotlin แบ่งเป็น 3 กลุ่ม ดังนี้

- Hard Keywords

as	else	in	object	try
as?	false	interface	return	typealias
break	for	is	super	val
class	fun	!is	this	var
continue	if	null	throw	when
do	in	package	true	while

- Soft Keywords

by	dynamic	get	property	where
catch	field	import	receiver	
constructor	file	init	set	
delegate	finally	param	setparam	

- Modifier Keywords

actual	data	inline	operator	reified
abstract	enum	inner	out	sealed
annotation	expect	internal	override	suspend
companion	external	lateinit	private	tailrec
const	final	noinline	protected	vararg
crossinline	infix	open	public	

การแสดงผลด้วยฟังก์ชัน print() และ println()

การเขียนโปรแกรมขั้นพื้นฐาน ส่วนใหญ่เราจะแสดงผลและติดต่อกับผู้ใช้ในรูปแบบข้อความผ่านทางคอนโซล โดยใน Kotlin มีคำสั่งสำหรับการแสดงผลที่ควรรู้จักในเบื้องต้น คือ

print(ข้อมูล)	ใช้ในการแสดงข้อมูลที่กำหนดออกไปที่คอนโซล
println(ข้อมูล)	ใช้ในการแสดงข้อมูลเช่นเดียวกัน แต่หลังจากการแสดงผล โคอร์เซอร์จะถูกเลื่อนไปอยู่บรรทัดถัดไป หรือขึ้นบรรทัดใหม่ให้โดยอัตโนมัตินั่นเอง

สำหรับข้อมูลที่จะกำหนดให้แก่วางสองคำสั่งนี้ หากเป็นตัวเลข ก็เขียนลงไปโดยตรงได้เลย แต่หากเป็นข้อความ (สตริง) ให้เขียนไว้ในเครื่องหมาย " " เช่น

```
fun main() {  
    print(12345)  
    println("Kotlin for Android")  
}
```

ข้อแตกต่างระหว่าง print() และ println() จะอยู่ที่ตำแหน่งเคอร์เซอร์หลังข้อมูลที่กำหนด หากใช้ print() เคอร์เซอร์จะอยู่ต่อท้ายข้อมูลที่แสดง ถ้าเราใช้คำสั่งต่อเนื่องกันหลายครั้ง ข้อมูลทั้งหมดจะถูกวางเรียงต่อเนื่องในบรรทัดเดียวกันไปเรื่อยๆ เช่น

```
print("Kotlin ")  
print("for ")  
print("Android")  
//ลักษณะผลลัพธ์คือ Kotlin for Android
```

แต่ถ้าเปลี่ยนมาใช้ println() จะได้ผลดังนี้

```
println("Kotlin ")  
println("for ")  
println("Android")  
//ลักษณะผลลัพธ์คือ  
Kotlin  
for  
Android
```

ชนิดข้อมูลพื้นฐานใน Kotlin

ชนิดของข้อมูลพื้นฐาน (Primitive Data Type) ที่มีใน Kotlin สามารถแบ่งเป็นประเภทย่อยๆ ได้ดังนี้ (ต้องขึ้นต้นด้วยตัวพิมพ์ใหญ่)

- ข้อมูลประเภทเลขจำนวนเต็ม (Signed Integral Type)

ชนิดข้อมูล	ช่วงข้อมูล	ขนาด (bit)	อักขระต่อท้าย
Byte	-128 ถึง 127	8	
Short	-32768 ถึง 32767	16	
Int	-2,147,483,648 ถึง 2,147,483,647	32	
Long	-9,223,372,036,854,775,808 ถึง 9,223,372,036,854,775,807	64	L

- ข้อมูลประเภทเลขจำนวนเต็มบวก (Unsigned Integral Type)

ชนิดข้อมูล	ช่วงข้อมูล	ขนาด (bit)	อักขระต่อท้าย
UByte	0 ถึง 255	8	u หรือ U
UShort	0 ถึง 65535	16	u หรือ U
UInt	0 ถึง 4,294,967,295	32	u หรือ U
ULong	0 ถึง 18,446,744,073,709,551,615	64	uL หรือ UL

- ข้อมูลประเภทเลขทศนิยม (Floating Point Type)

ชนิดข้อมูล	ช่วงข้อมูล	ขนาด (bit)	อักขระต่อท้าย
Float	1.4E-45 ถึง 3.4028235E38	32	f หรือ F
Double	4.9E-324 ถึง 1.7976931348623157E308	64	

- ข้อมูลประเภทจริงเท็จ (Boolean Type)

ชนิดข้อมูล	ช่วงข้อมูล	ขนาด (bit)	อักขระต่อท้าย
Boolean	ค่าที่เป็นไปได้ มี 2 อย่างคือ true หรือ false	1	

- ข้อมูลประเภทอักขระ

ชนิดข้อมูล	ช่วงข้อมูล	ขนาด (bit)	อักขระต่อท้าย
Char	เก็บข้อมูลที่เป็นอักขระเพียง 1 ตัว	16	
String	เก็บข้อมูลที่เป็นข้อความที่อาจมีอักขระได้มากกว่า 1 ตัว	16 * length	

- ข้อมูลประเภทใดก็ได้

ชนิดข้อมูล	ช่วงข้อมูล	อักขระต่อท้าย
Any	สามารถเก็บข้อมูลชนิดใดก็ได้ เช่น ตัวเลข สตริง หรือค่าจริงเท็จ	

ชนิดข้อมูลแบบ **String** และ **Any** ไม่ จัดว่าเป็นชนิดข้อมูลพื้นฐาน หรือ Primitive Type แต่สามารถนำมาใช้งานในแบบ Primitive Type ได้ ซึ่งในบางกรณี เราอาจต้องนำข้อแตกต่างนี้ไปพิจารณาร่วมด้วย และสำหรับชนิดที่เป็นตัวเลข สามารถตรวจสอบช่วงข้อมูลได้จากค่าคงที่ MIN_VALUE และ MAX_VALUE ของข้อมูลชนิดนั้น เช่น

```
print(Int.MIN_VALUE)
print(Int.MAX_VALUE)
print(ULong.MAX_VALUE)
```

การกำหนดตัวแปร

การตั้งชื่อตัวแปรใน Kotlin ก็ใช้แนวทางเดียวกับ Java โดยมีหลักเกณฑ์ที่สำคัญดังนี้

- ต้องไม่ซ้ำกับคีย์เวิร์ดของ Kotlin
- ต้องขึ้นต้นด้วยตัวอักษร หรือเครื่องหมาย _ เช่น myVar, value_
- สามารถใช้ตัวเลขร่วมได้ แต่ห้ามใช้ตัวเลขเป็นตัวขึ้นต้น เช่น num1, year2000

- การเขียนด้วยตัวพิมพ์เล็กใหญ่ต่างกัน เช่น abc, ABC, Abc ถือว่าไม่เหมือนกัน
- โดยทั่วไป เรานิยมตั้งชื่อตัวแปรให้ขึ้นต้นด้วยตัวพิมพ์เล็ก แล้วถ้าเป็นการนำหลายๆ คำมารวมกันเป็นชื่อตัวแปรเดียวกัน ให้เขียนติดกัน แล้วให้ตัวอักษรขึ้นต้นของแต่ละคำเป็นตัวพิมพ์ใหญ่ ยกเว้นคำแรก เช่น kotlinVersion, isValidNumber, numDayOfMonth

การประกาศตัวแปร

สำหรับการประกาศตัวแปรใน Kotlin จะต้องใช้คีย์เวิร์ด `var` ตามด้วยชื่อตัวแปร แล้วก็เครื่องหมาย : และชนิดข้อมูล ในรูปแบบดังนี้

```
var ชื่อตัวแปร: ชนิดข้อมูล
```

แนวทางการประกาศตัวแปร เช่น

```
var x: Int
var result: Double
var str: String
```

อย่างไรก็ตาม สำหรับใน Kotlin เราไม่สามารถประกาศหลายตัวแปรในคำสั่ง `var` อันเดียวกันได้ เช่น กรณีต่อไปนี้ก็จะเกิดข้อผิดพลาด

```
var x, y, z: Int //Error
var a, b, c: String //Error
```

ตัวแปรที่กำหนดชนิดข้อมูลอย่างใดอย่างหนึ่งให้กับมันแล้ว จะเปลี่ยนชนิดข้อมูลเป็นอย่างอื่นอีกไม่ได้ หรือกล่าวได้ว่า เราจะประกาศตัวแปรซ้ำกันไม่ได้ แม้จะกำหนดข้อมูลคนละชนิดก็ตาม เช่น

```
var x: Int
var x: Double //Error
```

การกำหนดค่าให้กับตัวแปร

การกำหนดค่าให้แก่ตัวแปรสำหรับข้อมูลแต่ละชนิดจะมีรูปแบบที่แตกต่างกัน ซึ่งสามารถจำแนกตามชนิดข้อมูลได้ดังนี้

ข้อมูลชนิดตัวเลข

- ในการกำหนดค่าที่เป็นข้อมูลชนิดตัวเลขทั้งหมด สามารถใส่ตัวเลขลงไปได้เลย แต่ต้องไม่มีเครื่องหมาย , คั่นหลักพันรวมอยู่ด้วย เช่น

```
var x: Byte
x = 100 //อาจประกาศตัวแปรไว้ก่อน แล้วค่อยมากำหนดค่าทีหลัง
var y: Int = 456 //หรืออาจประกาศพร้อมกำหนดค่าให้แก่ตัวแปร
var z: Double = 456.789
```

- ตัวแปรประเภทเลขจำนวนเต็มบวก ให้วางตัว u หรือ U ต่อท้ายตัวเลข ยกเว้นชนิด ULong ให้วาง uL หรือ UL ต่อท้าย เช่น

```
var a: UInt = 123u
var b: UShort = 9999U
var c: ULong = 1234567UL
```

- หากนำตัวเลขที่มีทศนิยมไปกำหนดให้ตัวแปรประเภทจำนวนเต็ม จะเกิดข้อผิดพลาด เช่น

```
var x: Int = 123.45 //Error
var y: Short = 10E8 //Error
```

- สำหรับตัวแปรชนิด Double เรา ไม่ สามารถกำหนดค่าที่เป็นจำนวนเต็มให้กับมันได้ (ซึ่งต่างจาก Java ที่ทำเช่นนี้ได้) เช่น กรณีต่อไปนี้จะเกิดข้อผิดพลาด

```
var d1: Double = 123 //Error
var d2: Double = 123.0 //OK
```

- กรณีที่เป็นตัวแปรชนิด Float ให้ใส่อักขระ f หรือ F ต่อท้ายข้อมูล ซึ่งอาจกำหนดค่าที่เป็นเลขทศนิยมหรือจำนวนเต็มให้กับมันก็ได้ เช่น

```
var f1: Float = 123.45F //OK
var f2: Float = 678f //OK
var f3: Float = 1.99 //Error (ต้องใส่ F หรือ f ต่อท้าย)
var f4: Float = 4 //Error (ต้องใส่ F หรือ f ต่อท้าย)
```

- เราอาจใช้เครื่องหมาย underscore (_) ร่วมกับตัวเลข ตามข้อกำหนดต่อไปนี้
 - เขียนแทรกไว้ระหว่างตัวเลข โดยไม่จำเป็นต้องใช้คั่นหลักพัน
 - ห้ามใช้เป็นตัวขึ้นต้นและตัวสุดท้าย
 - สามารถนำตัวเลขนั้นไปคำนวณได้ตามปกติ
 - เมื่อนำไปใช้งาน เช่น แสดงผลด้วย print() เครื่องหมาย _ จะไม่ปรากฏ

```
var a: Int = 1_234;           //OK
var b: Int = 1_234_567;      //OK
var c: Int = 1_2_3_4_5;      //OK
var d: Int = _1234;          //Error ห้ามวาง _ เป็นตัวขึ้นต้น
var e: Int = 1234_;          //Error ห้ามวาง _ เป็นตัวสุดท้าย
var f: Double = 123_456.789 //OK
print(c)                     //12345
```

ข้อมูลชนิดบูลีน

ข้อมูลชนิดบูลีนจะเป็นได้เพียงค่า true (จริง) หรือ false (เท็จ) ใดๆอย่างหนึ่ง ซึ่ง true และ false นี้ก็เป็นคีย์เวิร์ดอยู่แล้ว ดังนั้น จึงสามารถนำไปกำหนดให้แก่ตัวแปรได้เลย เช่น

```
var a: Boolean = true
var b: Boolean = false
```

ข้อมูลชนิดอักขระ (Char)

ข้อมูลชนิดนี้จะกำหนดเป็นอักขระใดๆ เพียง 1 ตัว หรือรหัส Unicode ซึ่งเราต้องเขียนอักขระดังกล่าวไว้ในเครื่องหมาย ' ' เท่านั้น เช่น

```
var grade: Char = 'A'
var size: Char = 'M'
var letter: Char = "K" //Error ต้องกำหนดด้วยเครื่องหมาย ' '
var a: Char = '\u0061'
```

ข้อมูลชนิดข้อความ (String)

สตริงใน Kotlin ก็เช่นเดียวกับ Java โดยเราต้องกำหนดจุดเริ่มต้นและสิ้นสุดของมันด้วยเครื่องหมาย " " เช่น

```
var str: String = "Kotlin for Android"
var strNum: Strig = "1234"
```

สำหรับรายละเอียดพื้นฐานที่ควรรู้เพิ่มเติมของข้อมูลชนิดสตริง จะกล่าวถึงในหัวข้อต่อไป

ข้อมูลชนิด Any

ข้อมูลชนิด Any เราจะกำหนดค่าใดๆ ให้กับมันก็ได้ เช่น

```
var a: Any = 123
a = 4.56
a = "Kotlin"
a = true
```

การกำหนดค่าคงที่

ค่าคงที่ (Constant) เป็นการกำหนดค่าให้กับตัวแปรด้วยค่าใดค่าหนึ่ง และต้องใช้ค่านั้นไปตลอด ไม่สามารถเปลี่ยนค่าเป็นอย่างอื่นได้ ซึ่งใน Kotlin จะกำหนดด้วยคีย์เวิร์ด `val` ดังรูปแบบต่อไปนี้

```
val ชื่อค่าคงที่: ชนิดข้อมูล = ค่าที่กำหนด
```

แนวทางการใช้งาน เช่น

```
val pi: Float = 3.141f
val pricePerUnit: Short = 1000
val lat: Double
val lon: Double
lat = 100.12345
lon = 80.67890
lon = 80.00 //Error เพราะค่าคงที่จะเปลี่ยนแปลงไม่ได้
```

Type Inference

การประกาศตัวแปรหรือค่าคงที่ใน Kotlin เราไม่จำเป็นต้องระบุชนิดข้อมูลใดๆ เลยก็ได้ แต่ต้องกำหนดค่าอย่างใดอย่างหนึ่งให้กับมันในขั้นตอนการประกาศนี้ทันที แล้วตัวแปรหรือค่าคงที่ก็จะเป็นชนิดเดียวกับค่านั้นโดยอัตโนมัติ ซึ่งเราเรียกลักษณะดังกล่าวนี้ว่า Type Inference เช่น กรณีต่อไปนี้

```
var x = 3.14          //x เป็นชนิด Double
var y = "Hello"       //x เป็นชนิด String
val z = true          //x เป็นชนิด Boolean
var a
a = 123                //Error กรณีนี้ต้องกำหนดค่าในขั้นตอนการประกาศตัวแปร เช่น var a = 123
```

หลังจากที่เรากำหนดค่าให้กับมันครั้งแรก ตัวแปรก็จะมีชนิดข้อมูลตามค่านั้นไปเลย และเราจะเปลี่ยนไปกำหนดค่าเป็นข้อมูลชนิดอื่นไม่ได้ เช่น

```
var x = 3.14
x = 4.56              //OK
x = "Hello"           //Error
var y = 123            //Int
y = 456U               //Error (เพราะค่าที่กำหนดเป็นชนิด UInt)
```

การแปลงชนิดข้อมูล

ตามปกติ เราต้องเก็บข้อมูลให้ตรงกับชนิดของตัวแปร แต่ในบางกรณี อาจจำเป็นต้องแปลงชนิดข้อมูลให้สอดคล้องกับเงื่อนไขการใช้งาน โดยวิธีที่ใช้ใน Kotlin คือ นำเมธอด toXxx() มาต่อท้ายตัวแปร หรือตัวเลขเมื่อ Xxx คือชนิดข้อมูลเป้าหมาย เช่น toInt(), toShort(), toLong(), toFloat(), toDouble(), toBoolean(), toString() เป็นต้น ดังแนวทางต่างๆ ต่อไปนี้

```
val x: Byte = 100
val y: Int = x.toInt()

val s: Short = 9999
val l: Long = s.toLong()

val i: Int = 123.45.toInt()      //i = 123
val d: Double = 567.toDouble()  //d = 567.0
```

```
val num1: Int = "1234".toInt() //OK
val num2: Int = "1234.5".toInt() //Error เพราะมีทศนิยม
val num3: Int = "1234.5".toDouble().toInt() //num3 = 1234

val str: String = 123.toString() //str = "123"

val bool1: Boolean = "true".toBoolean() //bool1 = true
val bool2: Boolean = "True".toBoolean() //bool2 = true
val bool3: Boolean = "Yes".toBoolean() //bool3 = false
//ทุกค่าที่ไม่เทียบเท่ากับคำว่า true เมื่อแปลงด้วย toBoolean() จะได้ค่าเป็น false ทั้งหมด
```

ในกรณีการแปลงจากชนิดที่ใหญ่กว่าไปเป็นชนิดที่เล็กกว่า หากข้อมูลมีค่ามากกว่าค่าสูงสุดของชนิดข้อมูลที่เล็กกว่า อาจได้ผลลัพธ์ที่ผิดพลาด ดังนั้น จึงควรพิจารณาให้รอบคอบ เช่น

```
var l: Long = 1234
var s: Short = l.toShort() //s = 1234
l = 123456 //l มีค่าเกินกว่า MAX_VALUE ของชนิด Short
s = l.toShort() //s = -7616
```

ลักษณะเพิ่มเติมเกี่ยวกับสตริง

สำหรับข้อมูลชนิดสตริง มีลักษณะพื้นฐานบางอย่างที่เราควรรู้เพิ่มเติม ดังนี้

การเชื่อมต่อสตริง

ใน Kotlin ก็เช่นเดียวกับ Java ที่สามารถนำสตริงย่อยๆ หรือค่าที่เก็บในตัวแปร มาเชื่อมต่อกันด้วยเครื่องหมาย + เช่น

```
val str1 = "IntelliJ" + " " + "IDEA"
val str2 = "108" + "1009" //1081009
val day = 25
val month = "มิถุนายน"
val now = "วันที่ " + day + " เดือน " + month
```

แต่ถ้าเริ่มต้นด้วยตัวเลข ต้องแปลงตัวเลขนั้นให้เป็นสตริงก่อน เช่น

```
val day = 25
val now = day.toString() + " มิถุนายน"
```

การเขียนสตริงหลายบรรทัด

เครื่องหมาย " " ในการกำหนดสตริงนั้น ต้องเขียนไว้ในบรรทัดเดียวกัน จะแยกสตริงที่กำหนดไว้ในเครื่องหมาย " " อันเดียวกันไปไว้คนละบรรทัดไม่ได้ เช่น กรณีต่อไปนี้เกิดข้อผิดพลาด

```
var str = "Write Once
          Run Anywhere"    //Error!
```

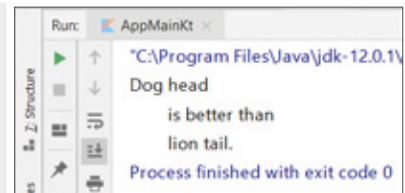
ถ้าเราไม่สะดวกที่จะเขียนสตริงที่ยาวๆ ไว้ในเครื่องหมาย " " เดียวกันทั้งหมด ให้แยกเป็นสตริงย่อยๆ แล้วนำมาต่อกันด้วยเครื่องหมาย + เช่น

```
var str = "Write Once " +
          "Run Anywhere"
```

นอกจากนี้ Kotlin ยังรองรับการกำหนดสตริงหลายบรรทัดด้วย Triple Quotes (""" """) ในลักษณะดังโค้ดถัดไป แต่เมื่อเรานำไปแสดงผล สตริงจะเยื้องเข้าไปตามโค้ดที่เขียน ดังภาพ

```
val str = """Dog head
            is better than
            lion tail."""

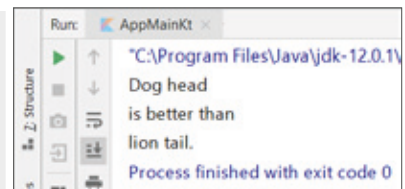
print(str)
```



หากต้องการแก้ปัญหการเยื้องดังกล่าว ให้ขึ้นต้นสตริงแต่ละบรรทัดด้วยเครื่องหมาย | แล้วต่อท้าย Triple Quote ด้วยเมธอด trimMargin() ทั้งนี้หาก IntelliJ ใส่เมธอด trimIndent() ไว้ล่วงหน้า ก็ให้เปลี่ยนเป็นเมธอด trimMargin() แทน เช่น

```
val str = """Dog head
            |is better than
            |lion tail.""".trimMargin()

print(str)
```



การแทรกค่าของตัวแปรในสตริง

จุดเด่นที่สำคัญอีกอย่างของ Kotlin ที่ไม่มีใน Java ก็คือ การแทรกค่าตัวแปรลงในสตริง ซึ่งเป็นสิ่งมีประโยชน์ต่อการใช้งานค่อนข้างมากทีเดียว โดยมีหลักการที่น่าสนใจดังนี้

- รูปแบบพื้นฐานคือ ให้วางเครื่องหมาย \$ นำหน้าตัวแปรเมื่อแทรกลงในสตริง เช่น

```
val name = "สมชาย"
val num = 5
print("ขอเชิญคุณ $name ที่ช่องบริการหมายเลข $num ค่ะ")
val mu = 2
val lp = 3
val r = "ManU $mu : $lp Liverpool" //ManU 2 : 3 Liverpool
```

- หากจำเป็นต้องเขียนตัวแปรติดกันกับอักขระที่ตั้งเป็นชื่อตัวแปร ก็ให้ใส่วงเล็บ { } ครอบตัวแปร แล้ววางเครื่องหมาย \$ ไว้หน้าวงเล็บดังกล่าว เช่น

```
val w = 100
val h = 200
val str = "width: ${w}px; height: ${h}px"
//width: 100px; height 200px
```

- เราสามารถใช้เครื่องหมาย \$ ร่วมกับวงเล็บ { } เพื่อกำหนดการกระทำกับตัวแปร เช่น การคำนวณทางคณิตศาสตร์ แล้วแทรกผลลัพธ์ลงที่ตำแหน่งนั้น ในลักษณะดังนี้

```
val x: Int = 10
val y: Int = 20
print("$x + $y = ${x + y}") //10 + 20 = 30
```

การเขียนอักขระพิเศษบางตัวในสตริง

หากจะเขียนอักขระพิเศษบางตัวลงในสตริง ต้องใส่เครื่องหมาย \ นำหน้า ซึ่งตัวที่น่าสนใจคือ

\n	สำหรับการขึ้นบรรทัดใหม่	\"	สำหรับแทรกเครื่องหมาย "
\t	สำหรับการเว้นระยะ 1 แท็บ	\'	กำหนดค่าให้ข้อมูลชนิด char เป็นเครื่องหมาย ' เช่น
\\	เมื่อเขียนเครื่องหมาย \ ไว้ในสตริง		val c: char = '\'
\\$	เมื่อเขียนเครื่องหมาย \$ ไว้ในสตริง		

แนวทางการนำอักขระ Escape ไปใช้ เช่น

```
print("Press \"Enter\" to continue")
//แสดงข้อความ => Press "Enter" to continue

print("บรรทัดที่ 1\nบรรทัดที่ 2\nบรรทัดที่ 3" + "\n" + "บรรทัดที่ 4" + "\nบรรทัดที่ 5")
//แสดงข้อความ => บรรทัดที่ 1 ถึง บรรทัดที่ 5 อยู่คนละบรรทัด

print("วันที่\tyอดขายสินค้าชนิดที่ 1\tยอดขายสินค้าชนิดที่ 2\tยอดขายสินค้าชนิดที่ 3")
//เว้นระยะ 1 แท็บ => วันที่   ยอดขายสินค้าชนิดที่ 1   ยอดขายสินค้าชนิดที่ 2   ยอดขายสินค้าชนิดที่ 3

val path = "C:\\temp\\images\\bird.png"
var msg = "สินค้าชิ้นนี้ ราคา \\\$200"
```

การรับข้อมูลทางคีย์บอร์ด

การรับข้อมูลทางคีย์บอร์ด ทำให้ผู้ใช้สามารถกำหนดค่าของข้อมูลได้ตามต้องการ โดยใน Kotlin สามารถทำได้ง่ายๆ ดังนี้

- สร้างตัวแปรโดยไม่ระบุชนิดข้อมูล (ความจริงมันคือชนิด String? แบบ Nullable ซึ่งจะกล่าวถึงเรื่องนี้ในบทต่อไป) แล้วให้รับค่าจากฟังก์ชัน `readLine()`
- เราควรแสดงข้อความด้วยฟังก์ชัน `print()` ก่อนรับข้อมูลด้วย `readLine()` เพื่อบ่งบอกให้ทราบว่าจะต้องใส่ข้อมูลเกี่ยวกับอะไร

```
print("กรุณาใส่ตัวเลข 1 - 10: ")
var n = readLine()
println("เลขที่ท่านระบุคือ $n")
```

- ค่าที่ได้จาก `readLine()` จะเป็นชนิด String เสมอ แม้เราจะใส่ค่าเป็นตัวเลขทั้งหมดก็ตาม ซึ่งหากต้องการนำไปคำนวณ อาจต้องแปลงเป็นชนิดตัวเลขก่อน เช่น

```
var n = readLine()
var num = n!!.toInt() //รายละเอียดของโอเปอเรเตอร์ !! อยู่ในบทที่ 5
```

อย่างไรก็ตาม การแปลงด้วยเมธอด `toXxx()` ก็อาจเกิดข้อผิดพลาด ถ้าข้อมูลนั้นไม่สอดคล้องหรือไม่อยู่ในหลักเกณฑ์ที่จะแปลงได้ ซึ่งเราจะได้เรียนรู้วิธีการตรวจสอบอีกครั้งในบทต่อไป



Kotlin



พัฒนา Mobile App บนระบบ Android ด้วย Kotlin

Kotlin เป็นหนึ่งในภาษาที่ใช้พัฒนาแอปบนระบบ Android เช่นเดียวกับ Java แต่การเขียนโค้ดจะกระชับและเข้าใจง่ายกว่า รวมทั้งมีเทคนิคที่ช่วยลดความยุ่งยากในหลายรูปแบบ ดังนั้นเพื่อให้ผู้อ่านสามารถเรียนรู้ทั้งการเขียนโปรแกรมด้วย Kotlin และการพัฒนาแอปบน Android ไปพร้อมๆ กัน ในหนังสือเล่มนี้จึงได้รวบรวมเนื้อหาที่จำเป็นต้องรู้ทั้งหมดไว้อย่างครบถ้วน ซึ่งผู้อ่านสามารถเรียนรู้ด้วยตนเองได้อย่างต่อเนื่อง ตั้งแต่ขั้นพื้นฐานไปจนถึงการประยุกต์ใช้งานเพื่อพัฒนา Mobile App บนระบบ Android ด้วย Kotlin ตามต้องการ

เนื้อหาในหนังสือเล่มนี้ประกอบด้วย :

บทที่ 1 : Kotlin Basic & Data Type
บทที่ 2 : Operator & Control Flow
บทที่ 3 : Array, String & Number
บทที่ 4 : Function & Lambda
บทที่ 5 : Null Safety & Exception
บทที่ 6 : Class & Object
บทที่ 7 : Inheritance & Interface
บทที่ 8 : Generic & Collection
บทที่ 9 : Android Studio
บทที่ 10 : Basic App Development
บทที่ 11 : Views & UI

บทที่ 12 : Toast, Snackbar & Layout
บทที่ 13 : Activity & Explicit Intent
บทที่ 14 : Fragment, ViewPager & Tab
บทที่ 15 : Menu & Navigation
บทที่ 16 : SQLite Database
บทที่ 17 : RecyclerView & CardView
บทที่ 18 : Permission & Implicit Intent
บทที่ 19 : Multimedia & Camera
บทที่ 20 : Alert & Custom Dialog
บทที่ 21 : Workshop (To-Do List)
บทที่ 22 : Workshop (Torchlight)



www.se-ed.com



SE-ED Publisher

พร้อมจำหน่ายในรูปแบบ

- | | |
|--|---|
| ☐ e-book (PDF) | ☐ audiobooks |
| ☐ e-book (EPUB) | ☐ audio CD / MP3 |
| ☒ ปกอ่อน | ☐ LARGE PRINT (ตัวอักษรขนาดใหญ่) |



คอมพิวเตอร์ - การเขียนโปรแกรม