مروری بر زبان
برنامه‌نویسی کاتلین

محمد فلاحی

مقدمه

در دنیای امروز سرعت پیشرفت تکنولوژی افزایش یافته و این افزونی پیشرفت در حوزه کامپیوتر بیشتر از سایر حوزه‌ها مشهود است؛ به طوری که ایجاد هوش‌مصنوعی، توسعه نرم‌افزارهای انعطاف‌پذیر و پر سرعت بیش از گذشته احساس می‌شود.

اما برای توسعه این موارد همواره به دنبال روش‌ها و زبان‌های برنامه‌نویسی بهنیه‌تر و بهتر هستیم تا به روند توسعه شکلی ساده‌تر و سریعتر ببخشیم. در این شرایط زبان کاتلین با شعار ساده، پرقدرت و سرگرم‌کننده وارد میدان بازی می‌شود تا روند توسعه را با چاشنی سادگی و سرعت به توسعه‌دهنده هدیه دهد.

تعامل آسان زبان کاتلین با جاوا سبب توجه بسیاری از توسعه‌دهندگان و استارتاپ‌ها به این زبان شده است. کاتلین با قابلیت‌های فراوانی که ارائه می‌دهد توانسته در مدت کوتاهی پس از تولد خود، تبدیل به زبان اول توسعه اپلیکیشن‌های موبایل به خصوص Android شود.

با ما همراه باشید تا در این کتابچه زبان برنامه‌نویسی کاتلین را زیر ذره‌بین قرار داده و مروری بر تاریخچه، ویژگی‌ها و کاربردهای این زبان شیرین داشته باشیم.

معرفی و تاریخچه

کاتلین یک زبان برنامه‌نویسی سطح بالا، همه‌منظوره (GPL) است که بر روی ماشین مجازی جاوا (JVM) اجرا می‌شود. همچنین علاوه بر JVM می‌تواند به زبان جاوا اسکریپت یا کد ماشین کامپایل شود. کاتلین نخستین بار در ۲۲ جولای سال 2011 توسط شرکت جت برینز تحت لایسنس Apache 2.0 منتشر شد. در نتیجه می‌توان گفت این زبان یک زبان کاملا اپن سورس است و سایر توسعه‌دهندگان نیز می‌توانند در توسعه و ارتقا این زبان نقش ایفا نمایند.

این زبان در ابتدا با هدف برتری و رفع نواقص Java منتشر شد. بنابراین این زبان برنامه نویسی به دلیل ویژگی‌های برجسته‌ای مانند خوانایی بالا، انعطاف‌پذیری و امنیت به سرعت توانست محبوب شود و در برخی حوزه‌ها از جاوا پیشی بگیرد. هرچند کاتلین می‌تواند با جاوا در تعامل باشد و همزمان امکان استفاده از جاوا و کاتلین میسر باشد.

نام این زبان از جزیره‌ای روسی در خلیج فنلاند در نزدیکی سن‌پترزبورگ گرفته شده است. توسعه‌دهنده اصلی و سابق زبان کاتلین تصمیم گرفت با الهام از توسعه‌دهندگان جاوا، نام زبان را از اسم یک جزیره انتخاب نماید. خود این اتفاق بیانگر هدف توسعه این زبان قدرتمند است!

آیا می‌دانید...؟

اسم زبان Java از اسم یک نوع قهوه با این نام متعلق به جزیره جاوه در اندونزی الهام گرفته شده است؟ هرچند در بسیاری از منابع به اشتباه گفته می‌شود نام این زبان از نام خود جزیره گرفته شده است.

ویژگی‌های زبانی

همانطور که قبل‌تر گفته شد زبان برنامه‌نویسی کاتلین با ویژگی‌های جذاب خود سبب شده است بین توسعه‌دهندگان محبوب شود.

کاتلین یک زبان برنامه نویسی سطح بالا (High Level)، سطح انتزاع بالا (High Abstraction) می‌باشد. این یعنی نزدیک به زبان انسانی بوده و نیاز به اضافه‌کاری هایی که در سایر زبان‌ها همچون C انجام می‌شود ندارد. در نتیجه از پیچیدگی‌ها جلوگیری کرده و امکان نوشتن کد با سرعت بالا را فراهم می‌کند.

کاتلین یک زبان کامپایلری است به بایت‌کد جاوا، کد ماشین و یا زبان جاوا اسکریپت کامپایل می‌شود. با توجه به اینکه روی JVM نیز قابلیت اجرا شدن دارد می‌تواند شعار اصلی زبان جاوا یعنی "یک بار بنویس و همه جا اجرا کن" را محقق سازد.

کامپایلری بودن این زبان سبب می‌شود نسبت به زبان‌های مفسری از سرعت بسیار بالاتری برخوردار باشد؛ بنابراین کاتلین به یک زبان مناسب برای اپلیکیشن‌ها و سکوهایی شده است که سرعت بالا از فاکتورهای مهم توسعه می‌باشد.

کاتلین را می‌توان ترکیبی از زبان‌های جاوا، اسکالا (Scala) و سوئیفت (Swift) دانست که سعی شده از ویژگی‌های مثبت این زبان‌ها الهام گرفته شده و نواقص آنان را برطرف سازد. کاتلین سازگاری کامل با جاوا و زیرساخت‌های آن دارد با این تفاوت که نسبت به جاوا بسیار خواناتر و ساده است.

از ویژگی‌های زبانی‌ای که کاتلین را نسبت به جاوا متمایز می‌سازد، پشتیبانی همزمان از برنامه‌نویسی شی‌گرا (Object-oriented programming) و تابعی (Functional programming) است. به طور کلی ویژگی‌های متمایزکننده کاتلین عبارت‌اند از:

ایمنی در برابر خطاهای Null **یا** Null Safety

این ویژگی که با نام Void Safety نیز شناخته می‌شود، یک ویژگی کاربردی مدرن است که به ما کمک می کند از موارد خاص (Exception) تهی (Null) در اپلیکیشن‌ها جلوگیری کنیم. برای درک این ویژگی بهتر است با مفهوم Null آشنا شویم.

هنگامی که ما متغیری را تعریف می‌نماییم این متغیر به همراه مقدار خود در حافظه کامپیوتر ذخیره می‌شود. اما فرض کنید ما متغیری را تعریف می‌کنیم که هیچ مقداری نداشته باشد. در برنامه‌های بزرگ با توجه به اینکه کد منبع (Source Code) نیز بزرگ‌تر خواهد بود، ممکن است باعث خطاهایی از نوع Null شود. اما در کاتلین برخلاف زبان جاوا به راحتی می‌توان تعیین کرد چه متغیری Null پذیر باشد و یا نباشد. در این صورت می‌توانیم از Null Pointer Exception جلوگیری کرده و کدنویسی ایمنی را تجربه کنیم.

توابع گسترش‌پذیر

ویژگی Extension Function در کاتلین قابلیتی است که امکان ایجاد توابع جدید در کلاس‌های موجود یا از پیش تعریف شده، بدون تغییر در کلاس اصلی را امکان‌پذیر می‌کند. این ویژگی باعث می‌شود بدون نیاز به کدنویسی پیچیده مانند ارث‌بری بتوان عملکردهای جدیدی را به کلاس‌های اپلیکیشن یا حتی کتابخانه‌ها اضافه کرد.

تعریف داده‌های مختصر

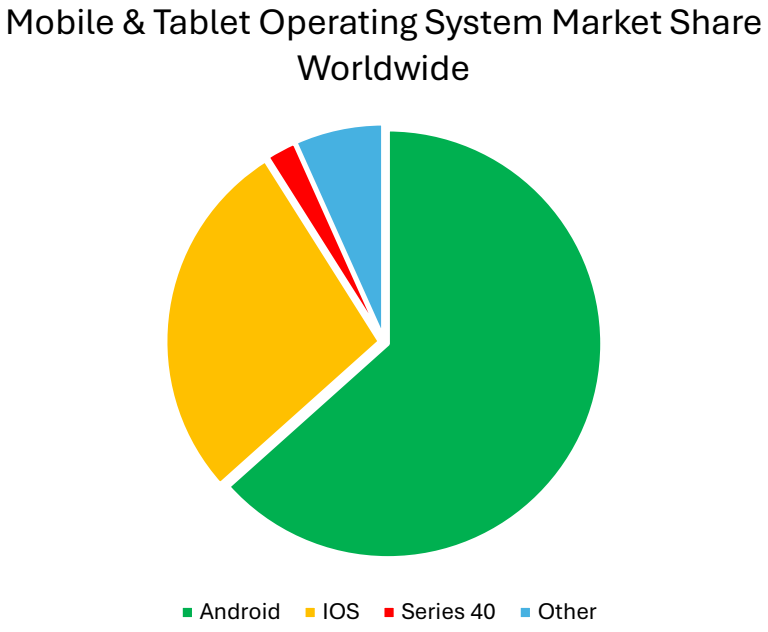
کلاس داده یا Data Classes نوع خاصی از کلاس‌ها هستند که برای ذخیره و مدیریت داده‌ها طراحی شده‌اند. این کلاس‌ها به شما اجازه می‌دهند تا کلاس‌هایی با حداقل کدنویسی ایجاد کنید که به طور خودکار قابلیت‌های مفیدی مانند مقایسه، کپی، و نمایش را فراهم کنند. هدف اصلی Data Class ساده‌سازی مدیریت داده‌ها و جلوگیری از نوشتن کدهای تکراری است.

کاربردها و زمینه‌های استفاده از کاتلین

با توجه به هدف از توسعه این زبان، به راحتی می‌توان کاربردها و زمینه‌های استفاده از کاتلین را حدس زد. کاتلین با سینتکس ساده خود و امکانات قدرتمندی که دارد به راحتی می‌تواند در حوزه‌های مختلف مورد استفاده قرار گیرد.

برنامه‌نویسی اندروید

موبایل‌ها از محبوب‌ترین لوازم دیجیتالی در حال حاضر هستند که به تقریب 6.84 میلیارد از جمعیت جهان آن را در اختیار دارد. در این میان سیستم‌عامل اندروید توانسته است تاکنون در میان اسمارت‌فون‌ها رهبری نماید و تبدیل به یکی از اصلی‌ترین سیستم‌عامل ها برای موبایل شود.



با گستردگی استفاده از موبایل‌های اندرویدی، استارتاپ‌ها و شرکت‌های خدماتی مانند بانک‌ها، فروشگاه‌ها و ... برای ارائه خدمات به کاربران و مشتریان خود، توجه خاصی به

توسعه اپلیکیشن‌های اندرویدی دارند. به طوری که حدودا ۳ میلیون اپلیکیشن اندرویدی فقط در گوگل‌پلی ارائه می‌شود!

در تاریخ 7 می ماه سال 2019 گوگل، زبان کاتلین را به زبان اصلی توسعه اپلیکیشن‌های اندرویدی بدل ساخت و به توسعه‌دهندگان موبایل پیشنهاد کرد با توجه به سادگی و امکانات زیاد کاتلین بهتر است اپلیکیشن های خود را با کاتلین توسعه داده و در اپ استورها منتشر نمایند.

طبق داکیومنت اصلی کاتلین، بیش از 50 درصد از توسعه‌دهندگان اپلیکیشن اندرویدی زبان کاتلین را به عنوان زبان اصلی استفاده می‌کنند این در حالی است که صرفا 30 درصد از توسعه‌دهندگان از Java به عنوان زبان اصلی توسعه اپلیکیشن‌های اندرویدی استفاده می‌کنند. به تقریب 70 درصد از این توسعه‌دهندگان که کاتلین را انتخاب نموده‌اند عقیده دارند که کاتلین سبب افزایش بازدهی فرایند توسعه و خلاقیتشان شده است.

دلایل انتخاب

توسط توسعه‌دهندگان اندروید

تعامل با جاوا

همراه زبان برنامه نویسی جاوا در برنامه های خود بدون نیاز به انتقال همه کدهای خود به کاتلین استفاده کنید

خطاهای کمتر

اپلیکیشن‌های توسعه یافته با کاتلین، بیست درصد کمتر احتمال دارد که از کار بیفتند

کدنویسی کمتر

زمان کمتری را صرف نوشتن و درک کد دیگران خواهید کرد

زبان و محیط بالغ

از زمان ایجاد آن در سال 2011، کاتلین به طور مداوم نه تنها به عنوان یک زبان بلکه به عنوان یک اکوسیستم کامل با ابزار قوی توسعه یافته است. اکنون به طور یکپارچه در Android Studio ادغام شده است و به طور فعال توسط بسیاری از شرکت ها برای توسعه برنامه‌های اندروید استفاده می شود.

پشتیبانی از Kotlin در کتابخانه های Jetpack

جت‌پک کامپوز (Jetpack Compose) ابزار مدرن توصیه شده اندروید برای ایجاد رابط کاربری نیتیو درKotlin است. برنامه‌های افزودنی KTX ویژگی‌های زبان Kotlin مانند coroutines را به کتابخانه‌های موجود اندروید اضافه می‌کنند.

برنامه‌نویسی سمت سرور (Server Side)

کاتلین برای توسعه برنامه های سمت سرور مناسب است. این به شما امکان می دهد کدهایتان را مختصر و رسا بنویسید و در عین حال سازگاری با استک‌ها و فریم‌ورک‌های مبتنی بر جاوا را نیز حفظ کنید. همه این ویژگی‌های گفته شده با یادگیری کاملا آسان و چون آب زلال امکان پذیر است؛ زیرا کاتلین رسا و آسان، دارای قابلیت مقیاس‌پذیری و مهاجرت گسترده می‌باشد.

رسا و آسان بودن: ویژگی‌های نوآورانه کاتلین، مانند پشتیبانی آن از Type Safety به توسعه ساختارهای انتزاعی قدرتمند برای استفاده کمک می‌کند.

مقیاس پذیری: پشتیبانی کاتلین برای اپلیکیشن‌های کاربردی به ساخت اپلیکیشن‌های سمت سرور کمک می کند تا به مقیاس‌پذیری اپلیکیشن‌های دارای کلاینت‌ها و کاربران زیاد در عین سخت‌افزار متوسط یاری رساند.

مهاجرت: کاتلین از مهاجرت تدریجی سورس کدهای جاوا به کاتلین پشتیبانی می کند. کاتلین کمک می‌کند حتی هنگامی که قصد تغییر کدهای نوشته شده به جاوا را ندارید بتوانید کدهای جدید را با کاتلین بنویسید.

فریم‌ورک‌های مختلفی برای توسعه اپلیکیشن‌های سرور ساید با کاتلین وجود دارد که مهمترین آنها شامل Spring، Ktor، Vertx و Javalin می‌باشد.

کاتلین نیتیو: بدون ماشین مجازی (Kotlin/Native)

کاتلین علاوه بر اجرا در ماشین‌های مجازی مانند JVM می‌تواند به صورت نیتیو نیز در پلتفرم‌های مختلف اجرا شود. البته این نوع پلتفرم‌ها اغلب پلتفرم‌هایی هستند که استفاده از VM ها امکان پذیر نیست؛ مانند امبدد سرویس‌ها و IOS.

کاتلین نیتیو سورس کد کاتلین را مستقیما به باینری‌های نیتیو پلتفرم مورد نظر کامپایل می‌کند. این کامپایل از طریق یک لایه زیرین که بر پایه LLVM می‌باشد صورت می‌گیرد.

قابلیت تعامل‌پذیری یا همان Interoperability جذاب‌ترین و کلیدی‌ترین ویژگی Kotlin/Native است که اجازه می‌دهد از کتابخانه‌ها و فریم‌ورک‌های سایر زبان‌های نیتیو مانند C و C++ یا Swift و Objective-C استفاده کرد.

برنامه‌نویسی کراس پلتفرم و مولتی‌پلتفرم (Cross & Multiplatform)

پشتیبانی از برنامه نویسی چند پلتفرمی یکی از مزایای کلیدی و ویژه زبان کاتلین است. این ویژگی زمان صرف شده برای نوشتن و حفظ یک کد برای پلتفرم های مختلف را کاهش می‌دهد و در عین حال انعطاف پذیری و مزایای برنامه نویسی Native را نیز حفظ می‌کند.

از مهمترین ابزارهای مهمی که کاتلین برای برنامه‌نویسی Multiplatform به توسعه‌دهندگان کمک می‌کند، کامپوز مولتی‌پلتفرم (Compose Multiplatform) نام دارد این ابزار در طراحی و توسعه رابط کاربری تعاملی اپلیکیشن کاربرد دارد. طراحی این رابط کاربری به دو صورت Native و Shared انجام می‌شود.

ایستگاه توضیح المسائل

رابط کاربری Native در KMP به گونه‌ای است که کدهای نوشته شده برای رابط کاربری با توجه به پلتفرمی که در آن اجرا می‌شود نمایش داده می‌شود. به عبارتی دیگر اپلیکیشن ویوهایی را نمایش می‌دهد که از قبل برای اپلیکیشن در قالب SDK کدنویسی شده است. در رابط کاربری Shared برخلاف Native همانطور که نوشته شده است در همه پلتفرم‌ها با همان طراحی قابل نمایش است و اجرا می‌شود.

به طور کلی KMP دارای ویژگی‌های توسعه سریع رابط کاربری، طراحی کامپوننت محور و قابلیت استفاده از کامپوننت‌های نیتیو می‌باشد.

توسعه سریع رابط کاربری: دیگر نیاز نیست زمان زیادی را صرف همگام نگه داشتن رابط کاربری و اپلیکیشن برای پلتفرم‌های مختلف نمایید.

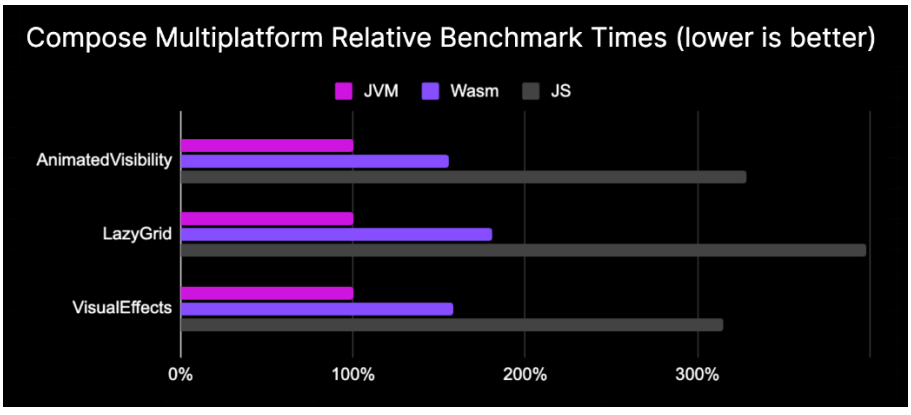
طراحی کامپوننت محور: می‌توانید برای UI اپلیکیشن کامپوننت‌های مختلفی را طراحی کنید و بدون نیاز به کدنویسی مجدد در پلتفرم‌های مورد نظر نمایش دهید.

استفاده از کامپوننت‌های Native**:** به آسانی می‌توان در هر زمان که خواستید از کامپوننت‌های Native نیز در کنار کامپوننت‌های سفارشی خود استفاده نمایید.

کامپایل به وب‌اسمبلی

وب‌اسمبلی (Web Assembly) یا Wasm یک فرمت باینری و مستقل از پلتفرم است که برای اجرای کدهای کامپایل شده در محیط‌های مختلف به خصوص مرورگرهای وب توسعه یافته است. وب‌اسمبلی می‌تواند کدها را نزدیک به زبان‌های Native مانند C و C++ اجرا کند. وب‌اسمبلی برای وب اپلیکیشن‌هایی که نیاز به سرعت بسیار بالا و ساختاری پیچیده دارند مناسب است.

در این میان کاتلین قابلیت تبدیل یا کامپایل شدن به Wasm را دارد که از این قابلیت با نام Kotlin/Wasm یاد می‌شود. می‌توان از Kotlin/Wasm در محیط‌ها یا پلتفرم‌های مختلفی مانند مرورگرها برای توسعه وب اپلیکیشن‌ها با استفاده از Compose Multiplatform استفاده کرد یا در خارج از مرورگر در ماشین‌های مجازی مستقل Wasm از آن استفاده کرد. در حالت خارج از مرورگر، رابط سیستم WebAssembly (WASI) دسترسی به APIهای پلتفرم را فراهم می‌کند.



مقایسه بنچمارک زمان JVM, Wasm و JS نشان می‌دهد که عملکرد Wasm نسبت به JS بسیار نزدیکتر به JVM است.1

کاتلین/وب‌اسمبلی از WASI استفاده می‌کند تا جزئیات خاص هر پلتفرم را حذف کند و امکان اجرای یک کد کاتلین یکسان را در پلتفرم‌های مختلف فراهم سازد. این ویژگی، استفاده از Kotlin/Wasm را بدون نیاز به تنظیمات خاص برای هر محیط اجرایی، فراتر از وب اپلیکیشن‌ها گسترش می‌دهد.

کتابخانه استاندارد Kotlin/Wasm شامل تعاریفی برای APIهای مرورگر، از جمله DOM API است. با استفاده از این تعاریف، می‌توان به‌طور مستقیم از API کاتلین برای دسترسی و استفاده از قابلیت‌های مختلف مرورگر استفاده کرد. به‌عنوان مثال، در برنامه‌های Kotlin/Wasm خود می‌توان عملیات‌هایی مانند تغییر عناصر DOM یا استفاده از Fetch API را بدون نیاز به تعریف این قابلیت‌ها از ابتدا انجام داد.

ایستگاه توضیح المسائل

مدل شیء سند (Document Object Model یا DOM) یک رابط مستقل از پلتفرم و زبان است که یک سند HTML یا XML را به‌صورت یک ساختار درختی نمایش می‌دهد. در این ساختار، هر گره (Node) یک شیء است که نمایانگر بخشی از سند می‌باشد. DOM سند را به‌صورت یک درخت منطقی نشان می‌دهد، به‌طوری که هر شاخه‌ی این درخت به یک گره ختم می‌شود و هر گره شامل اشیایی است.

کاتلین برای تجزیه و تحلیل داده

شاید با شنیدن این عنوان تعجب کنید؛ اما از مهمترین مهارت‌های یک توسعه‌دهنده نرم افزار تجزیه و تحلیل داده است. تحلیل داده‌ها در توسعه نرم‌افزار نقش کلیدی ایفا می‌کند؛ برای مثال تحلیل محتوای Collectionها در زمان دیباگ کردن، بررسی داده‌های موجود در پایگاه‌داده یا حافظه و کار با فایل‌های حجیم Json که در هنگام استفاده از Rest APIها وجود دارند.

ابزارهای تحلیل داده اکتشافی (EDA) در کاتلین، مانند Kotlin Notebooks، Kotlin DataFrame، و Kandy، مجموعه‌ای قدرتمند از قابلیت‌ها را در اختیار توسعه‌دهندگان قرار می‌دهند تا توسعه‌دهنده مهارت‌های تحلیلی خود را ارتقا دهد و در سناریوهای مختلف از آن‌ها بهره برد که عبارت‌اند از:

بارگذاری، تبدیل و تجسم داده‌ها در فرمت‌های مختلف: با استفاده از ابزارهای تحلیل داده اکتشافی (EDA) کاتلین، می‌توان داده‌ها را بارگذاری، تبدیل و تجسم کرد. با این ابزارها می‌توان اقداماتی مانند فیلتر کردن، مرتب‌سازی و تجمیع داده‌ها را انجام داد. همچنین این ابزارها به‌طور یکپارچه قادر به خواندن داده‌ها از فرمت‌های مختلف فایل مانند CSV، JSON و TXT به طور مستقیم در IDE هستند.

تحلیل کارآمد داده‌های ذخیره‌شده در دیتابیس‌های رابطه‌ای: ابزار Kotlin DataFrame به‌طور یکپارچه با دیتابیس‌های گوناگون ادغام می‌شود و قابلیت‌هایی مشابه به دستورات SQL را فراهم می‌کند. به طوری که می‌توان داده‌ها را به‌طور مستقیم از دیتابیس‌های مختلف بازیابی، ویرایش یا تجزیه و تحلیل نمود.

دریافت و تحلیل داده‌های زنده و پویا از وب‌سرویس‌ها و APIها: انعطاف‌پذیری ابزارهای EDA امکان ادغام با API های خارجی از طریق پروتکل‌هایی مانند Open API را فراهم می‌کند. این ویژگی کمک می‌کند تا داده‌ها را از وبAPIها دریافت کرده و سپس آن‌ها را برای نیازهای خود تبدیل کرد.

ایستگاه توضیح المسائل

در آمار، تحلیل داده‌های اکتشافی (EDA) رویکردی برای تحلیل مجموعه‌های داده به منظور خلاصه‌سازی ویژگی‌های اصلی آن‌ها است که معمولاً از گرافیک‌های آماری و سایر روش‌های تجسم داده استفاده می‌شود.

محیط توسعه و ابزارها

اکوسیستم قدرتمند کاتلین سبب شده است که برنامه‌نویسان و شرکت‌های مختلف محیط‌هایی را برای توسعه با کاتلین آماده سازند. شرکت مادر کاتلین یعنی Jetbrains ویرایشگرها یا IDEهایی (Integrated Development Environment) را ارائه می‌دهد که به طور رسمی از کاتلین پشتیبانی می‌کنند. در ادامه به معرفی این IDEها خواهیم پرداخت.

نرم‌افزار IntelliJ

این نرم‌افزار یک محیط توسعه یکپارچه (IDE) طراحی‌شده برای زبان‌های مبتنی بر JVM مانند کاتلین و جاوا است که هدف آن حداکثرسازی بهره‌وری توسعه‌دهندگان است. این IDE وظایف روتین و تکراری را با ارائه تکمیل خودکار هوشمند کد، تحلیل استاتیک کد و ابزارهای بازسازی (Refactoring) انجام می‌دهد. به این ترتیب، سبب تمرکز روی جنبه‌های مختلف توسعه نرم‌افزار و تجربه‌ای لذت بخش‌تر از این فرایند می‌شود.

افزونه کاتلین (Kotlin Plugin) به‌صورت پیش‌فرض با هر نسخه IntelliJ IDEA عرضه می‌شود. هر نسخه جدید این IDE ویژگی‌ها و ارتقاءهایی را ارائه می‌دهد که تجربه توسعه با کاتلین را بهبود می‌بخشد.

ویرایشگر Fleet

فلیت یا JetBrains Fleet یک ویرایشگر کد چندزبانه است که پشتیبانی پیشرفته‌ای برای کاتلین ارائه می‌دهد و تجربه‌ای ساده و کارآمد برای توسعه‌دهندگان کاتلین فراهم می‌کند. می‌توان از Fleet جهت ویرایش‌های سریع و هدفمند استفاده کرد یا با فعال کردن حالت هوشمند (Smart Mode) آن را به یک ابزار قدرتمند با ویژگی‌های

هوشمندی کد تبدیل نمود. افزونه کاتلین (Kotlin Plugin) به‌صورت پیش‌فرض با هر نسخه Fleet ارائه می‌شود.

Fleet از پروژه‌های Kotlin Multiplatform که پلتفرم‌های Android، iOS، وب و دسکتاپ را هدف قرار می‌دهند از جمله تست و دیباگ پشتیبانی می‌کند. در حالت هوشمند، موتور پردازش کد مناسب انتخاب می‌شود و امکان جابجایی بین کدهای Kotlin Multiplatform و کدهای نوشته‌شده به زبان‌های سازگار با کاتلین فراهم می‌گردد.

نرم افزار Android Studio

اندروید استودیو محیط توسعه رسمی (IDE) برای برنامه‌نویسی اندروید است که بر پایه IntelliJ IDEA ساخته شده است. علاوه بر ویرایشگر کد قدرتمند و ابزارهای توسعه IntelliJ، اندروید استودیو ویژگی‌های بیشتری ارائه می‌دهد که بهره‌وری را در ساخت اپلیکیشن‌های اندرویدی افزایش می‌دهد. افزونه کاتلین (Kotlin Plugin) به‌صورت پیش‌فرض با هر نسخه Android Studio عرضه می‌شود.

سایر محیط‌های توسعه

JetBrains افزونه‌های رسمی کاتلین برای سایر IDEها ارائه نمی‌دهد. با این حال، برخی از IDEها و ویرایشگرهای کد مانند Eclipse، Visual Studio Code و Atom افزونه‌های کاتلین خود را دارند که توسط جامعه کاتلین پشتیبانی می‌شوند.

نرم افزار Eclipse به توسعه‌دهندگان این امکان را می‌دهد تا برنامه‌های خود را با زبان‌های برنامه‌نویسی مختلف، از جمله کاتلین، بنویسند. این IDE همچنین افزونه

کاتلین را دارد؛ افزونه‌ای که در ابتدا توسط JetBrains توسعه داده شده است و اکنون توسط مشارکت‌کنندگان جامعه کاتلین پشتیبانی می‌شود.

بررسی فنی

در بخش‌های قبلی به طور کلی به ویژگی‌های زبانی و کاربرد کاتلین اشاره شد؛ ویژگی‌هایی که هر کدام مشان دهنده قدرتمندی این زبان خاص هستند. با ما در این بخش همراه باشید تا این زبان را از لحاظ فنی، سینتکس و عملکرد بررسی نماییم.

از برنامه‌نویسی شی‌گرا تا تابعی

حدودا 58 سال از زمانی که زبان Simula منتشر شد می‌گذرد. این زبان برای اولین بار ویژگی‌ای را به دنیای کامپیوتر معرفی نمود که در عصر حاضر نیز به عنوان یک ویژگی مهم برای مقایسه زبان‌ها صورت می‌گیرد؛ این ویژگی شی‌گرایی (Object Oriented) است که به اختصار OOP نیز نامیده می‌شود.

با گذشت زمان زبان‌های پیشرفته دیگری مانند جاوا و C# ظهور پیدا کردند که به واسطه این ویژگی مهم یعنی شی‌گرایی توانستند به شهرت دست یابند و بتوانند توجه بسیاری از توسعه‌دهندگان را به خود جلب کنند.

جاوا فارغ از اکوسیستم جامع و امکانات پیشرفته‌ای که داشت باز نتوانست توجه برخی از توسعه‌دهندگان را جلب کند؛ زیرا در جاوا امکان برنامه‌نویسی فانکشنال (Functional) یا تابعی وجود ندارد. همین سبب شده است که جاوا برای برخی توسعه‌دهندگان انعطاف‌ناپذیر جلوه کند.

برنامه‌نویسی تابعی (Functional Programming) یک پارادایم برنامه‌نویسی است که در آن برنامه‌ها با استفاده از اعمال (Functions) و ترکیب آن‌ها ساخته می‌شوند. این پارادایم یک شیوه برنامه‌نویسی اعلانی (Declarative) است که در آن تعریف توابع به‌صورت درختی از عبارات است که مقادیر را به مقادیر دیگر نگاشت می‌دهند، برخلاف برنامه‌نویسی دستوری (Imperative) که شامل توالی‌ای از دستورات برای به‌روزرسانی وضعیت جاری برنامه است.

با این حال کاتلین تمامی پارادایم‌های اصلی برنامه‌نویسی را به شکلی زیبا ترکیب می‌کند و این امکان را فراهم می‌سازد که از برنامه‌نویسی تابعی، دستوری (Imperative)، شیءگرا (Object-Oriented) یا رویه‌ای (Procedural) همگی در یک زبان استفاده کنید. با پشتیبانی کاتلین از Coroutineها، مفاهیم همزمانی (Concurrency) و پردازش موازی (Parallelism)، چند نخی (Multi Threading) به‌طور طبیعی و ساده پیاده‌سازی می‌شوند.

> شما قادر هستید برنامه‌نویسی رویه‌ای (Procedural Programming) را برای مبتدیان آموزش دهید بدون نیاز به توضیح کلاس‌ها. بنابراین، دوره آموزشی شما می‌تواند منسجم‌تر و ساده‌تر باشد.
>
> Alexey Mitsyuk, HSE university

> دانشجویان کاتلین من در واقع مفاهیم شیءگرا (OO) را بهتر از دانشجویان جاوا درک می‌کنند.
>
> San Skulrattanakulchai, Gustavus Adolphus College

مدیریت حافظه

کامپیوتر برای اجرای دستورات خود ابتدا نیاز دارد که این دستورات را به حافظه منتقل کند و سپس توسط CPU این دستورات انجام شوند. بنابراین حافظه یکی از بخش‌های مهم در حیات یک کامپیوتر سالم است.

پیشرفت کامپیوتر سبب ایجاد حافظه‌ها و الگوریتم‌های مختلفی برای مدیریت حافظه شده شده است. یکی از این حافظه‌های مهم Heap نام دارد که تحت عنوان حافظه داینامیک نیز شناخته می‌شود. این حافظه در کنار حافظه Stack به کار می‌رود.

حافظه Stack اندازه‌ای ثابت دارد و داده‌های غیر استاتیکی همچون پارامترها و آدرس‌های return شده توابع در خود ذخیره می‌کند.حافظه استک با الگوریتم LIFO داده‌ها را ذخیره و آزاد می‌کند. الگوریتم LIFO مخفف Last In First Out است که به معنی "آخرین ورودی اولین خروجی" می‌باشد. بنابراین واضح است که در استک اطلاعات پشت سر هم قرار گرفته و آخرین داده آن در بالاترین استک قرار می‌گیرد. حال اگر قصد گرفتن یا برداشتن (Pop) اطلاعات را داشته باشیم اولین خروجی همان آخرین داده وارد شده است.

در حافظه Heap برخلاف Stack داده‌های ذخیره شده به همراه یک آدرس اختصاصی ذخیره می‌شوند؛ به عبارتی دیگر این فضای داده توسط Pointer یا اشاره‌گر قابل دسترسی است. چنین روندی باعث می‌شود حافظه هیپ نسبت به استک کندتر باشد زیرا نیاز به محاسبات مورد نیاز برای یافتن Pointer مورد نظر در Heap دارد.

هنگامی که توسعه‌دهنده یا برنامه‌نویس Objectهایی را ایجاد می‌کند، این آبجکت‌ها در حافظه Heap ذخیره می‌شوند. وظیفه برنامه‌نویس این است که خود پس از پایان کار داده آن را از حافظه آزاد کند. در غیر این صورت سبب بروز Memory Leak یا نشت حافظه می‌شود.

این آزادسازی دستی در زبان‌هایی مانند C و C++ وجود دارد. به مرور زمان زبان‌های زیادی مانند Java با استفاده از Garbage Collection یا جمع‌آوری زباله این مشکل را حل کردند تا بار دیگر توسعه‌دهنده و برنامه‌نویس دچار خطاهای نشت حافظه نشود و بتواند با خیال راحت کدنویسی کند.

جمع‌آوری و بازیابی زباله یا همان Garbage Collection به فرایند بازیافت خودکار فضای مشترک حافظهٔ کامپیوتر اطلاق می‌شود. در طی این فرایند فضایی از حافظه‌ی کامپیوتر که قبلا درگیر نگهداری دیتای مورد نیاز یک برنامهٔ کامپیوتری بوده و اکنون آن برنامه دیگر نیازی به این دیتا ندارد، آزاد شده و برای ذخیره و نگهداری دیتای جدید مورد استفاده قرار می‌گیرد. همچنین این فرآیند سبب می‌شود تا برنامه‌ی در حال اجرا، تمام حجم حافظه‌ی از پیش تعیین شده‌ی مخصوص خود را درگیر نکند.

گاربج کالکشن‌ها الگوریتم‌ها و روش‌های مختلفی دارند که هر کدام معایب و مزایای خود را دارند. به طور کلی GC از بروز خطاهای زیر جلوگیری می‌کند:

پوینترهای سرگردان (Dangling Pointers**):** این مشکل زمانی رخ می‌دهد که پوینتری به آبجکتی اشاره کند که در بخشی از حافظه آزاد شده باشد.

نشت حافظه (Memory Leaks**):** زمانی رخ می دهد که برنامه حافظه‌ای از آبجکت‌هایی را که دیگر در دسترس نیستند را نمی‌تواند آزاد کند و این به مرور زمان باعث فرسودگی و ناکارآمدی حافظه می‌شود.

اما GCها همیشه همراه با مزیت نیستند؛ باعث دردسرهایی می‌شود و تاثیر منفی بر عملکرد برنامه می‌گذارد. این معایب عبارت‌اند از:

مدیریت حافظه کاتلین

کاتلین با توجه به محیط اجرایی خود رویکردهای متفاوتی را برای مدیریت حافظه در پیش می‌گیرد. در کاتلین/نیتیو از یک مدیریت‌کننده مدرن برای حافظه استفاده می‌کند که شبیه به JVM، Go و سایر فناوری‌های مطرح عصر حاضر است و شامل ویژگی‌های زیر است:

- اشیاء (Objects) در یک هیپ (Heap) مشترک ذخیره می‌شوند و می‌توانند از هر رشته‌ای (Threads) قابل دسترسی باشند.

- جمع‌آوری زباله با استفاده از ردیابی به طور دوره‌ای انجام می‌شود تا اشیائی که از ریشه‌ها (Roots) (مانند متغیرهای محلی و سراسری) قابل دسترسی نیستند، جمع‌آوری شوند.

مدیریت حافظه در JVM

در JVM مدیریت حافظه توسط یک Garbage Collection اختصاصی با الگوریتم‌های مختلف کار می‌کند. این رویه برخلاف زبان‌هایی چون C و C++ است لذا محدودیت بیشتری را بر توسعه‌دهندگان تحمیل می‌کند.

بررسی نحوه کار GC در JVM در ابتدا نیازمند یادگیری شیوه کلی کارکرد JVM است. ماشین مجازی جاوا یا JVM شامل موارد زیر می‌باشد:

بارگذاری کلاس (Classloader**):** این فرایند به بارگذاری کلاس‌ها به حافظه در زمان اجرای برنامه اشاره دارد. وظیفه این مورد برقراری ارتباط، بارگذاری و مقدار دهی اولیه به صورت داینامیک است.

هیپ (Heap**):** مقدار دهی‌های اولیه، آرایه‌ها و اشیاء (Objects) به صورت Heap در حافظه ذخیره می‌شوند. در زمان فعالیت برنامه جاوا، محتوای Heap میان چندین Thread تقسیم و به اشتراک گذاشته می‌شود.

رجیسترهای شمارنده‌ی برنامه (PC Registers**):** شمارنده برنامه (PC) یک رجیستر است که آدرس دستور بعدی که باید اجرا شود را نگهداری می‌کند. به عبارت دیگر، PC نشان‌دهنده موقعیت یا آدرس دستور بعدی در حافظه است که پردازنده باید آن را پردازش کند. هر Thread خود دارای یک PC Registers اختصاصی است.

محدوده متد یا دستورات (Method Area**):** این عنوان در JVM به فضایی از حافظه اطلاق می‌شود که برای ذخیره‌سازی اطلاعات مربوط به کلاس‌ها و متدها استفاده

می‌شود. این بخش از حافظه درواقع قسمتی از Heap است که به متادیتاها اختصاص دارد.

الگوریتم‌های GC **در** JVM

الگوریتم GC در JVM به طور کلی به فرایندهایی گفته می‌شود که داده‌های غیر قابل استفاده را به طور خودکار از Heap حذف می‌کند. مانند آبجکت‌هایی که دیگر توسط برنامه رفرنس داده نمی‌شوند.

رایج‌ترین الگوریتم مورد استفاده JVM برای GC، الگوریتم علامت‌گذاری و پاک‌سازی یا به انگلیسی Mark-and-Sweep است که در آن فرایند GC ابتدا از بین آبجکت‌ها عبور می‌کند و آن‌هایی که هنوز در حال استفاده هستند را علامت‌گذاری می‌کند؛ سپس آبجکت‌های بدون علامت را به عنوان زباله حذف می‌کند. زیرا دیگر قابل دسترسی نیستند. سایر الگوریتم‌ها به شرح زیر هستند:

Serial GC: ساده‌ترین الگوریتم، که از یک Thread برای جمع‌آوری زباله استفاده می‌کند و برای برنامه‌های کوچک در سیستم‌های تک‌هسته‌ای مناسب است.

Parallel GC**:** از چندین Thread برای انجام جمع‌آوری زباله به صورت موازی با برنامه استفاده می‌کند که توان عملیاتی را در سیستم‌های چند هسته‌ای بهبود می‌بخشد.

CMS **یا** Concurrent Mark Sweep**:** هدف این الگوریتم کاهش زمان توقف است، زیرا بیشتر کارهای جمع‌آوری زباله را به صورت هم‌زمان با برنامه انجام می‌دهد، اما ممکن است مشکلاتی مانند پراکندگی حافظه ایجاد کند.

Garbage First **یا** G1**:** یک الگوریتم پیشرفته‌تر است که Heap را به بخش‌های مختلف تقسیم می‌کند و اولویت جمع‌آوری بخش‌هایی را می‌دهد که بیشترین زباله را دارند. این الگوریتم تعادلی خوب بین توان عملیاتی و زمان‌های توقف فراهم می‌کند.

بررسی سینتکس

کاتلین با هدف بهبودی نسبت به جاوا، سینتکسی بسیار آسان‌تر، خواناتر دارد و برای کدهای یکسان، خط و کد کمتری برای نوشتن نیاز است.

HelloWorld.kt

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

HelloWorld.java

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

نمونه کد: کد نمایش سلام دنیا! در دو زبان جاوا و کاتلین

پسوند فایل‌های کاتلین به دو صورت .kt و .ktx است که به بسته به نوع کار از هر کدام استفاده می‌شود. پسوند .kt برای فایل‌های معمولی و رایج کاتلین است که ساختارهای برنامه‌نویسی در آنها قرار می‌گیرند و توسط JVM اجرا می‌شوند. پسوند .kts نیز برای

کدها و فایل‌های کاتلین که قصد داریم به صورت اسکریپتی اجرا شوند مورد استفاده قرار می‌گیرد.

کالکشنی از سینتکس

شما در ادامه شاهد بخش‌هایی از سینتکس کاتلین خواهید بود و متوجه خواهید شد که این زبان چقدر از لحاظ سینتکسی سرعت کدنویسی را بهبود می‌بخشد.

تعریف پکیج‌ها

در کاتلین مشابه آنچه در جاوا وجود دارد، یک سورس کد می‌تواند با یک Package Name آغاز شود. پکیج روشی برای سازماندهی کلاس‌ها، اینترفیس‌ها و ... در یک Namespace می‌باشد. شیوه پکیج‌بندی را می‌توان به همان شیوه کلاسیک فولدر تشبیه کرد که با جداسازی آنها مدیریت فولدرها و محتوایشان نیز آسانتر می‌شود.

HelloWorld-Package.kt

```kotlin
package my.demo /* تعریف به صورت پکیج */

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

نقطه آغازین

هر برنامه کاتلین نقطه آغازی دارد. این نقطه آغاز با تابع main مانند نمونه زیر تعریف می‌شود. تابع println آرگومانی که می‌گیرد را در صفحه چاپ می‌کند و نمایش می‌دهد. تفاوت تابع println با print این است که اگر در خط‌های مختلف تابع println با آرگومان در Nبار تکرار شود بین آنها خط شکسته اضافه می‌کند به عبارتی دیگر باعث می‌شود آرگومان تابع println در تکرار N+1 در خط بعدی چاپ شود.

PrintAndPrintln.kt

```kotlin
fun main() {
    print("Hello ")
    print("World!")

    /* Hello World! */

    println("Hello")
    println("World!")

    /*
    Hello
    World!
    */
}
```

تعریف متغییر و انواع تایپ‌ها

در کاتلین، همه چیز یک آبجکت است؛ به این معنا که می توانید توابع و ویژگی‌های عضو را روی هر متغیری فراخوانی کنید. کاتلین نوع متغییر را اجبار نمی‌کند و می‌توان متغییری بدون تایپ تعریف نمود. تایپ داده‌ها و متغییرها در کاتلین عبارت‌اند از:

- Numbers
- Booleans
- Characters
- Strings
- Arrays
- Any
- Nothing
- Unit

Variables.kt

```kotlin
fun main() {
    val inferredType = 42
    var inferredString = "Hello, Kotlin"
    val intNum: Int = 10
    val isKotlinFun: Boolean = true
    val isProgrammingHard: Boolean = false
}
```

برنامه‌نویسی ناهمزمان و کوروتین‌ها

در Kotlin، برنامه‌نویسی async و کوروتین‌ها ابزارهای قدرتمندی برای مدیریت وظایف همزمان (Concurrency) هستند که با جلوگیری از بلاک شدن تردها (Threads) عملکرد برنامه را بهینه می‌کنند. کوروتین‌ها نوعی ساختار سبک برای مدیریت همزمانی (Concurrency) هستند که می‌توانند وظایف را به‌صورت معلق (Suspend) اجرا کنند و از سربار تردها جلوگیری می‌کنند.

Coroutines.kt

```kotlin
import kotlinx.coroutines.*

fun main() = runBlocking {
    // اجرای وظیفه‌ای که نتیجه بازمی‌گرداند (async)

    val deferred = async {
        fetchDataFromServer() // فراخوانی تابع معلق
    }

    // اجرای وظیفه‌ای که نتیجه‌ای بازنمی‌گرداند (launch)

    launch {
        println("Loading data...")
    }

    // استفاده از نتیجه async
    println("Fetched Data: ${deferred.await()}")
}

// تابع معلق که داده‌ها را شبیه‌سازی می‌کند
suspend fun fetchDataFromServer(): String {
    delay(2000) // شبیه‌سازی تأخیر
    return "Hello from Server"
}
```

بحث و نتیجه‌گیری

در توسعه نرم‌افزار، توسعه‌دهندگان با توجه به شرایط فنی پروژه و حتی مالی تکنولوژی‌های مورد استفاده را برای آن پروژه تعریف می‌کنند. لذا برخی مواقع با یک تیر دو نشان زدن نه تنها کاربردی است بلکه سبب سرعت بخشیدن به روند کار می‌شود.

کاتلین با ویژگی‌های خود اثبات می‌کند که انتخاب خوبی برای آن دسته از توسعه‌دهندگان و استارتاپ‌ها است که نمی خواهند زیاد در حال یادگیری زبان و تکنولوژی های جدید باشند. زیرا کاتلین با وجود اینکه تنها یک زبان است کاربردهای فراوانی در تمام پلتفرم‌ها دارد.

به یقین کاتلین ارزش یادگیری دارد و آمار نشان می‌دهد کاتلین چشم‌انداز بسیار روشنی دارد؛ زیرا جامعه برنامه‌نویسان کاتلین در حال رشد است و در پشت حمایت‌کنندگان کاتلین شرکت Jetbrains قرار دارد. شرکتی که حتی گوگل طراحی و تولید کاربردی‌ترین نرم‌افزار توسعه اپلیکیشن اندروید را به آن سپرده است.

کاتلین بسیار آسان است و حتی با مراجعه به سایت کاتلین مستندات جامعی از آن وجود دارد و با توضیح شفافی که ارائه شده روند یادگیری را لذت بخش نیز می‌کند.

فارغ از مزایای بی‌شمار کاتلین، معایبی نیز دارد؛ محیط توسعه با این زبان بسیار محدود است. IDE هایی که از این زبان پشتیبانی می‌کنند حجیم هستند و نیاز به سخت‌افزار متوسطی دارند.

همچنین مشکلاتی چون تحریم برای ایرانیان می‌تواند بسیار گران تمام شود؛ گریدل یا Gradle سیستم ساخت اتوماتیک متن‌باز است که برای ساخت، تست و استقرار نرم‌افزارها استفاده می‌شود. برای توسعه اپلیکیشن‌های کاتلین و زبان‌های بر پایه JVM گریدل محبوبیت زیادی دارد. اما به دلیل آنچه که ما آن را تحریم می‌خوانیم روند سینک شدن آن با پروژه بسیار طولانی است و اغلب با ارورهای زیادی همراه است. ارورهایی که که اگر در کشور همسایه‌مان عراق بودیم احتمالا رخ نمی‌داد. لذا برای افراد مبتدی ممکن است چالش‌هایی را ایجاد کند. برای حل این مشکل ابزارهای تحریم‌شکنی چون Online 403 و Shekan وجود دارند اما از قدرت کافی برخوردار نیستند.

در نهایت کاتلین من را یاد مارشمالو می‌اندازد: شیرین، ساده و خوشمزه.

رفرنس‌ها

- Kotlin Programming Language Official Website
  https://kotlinlang.org/
- Kotlin Programming Language Official Documentation
  https://kotlinlang.org/docs/home.html
- https://en.wikipedia.org/wiki/Kotlin_(programming_language)
- https://en.wikipedia.org/wiki/Java_(programming_language)
- https://gs.statcounter.com/os-market-share/mobile/worldwide
- https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/
- https://medium.com/javarevisited/understanding-garbage-collection-algorithms-in-java-6d6e7ddf5272
- https://www.geeksforgeeks.org/how-many-types-of-memory-areas-are-allocated-by-jvm/
- https://en.wikipedia.org/wiki/Memory_management
- https://en.wikipedia.org/wiki/Document_Object_Model
- https://en.wikipedia.org/wiki/Exploratory_data_analysis
- https://kotlinlang.org/education/why-teach-kotlin.html
- https://appinventiv.com/blog/google-play-store-statistics/
- https://www.jetbrains.com/compose-multiplatform/