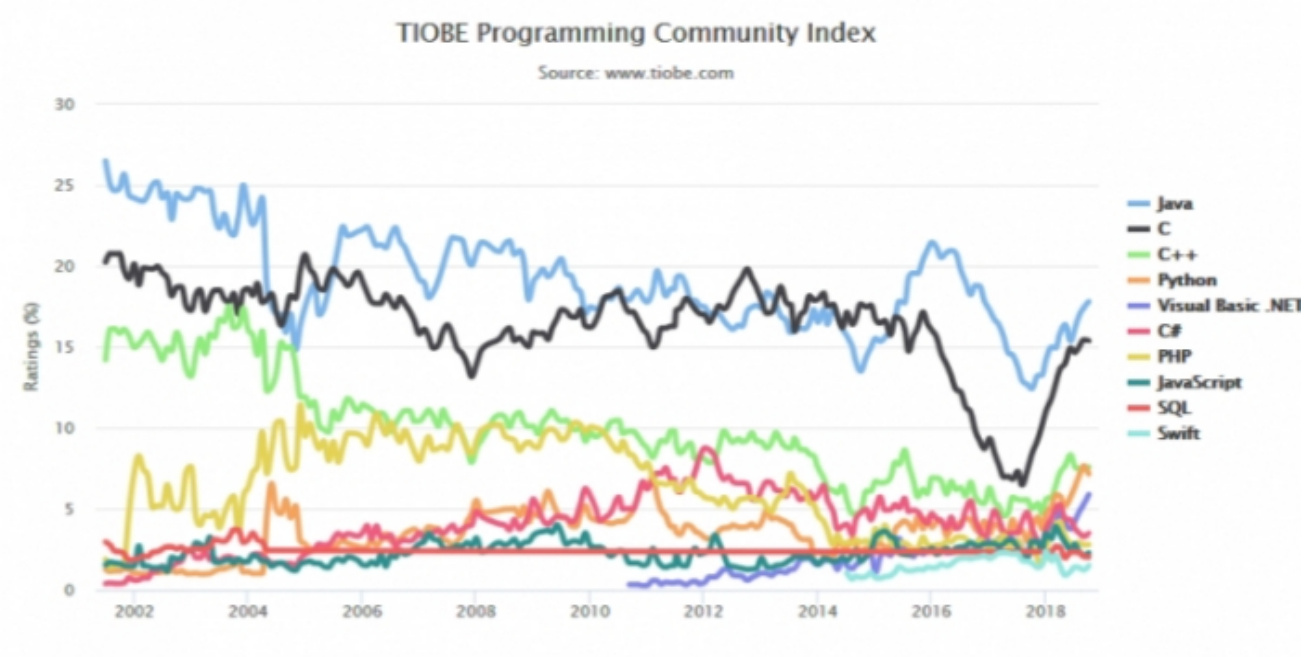
مروری اجمالی بر جاوا

نویسنده : علی زین الدینی

برنامه نویسی جاوا

زبان برنامه نویسی جاوا (JAVA) در 23 مه 1995 (برابر با 2 خرداد 1374) از سوی جیمز گاسلینگ (James Gosling) طراحی شده است. جاوا به گواهی سایت معتبر Tiobe از سال 2001 همواره به عنوان اولین یا دومین زبان برنامه نویسی دنیا مطرح بوده است. در تصویر زیر درصد رتبه کسب شده این زبان را بین همه زبان های برنامه نویسی در طی 18 سال اخیر مشاهده می کنید.



تا جایی که به جاوا مربوط می شود، همه چیز از اوایل دهه 1990 آغاز شد، یعنی زمانی که شرکت سان مایکروسیستمز (Sun Microsystems) شروع به توسعه نسخه بهتری از C++ کرد که بتواند به آسانی پرتابل شود، برای افراد مبتدی مطلوب باشد و به مدیریت خودکار حافظه بپردازد. تحقیقات این شرکت منجر به خلق یک

زبان کاملاً جدید شد که نام آن از میان ده ها نام پیشنهادی در اتاق جلسه معرفیش انتخاب شد. امروزه لوگوی جاوا به صورت فنجان قهوه ای که از آن بخار می دمد، به نماد جهانی برنامه نویسی تبدیل شده است. در واقع دیگر کسی نمی داند که ارتباط برنامه نویس ها با کافئین قدیمی تر است؛ یا ارتباطشان با جاوا که امروزه مترادف با قهوه است.

در این نوشته به بررسی کامل و جامع این زبان برنامه نویسی، ویژگی ها، مزایا، معایب، بازار کار، کتابخانه ها، کاربردها و هر آنچه که ممکن است در مورد جاوا بخواهید بدانید می پردازیم.

جاوا چیست؟

جاوا یک زبان برنامه نویسی چندمنظوره و شی گرا است که تا حدودی زیادی به C و C++ شباهت دارد؛ اما استفاده از آن آسان تر است و امکان ساخت برنامه هایی قدرتمند با آن وجود دارد. البته تعریفی که شرکت سان مایکروسیستمز در سال 2000 ارائه کرده است، شاید از

تعریف فوق گویاتر باشد:

جاوا زبان رایانه ای ساده، شی گرا، مناسب شبکه، تفسیرشدنی، مستحکم، امن، فارغ از معماری، پرتابل، با عملکرد بالا، چندنخی و دینامیک است.

در ادامه هر یک از خصوصیات فوق را به طرز جداگانه ای مورد بررسی قرار می دهیم:

جاوا یک زبان ساده است

جاوا در ابتدا بر اساس زبان های C و C++ و با حذف برخی ویژگی هایی که قابلیت سردرگم کننده داشتند مدلسازی شد. از جمله این ویژگی ها می توان به اشاره گرها، پیاده سازی چندباره وراثت و بارگذاری بیش از حد عملگرها اشاره کرد که در جاوا حذف شدند. یکی از ویژگی هایی که در C++/C وجود نداشت؛ اما جزو ویژگی های اساسی جاوا به شمار می آید، امکان بازیافت حافظه (garbage-collection) است که به طور خودکار اشیا و آرایه های بی استفاده را حذف می کند.

جاوا یک زبان شی گرا است

تمرکز شی گرایی جاوا موجب شده است که توسعه دهندگانی که از این زبان استفاده می کنند، از آن برای حل یک مسئله استفاده کنند و نه سروکله زدن با آن برای رفع محدودیت های مسئله. این وضعیت موجب

تمایز بین جاوا و C شده است. برای نمونه اگر می خواهید برنامه ای برای یک حساب بانکی بنویسید، در جاوا تنها باید به ذخیره سازی شی حساب بانکی بپردازید؛ اما در C باید وضعیت حساب (تراز حساب) و رفتارهایی مانند واریز یا برداشت را به طور مستقل برنامه نویسی کنید.

جاوا یک زبان مبتنی بر شبکه است

کتابخانه وسیع شبکه در جاوا باعث شده است که امکان کار با پروتکل کنترل انتقال/پروتکل اینترنت (TCP/IP) و پروتکل های شبکه مانند HTTP (پروتکل انتقال ابرمتن) و FTP (پروتکل انتقال فایل) ساده تر شود و همچنین وظیفه ایجاد اتصال های شبکه آسان تر گشته است. به علاوه برنامه های جاوا می توانند از طریق شبکه TCP/IP، از طریق URLها، به اشیا دسترسی داشته باشند و این دسترسی به همان سادگی دسترسی روی سیستم فایل محلی است.

جاوا یک زبان تفسیر شده است

یک برنامه جاوا در زمان اجرا به طور غیر مستقیم از طریق یک ماشین مجازی (که بازنمایی نرم افزاری از یک پلتفرم فرضی است) و محیط زمان اجرای مرتبط با آن با واسطه روی یک پلتفرم زیرساخت (مانند ویندوز یا لینوکس) اجرا می شود. این ماشین مجازی بایت کدها (دستورالعمل ها و داده های مرتبط) را از طریق تفسیر به دستورالعمل های خاص پلتفرم ترجمه می کند. منظور از تفسیر، عمل شناسایی معنی دستورالعمل ها و سپس انتخاب دستورالعمل های خاص پلتفرم برای اجرا است. در ادامه ماشین مجازی این دستورالعمل های

خاص پلتفرم را اجرا می کند. این ویژگی تفسیری بودن جاوا باعث شده است که خطایابی برنامه های جاوا آسان تر شود، چون اغلب اطلاعات در زمان کامپایل در واقع در محیط اجرا وجود دارند. تفسیری بودن جاوا همچنین امکان به تأخیر انداختن پیوند بین قطعات مختلف برنامه جاوا تا زمان اجرا را فراهم ساخته است و این مسئله موجب افزایش سرعت توسعه برنامه می شود.

جاوا یک زبان مستحکم است

برنامه های جاوا باید قابل اعتماد باشند، زیرا هم در اپلیکیشن های مصرفی و هم مأموریت های حیاتی استفاده می شوند که از پخش کننده های بلوری تا ناوبری خودرو یا سیستم های کنترل هوایی را شامل می شود. ویژگی های زبان جاوا که باعث استحکام آن می شوند، شامل اعلان ها، بررسی دوباره نوع داده، یک بار در زمان کامپایل و بار دیگر در زمان اجرا (برای جلوگیری از عدم تطبیق نسخه ها)، آرایه های واقعی با بررسی خودکار کران ها و کنار گذاشتن اشاره گرها است.

جنبه دیگری که موجب استحکام جاوا می شود، این است که حلقه ها به جای عبارت های عدد صحیح که در آن 0 برابر «نادرست» و مقادیر غیر صفر برابر با «درست» هستند، باید به وسیله عبارت های بولی کنترل شوند. برای مثال برخلاف C، در جاوا حلقه هایی مانند عبارت زیر

```
while (x) x++
```

مجاز نیستند؛ زیرا این حلقه ممکن است در جایی که انتظار می رود متوقف نشود. به جای آن باید عبارت های بولی صریحی مانند زیر

```
while (x!= 10) x++;
```

استفاده شود، یعنی حلقه تا زمانی که x برابر با 10 شود، اجرا خواهد شد.

جاوا یک زبان امن است

برنامه های جاوا در محیط های شبکه بندی شده/ توزیع یافته مورد استفاده قرار می گیرند. از آنجا که برنامه های جاوا می توانند روی پلتفرم های مختلف شبکه اجرا شوند، امن ساختن این پلتفرم ها در برابر کدهای مخرب که موجب گسترش ویروس ها، سرقت اطلاعات کارت های بانکی یا اجرای اعمال خرابکارانه می شوند، امری حائز اهمیت است. ویژگی هایی که موجب استحکام زبان جاوا می شوند شامل کنار گذاشتن اشاره گرها هستند که به همراه ویژگی های امنیتی مانند مدل امن sandbox جاوا و رمزنگاری کلید عمومی فعالیت می کنند. این دو نوع از ویژگی ها در کنار هم از تأثیر ویروس ها و دیگر کدهای خطرناک روی پلتفرم های مشکوک جلوگیری می کنند. جاوا از لحاظ تئوریک امن است؛ اما در عمل آسیب پذیری های امنیتی مختلفی شناسایی و مورد سوءاستفاده قرار گرفته است. در نتیجه در زمان های قبل، شرکت سان مایکروسیستمز و اینک شرکت اوراکل همواره اقدام به انتشار به روزرسانی های امنیتی برای جاوا می کنند.

جاوا یک زبان فارغ از معماری است

شبکه ها موجب اتصال پلتفرم هایی با معماری مختلف ریزپردازنده ها و سیستم های عامل می شوند. نمی توان انتظار داشت که جاوا دستورالعمل های خاص پلتفرم را ایجاد کند و انتظار داشته باشد که این دستورالعمل ها از سوی همه انواع پلتفرم هایی که بخشی از شبکه هستند درک شود. در عوض جاوا دستورالعمل های بایت کد مستقل از پلتفرم ایجاد می کند که تفسیر آن برای هر پلتفرم (از طریق پیاده سازی JVM) آسان است.

جاوا یک زبان پرتابل است

عدم وابستگی به معماری موجب پرتابل شدن جاوا شده است. با این حال پرتابل بودن جاوا چیزی فراتر از مستقل بودن دستورالعمل های بایت کدها از پلتفرم است. برای مثال در نظر بگیرید که اندازه نوع عدد صحیح روی پلتفرم های مختلف یکسان خواهد بود. برای نمونه یک نوع عدد صحیح 32 بیتی، صرف نظر از این که روی پلتفرم های با رجیسترهای 16 بیتی، 32 بیتی یا 64 بیتی پردازش شود؛ در هر حال به صورت علامت دار بوده و 32 بیت از حافظه را اشغال می کند. کتابخانه های جاوا نیز به پرتابل بودن آن کمک می کنند. این کتابخانه ها در موارد ضروری، انواع داده ای را ارائه می کنند که به روشی تا حد امکان پرتابل، کد جاوا را به قابلیت های خاص پلتفرم متصل می سازد.

جاوا یک زبان با عملکرد بالا است

ویژگی تفسیری بودن جاوا موجب شده است عملکرد بالایی داشته باشد که در اکثر موارد بیش از حد کفایت است. جاوا در مورد اپلیکیشن های با عملکرد بسیار بالا از کامپایل درجا (just-in-time) استفاده می کند یعنی دستورالعمل های بایت کد تفسیر شده را تحلیل می کند و دستورهای تفسیر شده با بسامد بالا را به دستورالعمل های خاص پلتفرم کامپایل می کند. تلاش های بعدی برای تفسیر این دستورالعمل های بایت کد موجب اجرای همان دستورالعمل های خاص پلتفرم می شود و به این ترتیب عملکرد نرم افزار را ارتقا می بخشد.

جاوا یک زبان چند نخی (multithread) است

جاوا برای بهبود عملکرد برنامه هایی که چندین وظیفه را به یک باره اجرا می کنند، از مفهوم اجرای چند نخی پشتیبانی می کند. برای نمونه برنامه ای که رابط گرافیکی کاربر (GUI) را مدیریت می کند و در همین حال منتظر ورودی از یک اتصال شبکه است، از نخ (thread) دیگری به جای نخ GUI برای این انتظار استفاده می کند. بدین ترتیب رابط گرافیکی برنامه همچنان پاسخگو است. ابتکارهای همگام سازی جاوا به نخ ها اجازه می دهد که داده ها را بدون هیچ تأثیر مخربی بین خود مبادله کنند.

جاوا یک زبان پویا (دینامیک) است

به دلیل ارتباط های متقابل بین کد برنامه و کتابخانه ها که در زمان

اجرا به صورت دینامیک صورت می پذیرند، نیازی به ایجاد لینک صریح بین آن ها وجود ندارد. در نتیجه زمانی که یک برنامه یا یکی از کتابخانه های آن تکامل می یابد (برای مثال باگ اصلاح می شود یا عملکرد بهبود می یابد) توسعه دهنده تنها باید برنامه یا کتابخانه به روزرسانی شده را منتشر سازد. با این که رفتار دینامیک جاوا موجب شده است به کد کمتری هنگام تغییر کد نیاز باشد؛ اما این روش انتشار می تواند موجب تداخل هایی نیز بشود. برای نمونه یک توسعه دهنده ممکن است یک نوع کلاس را از یک کتابخانه حذف کند یا نام آن را تغییر دهد. وقتی شرکتی کتابخانه به روز شده را منتشر می کند، برنامه های موجود که به آن نوع کلاس وابسته هستند از کار می افتند. برای حل این مشکل جاوا از نوع رابط (interface type) پشتیبانی می کند که مانند تعامل بین دو طرف است.

با توجه به ویژگی های فوق متوجه می شویم که جاوا علاوه بر نوعی زبان، یک پلتفرم برنامه نویسی نیز محسوب می شود. این پلتفرم از دو بخش مهم تشکیل یافته است که شامل ماشین مجازی جاوا و محیط اجرایی جاوا است.

ریشه های پیدایش جاوا

جاوا از سوی تیمی در شرکت سان مایکروسیستمز به رهبری جیمز گاسلینگ توسعه یافته و در سال 1995 منتشر شد. این زبان متعاقباً از سوی شرکت اوراکل خریداری شده است.

هدف اصلی خالقان جاوا این بوده که زبانی را ایجاد کنند که بتوانند آن را روی کاربردهای مصرفی اجرا کنند. این طراحان می توانسته اند دنیایی را تصور کنند که در آن کدها روی یخچال یا دستگاه توستر اجرا می شوند، یعنی آن چه که امروز به نام اینترنت اشیا می شناسیم. ما تنها در طی سال های اخیر دستگاه هایی ساخته ایم که چنین قابلیت هایی داشته باشند و از این رو باید گفت که این طراحان اولیه بسیار از زمان خود جلوتر بوده اند. هدف طراحی این زبان منجر به چنین معماری برای آن شده است. یکی از شعارهای مهم زبان برنامه جاوا چنین است: «یک بار بنویس، همه جا اجرا کن». به بیان دیگر شما با جاوا می توانید کدی بنویسید که آن را برای اجرای روی هر نوع دستگاهی کامپایل کنید.3

اما نکته جالب این است که جاوا به دلیل این ویژگی خود محبوب نشده است؛ بلکه از مزیت فناوری نوظهوری که در همان نیمه های دهه 90 میلادی ظهور یافت و چهره دنیا را دگرگون ساخت بهره گرفت؛ منظور ما فناوری وب است. جاوا این قابلیت را داشت که با آن می شد برنامه هایی به نام applet نوشت. این اپلت ها برنامه های کوچکی بودند که می شد داخل مرورگرهای وب آن ها را اجرا کرد. با رشد خیره کننده وب جاوا نیز سوار این موج شد و به یک زبان برنامه نویسی بسیار محبوب تبدیل شد. بدین ترتیب علی رغم این که قصد اولیه طراحان این زبان چیز دیگری بود؛ اما بسیاری از وب اپلیکیشن ها به زبان جاوا نوشته شدند.

واقعیت این است که طراحان اولیه زبان برنامه نویسی تا حدود زیادی تحت تأثیر زبان هایی مانند C و C++ بوده اند و جاوا نیز شباهت های

دستور زبانی زیادی با این زبان ها دارد. خالقان جاوا از این زبان ها به عنوان نمونه ای برای انجام کارها استفاده کردند و از این رو ویژگی های خاصی وجود داشتند که طراحان جاوا قصد نداشتند در جاوا آن ها را پیاده سازی کنند، چون قبلاً ثابت شده بود که در C و C++ موجب بروز مشکلاتی می شوند.

نسخه های مختلف جاوا

شرکت سان مایکروسیستمز، کیت توسعه نرم افزاری (JDK) شماره 1.0 جاوا را در سال 1995 منتشر کرده است. این JDK نخست برای توسعه اپلیکیشن های دسکتاپ و اَپلت (applet) مورد استفاده قرار گرفت. متعاقباً جاوا برنامه نویسی دستگاه های موبایل و سرورهای تجاری را نیز در این کیت میسر ساخت. ذخیره سازی همه کتابخانه ها در یک JDK منفرد باعث شده که این کیت بسیار بزرگ تر از حد مناسب برای توزیع شود. البته باید این نکته را در نظر داشته باشید که توزیع نرم افزارها در دهه 1990 توسط CD های اندازه کوچک و یا از طریق سرعت های پایین شبکه صورت می گرفته است. از آنجا که اغلب توسعه دهندگان به همه API ها نیاز نداشتند (یک توسعه دهنده اپلیکیشن های دسکتاپ به ندرت به API های سرورهای تجاری نیاز پیدا می کند) شرکت Sun این مشکل توزیع را با تقسیم جاوا به سه نسخه حل کرد. این نسخه ها نهایتاً به نام JAVA EE، JAVA SE و JAVA ME نامیده شدند که در ادامه هر کدام را توضیح داده ایم.

پلتفرم جاوا، نسخه استاندارد (Java SE)

این نسخه از جاوا برای توسعه اپلیکیشن های سمت کلاینت که روی رایانه های رومیزی اجرا می شوند، و اپلت ها که روی مرورگرهای وب اجرا می شوند، طراحی شده است.

پلتفرم جاوا نسخه انترپرایز (Java EE)

این نسخه از جاوا بر مبنای JAVA SE طراحی شده و به طور انحصاری برای توسعه اپلیکیشن های سرور با گرایش سازمانی استفاده می شود. اپلیکیشن های سمت سرور شامل سرولت ها (Servlet) می شود که برنامه های جاوای مشابه اپلت هستند؛ اما به جای کلاینت روی سرور اجرا می شوند. سرولت ها از API Java EE Servlet استفاده می کنند.

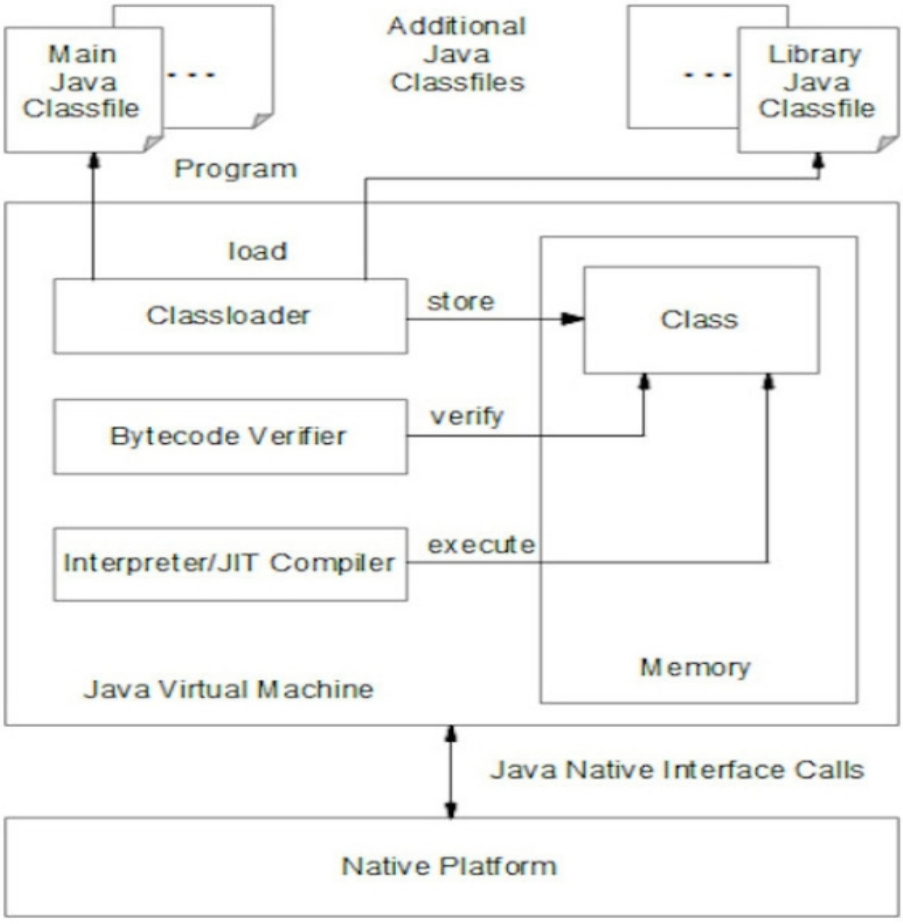
پلتفرم جاوا، نسخه میکرو (Java ME)

این نسخه از جاوا بر مبنای JAVA SE طراحی شده است. این پلتفرم برای توسعه میدلت ها (MIDlet) استفاده می شود که برنامه های جاوایی هستند که روی دستگاه های اطلاعاتی موبایل اجرا می شوند. ایکس لت (Xlet) ها نیز برنامه های جاوایی هستند که روی دستگاه های مونتاژی اجرا می شوند.

به هر حال java SE پلتفرم اصلی زبان جاوا میباشد و پلتفرم های دیگر بر مبنا آن طراحی شده اند

مروری بر پلتفرم جاوا

جاوا هم یک زبان برنامه نویسی و هم پلتفرمی برای اجرای کد جاوای کامپایل شده است. این پلتفرم به طور عمده شامل JVM است؛ اما محیط اجرایی را نیز شامل می شود که از اجرای JVM روی پلتفرم های زیرساختی بومی پشتیبانی می کند. JVM خود شامل چند جزء برای بارگذاری، تأیید و اجرای کد جاوا است. در تصویر زیر شیوه اجرای یک کد جاوا روی این پلتفرم را مشاهده می کنید



در بخش فوقانی نمودار فوق یک سری از فایل های کلاس برنامه را مشاهده می کنید که یکی از آن ها به نام فایل کلاس اصلی نامیده شده است. برنامه جاوا دست کم باید یک فایل کلاس اصلی داشته باشد که

نخستین کلاسی است که بارگذاری، تأیید و اجرا خواهد شد

JVM بارگذاری کلاس را به مولفه classloader می سپارد.
Classloader فایل های کلاس را از منابع مختلف مانند سیستم های فایل، شبکه ها و فایل های فشرده بارگذاری می کند. این مولفه، JVM را از مسائل و مشکلات مرتبط با بارگذاری کلاس بر حذر می دارد.

یک فایل کلاس بارگذاری شده در حافظه نگهداری می شود و به صورت یک شی ایجاد شده از کلاس Class نمایش می یابد. هنگام بارگذاری، bytecode verifier دستورالعمل های مختلف بایت کد را بررسی می کند تا مطمئن شود که معتبر هستند و امنیت را به مخاطره نمی اندازند.

اگر بایت کد فایل کلاس معتبر نباشد، JVM خاتمه می یابد. در غیر این صورت مولفه مفسر آن، بایت کد را یک به یک به دستورالعمل ها تفسیر می کند. در این فرایند تفسیر، دستورالعمل های بایت کد شناسایی شده و دستورالعمل های بومی معادل آن ها اجرا می شوند.

برخی توالی های دستورالعمل های بایت کد بیش از دیگر موارد تکرار می شوند. وقتی مفسر این موقعیت را تشخیص می دهد، کامپایلر درجای JVM به نام JIT این توالی های بایت کد را به کد بومی کامپایل می کند تا سریع تر اجرا شوند.

در زمان اجرا، مفسر معمولاً با درخواستی برای اجرای بایت کد فایل

کلاس دیگر مواجه می شود که به این برنامه یا یک کتابخانه تعلق دارد. در چنین مواردی classloader فایل کلاس را بارگذاری می کند و bytecode verifier بایت کد فایل کلاس بارگذاری شده را پیش از اجرا تأیید می کند. همچنین در زمان اجرا نیز دستورالعمل های بایت کد ممکن است از JVM بخواهند که یک فایل را باز کند، چیزی را روی صفحه نمایش دهد، صدایی ایجاد کند یا وظیفه دیگری انجام دهد که نیازمند همکاری با پلتفرم بومی است. در این موارد JVM با استفاده از رابط بومی جاوا (JNI) که یک پل فناوری برای تعامل با پلتفرم بومی برای اجرای وظایف است این کار را انجام می دهد.

نوع دینامیک یا نوع استاتیک

جاوا یک زبان برنامه نویسی با نوع استاتیک است در حالی که زبان های برنامه نویسی دیگری مانند روبی (Ruby)، پایتون (Python)، و جاوا اسکریپت زبان هایی از نوع دینامیک محسوب می شوند. اکثر افراد به بحث تفاوت بین زبان های با نوع استاتیک و دینامیک علاقه مند هستند.

نوع دینامیک

اگر تاکنون تجربه عملی برنامه نویسی داشته باشید، با مفهوم متغیر آشنا هستید. در زبان های با نوع دینامیک مانند روبی یا جاوا اسکریپت می توان متغیری را بدون این که نوع داده ذخیره شده اش معین باشد، اعلان کرد. این متغیر از نوع دینامیک است و مقدار آن می تواند هر چیزی مانند یک عدد، جمله و یا غیره باشد.

نوع استاتیک

هنگامی که با داده ها در زبان های از نوع استاتیک مانند جاوا سر و کار داریم، باید در زمان اعلان متغیر دقیقاً مشخص کنیم که متغیر چه نوع داده ای را نگهداری خواهد کرد. برای نمونه این متغیر حاوی عدد خواهد بود یا متغیر دیگر متن را ذخیره می کند و متغیر سوم حاوی تاریخ خواهد بود. این بدان معنی است که یک زبان از نوع استاتیک ساختار بیشتری را شامل می شود. برخی از انواع خطاهایی که برنامه نویس ها مرتکب می شوند، توسط ابزارهایی که جاوا در اختیار ما قرار می دهد، حتی پیش از اجرای برنامه، قابل تشخیص هستند. با این حال در صورتی که در زبان های از نوع دینامیکی مانند روبی یا پایتون برنامه نویسی کنید، چنین خطاهایی تا زمان اجرای برنامه و مواجهه با از کار افتادن برنامه قابل تشخیص نخواهند بود.

بدین ترتیب در زبان های برنامه نویسی از نوع استاتیک یک لایه اضافی از کدنویسی وجود دارد که باید انواع همه متغیرها از قبل مورد تفکر قرار گرفته و مشخص شود. بنابراین یادگیری زبان های از نوع دینامیک برای کسانی که اولین زبان برنامه نویسی شان از نوع استاتیک بوده است، بسیار آسان تر از مسیر معکوس آن خواهد بود.

چه زبان هایی از جاوا مشتق شده اند؟

برخی زبان ها مانند اسکالا (Scala) و گرووی (Groovy) وجود دارند که برای اجرا روی JVM طراحی شده اند و یا زبان هایی هستند که برای اجرا روی محیط جاوا توسعه یافته اند. همچنین ممکن است

برخی افراد ادعا کنند که زبان C# تا حدود زیادی تحت تأثیر جاوا توسعه یافته است. زبان سی شارپ مایکروسافت پس از جاوا توسعه یافت و به طور خاص به مقدار زیادی از جاوا الهام گرفته است. سی شارپ موجب برخی بهبودها در جاوا نیز شده است و از این رو این دو زبان به طور متقابل بر هم تأثیرگذار بوده اند.

جاوا چه تفاوتی با جاوا اسکریپت دارد؟

هیچ رابطه فنی بین جاوا و جاوا اسکریپت وجود ندارد. جاوا اسکریپت از سوی نت اسکیپ (Netscape) در دهه 90 میلادی توسعه یافته است و در ابتدا LiveScript نامیده می شد. زمانی که نت اسکیپ دید هیچ کس از LiveScript استفاده نمی کند و جاوا محبوبیت روزافزونی دارد، نام آن را به جاوا اسکریپت تغییر داد تا بتوانند از این موج محبوبیت جاوا بهره مند شوند. در واقع این ایده موفق بود و جاوا اسکریپت نیز محبوب شد؛ اما از منظر فنی هیچ رابطه ای بین این دو وجود ندارد و صرفاً دارای تشابه اسمی هستند. شاید تنها مشابهت فنی بین جاوا و جاوا اسکریپت را در این بدانیم که هر دو آن ها دستور زبانشان را از زبان برنامه نویسی C گرفته اند. به همین دلیل اگر با جاوا آشنا باشید، در این صورت یادگیری جاوا اسکریپت آسان خواهد بود و برعکس.

فریمورک های جاوا

نکته جالب در مورد جاوا این است که یک زبان برنامه نویسی چندمنظوره محسوب می شود و از این رو در محیط های بسیار متفاوتی

مورد استفاده قرار می گیرد. جاوا اصولاً به منظور استفاده روی پلتفرم های مختلف طراحی شده است و بنابراین می توان آن را روی ماشین های لینوکس، یونیکس باکس، مک، ویندوز یا حتی گوشی تلفن همراه اجرا کرد. در واقع می توان گفت که جاوا را می شود همه جا استفاده کرد.

با این که جاوا شاید زبان چندان سرراستی نباشد؛ اما شما مجبور نیستید کدهای جاوا را از صفر بنویسید. فریمورک های عالی زیادی برای جاوا وجود دارند که با آن ها می توان اپلیکیشن های وب، موبایل، میکروسرویس و REST API هایی نوشت که روی ماشین مجازی جاوا اجرا می شوند.

فریمورک های جاوا امکان تمرکز روی منطق تجاری اپلیکیشن به جای نوشتن کارکردهای ابتدایی مانند ایجاد اتصال به پایگاه داده یا مدیریت خطاها را فراهم می سازند. ضمناً اگر تجربه کدنویسی با جاوا را داشته باشید، با استفاده از فریمورک ها می توانید بسیار سریع تر کار برنامه نویسی را آغاز کنید. همه فریمورک ها از دستور زبان یکسانی استفاده می کنند و اصطلاح ها، پارادایم ها و مفاهیم یکسانی از زبان برنامه نویسی جاوا ارائه می کند. در ادامه برخی از محبوب ترین فریمورک های جاوا را معرفی می کنیم:

BLADE : فریمورک ساده اپلیکیشن با کمترین اثرات جانبی

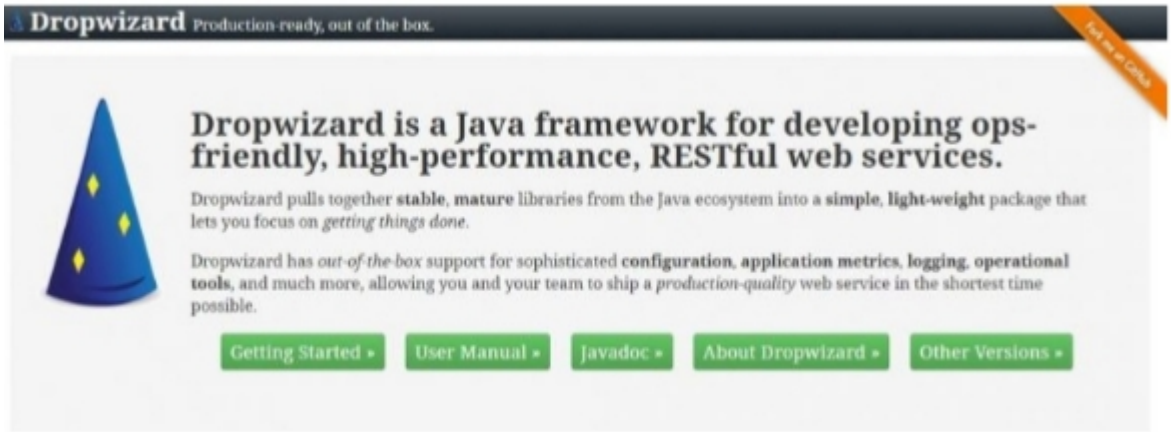
Blade یک فریمورک سبک و با عملکرد بالا برای جاوا است که امکان ساخت اپلیکیشن های وب سریع به روشی سرراست را فراهم می سازد. خالقان این فریمورک هدفشان این بوده است که کاربران آن بتوانند در طی یک روز آن را درک کنند. از این رو blade روی سادگی و ظرافت متمرکز شده است.

فریمورک blade از الگوی طراحی نرم افزار MVC یعنی «مدل-نما-کنترلر» (Model-View-Controller) استفاده می کند. درک طراحی این فریمورک آسان است، چون به هیچ کتابخانه شخص ثالث وابسته نیست و لایه های زیادی را نیز شامل نمی شود. blade بر مبنای جاوا 8 طراحی شده است و یک وب سرور Netty و موتور templete نیز در داخل فریمورک جاسازی شده است. این فریمورک اثرات جانبی کمی دارد و کل کد منبع آن کمتر از 500 کیلوبایت است.

شما با استفاده از blade به رابط مسیریابی به سبک RESTful

دسترسی می یابید و می توانید اپلیکیشن خود را به عنوان یک پروژه maven ابتدایی توزیع کنید. این فریمورک ویژگی های امنیتی داخلی دارد که برای مثال از دفاع CSRF و XSS استفاده می کند.
blade فریمورک کارآمدی است چون از افزونه ها و منابع webjar پشتیبانی می کند. مستندات روی سایت اصلی (https://lets-blade.com) به زبان چینی است. با این حال روی ریپو گیت هاب (https://github.com/lets-blade/blade) مستندات به زبان انگلیسی نیز وجود دارد.

DROPWizard : سرویس های وب RESTful آماده Production



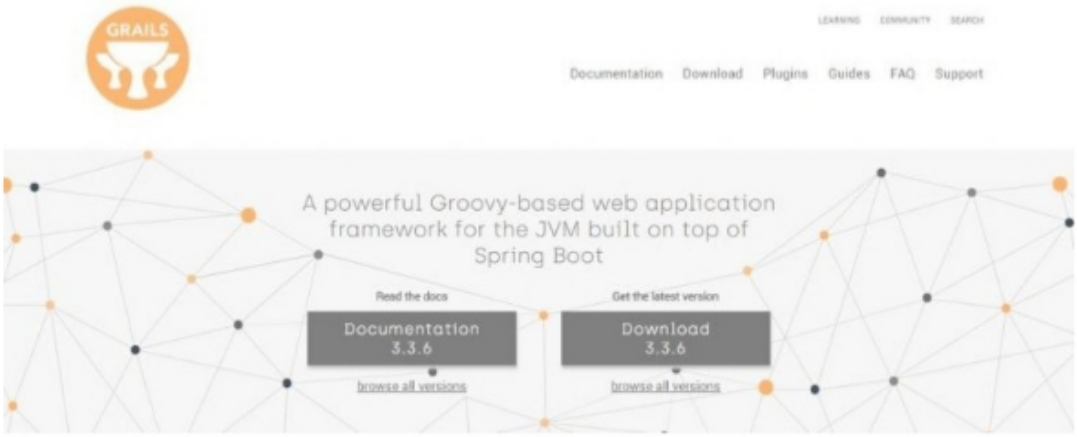
Dropwizard یک فریمورک با عملکرد بالا؛ اما سرراست محسوب می شود که برای توسعه سریع سرویس های وب RESTful مورد استفاده قرار می گیرد. این فریمورک به طور خاص برای ایجاد

میکروسرویس های جاوا مناسب است.

این فریمورک Dropwizard چند کتابخانه جاوا کاملاً تثبیت شده برای ارائه پلتفرمی سریع و بدون مزاحمت را گرد هم آورده است. این فریمورک به همراه یک سرور jetty توکار، Google Guava، Joda Time، Hibernate Validator، Logback و بسیاری کتابخانه های محبوب دیگر جاوا ارائه شده است. به علاوه Dropwizard شامل jersey نیز هست که با آن می توان وب سرویس های RESTful ساخت و Jackson که برای پردازش JSON مورد استفاده قرار می گیرد. Dropwizard را می توان به عنوان یک اکوسیستم مجزا در نظر گرفت که شامل همه وابستگی های اشاره شده بالا است که در یک بسته منفرد گرد هم آمده اند.

اگر می خواهید از Dropwizard استفاده کنید، دیگر نیاز نیست زمان زیادی را صرف کارکردهای ثانویه ای مانند نوشتن کد برای پیکربندی، معیارها، یا گزارش گیری بکنید. به جای آن می توانید روی منطق تجاری اولیه اپلیکیشن تمرکز کنید و به بالاترین بهره وری دست یابید. به همین دلیل است که Dropwizard در اغلب موارد به صورت یک فریمورک جاوای دوستدار عملیات یاد می شود. آغاز کار با آن در صورتی که قبلاً کد جاوا نوشته باشید، دشوار نیست چون مستندات Dropwizard مثال Hello World ساده ای ارائه کرده است که می تواند در گام های نخست به شما کمک کند.

Grails : فریمورک وب اپلیکیشن مبتنی بر Groovy



Grails یک فریمورک وب اپلیکیشن است که از زبان برنامه نویسی Groovy استفاده می کند. Grails یک زبان شی گرا برای پلتفرم جاوا است که به منظور بهبود بهره وری توسعه دهندگان طراحی شده است. دستور زبان آن با جاوا مطابقت دارد و به بایت کد JVM (ماشین مجازی جاوا) کامپایل می شود.

با این که در Grails مجبور هستیم کد خود را به زبان Groovy بنویسیم؛ اما Grails با فناوری های دیگر جاوا مانند کیت توسعه جاوا، کانتینرهای JAVE EE، Hibernate یا Spring سازگار است.
Grails در لایه های زیرین خود بر مبنای SPRING BOOT ساخته شده است و می تواند از ویژگی هایی از قبیل تزریق وابستگی SPRING که به بهره وری کمک می کنند بهره بجوید. شاید بهترین نکته در مورد Grails این است که به لطف زبان Groovy در آن می توان نتایج یکسانی با فریمورک های دیگر را در کدهای کمتری به دست آورد.

Grails از تعدادی از مفاهیم مدرن توسعه نرم افزار پیروی می کند که شامل اصل ترجیح قرارداد بر پیکربندی (convention over configuration)، API opinionated برای الزام به بهترین رویه ها و مقادیر پیش فرض ملموس است. همچنین بسیار محیط مناسبی برای توسعه دهنده دارد، چون به همراه مستندات، راهنمای گام به گام و کتابخانه گسترده افزونه ارائه شده است. شما می توانید افزونه های خودتان را نیز بسازید و از پشتیبانی IDE برای ایکلیپس (Eclipse)، سابلایم (Sublime)، تکست میت (TextMate)، انتلیج (IntelliJ IDEA) و دیگر پلتفرم ها استفاده کنید.

GWT : کیت ابزار وب گوگل – اپلیکیشن های سمت کلاینت جاوا که به وسیله جاوا اسکریپت توزیع می شوند

GWT یا کیت ابزار وب گوگل یک فریمورک وب عالی ایجاد شده از سوی گوگل است. در واقع GWT رویای هر توسعه دهنده ای که می خواهد اپلیکیشن های جاوا را برای وب بسازد برآورده ساخته است، چون امکان نوشتن کدهای سمت کلاینت جاوا و توزیع آن به صورت

جاوا اسکریپت روی مرورگر را فراهم می سازد.

GWT که «گیویت» نیز تلفظ می شود، یک فریمورک پایدار و با نگهداری مناسب جاوا است. برای بیان این پایداری چیزی بهتر از این نیست که این فریمورک در چند مورد از محصول های گوگل مانند Blogger، AdSense، AdWords و Google Wallet مورد استفاده قرار می گیرد. کیت ابزار وب گوگل یک وب سایت خاص است که همه ابزارها و منابعی مانند راهنماها، یک اپلیکیشن اولیه و افزونه ایکلیپس که ممکن است نیاز داشته باشید را ارائه می کند.

نکته جالب در مورد GWT این است که با آن می توانید اپلیکیشن های کاملاً مبتنی بر مرورگر را بدون تخصصی در فناوری های فرانت اند مانند بهینه سازی جاوا اسکریپت یا طراحی واکنش گرا بنویسید. بدین ترتیب می توانید از GWT به جای فریمورک های سمت کلاینت جاوا اسکریپت که گاهی با سرعت وارد بازار شده و با همان سرعت آن را ترک می کنند، استفاده کنید. GWT ویژگی های پیشرفته زیادی مانند بین المللی سازی، پرتابل بودن میان مرورگرهای مختلف، تجرید UI، بوکمارک کردن، و مدیریت تاریخچه دارد.

HIBERNATE : فریمورک نگاشت شی-رابطه ای برای ارتباط بهتر با پایگاه داده

Hibernate یک فریمورک نگاشت شی -رابطه ای است که ارتباط بهتر بین زبان برنامه نویسی جاوا و سیستم های مدیریت پایگاه داده رابطه ای (RDBMS) را ارائه می کند.

هنگامی که با زبان های برنامه نویسی شی گرایی مانند جاوا کار می کنیم، با مشکلی به نام عدم مطابقت امپدانس شی-رابطه ای (که در برخی موارد پارادایم عدم مطابقت نیز نامیده می شود) مواجه می شویم. زبان های شی گرا و RDBMS ها داده ها را به طرز متفاوتی مدیریت می کنند که این امر می تواند منجر به مسائل عدم تطبیق شود. با این که زبان های شی گرا داده ها را به شکل سلسله مراتبی از اشیا سازماندهی می کنند؛ اما پایگاه های داده رابطه ای داده ها را در قالب جدول ارائه می کنند. برای نمونه یکی از مسائل عدم مطابقت هنگامی روی می دهد که مدل شی کلاس هایی بیش از تعداد جدول های موجود در پایگاه های داده رابطه ای داشته باشد.

Hibernate فریمورکی ارائه کرده است که بر مسائل عدم مطابقت جاوا فائق می آید. منظور از ارائه Hibernate دستیابی به ثبات (Persistence) بوده است، یعنی داده هایی که توسط اپلیکیشن تولید یا استفاده می شوند باید عمری بلندتر از پردازشی که آن ها را تولید کرده است داشته باشند. با این که Hibernate برای پایگاه های داده رابطه ای ساخته شده است؛ اما نسخه جدیدتر آن از انبارهای داده ای NoSQL نیز پشتیبانی می کند. همچنین ابزارهای توسعه ای خوبی مانند ویرایشگر نگاشت، کنسول Hibernate و یک ابزار جالب مهندسی معکوس پایگاه داده دارد.

JavaServer Face : فریمورک UI مبتنی بر مولفه

JAVAServer Face یا به اختصار JSF از سوی اوراکل به عنوان یک شاخص برای ساخت رابط های کاربر برای وب اپلیکیشن های جاوا ارائه شده است. این استاندارد رسمی برای ابتکار پردازش جامعه جاوا (JCP) نیز محسوب می شود.

نسخه اول JavaServer Faces در سال 2004 منتشر شده است و از این رو فریمورک کاملاً پایداری محسوب می شود. این فریمورک از الگوی طراحی نرم افزار MVC پشتیبانی می کند و یک معماری مبتنی بر مولفه دارد. با استفاده از JavaServer Faces می توان رابط های

کاربری برای مولفه های قابل استفاده مجدد ساخت، وضعیت مولفه ها را مدیریت کرد، آن ها را به منابع داده متصل ساخت و رویدادهای ایجاد شده از سوی کاربر را به شنونده های رویداد در سمت سرور اتصال داد.

سیستم قالب بندی پیش فرض JSF به صورت Facelet است که به طور مشخص برای این پروژه ساخته شده است. با استفاده از Facelet می توان از XML به جای جاوا برای مدیریت view استفاده کرد. با این حال، می توان view ها را با استفاده از فناوری های دیگر مانند XUL (زبان رابط کاربری XML) و جاوای خالی نیز ساخت. وب اپلیکیشن های ایجاد شده از سوی JavaServer Faces پرتابل هستند و روی سرورهای مختلف اپلیکیشن JAVA EE نیز کار می کنند.

JHiPSter : اپلیکیشن های وب و میکروسرویس ها با Spring Boot و Angular/React

JHipster یک فریمورک نسبتاً جدیدتر (در سال 2013 انتشار یافته) جاوا است که Spring Boot و دو فریمورک محبوب فرانت اند یعنی Angular و React را در یک نرم افزار تولید اپلیکیشن کنار هم گردآورده است. با استفاده از JHipster می توانید به سرعت اپلیکیشن ها و میکروسرویس های مدرن وب مبتنی بر جاوا بسازید.

Spring Boot امکان ایجاد اپلیکیشن های مبتنی بر Spring در رده production را ارائه می کند که با کمترین پیکربندی کار می کنند. JHipster این وضعیت را با انگولار، ری اکت و بوت استرپ (Bootstrap) در سمت کلاینت ترکیب کرده است تا یک معماری فول استک را در اختیار شما قرار دهد. اگر می خواهید ببینید که اپلیکیشن های JHipster در دنیای واقعی چگونه به نظر می رسند برخی از نمونه اپلیکیشن ها برای انگولار و ری اکت را که هر دو از سوی تیم JHipster توسعه یافته اند ملاحظه کنید.

JHipster امکان انتخاب از میان دو سبک معماری را می دهد. نخست می توانید از معماری monolithic استفاده کنید که در آن بخش های فرانت اند و بک اند اپلیکیشن در یک اپلیکیشن منفرد ترکیب شده اند. در روش دوم می توانید از معماری میکروسرویس استفاده کنید که فرانت اند را از بک اند جدا می کند. JHipster شامل چندین ابزار نیز می شود و گزینه های بسیار زیادی برای کدنویسی سمت کلاینت و سرور، بسته بندی، به همراه وظایف مختلف DevOps ارائه کرده است. در نهایت باید گفت که تصادفی نیست که همه برندهای مشهوری مانند HBO، Bosch، Siemens، Adobe و Google از JHipster استفاده می کنند.

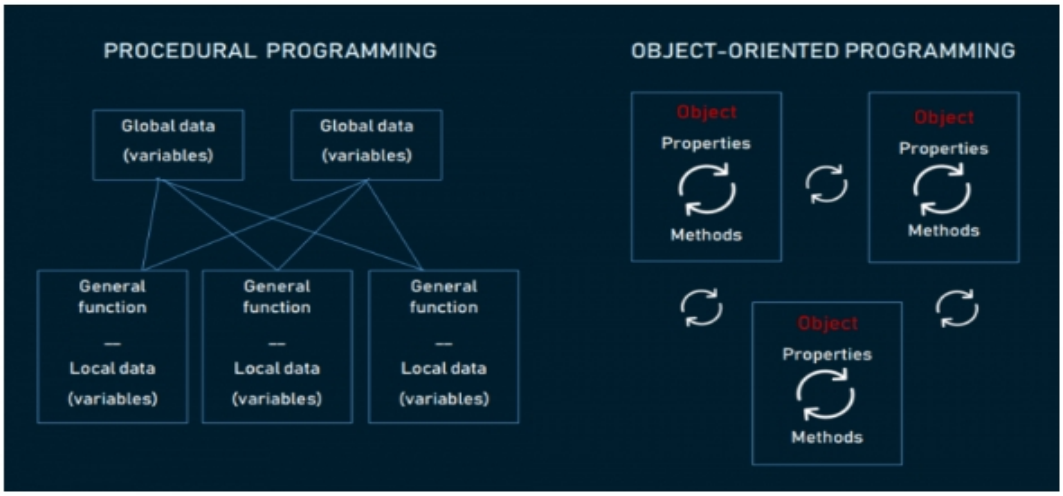
مزیت های جاوا

با این که جاوا دیگر تنها زبانی نیست که برای توسعه اپلیکیشن های اندرویدی استفاده می شود؛ و دیگر به هیچ وجه تنها گزینه برای برنامه نویسی وب محسوب نمی شود؛ اما جاوا همچنان در این عرصه ها حضور دارد. همه این توفیق را نمی توان صرفاً به قدمت آن نسبت داد، فلذا در ادامه برخی از مزیت های جاوا را بررسی می کنیم

برنامه نویسی شی گرا

جاوا، برنامه نویسی شی گرا (Object-oriented) را کاملاً پذیرفته است. منظور از برنامه نویسی شی گرا سبکی از کدنویسی است که در آن نه تنها انواع داده و ساختمان داده تعریف می شوند؛ بلکه مجموعه توابع مورد استفاده این داده ها نیز تعریف می شوند. بدین ترتیب ساختمان داده تبدیل به یک شی می شود که می توان آن را برای ایجاد روابطی بین شی های مختلف دستکاری کرد.

برخلاف رویکرد متضادش یعنی برنامه نویسی رویه ای (procedural programming) که در آن یک توالی از دستورالعمل ها با استفاده از متغیرها و توابع استفاده می شوند؛ در برنامه نویسی شی گرا امکان گروه بندی این متغیرها و توزیع بر اساس زمینه ارائه شده است و از این رو می توان آن ها را بر اساس زمینه هایی که هر شی خاص دارد، نامگذاری کرده و مورد ارجاع قرار داد.



مقایسه برنامه‌نویسی رویه‌ای با برنامه‌نویسی شی‌گرا

چرا شی گرایی یک مزیت محسوب می شود؟

- در این روش می توان شی ها را در برنامه های دیگر به آسانی استفاده کرد

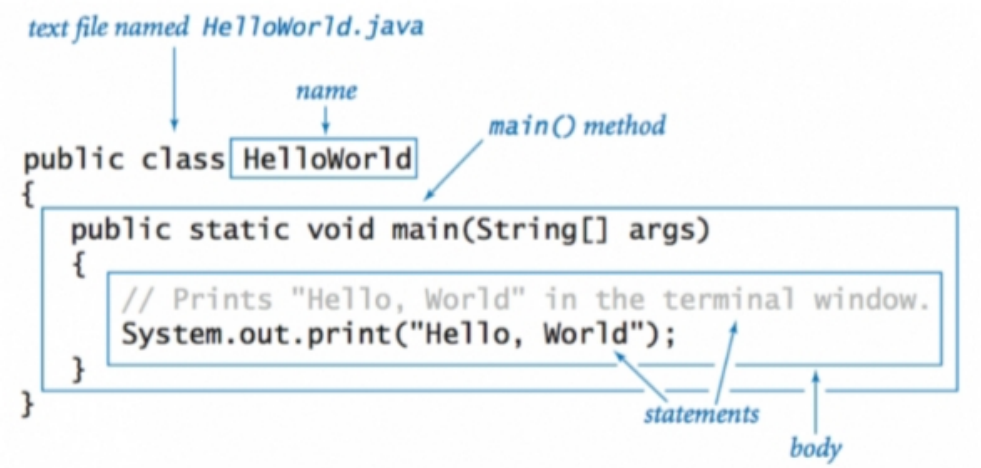
- با واداشتن شی ها به مخفی کردن اطلاعاتی که نباید به آسانی مورد دسترسی قرار گیرند، از برخی خطاها جلوگیری می شود

- در این روش برنامه ها به خصوص برنامه های پیچیده و بزرگ، به روش سازمان یافته تر و از پیش برنامه ریزی شده تری ایجاد می شوند

- در این روش نگهداری کد و نوسازی کدهای قدیمی آسان تر است.

زبان سطح بالا با دستور زبان ساده و یادگیری نسبتاً آسان

جاوا یک زبان سطح بالا است، یعنی شباهت زیادی با زبان انسان دارد. برخلاف زبان های سطح پایین که به کد ماشینی شبیه هستند، زبان های سطح بالا باید با استفاده از کامپایلرها و مفسرها تبدیل شوند. این فرایند ساده سازی موجب می شود که نوشتن، خواندن و نگهداری زبان آسان تر شود.



کد Hello World در زبان جاوا

جاوا دستور زبان (به معنی قواعد و ساختارهای مورد استفاده از سوی برنامه نویسان) خود را از C++ اخذ کرده است و به همین دلیل ساختاری شبیه به کد C دارد. با این وجود بسیار ساده تر است و به افراد مبتدی امکان یادگیری سریع تر فناوری و کدنویسی مؤثرتر برای رسیدن به نتایج مشخص را ارائه می کند.

جاوا ممکن است به قدر پایتون برای افراد مبتدی مطلوب نباشد؛ اما هر

توسعه دهنده ای با درکی مقدماتی از فریمورک ها، بسته ها، کلاس ها و اشیا می تواند الگوی آن را خیلی زود متوجه شود. جاوا سرراست و دارای نوع های کاملاً تعریف شده است که انتظارات کاملاً مشخصی دارد و بدین سبب باعث می شود خیلی زود تفکر شما در مسیر صحیح قرار گیرد. علاوه بر آن راهنماها و دوره های آموزشی رایگان بسیار زیادی روی اینترنت وجود دارد که به افراد مبتدی کمک می کند جاوا را بسیار سریع بیاموزند.

استاندارد برای محاسبات سازمانی

اپلیکیشن های سازمانی (Enterprise) بزرگ ترین دارایی جاوا هستند. این روند به دهه 90 میلادی باز می گردد که سازمان ها شروع به جستجوی ابزارهای برنامه نویسی مستحکمی کردند که از جنس C نباشند. جاوا از کتابخانه های زیادی پشتیبانی می کند که بلوک های سازنده هر سیستم سازمانی هستند و این مسئله به توسعه دهندگان کمک کرد تا هر کارکردی که یک شرکت لازم داشت را با استفاده از جاوا بنویسند. البته افراد با استعداد زیادی که جاوا می دانستند نیز بی تأثیر نبود. جاوا زبانی است که در اغلب مدارس و دانشگاه ها به عنوان یک مقدمه برای برنامه نویسی رایانه ای محسوب می شود. علاوه بر این قابلیت های یکپارچه شدن جاوا بسیار وسیع است، چون اغلب ارائه دهنده های خدمات میزبانی از جاوا پشتیبانی می کنند. نکته آخر این که نگهداری جاوا نسبتاً ارزان است، چون به سخت افزار خاصی وابسته نیست و می تواند روی سرورهایی از هر نوع اجرا شود.

کاهش ریسک های امنیتی

ممکن است شنیده باشید که جاوا زبان برنامه نویسی امنی است؛ اما این گزاره کاملاً صحیح نیست. خود زبان نمی تواند در برابر
آسیب پذیری ها حفاظتی ایجاد کند، بلکه برخی از ویژگی های آن هستند که می توانند شما را در برابر رخنه های امنیتی رایج محافظت کنند.
نخستین ویژگی در قیاس با C این است که جاوا از اشاره گرها استفاده نمی کند. اشاره گر شیئی است که آدرس حافظه یک مقدار دیگر را نگهداری می کند و موجب می شود که دسترسی ناخواسته ای به حافظه ایجاد شود. ویژگی دوم است که جاوا یک ابزار مدیریت امنیت دارد که نوعی سیاست امنیتی برای هر اپلیکیشن است و در آن قواعد دسترسی خاصی تعریف می شود. بدین ترتیب امکان اجرای اپلیکیشن های جاوا درون یک sandbox وجود دارد و بدین ترتیب ریسک آسیب حذف می شود.

عدم وابستگی به پلتفرم (یک بار بنویس، همه جا اجرا کن)

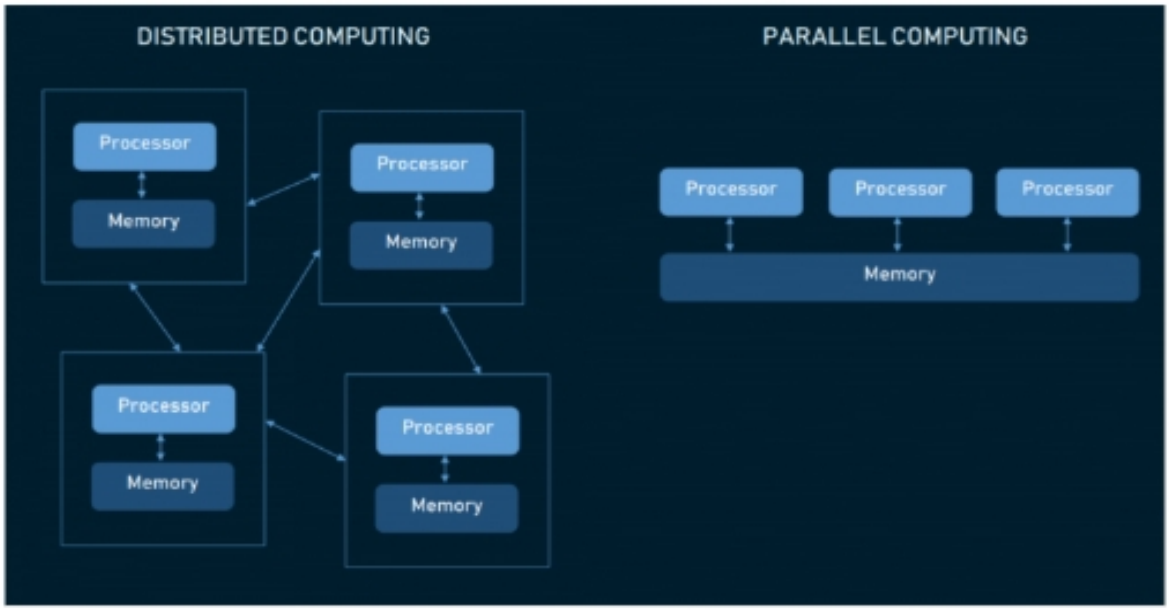
شعار «یک بار بنویس، همه جا اجرا کن» به اختصار WORA یک عبارت مشهور برنامه نویسی است که شرکت سان مایکروسیستمز معرفی کرد تا قابلیت های چند پلتفرمی جاوا را توصیف کند. معنی این شعار آن است که می توان برنامه جاوا را برای مثال روی ویندوز نوشت، آن را به بایت کد تبدیل کرد و این اپلیکیشن را روی هر پلتفرم دیگری که از ماشین مجازی جاوا (JVM) پشتیبانی می کند، اجرا کرد.
در این حالت JVM یک سطح تجرید بین کد و سخت افزار ایجاد می کند.

طرز کار WORA در جاوا

همه سیستم های عامل عمده شامل ویندوز، مک اواس، و لینوکس از JVM پشتیبانی می کنند. به جز در مواردی که برنامه شما تکیه زیادی روی ویژگی های خاص پلتفرم و UI داشته باشد، می توانید اگر چه نه همه؛ اما بخش عمده ای از بایت کد را روی همه این سیستم های عامل به اجرا بگذارید.

زبان توزیع یافته برای همکاری ریموت آسان

جاوا به عنوان یک زبان توزیع یافته طراحی شده است، یعنی سازوکاری درونی برای اشتراک داده ها و برنامه ها میان چند رایانه برای بهبود عملکرد و کارایی دارد.



محاسبات توزیع یافته در برابر محاسبات موازی

برخلاف زبان های دیگر که باید از API های بیرونی برای توزیع استفاده شود، جاوا این فناوری را در هسته خود ارائه کرده است. روش خاص جاوا برای محاسبات توزیع یافته به نام فراخوانی متد از راه دور یا «RMO» (Remote Method Invocation» نامیده می شود. با استفاده از RMI می توان همه مزیت های جاوا مانند امنیت، عدم وابستگی به پلتفرم، و برنامه نویسی شی گرا را در محاسبات توزیع یافته وارد کرد. علاوه بر این جاوا از برنامه نویسی Socket و روش توزیع CORBA برای اشتراک شی ها بین برنامه های نوشته شده به زبان های مختلف، پشتیبانی می کند.

مدیریت خودکار حافظه

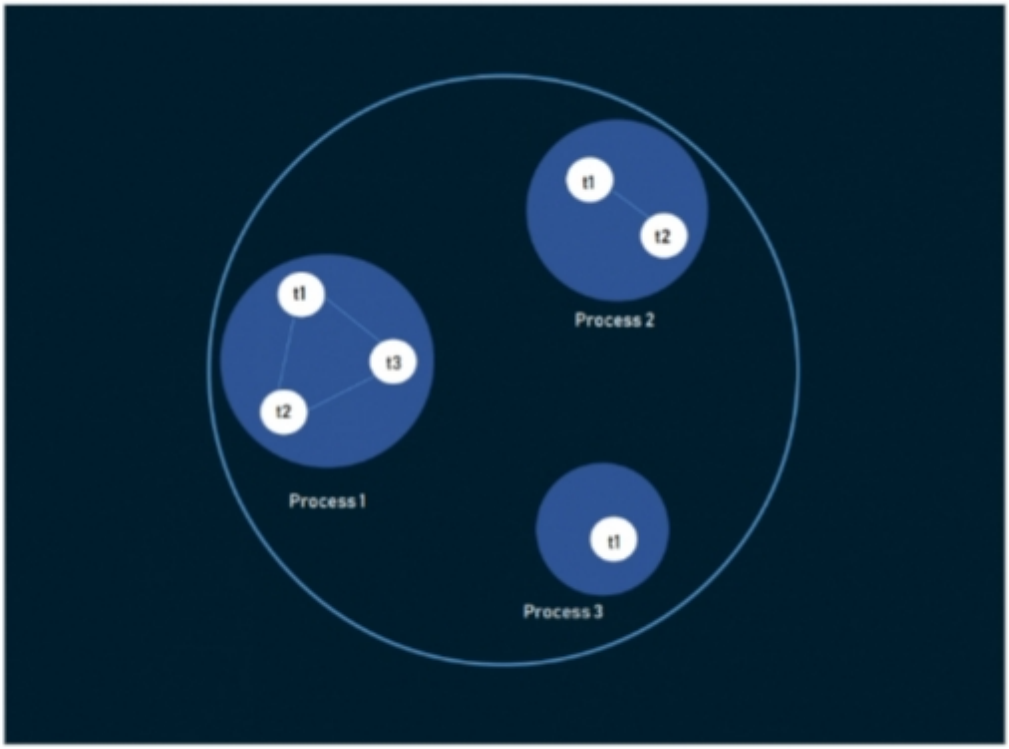
توسعه دهنده های جاوا به لطف مدیریت حافظه خودکار آن (AMM) که در زبان برنامه نویسی سوئیفت (Swift) نیز استفاده می شود، نباید نگران نوشتن دستی کد برای وظایف مدیریت حافظه باشند. ضمناً بازیافت حافظه (garbage collection) اپلیکیشنی است که به طور خودکار تخصیص و آزادسازی حافظه را مدیریت می کند. شاید بپرسید بازیافت حافظه دقیقاً به چه معنا است؟

کارایی یک برنامه به طور مستقیم با حافظه ارتباط دارد و حافظه محدود است. هنگام استفاده از زبان های برنامه نویسی با مدیریت دستی حافظه، توسعه دهندگان با ریسک فراموش کردن تخصیص حافظه مواجه هستند که منجر به افزایش استفاده از حافظه و کند شدن برنامه و سیستم می شود. بازیاب حافظه (garbage collector) می تواند شی هایی که دیگر از سوی برنامه شما ارجاعی ندارند را شناسایی کرده و حذفشان کند. علی رغم این واقعیت که این وضعیت روی جنبه CPU برنامه تأثیر منفی می گذارد؛ اما می توان با بهینه سازی یا تنظیم دقیق این تأثر منفی را کاهش داده یا به کلی از آن جلوگیری کرد.

چندنخی (Multithreading)

نخ (thread) در برنامه نویسی به کوچک ترین واحد پردازشی گفت می شود. جاوا برای بیشینه ساختن بهره برداری از زمان CPU امکان اجرای همزمان نخ ها را می دهد و این فرایند اجرای چند نخی نام دارد.

نخ ها ناحیه مشترکی از حافظه را اشغال می کنند، بنابراین سوئیچ کردن بین آن ها به زمان اندکی نیاز دارد. با این حال نخ ها مستقل از هم هستند و از این رو اگر نخی با خطا مواجه شود، روی نخ های دیگر تأثیری ندارد. این وضعیت به طور خاص در مورد بازی ها و برنامه هایی که انیمیشن های سنگین دارند، مفید است.



مثالی از چند نخی

ثبات و جامعه عظیم

جاوا به لطف جامعه بزرگ پشتیبانی کننده، حمایت اوراکل و همچنین نعمت اپلیکیشن ها و زبان هایی که همچنان روی JVM اجرا می شوند چنین عمر درازی یافته است. علاوه بر این، نسخه های جدیدتر جاوا با

ویژگی های جدید و جذاب به طور منظمی منتشر می شوند.

جامعه توسعه دهنده های جاوا نیز بی همتا است. در حدود 45 درصد از افرادی که در پیمایش سال 2018 استک اورفلو شرکت کرده اند از جاوا استفاده می کنند. جاوا اکوسیستم بزرگی از کتابخانه ها و فریمورک های کاملاً تست شده برای هر نوع کاربردی دارد. به احتمال بالا جاوا نخستین زبانی است که اغلب توسعه دهندگان می آموزند، چون بیش از 1000 دوره آموزشی مرتبط با جاوا روی Udemy و بیش از 300 مورد روی Coursera وجود دارد.

معایب برنامه نویسی جاوا

اگر همه مزایای جاوا را تا اینجا خوانده باشید، شاید مشتاقانه قصد دارید در پروژه بعدی خود از زبان برنامه نویسی جاوا استفاده کنید؛ اما از آنجا که هیچ گلی بی خار نیست، باید گفت که جاوا نیز معایبی دارد که در ادامه به برخی از آن ها اشاره کرده ایم.

لایسنس تجاری پولی

شرکت اوراکل اخیراً اعلام کرده است که از ابتدای سال 2019 استفاده از نسخه JAVE SE 8 در محیط های تجاری، کسب و کار یا production نیازمند پرداخت هزینه خواهد بود. بدین ترتیب برای دریافت همه اصلاحیه های باگ و به روزرسانی ها باید بر اساس تعداد کاربران یا تعداد پردازنده ها هزینه ای پرداخت کنید.

امروزه نسخه کنونی جاوا رایگان است و امکان بازتوزیع آن برای محاسبات با مقاصد عمومی وجود دارد. اما در جهت آماده سازی برای طرح پولی، هر شرکت باید میزان استفاده از جاوا را ارزیابی کرده و در صورتی که ارتقا به نسخه پولی مقرون به صرفه نباشد، به دنبال فناوری های جایگزین برود.

عملکرد پایین

هر زبان سطح بالایی خواه ناخواه به دلیل مسئله کامپایل و سطح تجرید به یک ماشین مجازی باید با مشکل عملکرد پایین دست و پنجه نرم کند. با این وجود، این تنها دلیل انتقاد عمومی از سرعت پایین جاوا نیست. ابزار بازیاب حافظه (garbage collector) جاوا یک ویژگی مفید است؛ اما متأسفانه در صورتی که بیش از 20 درصد زمان CPU را اشغال کند، می تواند منجر به مسائل مهم عملکردی بشود. پیکربندی گش بد نیز می تواند منجر به استفاده بیش از حد از حافظه و کاربرد زیاد ابزار بازیابی حافظه شود. همچنین برخی بن بست های نخ وجود دارند که وقتی چند نخ سعی می کنند به منابع یکسانی دسترسی یابند رخ می دهند و در این وضعیت کابوس هر توسعه دهنده جاوا یعنی خطاهای «خارج از حافظه» (out-of-memory) رخ می دهند. با این که هر یک از این مسائل با برنامه ریزی ماهرانه قابل پیشگیری هستند؛ اما در برخی موارد روی هم جمع می شوند و آسیب های مختلفی ایجاد می کنند.

جاوا روی دسکتاپ ظاهر و شمایل یکسانی ندارد

توسعه دهندگان برای ایجاد رابط گرافیکی برنامه (GUI) از ابزارهای مختلفی که خاص هر زبان استفاده می کنند. بدین ترتیب برای اپلیکیشن های اندروید نرم افزار اندروید استودیو وجود دارد که به ایجاد اپلیکیشن هایی که حس و ظاهر بومی دارند کمک می کند. با این حال وقتی در مورد UI اپلیکیشن های دسکتاپ صحبت می کنیم، جاوا فاقد شکل و شمایل شایان ذکری است.

چند سازنده UI مانند JSF، JavaFX، SWT، Swing وجود دارند که برنامه نویس های جاوا می توانند از میان آن ها انتخاب کنند و این مورد آخر از همه محبوب تر است. Swing یک سازنده GUI قدیمی؛ اما مطمئن چند پلتفرمی و ادغام شده در چندین IDE شامل ایکلیپس و NetBeans است. اما به جز در موارد استفاده از قالب های از پیش ساخته مواردی از ناسازگاری در رابط مشاهده خواهید کرد.
SWT از مولفه های بومی استفاده می کند؛ اما برای UI های پیچیده مناسب نیست. JAVAFX ظاهر تمیز و مدرنی دارد؛ اما چندان به بلوغ نرسیده است. در مجموع انتخاب یک رویکرد کاملاً مناسب برای ساخت GUI اپلیکیشن نیازمند تحقیقات بیشتری است.

کد طولانی و پیچیده

منظور ما از کد طولانی این است که کدهای جاوا از کلمات زیادی استفاده می کنند. با این که این وضعیت هنگام تلاش برای درک زبان برنامه نویسی، شاید یک مزیت به حساب بیاید؛ اما جمله های طولانی و بسیار پیچیده باعث می شوند که کد خوانایی کمتری داشته باشد و نتوان به سادگی آن را اسکن کرد. زبان های سطح بالای زیادی در تلاش

برای تقلید از زبان انگلیسی موجب شده اند که شلوغی زیادی در کد ایجاد شود. جاوا برای کاهش کد غیر قابل درک C++ ایجاد شده تا برنامه نویسان را وادارد دقیقاً آنچه در نظر دارند را تایپ کنند. همین نکته باعث شده است که این زبان شفافیت بیشتری برای افراد غیر خبره داشته باشد؛ اما از سوی دیگر متأسفانه فشردگی آن کاهش یافته است.

اگر جاوا را با رقیبش پایتون مقایسه کنیم، می بینیم که کد پایتون نیاز به نقطه ویرگول انتهایی ندارد و از «or»، «and» و «not» به جای عملگرهای «&&»، «||» و «!» در جاوا به عنوان عملگر استفاده می کند. به طور کلی پایتون تشریفات کمتری مانند پرانتز و آکولاد دارد.

کد جاوا در برابر کد پایتون

جاوا کجا استفاده می شود؟

اغلب سازمان ها از جاوا به روش های مختلف بهره می گیرند. گستره وسیع استفاده از جاوا باعث شده است که این کاربردها تقریباً از چشم

پنهان بمانند و به همین دلیل غالباً پرسیده می شود که از جاوا در کجا می توان استفاده کرد. در ادامه برخی از زمینه های کاربرد جاوا را فهرست کرده ایم:

اپلیکیشن های اندرویدی

با وجود رشد گسترده کاتلین (Kotlin)؛ جاوا همچنان زبان پیش فرض برای اپلیکیشن های اندرویدی محسوب می شود که به طور خودکار جمع عظیمی از توسعه دهندگان جاوا را به برنامه نویسان اندروید تبدیل کرده است. با این که اندروید از Android SDK به جای JDK استفاده می کند؛ اما کد آن همچنان به صورت جاوا نوشته می شود.

محصولات نرم افزاری

علاوه بر Hadoop و Apache Storm، جاوا برای ایجاد Eclipse، Atlassian، Gmail، OpenOffice و بسیاری از موارد دیگر مورد استفاده قرار گرفته است.

برنامه های مالی

جاوا با توجه به این که یکی از پر تقاضاترین مهارت های زبان در بخش مالی محسوب می شود، هم در سمت سرور و هم کلاینت برای ساخت وب سایت های مطمئن، سریع و ساده مورد استفاده قرار می گیرد. از جاوا به عنوان زبان برنامه نویسی مناسب برای شبیه سازی و مدلسازی داده ها نیز یاد می شود.

سیستم های نقطه فروش

بسیاری از کسب و کارها از جاوا برای ایجاد سیستم های PoS استفاده می کنند، زیرا چنین سیستم هایی معمولاً نیازمند عدم وابستگی به پلتفرم و منابع گسترده هستند.

اپلیکیشن های تجاری

Murex یک برنامه مدیریت بانکی محبوب برای اتصال فرانت و بک اند است که به زبان جاوا نوشته شده است.

برنامه های کلان داده

HadOPP به زبان جاوا نوشته شده است. Kafka، Scala و Spark از JVM استفاده می کنند. ضمناً جاوا امکان دسترسی به هزاران کتابخانه، دیباگر و ابزارهای نظارتی کاملاً تست شده را فراهم ساخته است.

موقعیت های شغلی برای برنامه نویسان جاوا

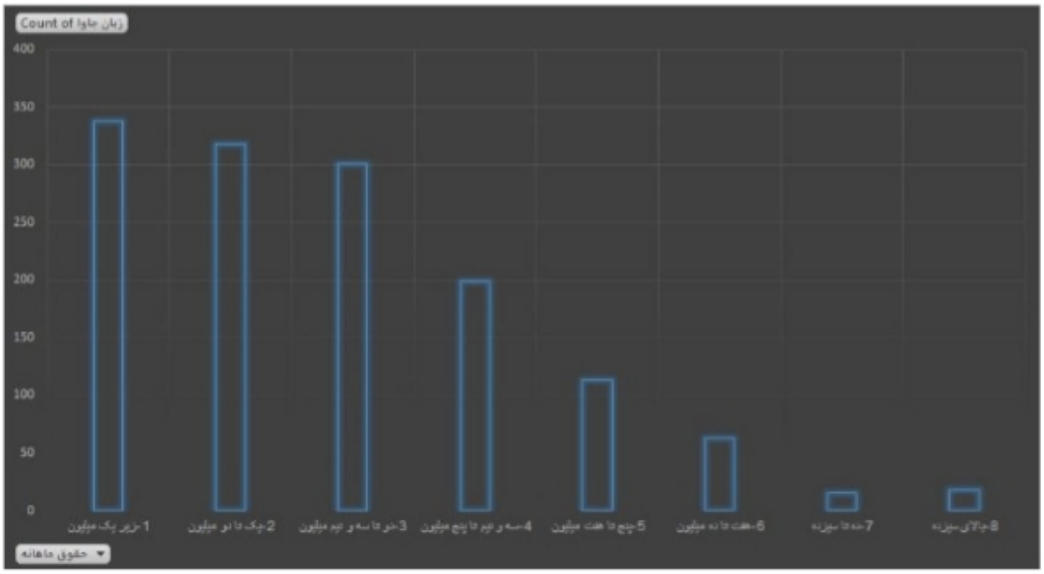
یک اصل مهم در مورد یادگیری زبان های برنامه نویسی وجود دارد که هرگز نباید فراموش کرد، و آن این است که یادگیری یک زبان برنامه نویسی خاص مانند جاوا، صرفاً یادگیری آن زبان نیست؛ بلکه بدین ترتیب شما با همه زبان های دیگر برنامه نویسی نیز آشنا

می شوید. بنابراین جاوا به عنوان یک زبان با مشخصاتی که در بخش های پیشین اشاره کردیم یک زبان خوب برای شروع یادگیری برنامه نویسی محسوب می شود. کسانی که جاوا را به خوبی یاد می گیرند، می توانند هر زبان دیگری شامل C#، پایتون و حتی روبی را نیز به سادگی بیاموزند. بنابراین باید این نکته را به خاطر داشته باشید که یادگیری جاوا به عنوان زبان برنامه نویسی شماره یک دنیا در طی قریب به دو دهه اخیر ضرورتی فراتر از یافتن شغل مرتبط دارد.

برنامه نویسان جاوا به طور عمده از جاوا برای طراحی اپلیکیشن ها و وب سایت هایی استفاده می کنند که اجزای دینامیک دارند. برخی از آن ها روی اپلیکیشن ها و برخی دیگر روی وب سایت ها کار می کنند اما در هر صورت اغلب توسعه دهندگان جاوا باید با مفهوم پروژه بودن کارشان آشنا باشند.

وضعیت درآمد

بر اساس آخرین پیمایش وب سایت معتبر StackOverflow از میان 64000 برنامه نویس در سال 2018 میانگین حقوق کاربرانی که به زبان جاوا برنامه نویسی می کنند در سرتاسر دنیا برابر با 52000 دلار/سالانه بوده است. همچنین بر اساس پیمایش سال 1396 وب سایت jadi.net از میان 1950 برنامه نویس ایرانی، 1365 نفر اعلام کرده اند که به زبان جاوا نیز برنامه نویسی می کنند. بدین ترتیب جاوا با 70% محبوبیت، در میان توسعه دهندگان ایرانی نیز محبوب ترین زبان محسوب می شود. وضعیت درآمد اعلامی از سوی برنامه نویسان ایرانی جاوا نیز مطابق تصویر زیر بوده است:



میانگین وزنی دسته‌های مختلف نشان می‌دهد که میانگین حقوق ماهانه برنامه‌نویسان جاوا در ایران در سال ۱۳۹۶ برابر با ۲،۹۳۲،۰۰۰ تومان بوده است.

جمع بندی زبان برنامه نویسی جاوا

در این بخش برخی از انتقادهایی که معمولاً به زبان جاوا وارد می شود را مطرح کرده و آن ها را بررسی می کنیم.

جاوا قدیمی است

برخی بر این باور هستند که جاوا قدیمی است و کم کم از روند زبان های محبوب خارج می شود. جاوا در سال 1995 متولد شده است و شاید اینک حتی از برخی از برنامه نویسانش نیز سن بیشتری داشته باشد؛ اما زبان های بسیار قدیمی تری نیز وجود دارند که همچنان در

صدر زبان های محبوب برنامه نویسی هستند. واقعیت این است که زبان برنامه نویسی جاوا همچنان در صدر لیست معتبر Tiobe قرار دارد و به نظر نمی رسد به این زودی ها هم آنجا را ترک کند و در واقع به طور مداوم نیز بر محبوبیت آن افزوده می شود. البته زبان های دیگری هستند که میزان رشد محبوبیت بالاتری از جاوا دارند؛ ولی این نکته که جاوا در حال خروج از رده زبان های محبوب است واقعیت ندارد.

زبان های JVM بهتری مانند Clojure، Scala و Kotlin وجود دارند

این نکته جالبی است و با توجه به این که کاتلین به سرعت در حال رشد است، شاید برای بسیاری از افراد این سؤال را مطرح کند که آیا باید شروع به یادگیری جاوا بکنند یا کاتلین را بیاموزند؟

پاسخ این است که یادگیری جاوا در صورتی که بخواهید یک توسعه دهنده JVM باشید، یک ضرورت محسوب می شود. البته شما می توانید زبان های دیگری که از JVM استفاده می کنند را نیز بیاموزید؛ اما در صورتی که جاوا را یاد نگرفته باشید، چیز مهمی را از دست داده اید. اغلب این زبان ها به طور گسترده ای به کتابخانه های جاوا اتکا دارند و عدم یادگیری جاوا به مهارت های ما لطمه خواهد زد.

دانستن جاوا یک مزیت عمده محسوب می شود و پایه ای برای زبان های دیگر محسوب می شود. در واقع JVM با وجود زبان هایی مانند Clojure، Scala، Groovy و Kotlin بسیار غنی تر شده است؛

اما همه آن ها علاوه بر JVM، به نوعی از جاوا الهام گرفته اند یا با آن رابطه دارند.

بررسی زبان های JVM دیگر به جز جاوا بسیار مناسب است، چون موجب بروز و ظهور ابداع ها و نوآوری ها می شود؛ اما این دلیل نمی شود که کسی یادگیری جاوا را کنار بگذارد، چون جاوا نقش بنیادینی برای همه این زبان ها دارد و یادگیری آن سرمایه گذاری ارزشمندی محسوب می شود.

آیا NodeJS برای برنامه نویسان فرانت اند بهتر نیست؟

این سؤال را می توان به همه توسعه دهنده های فرانت اند تعمیم داد و پرسید آیا ضرورتی برای یادگیری یک زبان سمت سرور مانند جاوا برای آن ها وجود دارد؟

NodeJs بسیار عملی و محبوب است و با آن می توان سرویس ها را با سرعت و به طرز کارآمدی ایجاد کرد. با این حال جاوا در سمت سرور گزینه تثبیت شده تری محسوب می شود.

پاسخ به سؤال فوق تا حدودی زیادی به شخص برنامه نویس بستگی دارد که آیا می خواهد برای همیشه یک توسعه دهنده فرانت اند بماند و یا ترجیح می دهد با برنامه نویسی سمت سرور نیز آشنا شود. کدهای سمت سرور زیادی با جاوا نوشته شده اند و حتی اگر کسی قصد نداشته باشد خودش کد جاوا بنویسد؛ اگر به یادگیری جاوا نپردازد از خواندن این کدها نیز محروم خواهد شد.

اگر شما روی یک پروژه NodeJS با استفاده از جاوا اسکریپت هم در سمت سرور و هم کلاینت کار می کنید، یادگیری جاوا برای شما بسیار مناسب خواهد بود و یک سرمایه گذاری ارزشمند برای آینده محسوب می شود.

کار با جاوا راحت نیست

نسخه Enterprise جاوا به دلیل استفاده از XML برای پیکربندی bean شهرتی منفی کسب کرد و این شهرت منفی سال ها با آن همراه بوده است؛ گرچه نکته فوق دیگر صحت ندارد.

امروزه فریمورک های زیادی برای نوشتن پروژه های جاوا معرفی شده اند که ما نیز در بخش های پیشین این نوشته برخی از آن ها را معرفی کردیم. بدین ترتیب نوشتن کدهای جاوا دیگر کار دشواری محسوب نمی شود. برای مثال در قطعه کد زیر برنامه Hello World را با استفاده از Spark JAVA نوشته ایم:

```
1 import static spark.Spark.*;
2 public class HelloWorld {
3 public static void main(String[]
4 args) {
5 get("/hello", (req, res) -> "Hello
6 World");
  }
  }
```

احتمالاً شما نیز بر این باور هستید که کد فوق به هیچ وجه ناخوشایند نیست و تا حدود زیادی جالب هم به نظر می رسد. با استفاده از Spring Boot این کد می تواند از این هم جذاب تر نوشته شود. نکته دیگری که باید اشاره کرد این است که جاوا ابزارهای با کیفیت، مواد آموزشی و پشتیبانی زیادی دارد که باعث می شود حل بسیاری از این مسائل در جاوا کار آسانی باشد.

جاوا خیلی کُند است و حافظه زیادی مصرف می کند

جاوا روی ماشین مجازی جاوا (JVM) اجرا می شود و از این رو زمان راه اندازی کندتری دارد. بدیهی است که در قیاس با برنامه نوشته شده به زبان C که ماهیتی شبیه به bash دارد، زمانی که برای شروع به کار JVM مورد نیاز است بسیار طولانی محسوب می شود. مسلم است که اپلیکیشن های بسیار کوچک و سبک که به زبان بومی پلتفرم نوشته شده اند در رقابت با JVM برنده خواهند بود. اما آیا این دلیلی بر عدم استفاده از JVM است؟ اگر هم چنین باشد، فقط در همین مواردی که اشاره کردیم خواهد بود.

سؤال دیگر این است که پس جاوا برای چه چیزی مناسب است؟ و آیا این روزها سریع تر شده است؟

جاوا در فضای کلان داده استفاده گسترده ای دارد. برای نمونه در ابزارهایی مانند Apache Hadoop که در عمل به زبان جاوا نوشته شده است، مورد استفاده قرار می گیرد.سازمان های بزرگ مالی و بانک های بزرگ دنیا از جاوا برای برنامه های بک اند خود بهره می گیرند.جاوا در اپلیکیشن های تجاری که کارکردهایی با بسامد بالا

دارند، استفاده می شود و در این حوزه عملکردی مشابه C++ دارد.جاوا به طور گسترده روی دستگاه های اندرویدی استفاده می شود.جاوا در فضای دستگاه های مختلف استفاده گسترده ای دارد.البته اگر بخواهید بازی های ویدئویی بنویسید، شاید جاوا گزینه مناسبی نباشد؛ در واقع جاوا آن قدر کاربردهای زیادی دارد که شاید عملکرد اصلاً در اولویت اهمیت نباشد

چرا باید به جای هر زبان برنامه نویسی دیگری جاوا آموخت؟

جاوا زبانی شگفت انگیز است. جاوا به عنوان محبوب ترین زبان برنامه نویسی دنیا در این لحظه یکی از مهارت های اصلی برای توسعه نرم افزار محسوب می شود.

البته شما مجبور نیستید به جای هر زبان دیگری جاوا بیاموزید. برای اغلب افراد تبدیل شدن به یک برنامه نویس چه به صورت سرگرمی و چه حرفه ای ماه ها زمان می برد. بنابراین نباید خود را به جاوا محدود کنید. اما عدم یادگیری جاوا باعث می شود از جامعه عظیم و دینامیک آن محروم شوید.

از طرف دیگر فرایند توسعه جاوا نیز سرعت بیشتری یافته است و انتشار نسخه های اصلی از سالی یک بار به سالی دو بار افزایش یافته است. در نهایت باید اشاره کرد جاوا امروزه بی دلیل به عنوان محبوب ترین زبان برنامه نویسی دنیا مطرح نشده است. دلیل این محبوبیت، سادگی، مدرن بودن، سریع بودن و فرایند تکامل مداوم آن است. همچنین کتابخانه های بی شماری هستند که به نوشتن کدهای جذاب کمک می کنند و دسترسی گسترده ای به منابع آموزشی و کمکی

وجود دارد. در ادامه برخی از دوره هایی که برای یادگیری جاوا می توانید مورد استفاده قرار دهید معرفی می کنیم.

پایان