

جزوه درس برنامه نویسی موبایل ۱

ساخت نخستین پروژه با

کاتلین

Contents

معرفی کاتلین	۳
چرا کاتلین؟	۳
این کتاب برای چه کسانی است؟	۴
نحوه استفاده از این کتاب	۴
برای افراد کنجکاوتر	۵
چالش‌ها	۵
قراردادهای تایپوگرافی	۵
استفاده از کتاب الکترونیکی	۶
نگاه به آینده	۶
بخش اول: شروع کار	۷
اولین برنامه کاتلین شما	۷
نصب IntelliJ IDEA	۷
اولین پروژه کاتلین شما	۸
اجرای فایل کاتلین شما	۱۸
قابلیت REPL در کاتلین	۲۰
برای افراد کنجکاوتر: چرا از IntelliJ استفاده کنیم؟	۲۲
برای افراد کنجکاوتر: هدف‌گیری JVM	۲۳
چالش: محاسبات ریاضی در REPL	۲۴

معرفی کاتلین

در سال ۲۰۱۱، شرکت JetBrains توسعه زبان برنامه‌نویسی کاتلین را به عنوان جایگزینی برای نوشتن کد در زبان‌هایی مانند Java یا Scala جهت اجرا بر روی ماشین مجازی جاوا (Java Virtual Machine) اعلام کرد. شش سال بعد، Google اعلام کرد که کاتلین به یک مسیر توسعه با پشتیبانی رسمی برای سیستم‌عامل اندروید تبدیل خواهد شد.

دامنه کاربرد کاتلین به سرعت از زبانی با آینده‌ای روشن، به زبانی که برنامه‌های کاربردی را در برترین سیستم‌عامل موبایل جهان قدرت می‌بخشد، گسترش یافت. امروزه، شرکت‌های بزرگی مانند Google، Uber، Netflix، Capital One، Amazon و غیره، کاتلین را به خاطر مزایای فراوانش از جمله نحو (syntax) مختصر، ویژگی‌های مدرن و تعامل‌پذیری یکپارچه با کدهای قدیمی جاوا پذیرفته‌اند.

چرا کاتلین؟

برای درک جذابیت کاتلین، ابتدا باید نقش جاوا را در چشم‌انداز توسعه نرم‌افزار مدرن درک کنید. این دو زبان ارتباط تنگاتنگی با یکدیگر دارند، زیرا کد کاتلین در اغلب موارد برای ماشین مجازی جاوا نوشته می‌شود.

جاوا یک زبان قدرتمند و آزمون‌پس‌داده است و سال‌هاست که یکی از رایج‌ترین زبان‌های نوشته‌شده در پایگاه‌های کد محیط تولید (production codebases) به شمار می‌رود. با این حال، از زمان انتشار جاوا در سال ۱۹۹۵، دانش زیادی در مورد ویژگی‌های یک زبان برنامه‌نویسی خوب به دست آمده است. جاوا فاقد بسیاری از پیشرفت‌هایی است که توسعه‌دهندگان شاغل با زبان‌های مدرن‌تر از آن‌ها بهره‌مند هستند.

کاتلین از تجربیات به‌دست‌آمده بهره می‌برد، زیرا برخی از تصمیمات طراحی اتخاذشده در جاوا (و زبان‌های دیگر مانند Scala) با گذشت زمان کارایی خود را از دست داده‌اند. این زبان فراتر از آنچه با زبان‌های قدیمی‌تر امکان‌پذیر بود تکامل یافته و مشکلات آزاردهنده آن‌ها را برطرف کرده است. در فصل‌های آینده در مورد اینکه چگونه کاتلین فراتر از جاوا عمل می‌کند و تجربه توسعه قابل‌اطمینان‌تری را ارائه می‌دهد، بیشتر خواهید آموخت.

علاوه بر این، کاتلین صرفاً یک زبان بهتر برای نوشتن کد جهت اجرا بر روی ماشین مجازی جاوا نیست. این یک زبان چندپلتفرمی (multiplatform) است که هدف آن کاربرد همه‌منظوره (general purpose) است: از کاتلین می‌توان برای نوشتن برنامه‌های بومی (native) برای iOS، macOS و Windows؛ برنامه‌های JavaScript؛ و البته برنامه‌های اندروید استفاده کرد. اخیراً، JetBrains در حال سرمایه‌گذاری بر روی این قابلیت‌های چندپلتفرمی (cross-platform) بوده است؛ قابلیت چندپلتفرمی کاتلین (Kotlin Multiplatform) روشی منحصربه‌فرد برای اشتراک‌گذاری کد بین برنامه‌های مختلف ارائه می‌دهد و منجر به افزایش استفاده از کاتلین فراتر از ماشین مجازی جاوا شده است.

این کتاب برای چه کسانی است؟

ما این کتاب را برای انواع توسعه‌دهندگان نوشته‌ایم: توسعه‌دهندگان باتجربه اندروید که خواهان ویژگی‌های مدرنی فراتر از امکانات جاوا هستند، توسعه‌دهندگان سمت سرور که علاقه‌مند به یادگیری ویژگی‌های کاتلین می‌باشند، توسعه‌دهندگانی که به دنبال اشتراک‌گذاری کد کاتلین بین برنامه‌های بومی یا تحت وب خود هستند، و توسعه‌دهندگان تازه‌کاری که می‌خواهند وارد دنیای یک زبان کامپایل‌شونده (compiled) با عملکرد بالا شوند.

پشتیبانی از اندروید ممکن است دلیل مطالعه این کتاب توسط شما باشد، اما این کتاب محدود به برنامه‌نویسی کاتلین برای اندروید نیست. در واقع، تمام کدهای کاتلین در این کتاب مستقل از چارچوب (framework) اندروید هستند. با این وجود، اگر علاقه‌مند به استفاده از کاتلین برای توسعه برنامه‌های اندروید هستید، این کتاب الگوهای رایجی را به نمایش می‌گذارد که نوشتن برنامه‌های اندروید با کاتلین را بسیار آسان می‌سازد.

اگرچه کاتلین تحت تأثیر تعدادی از زبان‌های دیگر قرار گرفته است، اما برای یادگیری کاتلین نیازی به دانستن جزئیات هیچ زبان دیگری ندارید. هر از گاهی، ما در مورد معادل کد جاوای قطعه‌کدهایی که به زبان کاتلین نوشته‌اید بحث خواهیم کرد. همچنین در صورت لزوم به شباهت‌هایی با زبان‌های دیگر اشاره خواهیم نمود. اگر در این زبان‌ها تجربه دارید، این امر به شما کمک می‌کند تا رابطه بین کاتلین و پلتفرم‌هایی که از آن‌ها پشتیبانی می‌کند را درک کنید. حتی اگر این مقایسه‌ها برای شما کمتر آشنا باشند، مشاهده اینکه چگونه یک زبان دیگر با مشکلات مشابه برخورد می‌کند، می‌تواند به شما در درک اصولی که توسعه کاتلین را شکل داده‌اند کمک کند.

نحوه استفاده از این کتاب

این کتاب یک راهنمای مرجع نیست. هدف ما راهنمایی شما در مهم‌ترین بخش‌های زبان برنامه‌نویسی کاتلین است. شما روی پروژه‌های نمونه کار خواهید کرد و با پیشرفت خود، دانشتان را افزایش خواهید داد. برای بهره‌گیری حداکثری از این کتاب، توصیه می‌کنیم در حین خواندن، مثال‌های کتاب را خودتان تایپ کنید. کار کردن روی پروژه‌ها به ایجاد حافظه عضلانی کمک کرده و اندوخته‌ای برای انتقال از یک فصل به فصل دیگر به شما می‌دهد.

همچنین، هر فصل بر مبنای مباحث ارائه‌شده در فصل قبل بنا شده است، بنابراین توصیه می‌کنیم مطالب را پرش نکنید. حتی اگر احساس می‌کنید با یک موضوع در زبان‌های دیگر آشنا هستید، پیشنهاد می‌کنیم آن را به طور کامل مطالعه کنید - کاتلین بسیاری از مشکلات را به روش‌های منحصربه‌فردی مدیریت می‌کند. شما با مباحث مقدماتی مانند متغیرها و روند کنترل (control flow) شروع خواهید کرد، مسیر خود را از طریق تکنیک‌های برنامه‌نویسی شیء‌گرا و تابعی ادامه می‌دهید، رویکرد اختصاصی (first-party) کاتلین برای اجرای کدهای ناهمگام (asynchronous) را امتحان کرده و به قابلیت‌های چندپلتفرمی کاتلین ورود خواهید کرد. در پایان کتاب، دانش خود را در مورد کاتلین از سطح یک فرد مبتدی به یک توسعه‌دهنده پیشرفته‌تر ارتقا خواهید داد.

با این وجود، عجله نکنید: موضوعات را بسط دهید، از مرجع کاتلین در kotlinlang.org/docs/reference برای پیگیری هر چیزی که کنجکاوی شما را برانگیخته است استفاده کنید و به آزمایش و تمرین بپردازید.

برای افراد کنجکاوتر

بیشتر فصل‌های این کتاب دارای یک یا دو بخش با عنوان «برای افراد کنجکاوتر» هستند. بسیاری از این بخش‌ها مکانیسم‌های زیربنایی زبان کاتلین را روشن می‌سازند. مثال‌های موجود در فصل‌ها به اطلاعات این بخش‌ها وابسته نیستند، اما اطلاعات اضافی و مکملی را فراهم می‌کنند که ممکن است برایتان جالب یا مفید باشد.

چالش‌ها

بسیاری از فصل‌ها با یک یا چند چالش به پایان می‌رسند. این‌ها مسائل اضافی برای حل کردن هستند که جهت تعمیق درک شما از کاتلین طراحی شده‌اند. ما شما را تشویق می‌کنیم که برای افزایش تسلط خود بر کاتلین، آن‌ها را امتحان کنید.

توصیه می‌کنیم قبل از کار روی یک چالش، از پروژه‌های خود یک کپی بگیرید؛ فصل‌های دیگر اغلب بر پایه راه‌حل‌های قبلی ساخته می‌شوند و شما نمی‌خواهید تغییراتتان در ادامه مسیر مانعی ایجاد کنند. همچنین می‌توانید راه‌حل‌های تمرین‌های این کتاب را از آدرس bignerdranch.com/kotlin-2e-solutions دانلود کنید.

<https://bignerdranch.com/kotlin-2e-solutions/>

قراردادهای تایپوگرافی

در حین ساخت پروژه‌های این کتاب، ما شما را با معرفی یک موضوع و سپس نشان دادن نحوه به‌کارگیری دانش جدیدتان راهنمایی خواهیم کرد. برای وضوح بیشتر، ما به قراردادهای تایپوگرافی زیر پایبند هستیم.

متغیرها، مقادیر و نوع‌ها (types) با فونت با عرض ثابت (fixed-width font) نشان داده می‌شوند. نام کلاس‌ها، توابع و رابط‌ها (interfaces) با فونت پررنگ نشان داده شده‌اند.

تمام قطعه کدها با فونت با عرض ثابت نشان داده شده‌اند. اگر قرار است کدی را در یک قطعه کد تایپ کنید، آن کد به صورت پررنگ مشخص شده است. اگر قرار است کدی را در یک قطعه کد حذف کنید، روی آن کد خط کشیده شده است. در مثال زیر، به شما دستور داده می‌شود خطی که متغیر `y` را تعریف می‌کند حذف کرده و متغیری به نام `z` اضافه کنید:

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

کاتلین زبانی در حال بلوغ است و قراردادهای کدنویسی آن با گذشت زمان همچنان در حال تکامل هستند. احتمالاً شما سبک خاص خود را توسعه خواهید داد، اما ما تمایل داریم به راهنماهای سبک کاتلین متعلق به JetBrains و Google پایبند باشیم:

قراردادهای کدنویسی JetBrains: kotlinlang.org/docs/coding-conventions.html

راهنمای سبک Google: developer.android.com/kotlin/style-guide

استفاده از کتاب الکترونیکی

اگر این کتاب را روی یک کتابخوان الکترونیکی (eReader) می‌خوانید، باید اشاره کنیم که خواندن کدها ممکن است گاهی دشوار باشد. خطوط طولانی‌تر کد ممکن است بسته به اندازه فونت انتخابی شما به خط دوم شکسته شوند (wrap شوند).

طولانی‌ترین خطوط کد در این کتاب ۸۶ کاراکتر با عرض ثابت (monospace) هستند، مانند مورد زیر.

```
println(playerCreateMessage(nameIsLong("Polarcubis, the Supreme Master of NyetHack")))
```

می‌توانید تنظیمات کتابخوان الکترونیکی خود را تغییر دهید تا بهترین حالت را برای مشاهده خطوط طولانی کد پیدا کنید.

اگر در حال مطالعه روی یک iPad با برنامه Books اپل هستید، توصیه می‌کنیم به برنامه Settings بروید، Books را انتخاب کنید و تنظیمات Full Justification و Auto-hyphenation را در وضعیت OFF (خاموش) قرار دهید.

وقتی به مرحله‌ای رسیدید که در واقع در حال تایپ کردن کد هستید، پیشنهاد می‌کنیم کتاب را روی رایانه شخصی (PC) یا Mac خود در نرم‌افزار Adobe Digital Editions باز کنید. (نرم‌افزار Adobe Digital Editions یک برنامه کتابخوان الکترونیکی رایگان است که می‌توانید از www.adobe.com/products/digitaleditions دانلود کنید). پنجره برنامه را به اندازه‌ای بزرگ کنید که بتوانید کدها را بدون شکسته شدن خطوط مشاهده کنید. همچنین قادر خواهید بود اشکال را با جزئیات کامل ببینید.

نگاه به آینده

برای کار با مثال‌های این کتاب وقت بگذارید. زمانی که با نحو (syntax) کاتلین آشنا شدید، فکر می‌کنیم فرآیند توسعه را واضح، عمل‌گرایانه و روان خواهید یافت. تا آن زمان، به تلاش خود ادامه دهید؛ یادگیری یک زبان جدید می‌تواند بسیار نتیجه‌بخش و لذت‌بخش باشد.

بخش اول: شروع کار

دو فصل اول این کتاب شما را با اصول اولیه استفاده از IntelliJ IDEA، محیط توسعه یکپارچه (IDE) رسمی برای توسعه برنامه‌های کاربردی کاتلین، آشنا می‌کند. شما کار خود را با ایجاد یک پروژه ساده آغاز خواهید کرد تا با ویژگی‌های اساسی این زبان احساس راحتی کنید. برای شروع، به بررسی انواع داده (types) در کاتلین می‌پردازید که داده‌هایی را که در یک برنامه با آن‌ها کار می‌کنید دسته‌بندی می‌کنند.

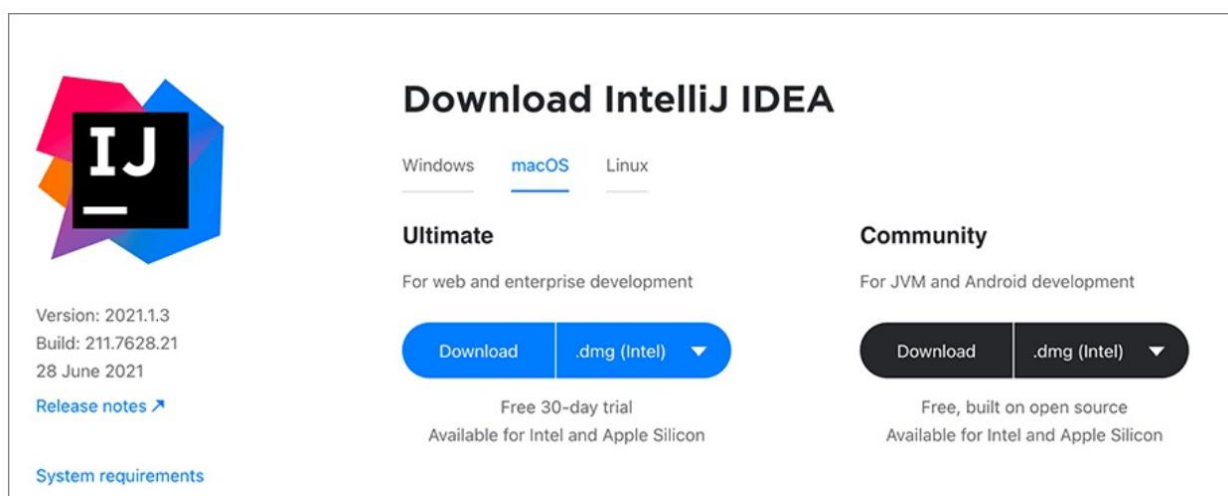
اولین برنامه کاتلین شما

در این فصل، اولین برنامه کاتلین خود را با استفاده از IntelliJ IDEA خواهید نوشت. در حین انجام این مرحله‌ی گذار برنامه‌نویسی، با محیط توسعه خود آشنا می‌شوید، یک پروژه جدید کاتلین ایجاد می‌کنید، کد کاتلین را نوشته و اجرا می‌کنید، و خروجی حاصل را بررسی می‌نمایید. پروژه‌ای که در این فصل ایجاد می‌کنید، به عنوان بستری آزمایشی برای ویژگی‌های بنیادی زبان که در سراسر برنامه‌های کاربردی کاتلین خود از آن‌ها استفاده خواهید کرد، عمل می‌کند.

نصب IntelliJ IDEA

نرم‌افزار IntelliJ IDEA یک محیط توسعه یکپارچه (IDE) برای کاتلین است که توسط شرکت JetBrains (که زبان کاتلین را نیز ایجاد کرده است) ساخته شده است. برای شروع، نسخه Community نرم‌افزار IntelliJ IDEA را از وبسایت JetBrains به آدرس jetbrains.com/idea/download دانلود کنید (شکل ۱.۱).

Figure 1.1 Downloading IntelliJ IDEA Community Edition



پس از دانلود، دستورالعمل‌های نصب مربوط به پلتفرم خود را که در صفحه نصب و راه‌اندازی JetBrains در آدرس jetbrains.com/help/idea/installationguide.html#standalone توضیح داده شده است، دنبال کنید.

IntelliJ IDEA که به اختصار IntelliJ نامیده می‌شود، به شما کمک می‌کند تا کدهای کاتلین را با ساختار مناسب بنویسید. این محیط همچنین فرآیند توسعه را با ابزارهای داخلی برای اجرا، اشکال‌زدایی (debugging)، بازرسی و بازآرایی (refactoring) کد شما ساده می‌کند. می‌توانید در بخش «برای افراد کنجکاوتر: چرا از IntelliJ استفاده کنیم؟» در اواخر این فصل، درباره دلایل توصیه ما برای استفاده از IntelliJ جهت نوشتن کد کاتلین بیشتر بخوانید.

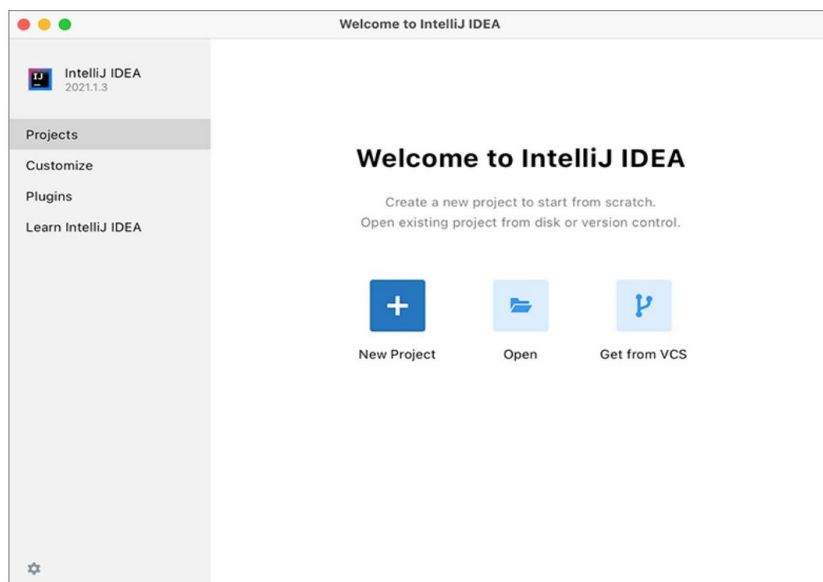
اولین پروژه کاتلین شما

تبریک می‌گوییم، اکنون زبان برنامه‌نویسی کاتلین و یک محیط توسعه قدرتمند برای نوشتن آن را در اختیار دارید. حالا تنها یک کار باقی مانده است: یاد بگیرید به روانی به زبان کاتلین صحبت کنید. اولین قدم - ایجاد یک پروژه کاتلین است.

در بیشتر بخش‌های این کتاب، پروژه‌هایی که روی آن‌ها کار می‌کنید بر روی یک دنیای بازی فانتزی تمرکز دارند که در آن یک قهرمان وارد مأموریت‌های قهرمانانه می‌شود، شیاطین شرور را شکست می‌دهد، شهرها را از خطرات حتمی نجات می‌دهد و به طور کلی کارهایی را انجام می‌دهد که قهرمانان انجام می‌دهند. اولین پروژه شما یک «تخته پاداش» (bounty board) می‌سازد، یک سیستم مأموریت که قهرمان داستان به نام Madrigal را به سمت وظایفی هدایت می‌کند که به توجه او نیاز دارند. شما تا فصل ۷ روی این پروژه کار خواهید کرد.

IntelliJ را باز کنید. پنجره خوش‌آمدگویی IntelliJ IDEA به شما نمایش داده خواهد شد (شکل ۱.۲).

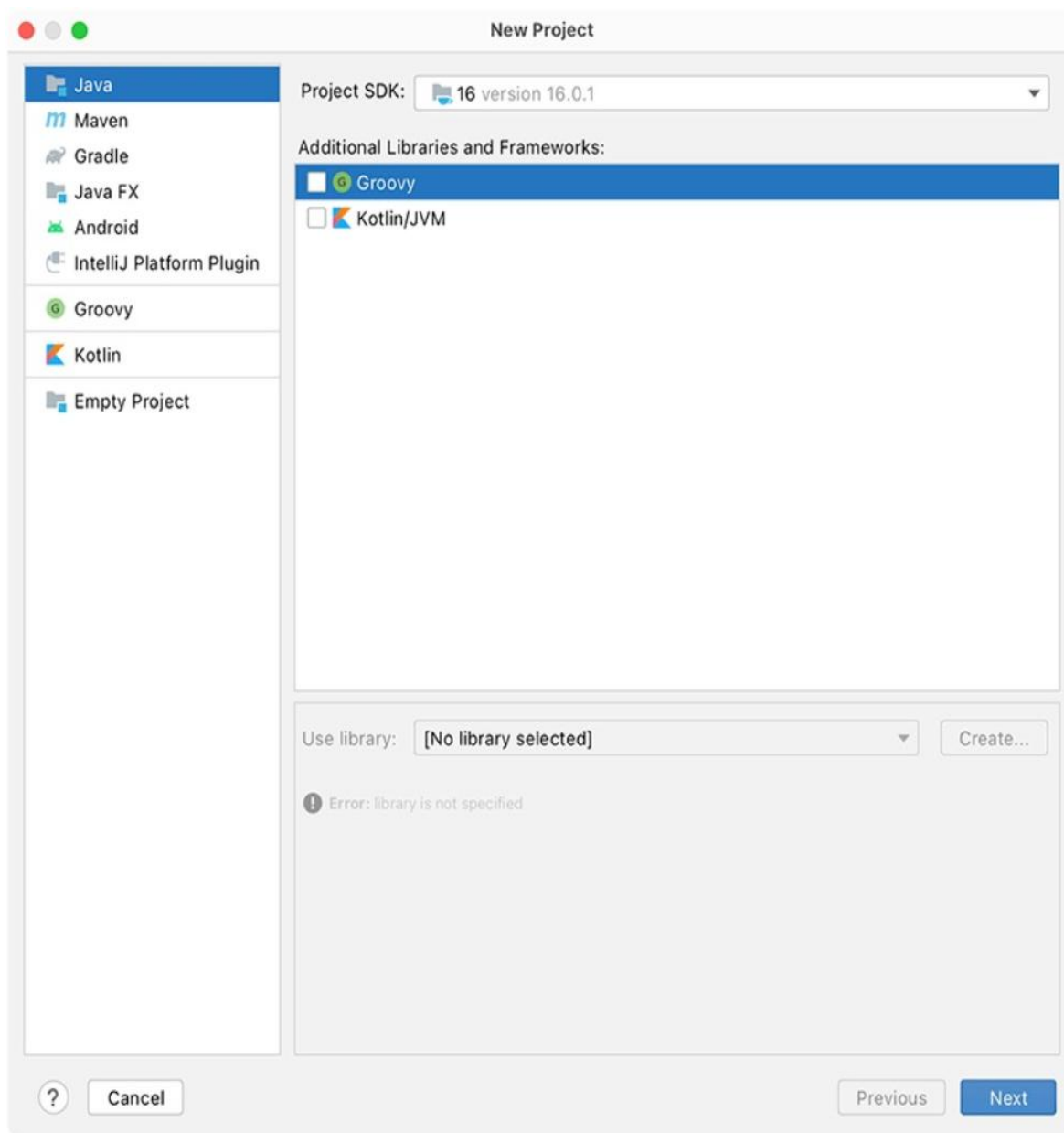
Figure 1.2 Welcome dialog



(اگر این اولین باری نیست که IntelliJ را پس از نصب باز می‌کنید، ممکن است مستقیماً به آخرین پروژه‌ای که باز کرده بودید هدایت شوید. برای بازگشت به صفحه خوش‌آمدگویی، پروژه را با استفاده از File سپس Close Project ببندید.)

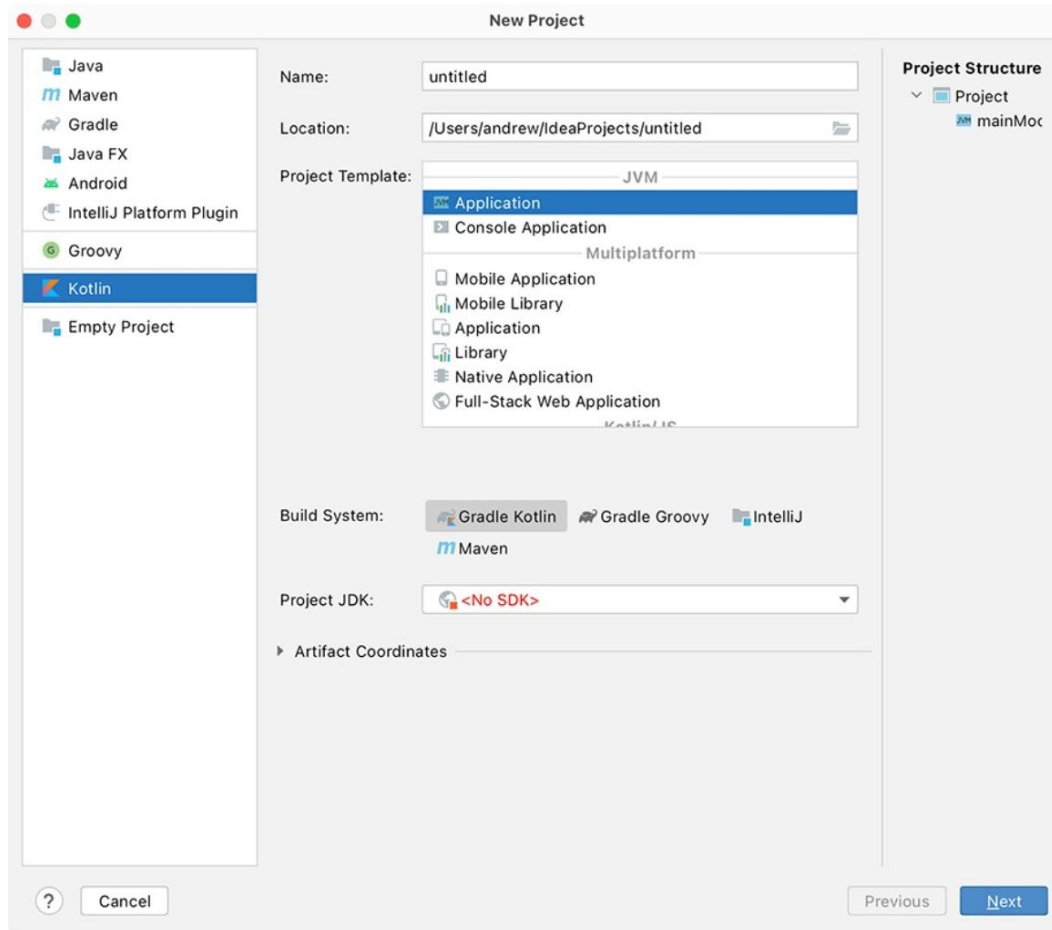
روی New Project کلیک کنید. IntelliJ پنجره New Project را نمایش می‌دهد، همانطور که در شکل ۱.۳ نشان داده شده است.

Figure 1.3 New Project dialog



در پنجره New Project، گزینه Kotlin را در سمت چپ انتخاب کنید، همانطور که در شکل ۱.۴ نشان داده شده است.

Figure 1.4 Creating a Kotlin project



شما می‌توانید به طور پیش‌فرض از IntelliJ برای نوشتن کد به زبان‌هایی غیر از کاتلین، از جمله Java و Groovy استفاده کنید. با استفاده از افزونه‌های (plugins) اضافی، IntelliJ می‌تواند برای نوشتن کد به زبان‌هایی مانند Python، Scala، Dart و Rust نیز مورد استفاده قرار گیرد. انتخاب Kotlin از بین گزینه‌های زبان در سمت چپ پنجره New Project، به IntelliJ می‌گوید که شما قصد دارید از کاتلین استفاده کنید.

اکنون نگاهی به تنظیمات پروژه در بخش مرکزی پنجره بیندازید.

در بالای پنجره New Project، برای Name مقدار bounty-board را وارد کنید. فیلد Location به طور خودکار پر می‌شود. می‌توانید مکان را همانطور که هست رها کنید یا با کلیک روی نماد پوشه در سمت راست فیلد، مکان جدیدی را انتخاب نمایید.

منوی Project Template گزینه‌های مختلفی را در زیر سه عنوان اصلی دارد: JVM، Multiplatform و Kotlin/JS (و همچنین یک بخش Experimental). گزینه Application را در زیر عنوان JVM انتخاب کنید. این کار به IntelliJ می‌گوید که قصد دارید کد کاتلینی بنویسید که هدف آن (تارگت)، اجرا بر روی ماشین مجازی جاوا (Java Virtual Machine) است.

کد کاتلین می‌تواند برای هر یک از سه هدفی که توسط این زبان پشتیبانی می‌شود، کامپایل شود: ماشین مجازی جاوا، پلتفرم‌های بومی x86 و ARM (که در بخش Multiplatform در لیست Project Template قرار دارند) و جاوا اسکریپت (با نام اختصاری JS). از آنجایی که کاتلین می‌تواند در هر یک از این سه هدف کامپایل شود، گاهی اوقات اصطلاحات Kotlin/JVM، Kotlin/JS و Kotlin/Native را مشاهده خواهید کرد که “نوع” کاتلین را بر اساس پلتفرم هدف مشخص می‌کنند.

Kotlin/JVM همان چیزی است که بیشتر مردم هنگام گفتن “کاتلین” به آن فکر می‌کنند. هر زمان که کدی می‌نویسید که ماشین مجازی جاوا را هدف قرار می‌دهد، از Kotlin/JVM استفاده می‌کنید. به همین ترتیب، زمانی که می‌خواهید کدهای جاوا اسکریپت را از کد کاتلین خود خروجی بگیرید، از Kotlin/JS استفاده خواهید کرد. و Kotlin/Native یک اصطلاح کلی برای تمام برنامه‌های کاتلین است که به کد ماشین بومی کامپایل می‌شوند. همانطور که از عنوان Multiplatform پیداست، می‌توانید از Kotlin/Native برای ساخت نرم‌افزار برای پلتفرم‌های بسیاری استفاده کنید، مانند کتابخانه‌های iOS، برنامه‌های دسکتاپ بومی و دستگاه‌های تعبیه‌شده (embedded devices) در صورت تمایل و بلندپروازی.

(از این پس، ما به ماشین مجازی جاوا با عنوان “JVM” اشاره خواهیم کرد، زیرا در جامعه توسعه‌دهندگان جاوا معمولاً به این نام خولنده می‌شود. در بخش “برای افراد کنجکاوتر: هدف‌گیری JVM” در اواخر این فصل، می‌توانید در مورد هدف قرار دادن JVM بیشتر بیاموزید.)

پلتفرم Kotlin/JVM بالغ‌ترین پلتفرم از بین این سه مورد است. این پلتفرم اولین موردی بود که توسط کاتلین پشتیبانی شد و همچنان یکی از رایج‌ترین پلتفرم‌ها برای توسعه کاتلین به شمار می‌رود. برای بخش عمده‌ای از این کتاب، شما از Kotlin/JVM استفاده خواهید کرد. دلایل متعددی برای این تصمیم وجود دارد:

پلتفرم JVM دستیابی به استقلال از پلتفرم را بسیار آسان‌تر می‌کند: در هر جایی که JVM اجرا شود، کد شما نیز می‌تواند اجرا گردد. شما می‌توانید یک باینری (binary) توزیع کنید که روی هر کامپیوتری، صرف نظر از معماری آن، کار کند. جاوا یک زبان بسیار بالغ است و رابط‌های برنامه‌نویسی کاربردی (API) فراوانی دارد که در سراسر این کتاب از آن‌ها استفاده خواهید کرد. این API‌های سطح بالا انجام وظایف خاصی را در مقایسه با API‌های سطح پایین موجود در پلتفرم‌های بومی بسیار آسان‌تر می‌کنند. مثال‌های این کتاب در داخل IDE شما اجرا می‌شوند، اما قابلیت اجرا از هر ترمینالی (terminal) را نیز دارند. این موضوع جاوا اسکریپت را به هدفی با مطلوبیت کمتر تبدیل می‌کند، زیرا در درجه اول در یک مرورگر اجرا می‌شود تا یک ترمینال.

کاتلین در سراسر این سه پلتفرم هدف رفتار بسیار مشابهی دارد، بنابراین دانش مربوط به Kotlin/JVM مستقیماً برای Kotlin/JS و Kotlin/Native نیز کاربرد دارد. همه API‌ها در تمام پلتفرم‌های هدف در دسترس نیستند، بنابراین ما در زمان استفاده از API‌ای که فقط روی JVM قابل استفاده است، به آن اشاره خواهیم کرد. در فصل‌های ۲۴، ۲۵ و ۲۶ درباره نحوه استفاده از کاتلین روی سایر پلتفرم‌ها بیشتر خواهید آموخت.

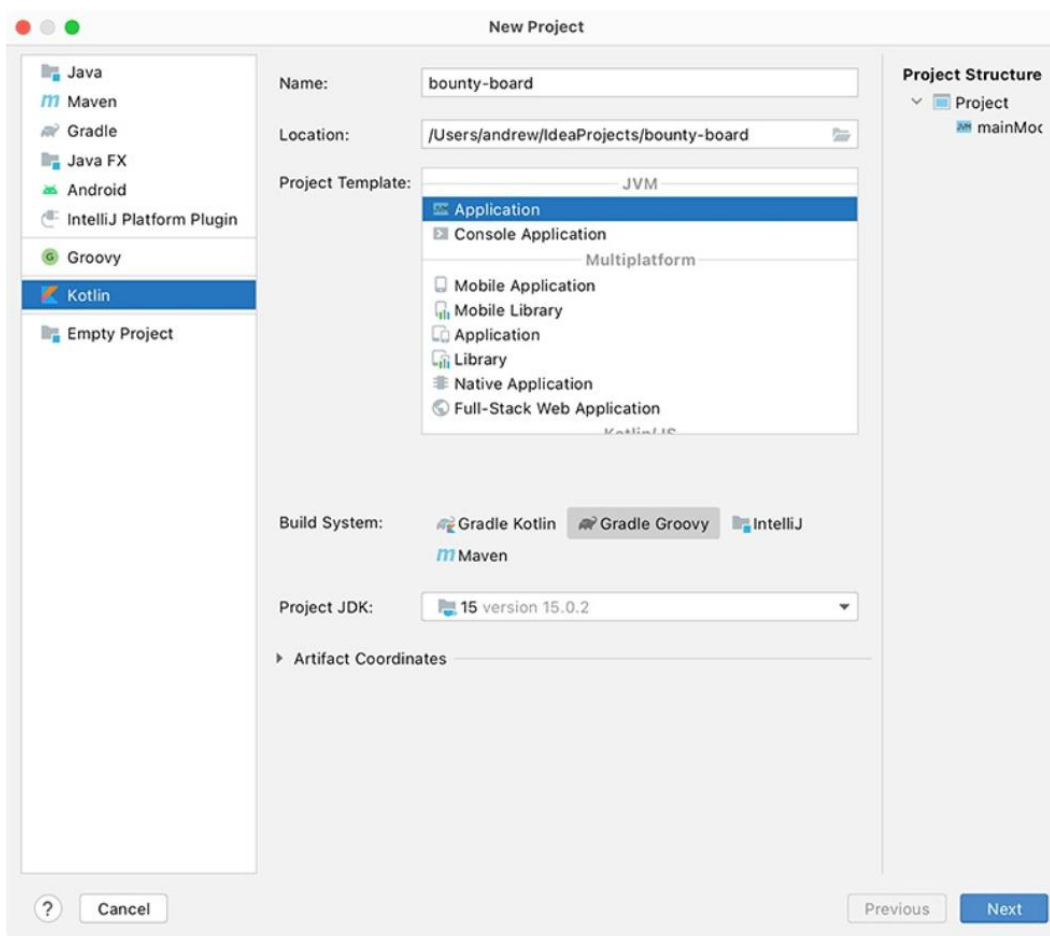
به راه‌اندازی پروژه جدیدتان بازگردیم. برای سیستم ساخت (Build System)، گزینه Gradle Groovy را انتخاب کنید. در نهایت، برای Project JDK، نسخه‌ای از جاوا را انتخاب نمایید تا پروژه شما به کیت توسعه جاوا (JDK) متصل شود. ما استفاده از نسخه‌ای بین جاوا ۸ (که اغلب با عنوان Java 1.8 لیست می‌شود) و جاوا ۱۵ را توصیه می‌کنیم.

اگر نسخه مناسبی از جاوا را در لیست کشویی مشاهده نمی‌کنید، IntelliJ نصبی را در کامپیوتر شما شناسایی نکرده است. اگر مطمئن هستید که یک JDK نصب کرده‌اید و می‌خواهید از آن نسخه استفاده کنید، می‌توانید از گزینه Add JDK ... استفاده کرده و محل نصب را در دیسک پیدا کنید. در غیر این صورت، IntelliJ می‌تواند از طریق گزینه Download JDK ... یکی را برای شما نصب کند. (توصیه می‌کنیم Version را روی ۱۵ و Vendor را روی AdoptOpenJDK (HotSpot) تنظیم کنید، اما هر ارائه‌دهنده‌ای (vendor) باید کار کند.) به محض اینکه IntelliJ نصب را به پایان رساند، آماده کار هستید.

چرا برای نوشتن یک برنامه کاتلین به JDK نیاز دارید؟ IntelliJ به JVM دسترسی به ابزارهای جاوا را می‌دهد که برای تبدیل کد کاتلین شما به بایت‌کد (bytecode) ضروری هستند (در این مورد کمی بعد بیشتر توضیح خواهیم داد). از نظر فنی، هر نسخه ۶ یا بالاتر کار خواهد کرد. اما تجربه ما، تا زمان نگارش این کتاب، نشان می‌دهد که JDK 8 یا نسخه‌های جدیدتر به روان‌ترین شکل ممکن کار می‌کنند.

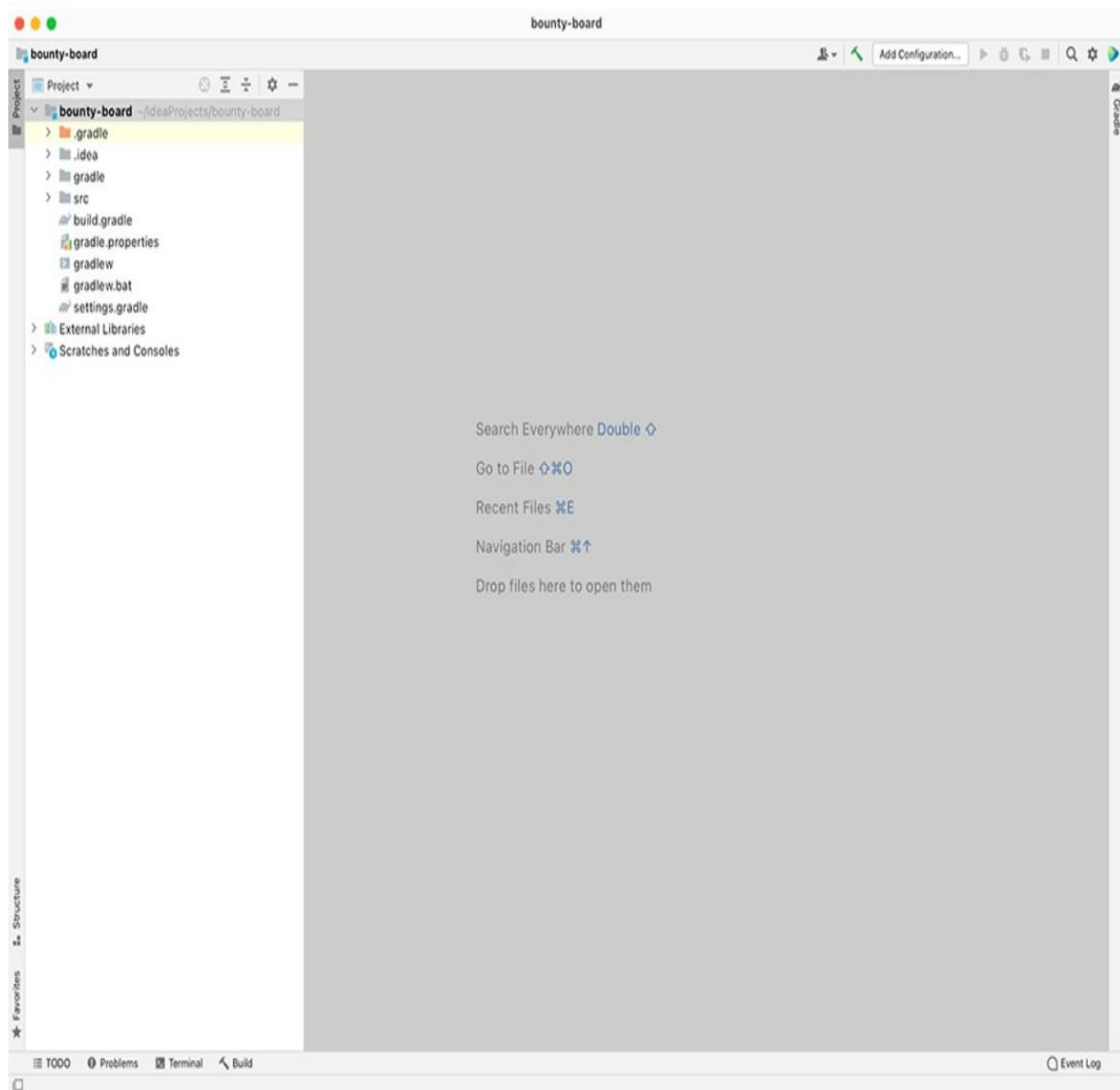
وقتی پنجره تنظیمات شما شبیه شکل ۱.۵ شد، روی Next و سپس روی Finish در پنجره بعدی کلیک کنید تا تنظیمات تأیید شوند.

Figure 1.5 Setting up a project



نرم افزار IntelliJ یک پروژه به نام bounty-board تولید کرده و پروژه جدید را در یک نمای پیش فرض با دو پنل (شکل 1.6) نمایش می دهد. روی دیسک، IntelliJ یک پوشه و مجموعه ای از زیرپوشه ها و فایل های پروژه را در مکانی که در فیلد Project location مشخص شده، ایجاد می کند.

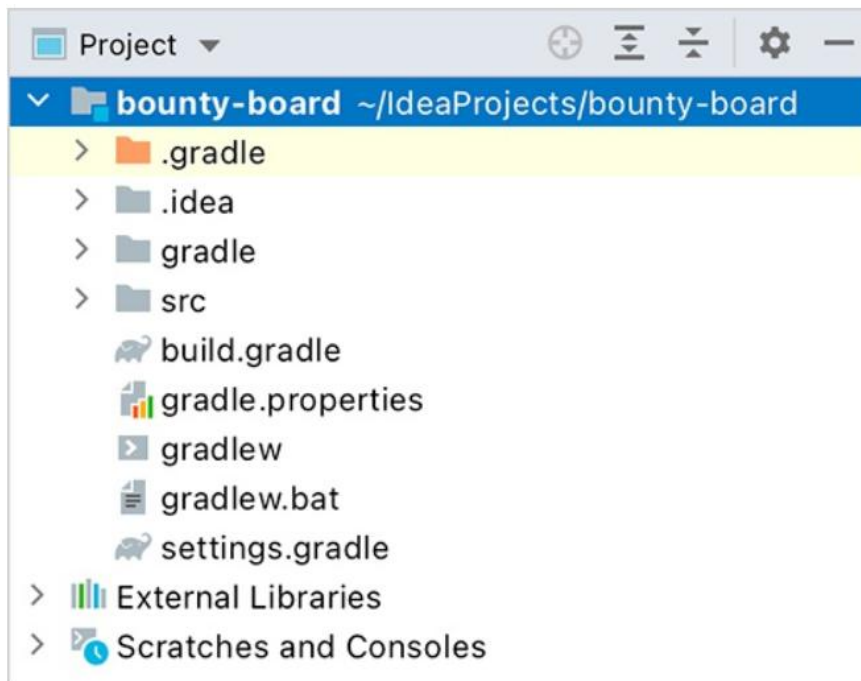
Figure 1.6 Default two-pane view



پنل سمت چپ پنجره ابزار پروژه (project tool window) را نشان می دهد. پنل سمت راست در حال حاضر خالی است. این قسمت جایی است که محتویات فایل های کاتلین خود را در ویرایشگر (editor) مشاهده و ویرایش خواهید کرد.

پنجره ابزار پروژه فایل های موجود در پروژه bounty-board را نمایش می دهد، همانطور که در شکل ۱.۷ مشاهده می کنید.

Figure 1.7 Project view



یک پروژه شامل تمام کد منبع (source code) برنامه شما، همراه با اطلاعات مربوط به وابستگی‌ها (dependencies) و پیکربندی‌هاست. یک پروژه می‌تواند به یک یا چند ماژول (module) که مانند پروژه‌های فرعی هستند، تقسیم شود. به طور پیش‌فرض، یک پروژه جدید دارای یک ماژول است که این تمام چیزی است که برای اولین پروژه ساده خود نیاز دارید.

پوشه‌های `gradle` و `idea` که در شکل ۱.۷ نشان داده شده‌اند، ممکن است هنگام باز کردن پروژه در یک مرورگر فایل (file explorer) پنهان باشند. آن‌ها پوشه‌های «پشت صحنه» هستند که نیازی به دسترسی به آن‌ها نخواهید داشت؛ `gradle` حاوی حافظه پنهان (caches) مورد استفاده توسط سیستم ساخت است، و `idea` شامل فایل‌های تنظیمات برای پروژه و IDE شما است.

پوشه `gradle` و فایل‌هایی مانند `gradlew` و `gradlew.bat` مربوط به Gradle Wrapper هستند؛ یک کپی خودکفا از سیستم ساخت Gradle که می‌توانید بدون نیاز به نصب Gradle روی کامپیوترتان، از آن استفاده کنید. فایل‌های `build.gradle`، `gradle.properties` و `settings.gradle`، فایل‌های پیکربندی سیستم ساخت Gradle هستند. آن‌ها بخش‌های مختلفی از اطلاعات مانند نام پروژه، سطح زبان (language level)، ماژول‌های پروژه و وابستگی‌های پروژه شما را تعریف می‌کنند. این فایل‌های تولید شده به طور خودکار را همان‌طور که هستند رها کنید.

بخش External Libraries شامل اطلاعات مربوط به کتابخانه‌هایی است که پروژه به آن‌ها وابسته است. اگر این بخش را باز کنید، خواهید دید که IntelliJ به طور خودکار جاوا و چندین بسته از کتابخانه استاندارد کاتلین را به عنوان وابستگی برای پروژه شما اضافه کرده است.

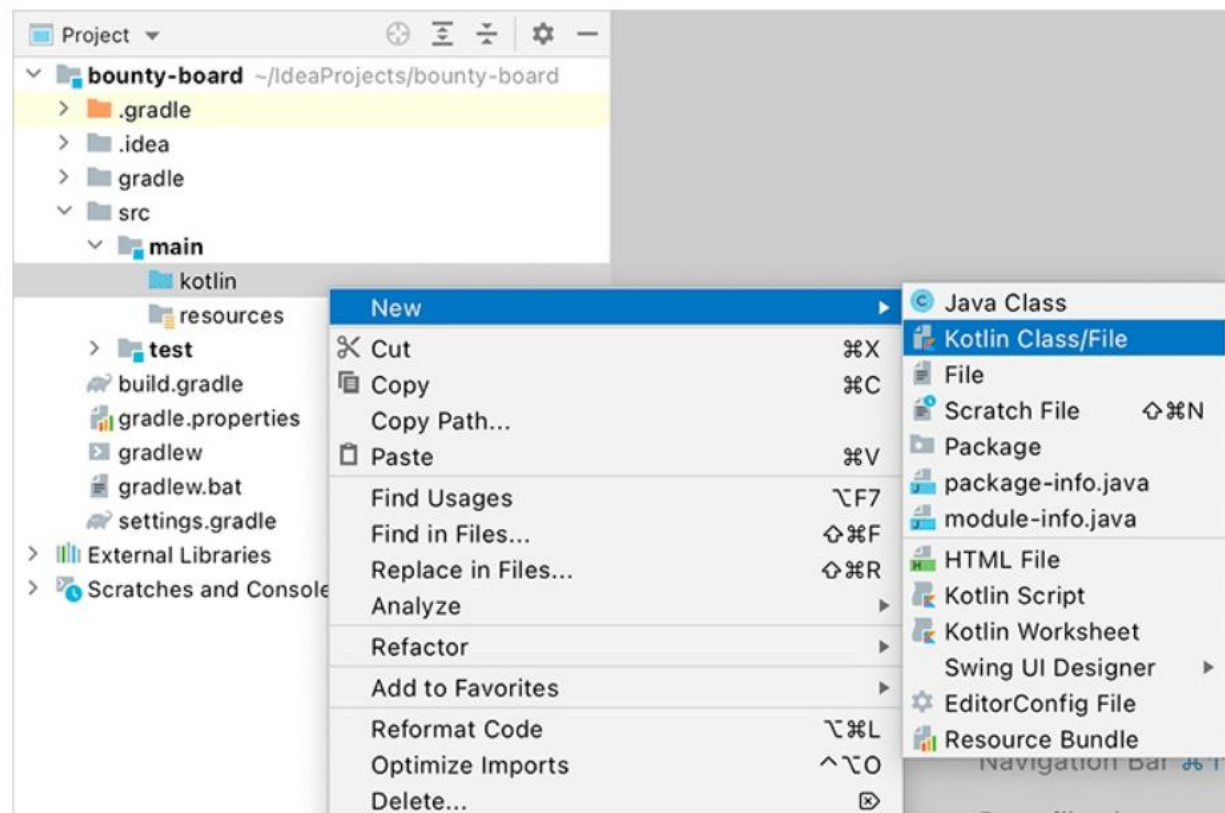
(شما می‌توانید در وبسایت مستندات JetBrains به آدرس jetbrains.org/intellij/sdk/docs/basics/project_structure.html درباره ساختار پروژه IntelliJ اطلاعات بیشتری کسب کنید. همچنین می‌توانید در وبسایت Gradle به آدرس docs.gradle.org/current/userguide/organizing_gradle_projects.html درباره ساختار یک پروژه Gradle بیشتر بیاموزید.)

پوشه src/main/kotlin مکانی است که شما تمام فایل‌های کاتلینی که برای پروژه‌تان ایجاد می‌کنید را در آن قرار خواهید داد. و با انجام این کار، زمان ایجاد و ویرایش اولین فایل کاتلین شما فرا رسیده است.

ایجاد اولین فایل کاتلین شما

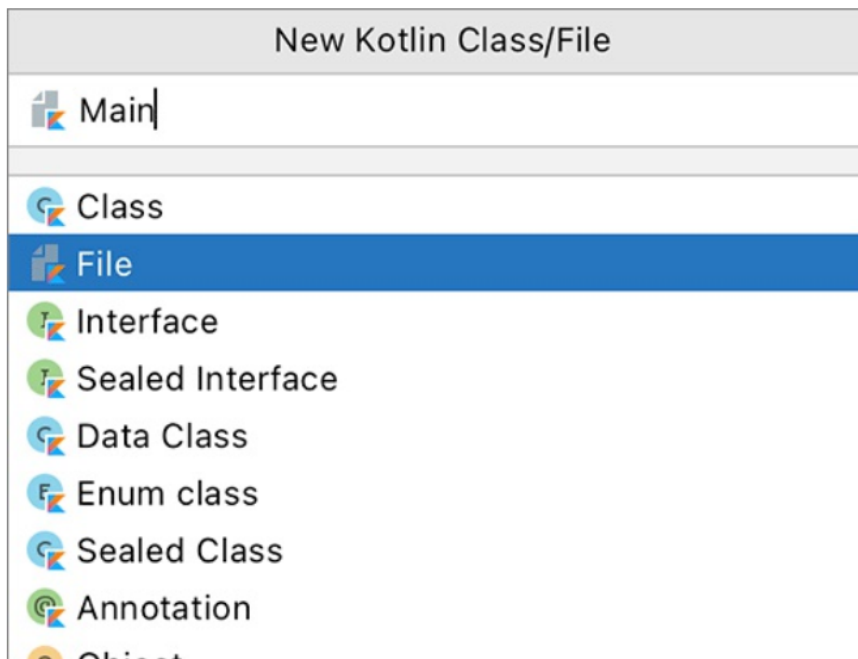
در پنجره ابزار پروژه پوشه src/main/kotlin را باز کرده و روی آن راست کلیک کنید. از منوی ظاهر شده، New و سپس Kotlin Class/File را انتخاب کنید (شکل ۱.۸).

Figure 1.8 Creating a new Kotlin file



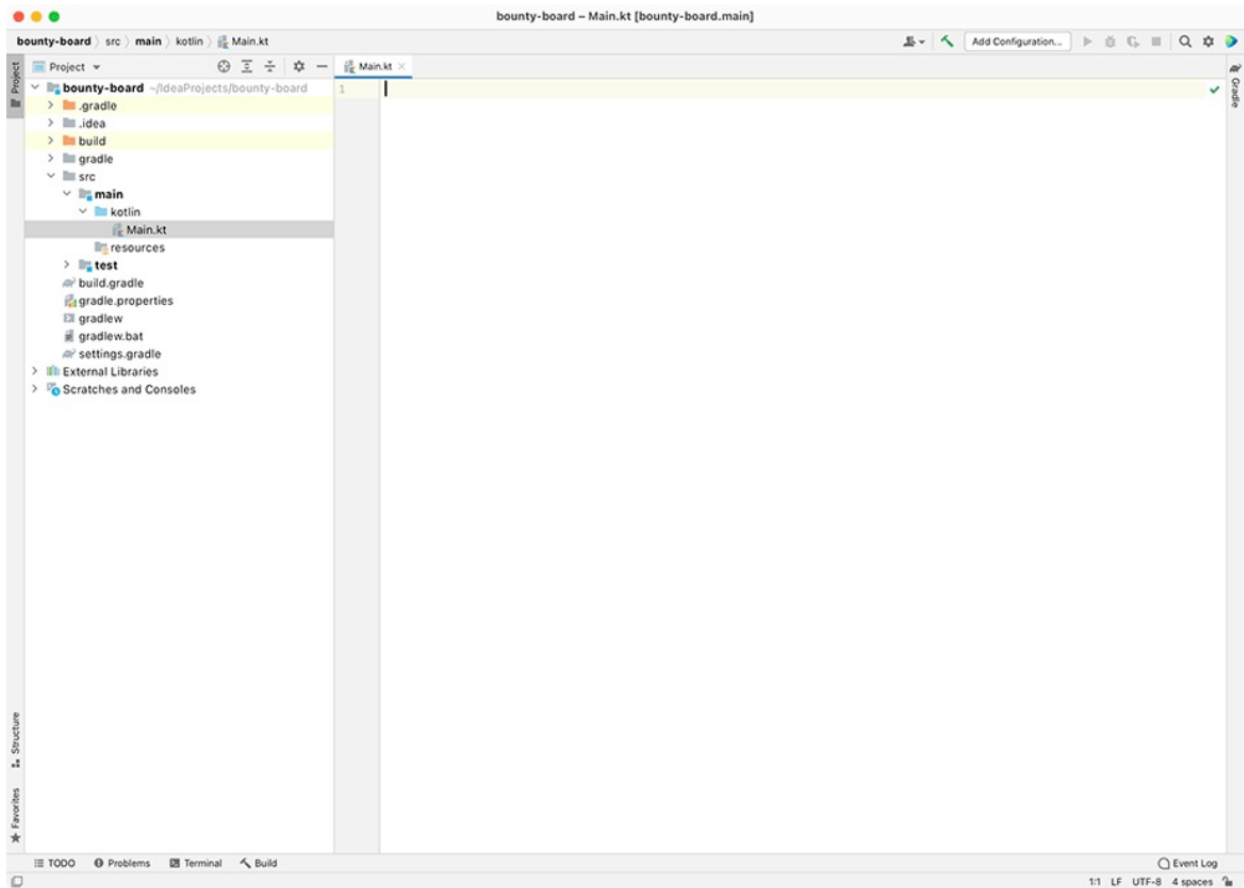
در پنجره New Kotlin Class/File، مقدار Main را در فیلد Name تایپ کنید و در لیست نوع فایل‌های زیرین، روی File دابل کلیک کنید (شکل ۱.۹).

Figure 1.9 Naming the file



نرم‌افزار IntelliJ فایل جدیدی را در پروژه شما ایجاد خواهد کرد، به آدرس `src/main/kotlin/Main.kt`، و محتویات فایل را در ویرایشگر سمت راست پنجره IntelliJ نمایش می‌دهد (شکل ۱.۱۰). پسوند `kt` نشان می‌دهد که فایل حاوی کدهای کاتالین است، دقیقاً همان‌طور که از پسوند `java` برای فایل‌های جاوا و `py` برای فایل‌های پایتون استفاده می‌شود.

Figure 1.10 Empty Main.kt file displays in editor



سرانجام، آماده نوشتن کد کاتلین هستید. کمی به انگلستان خود کشش دهید و دست به کار شوید. کد زیر را در ویرایشگر Main.kt تایپ کنید. (به یاد داشته باشید که در سراسر این کتاب، کدی که باید وارد کنید با فونت پررنگ نشان داده شده است.)

Listing 1.1 “Hello, world!” in Kotlin (Main.kt)

```
fun main() {  
    println("Hello, world!")  
}
```

کدی که به تازگی نوشتید ممکن است ناآشنا به نظر برسد. نترسید - تا پایان این کتاب، خواندن و نوشتن کاتلین برای شما مانند طبیعت ثانویه خواهد شد. در حال حاضر، کافایت کد را در یک سطح کلی درک کنید.

کد در Listing 1.1 یک تابع (function) جدید را تعریف می‌کند. یک تابع، گروهی از دستورالعمل‌هاست که می‌توانند بعداً اجرا شوند. شما در فصل ۴ به طور مفصل در مورد نحوه تعریف و کار با توابع خواهید آموخت.

این تابع خاص - تابع main - در کاتلین معنای ویژه‌ای دارد. تابع main نقطه شروع برنامه شما را مشخص می‌کند. این نقطه، نقطه ورود برنامه (application entry point) نامیده می‌شود و برای قابل اجرا بودن پروژه bounty-board (یا هر برنامه‌ای)، یک نقطه ورود از این دست باید تعریف شود. هر پروژه‌ای که در این کتاب می‌نویسید با یک تابع main شروع خواهد شد.

تابع `main` شما شامل یک دستورالعمل (که به عنوان `statement` نیز شناخته می‌شود) است: `println("Hello, world!")`. دستور `println()` نیز یک تابع است؛ تابعی که در کتابخانه استاندارد کاتلین تعبیه شده و در تمام پلتفرم‌های هدفی که کاتلین پشتیبانی می‌کند در دسترس است. هنگامی که برنامه اجرا شده و `println("Hello, world!")` پردازش می‌شود، IntelliJ محتویات داخل پرانتز را (بدون علامت‌های نقل قول، بنابراین در این مورد `Hello, world!`) روی صفحه چاپ خواهد کرد.

اجرای فایل کاتلین شما



مدت کوتاهی پس از اینکه تایپ کد موجود در Listing 1.1 را به پایان رساندید، IntelliJ یک فلش سبز رنگ، معروف به “دکمه اجرا” (`run button`) را در سمت چپ خط اول نمایش می‌دهد (شکل ۱.۱۱). (اگر نماد ظاهر نشد، یا اگر خط قرمزی را در زیر نام فایل در تب یا زیر هر یک از کدهای وارد شده مشاهده کردید، این بدان معناست که خطایی در کد خود دارید. دوباره بررسی کنید که کد را دقیقاً همانطور که در Listing 1.1 نشان داده شده است تایپ کرده باشید).

Figure 1.11 Run button



زمان آن رسیده است که برنامه شما جان بگیرد و به دنیا سلام کند. روی دکمه اجرا کلیک کنید. از منوی ظاهر شده `Run` `MainKt` را انتخاب کنید (شکل ۱.۱۲). این کار به IntelliJ می‌گوید که می‌خواهید برنامه خود را در حال اجرا ببینید.

Figure 1.12 Running Main.kt



هنگامی که برنامه خود را اجرا می‌کنید، IntelliJ کدهای داخل آکولادها ({}) را خط به خط اجرا کرده و سپس اجرا را خاتمه می‌دهد. همچنین یک پنجره ابزار جدید در پایین پنجره IntelliJ نمایش می‌دهد (شکل ۱.۱۳).

Figure 1.13 Run tool window (console)



این پنجره، پنجره ابزار اجرا (run tool window) است که با عنوان کنسول (console) نیز شناخته می‌شود (که از این پس ما نیز آن را همین‌گونه خواهیم نامید). این بخش اطلاعاتی در مورد آنچه با اجرای برنامه شما توسط IntelliJ اتفاق افتاده و همچنین هرگونه خروجی که برنامه شما چاپ می‌کند را نمایش می‌دهد. باید عبارت Hello, world! را مشاهده کنید که در کنسول شما چاپ شده است. شما همچنین باید عبارت Process finished with exit code 0 را ببینید که نشان دهنده تکمیل موفقیت آمیز اجراست. این خط در صورت عدم وجود خطا در انتهای تمام خروجی‌های کنسول ظاهر می‌شود؛ ما از این به بعد آن را در نتایج کنسول نشان نخواهیم داد.

کامپایل و اجرای کد Kotlin/JVM

در زمان کوتاه بین زمانی که گزینه 'MainKt' 'Run' دکمه اجرا را انتخاب می‌کنید تا زمانی که Hello, world! در کنسول چاپ شود، اتفاقات زیادی می‌افتد.

ابتدا، IntelliJ کد کاتلین را با استفاده از کامپایلر kotlin-jvm کامپایل می‌کند. کامپایلر برنامه‌ای است که کد منبع را به یک زبان سطح پایین‌تر ترجمه می‌کند تا یک برنامه قابل اجرا ایجاد کند. اگر در حال هدف‌گذاری پلتفرم متفاوتی بودید، IntelliJ حسب مورد از kotlin-js یا kotlin-native استفاده می‌کرد.

هر سه نسخه از کامپایلر `kotlin` تقریباً به یک روش کار می‌کنند. هنگامی که `kotlin-jvm` اجرا می‌شود، کد کاتلینی را که شما نوشته‌اید به بایت‌کد (bytecode) ترجمه می‌کند؛ زبانی که JVM با آن “صحبت” می‌کند. اگر `kotlin` برای ترجمه کد کاتلین شما با مشکلی روبرو شود، پیام (یا پیام‌های) خطایی را نمایش می‌دهد و به شما راهنمایی می‌کند که چگونه مشکلات را برطرف کنید. در غیر این صورت، اگر فرآیند کامپایل به آرامی پیش برود، IntelliJ به مرحله اجرا (execution phase) می‌رود.

در مرحله اجرا، بایت‌کدی که توسط `kotlin-jvm` تولید شده است روی ماشین مجازی جاوا (JVM) اجرا می‌شود. کنسول هرگونه خروجی از برنامه شما را نمایش می‌دهد، مانند چاپ متنی که در فراخوانی خود برای تابع `println()` مشخص کرده‌اید، همان‌طور که JVM دستورالعمل‌ها را اجرا می‌کند.

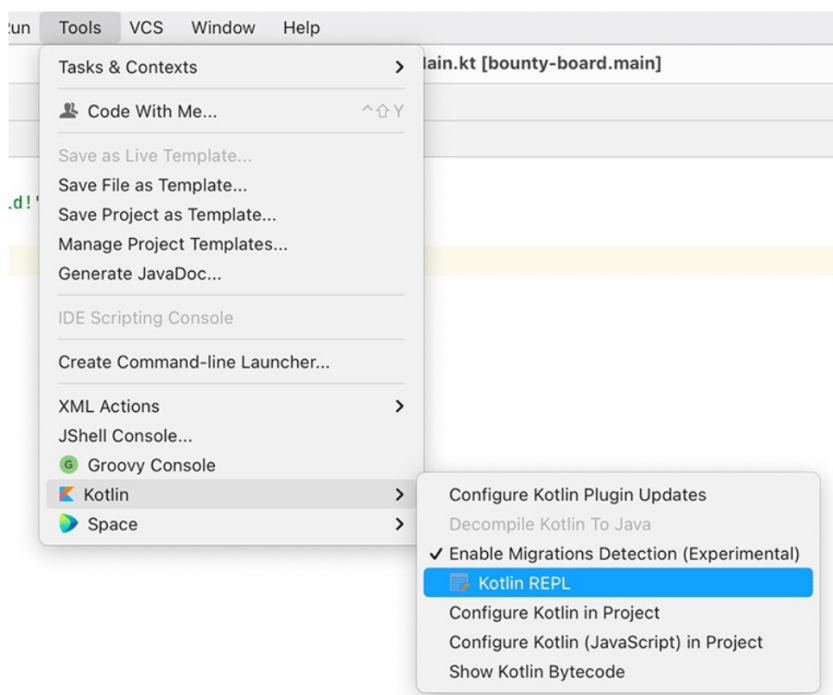
وقتی دیگر هیچ دستورالعمل بایت‌کدی برای اجرا وجود نداشته باشد، JVM متوقف می‌شود. IntelliJ وضعیت خاتمه را در کنسول نشان می‌دهد و به شما اطلاع می‌دهد که آیا اجرا با موفقیت به پایان رسیده یا با کد خطا خاتمه یافته است.

برای کار با این کتاب به درک جامعی از فرآیند کامپایل کاتلین نیاز نخواهید داشت. با این حال، ما در فصل ۲ درباره بایت‌کد با جزئیات بیشتری بحث خواهیم کرد.

قابلیت REPL در کاتلین

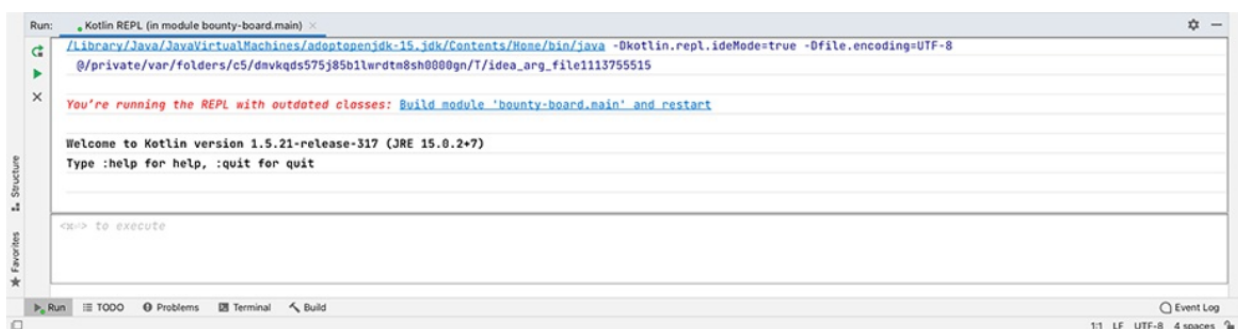
گاهی اوقات ممکن است بخواهید بخش کوچکی از کد کاتلین را تست کنید تا ببینید هنگام اجرای آن چه اتفاقی می‌افتد، مشابه اینکه چگونه از یک تکه کاغذ چک‌نویس برای یادداشت مراحل یک محاسبه کوچک استفاده می‌کنید. این امر به خصوص زمانی که در حال یادگیری زبان کاتلین هستید بسیار مفید است. خوشبختانه برای شما، IntelliJ ابزاری را برای تست سریع کد بدون نیاز به ایجاد فایل ارائه می‌دهد. این ابزار `Kotlin REPL` نامیده می‌شود. در ادامه در مورد این نام توضیح خواهیم داد – فعلاً آن را باز کنید و ببینید چه کاری می‌تواند انجام دهد.

در IntelliJ، با انتخاب `Tools` سپس `Kotlin` و بعد `Kotlin REPL` پنجره ابزار REPL را باز کنید (شکل ۱.۱۴).



نرم افزار IntelliJ پنجره REPL را در پایین صفحه نمایش می دهد (شکل ۱.۱۵).

Figure 1.15 The Kotlin REPL tool window



شما می توانید کدهای خود را مستقیماً در REPL تایپ کنید، درست مانند ویرایشگر. تفاوت در این است که می توانید آن را به سرعت ارزیابی کنید، بدون اینکه نیاز به کامپایل کل پروژه داشته باشید.

(ممکن است هنگام شروع کار با REPL متن هشدار قرمزی را مشاهده کنید یا هشدارهایی درباره "اجرای REPL با کلاس های قدیمی (outdated classes)". به طور کلی می توانید این گونه هشدارها را نادیده بگیرید. مشکلات عمومی مربوط به پلتفرم یا JVM هنگام استفاده از REPL هیچ آسیبی وارد نمی کند و از آنجایی که شما در REPL به کد پروژه bounty-board دسترسی نخواهید داشت، نیازی به نگرانی در مورد هشدار کلاس های قدیمی ندارید.)

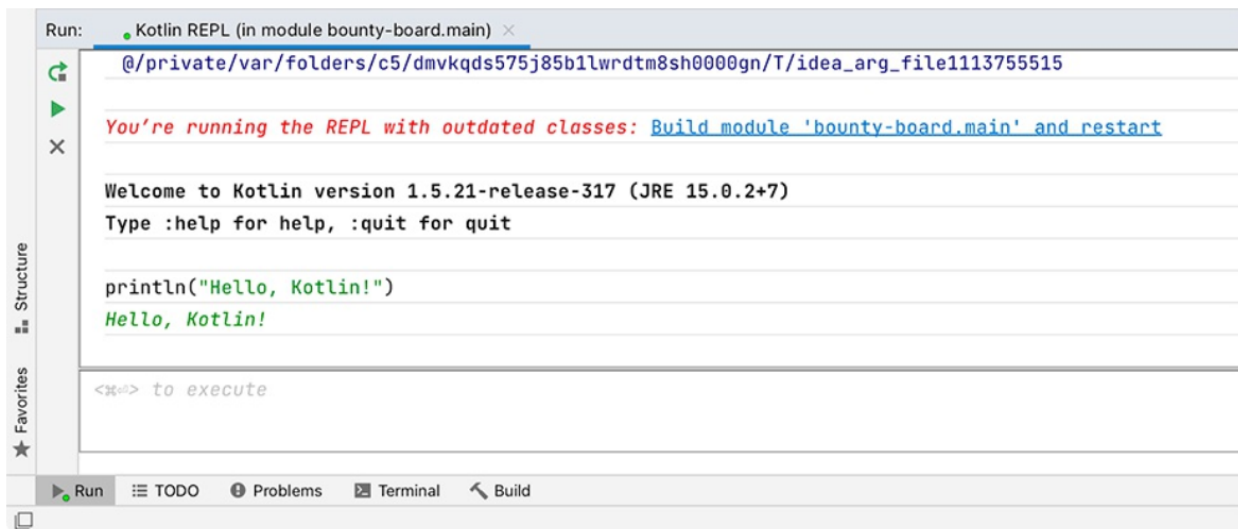
کد زیر را در REPL وارد کنید:

Listing 1.2 “Hello, Kotlin!” (REPL)

```
println("Hello, Kotlin!")
```

پس از وارد کردن متن، دکمه Command-Return (یا Ctrl-Enter) را فشار دهید تا کد در REPL ارزیابی شود. پس از یک لحظه، خروجی حاصل را در زیر آن مشاهده خواهید کرد که باید Hello, Kotlin! باشد (شکل ۱.۱۶).

Figure 1.16 Evaluating the code



کلمه REPL مخفف عبارت “read, evaluate, print, loop” (بخوان، ارزیابی کن، چاپ کن، حلقه تکرار) است. شما قطعه کدی را در خط فرمان تایپ می‌کنید و آن را با کلیک روی دکمه سبز رنگ اجرا در سمت چپ REPL یا با فشردن کلیدهای Command-Return (Ctrl-Enter) ارسال می‌نمایید. REPL سپس کد را می‌خواند (read)، آن را ارزیابی و اجرا می‌کند (evaluate)، و مقدار حاصل یا اثر جانبی آن را چاپ می‌کند (print). هنگامی که اجرای REPL به پایان می‌رسد، کنترل را به شما بازمی‌گرداند و حلقه فرآیند دوباره از ابتدا آغاز می‌شود (loop).

سفر کاتلینی شما آغاز شده است! شما در این فصل کار بزرگی انجام دادید و پایه‌های دانش در حال رشد خود از برنامه‌نویسی کاتلین را بنا نهادید. در فصل بعد، با یادگیری نحوه استفاده از متغیرها، ثابت‌ها و انواع داده‌ها برای نمایش اطلاعات، شروع به کندوکاو در جزئیات این زبان خواهید کرد.

برای افراد کنجکاوتر: چرا از IntelliJ استفاده کنیم؟

کد کاتلین را می‌توان با استفاده از هر ویرایشگر متن ساده‌ای (plain text editor) نوشت. با این حال، ما استفاده از IntelliJ را توصیه می‌کنیم، مخصوصاً زمانی که در حال یادگیری هستید. همان‌طور که نرم‌افزار ویرایش متنی که دارای بررسی املا و دستور

زبان (spell and grammar check) است، نوشتن یک مقاله نثری خوش ساخت را آسان تر می کند، IntelliJ نیز نوشتن کاتلین با ساختار صحیح را آسان تر می سازد. IntelliJ به شما در این موارد کمک می کند:

نوشتن کدهایی با صحت نحوی (syntactically) و معنایی (semantically) به کمک ویژگی هایی مانند برجسته سازی نحو (syntax highlighting)، پیشنهاد های حساس به متن (context-sensitive suggestions) و تکمیل خودکار کد (automatic code completion).

اجرا و اشکال زدایی (debug) کد با ویژگی هایی مانند نقاط توقف اشکال زدایی (debug breakpoints) و پیمایش کدهای به صورت بلادرنگ (real-time code stepping) هنگام اجرای برنامه تان.

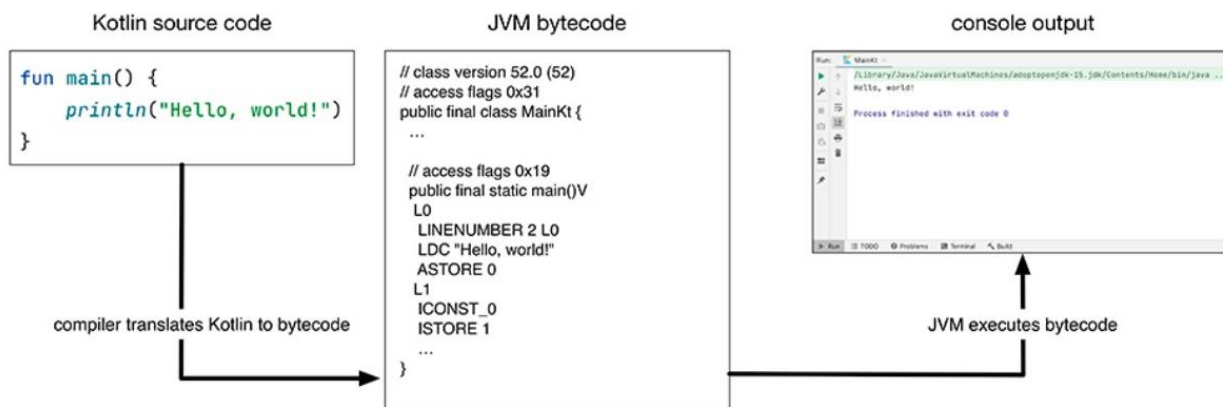
بازسازی و تغییر ساختار کدهای موجود با استفاده از میانبرهای بازآرایی کد (refactoring) (مانند تغییر نام و استخراج ثابت ها) و قالب بندی کد برای مرتب کردن دندانه گذاری (indentation) و فاصله ها.

همچنین از آنجایی که کاتلین توسط JetBrains ایجاد شده است، ادغام بین IntelliJ و کاتلین به دقت طراحی شده و اغلب منجر به تجربه ویرایش لذت بخشی می گردد. به عنوان یک مزیت اضافی، IntelliJ پایه و اساس نرم افزار Android Studio است، بنابراین در صورت تمایل شما، میانبرها و ابزارهایی که در اینجا یاد می گیرید قابل انتقال به محیط Android Studio خواهند بود.

برای افراد کنجکاوتر: هدف گیری JVM

ماشین مجازی جاوا (JVM) نرم افزاری است که می داند چگونه مجموعه ای از دستورالعمل ها را به نام بایت کد (bytecode) اجرا کند. "هدف گیری JVM" به معنای کامپایل کردن یا ترجمه کد منبع کاتلین شما به بایت کد جاوا، با هدف اجرای آن بایت کد روی ماشین مجازی جاوا است (شکل ۱.۱۷).

Figure 1.17 Compilation and execution flow



هر پلتفرمی مانند Windows یا macOS، مجموعه دستورالعمل های خاص خود را دارد. ماشین مجازی جاوا (JVM) به عنوان پلی بین بایت کد و محیط های سخت افزاری و نرم افزاری مختلفی که روی آن ها اجرا می شود، عمل می کند و با خواندن یک

قطعه بایت کد، دستورالعمل(های) مربوطه به آن پلتفرم خاص را که به آن بایت کد نگاشت می‌شود، فراخوانی می‌کند. بنابراین، نسخه‌های مختلفی از JVM برای پلتفرم‌های مختلف وجود دارد. این همان چیزی است که به توسعه‌دهندگان کاتلین اجازه می‌دهد تا کدهای مستقل از پلتفرمی بنویسند که تنها یک بار نوشته شده و سپس به بایت کد کامپایل شود و در دستگاه‌های مختلف صرف‌نظر از سیستم‌عامل‌هایشان اجرا گردد.

از آنجا که کاتلین می‌تواند به بایت‌کدی تبدیل شود که JVM قادر به اجرای آن است، کاتلین یک زبان مبتنی بر JVM محسوب می‌شود. جاوا احتمالاً شناخته‌شده‌ترین زبان مبتنی بر JVM است، زیرا اولین مورد بوده است. با این حال، دیگر زبان‌های JVM مانند Scala، Groovy و Kotlin به منظور رفع برخی از کاستی‌های جاوا از منظر توسعه‌دهندگان، پدیدار شده‌اند.

چالش: محاسبات ریاضی در REPL

بسیاری از فصل‌های این کتاب با یک یا چند چالش به پایان می‌رسند. این چالش‌ها برای این هستند که خودتان روی آن‌ها کار کنید تا درک خود را از کاتلین عمیق‌تر کرده و کمی تجربه بیشتر کسب کنید.

از REPL برای بررسی نحوه عملکرد عملگرهای ریاضی در کاتلین استفاده کنید: `+, -, *, /` و `%`. به عنوان مثال، $2 * (9 + 12)$ را در REPL تایپ کنید. آیا خروجی با چیزی که انتظار داشتید مطابقت دارد؟

اگر می‌خواهید عمیق‌تر شوید، توابع ریاضی موجود در کتابخانه استاندارد کاتلین را در آدرس kotlinlang.org/api/latest/jvm/stdlib/kotlin.math بررسی کنید و آن‌ها را در REPL امتحان کنید. به عنوان مثال، `min(۹۴, ۹۹)` را امتحان کنید که حداقل مقدار از دو عدد ارائه شده در پرانتز را به شما می‌گوید.

شما باید با وبسایت kotlinlang.org و به خصوص مستندات زبان موجود در آن احساس راحتی کنید. اگر قصد دارید از کاتلین در خارج از محیط JVM استفاده کنید، هنگام خواندن مراجع API به دایره‌های رنگی توجه کنید. این موارد نشان می‌دهند که یک API داده شده از چه پلتفرم‌هایی پشتیبانی می‌کند. بسیاری از API‌ها مانند `absoluteValue` (نشان داده شده در شکل ۱.۱۸) به عنوان API‌های مشترک (common) در نظر گرفته می‌شوند و روی تمامی پلتفرم‌های هدفی که کاتلین پشتیبانی می‌کند، کار می‌کنند.

CommonJVMJSNative

Version1.5

[kotlin-stdlib](#) / [kotlin.math](#)

Package kotlin.math

Mathematical functions and constants.

The functions include trigonometric, hyperbolic, exponentiation and power, logarithmic, rounding, sign and absolute value.

Properties

absoluteValue

CommonJVMJSNative1.2

Returns the absolute value of this value.

```
val Double.absoluteValue: Double
val Float.absoluteValue: Float
val Int.absoluteValue: Int
val Long.absoluteValue: Long
```

E

1.2

Base of the natural logarithms, approximately 2.71828.

```
const val E: Double
```

Figure 1.18 Platform support in the Kotlin API documentation

