بدون اغراق زبان برنامه نویسی جاوا به منزله یکی از معروف ترین زبان های برنامه نویسی در دنیا است چرا که از این زبان برنامه نویسی علاوه بر ساخت برنامه های کامپیوتری Enterprise یا **تجاری**، در ساخت اپلیکیشن ها و بازی های ابزارهای هوشمند اندرویدی و همچنین برنامه نویسی برای وب سایت های تعاملی مثل وب سایت بانک ها و موسسات بزرگ نیز استفاده می شود. در حقیقیت یکی از علل میزان محبوبیت این زبان برنامه نویسی در میان هزاران برنامه نویس حرفه ای در سراسر جهان این است که این زبان برنامه نویسی غیرممکن ها را ممکن ساخته است. هیچ جای شگفتی نیست که آشنایی با زبان برنامه نویسی سیستم عامل محبوبی همچون اندروید که بیش از ۸۰ درصد سهم بازار جهانی را در میان تلفن های همراه و تبلت ها به خود اختصاص داده است می تواند امید بخش آینده مالی خوبی نیز برای برنامه نویسان این حوزه باشد.

زبان برنامه نویسی جاوا زبان قدرتمندی است و برای کسانی که از پیشینه برنامه نویسی با زبان های C و ++C برخوردار باشند، منحنی یادگیری پرشیبی نخواهد داشت. زبان جاوا یک زبان OOP یا **شیئ گرا** است و همین مسئله برنامه نویسی با زبان جاوا را بسیار اثربخش می سازد. یکی دیگر از ویژگی های منحصر به فرد جاوا، مدیریت حافظه است که حافظه تخصیص داده شده را مدیریت کرده و به موقع آزاد می کند تا این حافظه ی آزاد شده بتواند توسط سایر بخش های سیستم مورد استفاده قرار گیرد. از دیگر مزایای این زبان برنامه نویسی می توان به امنیت بالای برنامه هایی اشاره کرد که با جاوا نوشته می شوند چرا که جاوا با ایجاد یک Firewall مابین سیستم و شبکه، از حمله ویروس ها به برنامه ی شما جلوگیری به عمل خواهد آورد. برنامه های نوشته شده توسط جاوا اصطلاحا Multi Platform هستند به این معنی که روی هر سیستمی که جاوا رویش نصب باشد -از ویندوز گرفته تا مکینتاش و گنو/لینوکس و ...- قابل اجرا هستند.

در این سری از آموزش های زبان برنامه نویسی جاوا، حداکثر تلاش صورت گرفته است تا مطالب آموزشی برای گروهی از مخاطبین تالیف شوند که تاکنون حتی یک خط کد هم در زندگی خود ننوشته اند! به عبارت دیگر، این امید را به دانشجویان این دوره می توان داد که چنانچه یک رایانه شخصی، کمی زمان آزاد اما در عین حال انگیزه بسیار زیادی داشته باشند قادر گردند پس از مطالعه ی کلیه آموزش ها، اصول کار با زبان برنامه نویسی قدرتمند جاوا را به خوبی فرا گیرند.

**تاریخچه زبان برنامه نویسی جاوا**

زبان برنامه نویسی جاوا در اوایل دهه ی ۱۹۹۰ میلادی توسط جیمز گاسلینگ و همکارانش در شرکت سان مایکروسیستمز پایه ریزی شد که در ابتدا Oak به معنی «**بلوط»** نام داشت. در حقیقت آقای گاسلینگ از اصول برنامه نویسی در زبان ++C خیلی احساس رضایت نمی کرد و یک سری نارسایی های این زبان برنامه نویسی موجب شد که وی زبان جاوا را بر مبنای زبان ++C طراحی کند به نحوی که بتواند ایده های مد نظر او را به نحو بهتری اجرا کند. از همان ابتدا شعار آقای گاسلینگ و سایر توسعه دهندگان این زبان برنامه نویسی این بود که «**یک بار بنویس، همه جا اجراش کن!**» و همین شعار بود که توانست انقلابی در دنیای برنامه نویسی ایجاد کند.

داستان های زیادی در ارتباط با وجه تسمیه ی زبان جاوا وجود دارد که یکی از آنها به این داستان اشاره دارد که جیمز گاسلینگ در حین طراحی این زبان برنامه نویسی جدید، اتاق کارش مشرف به یک درخت بلوط بود که این درخت همواره نظر وی را به خود جلب می کرد. اما پس از مدتی مدیران شرکت سان مایکروسیستمز تصمیم گرفتند نام بهتری برای این زبان برنامه نویسی جدید انتخاب نمایند از این رو توسعه دهندگان این زبان به کافی شاپی در نزدیکی شرکت رفتند تا برای انتخاب نام جدید با یکدیگر به توافق برسند. پس از نیم روزی بحث و تبادل نظر، نام Java که مخفف نام های James Gosling و Arthur Van Hoff و همچنین Andy Bechtolsheim بود انتخاب شد و از آنجا که مراسم نام گذاری در یک کافی شاپ صورت گرفته بود نماد این زبان **یک فنجان قهوه داغ** در نظر گرفته شد. در عین حال، معتبر‌ترین داستان این است که جیمز گاسلینگ به نوعی قهوه علاقه داشت که در جزیره‌ای به نام جاوا که در اندونزی در جنوب شرقی آسیا است می روید.

به هر حال، این زبان برنامه نویسی موفق که در سال ۱۹۹۵ به طور رسمی به بازار معرفی شد، توانست چنان محبوبیتی در جهان پیدا کند که در حال حاضر در بیش از ۳ میلیارد سیستم مورد استفاده قرار گرفته و تاکنون بیش از ۱۰۰۰ جلد کتاب پیرامون آن به رشته تحریر درآمده است. این زبان داری ساختار یا Syntax مشابه زبان های C و ++C می باشد اما از سادگی بیشتری برخوردار است.

ایده‌ی اولیه‌ی ساخت جاوا بر این بود که این زبان به گونه‌ای ساخته شود که پیچیدگی‌های زبان مطرح سی پلاس پلاس را نداشته باشد. یعنی جاوا زبانی شیء گرا است (OOP) اما مدل شیء گرایی ساده‌تری نسبت به سی پلاس پلاس دارد. به عنوان مثال در جاوا ارث بری چندگانه به شکلی که در ++C قابل پیاده‌سازی است، وجود ندارد و یا جاوا دارای Garbage Collector است که مدیریت حافظه‌ی هیپ را بر عهده دارد. همچنین این زبان مستقل از سخت افزار و نرم‌افزار است. از آنجایی که شعار جاوا «یکبار بنویس و همه جا اجرا کن» بود، یکباره جاوا به یکی از محبوب‌ترین زبان های برنامه نویسی جهان تبدیل شد. در نهایت هم شرکت سان مایکروسیستمز در سال ۲۰۱۰ توسط شرکت بزرگ اوراکل خریداری شد و از آن زمان تاکنون، توسعه ی نسخه های جدید این زبان برنامه نویسی به عهده ی شرکت اوراکل است.

**ابزارهای برنامه نویسی با زبان جاوا**

محیط های توسعه ی یکپارچه برنامه نویسی یا IDE های زیادی همچون  Eclipse، IntelliJ IDEA و NetBeans هستند که به منظور نوشتن برنامه های جاوا و دیگر زبان های برنامه نویسی می توان از آن ها استفاده کرد. در این سری از آموزش ها از محیط برنامه نویسی قدرتمند Eclipse استفاده می کنیم که علاقمندان می توانند به صورت رایگان آن از سایت رسمی[Eclipse](http://www.eclipse.org/downloads/" \t "_blank) بسته به نوع سیستم عامل خود دانلود کنند. یکی از دلایلی که شروع برنامه نویسی جاوا را با محیط برنامه نویسی اکلیپس آغاز خواهیم کرد این است که چنانچه بخواهیم در آینده به برنامه نویسی اندروید بپردازیم با هیچ گونه مشکلی مواجه نخواهیم شد زیرا با توجه به این که این نرم افزار رایگان است، شرکت گوگل هم شرایط نصب SDK اندروید را روی این نرم افزار فراهم نموده و ما برای توسعه ی اپلیکیشن های اندروید با هیچ گونه مشکلی مواجه نخواهیم شد.

در واقع برنامه Eclipse نیازی به نصب نداشته و صرفا پس از خارج کردن فایل دانلود شده از حالت زیپ با کلیک کردن بر روی آیکان eclipse.exe می توانید آن را اجرا نمایید. به منظور آماده سازی محیط برنامه نویسی جاوا شما حتماً به JRE و JDK نیاز دارید که می توانید آخرین نسخه آن ها را از سایت شرکت [Oracle](http://www.oracle.com/technetwork/java/javase/downloads/index.html) دریافت نمایید. (لازم به ذکر است چنانچه شما نسخه JDK نصب نمایید دیگر نیازی به نصب JRE نخواهید داشت چرا که JDK در برگیرنده JRE نیز می باشد.)

پس از اجرای برنامه اکلیپس از شما درخواست می شود تا مسیری را مشخص نمایید تا پروژه های جاوایی که می نویسید را در آن ذخیره سازد. این مسیر به دلخواه شما می تواند هرجایی رو سیستم شما باشد. پس از مشخص کردن مسیر، گزینه OK را انتخاب نموده و این مرحله از کار به پایان می رسد. از آنجا که این اولین باری است که شما نرم افزار را اجرا می کنید، یک صفحه خوش آمد گویی یا Welcome Screen نمایان خواهد شد. پس از نگه داشتن نشانگر موس خود روی تک تک گزینه ها، گزینه مرتبط با Workbench یا صفحه اصلی نرم افزار را پیدا نموده و روی آن کلیک نمایید.

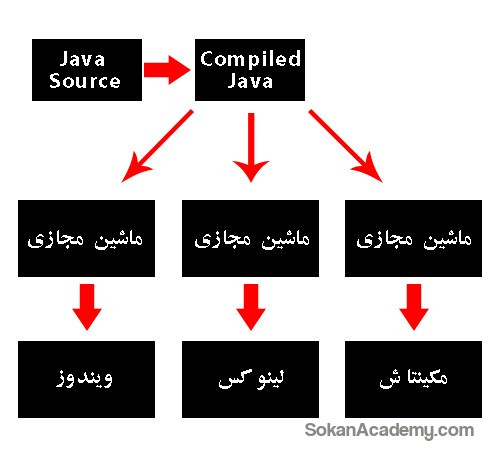
حال شما در محیط اصلی نرم افزار قرار دارید. از آنجا که ممکن است شما بیش از یک نسخه از زبان جاوا را روی سیستم خود نصب داشته باشید، نیاز است که در این مرحله برای اکلیپس مشخص کنید که از کدام نسخه جاوا برای اجرای برنامه ها استفاده کند. در منوی اصلی اکلیپس، با انتخاب گزینه Windows زیرشاخه Preferences را انتخاب نمایید. سپس در قسمت چپ پنجره، منوی Java را انتخاب نمایید. با کلیک کردن روی مثلث مشکی رنگ کنار واژه Java، قادر خواهید بود تا زیرشاخه های آن را نیز مشاهده کنید. از میان زیرشاخه ها، گزینه Installed JREs را انتخاب کنید. پس از انتخاب این زیرشاخه در منوی سمت راست، پنجره ای را مشاهده خواهید کرد که کلیه نسخه های JRE جاوا که بر روی سیستم شما نصب است را لیست کرده است. به طور مثال چنانچه نسخه JRE 7 بر روی سیستم عامل شما نصب باشد، این نسخه از جاوا را مشاهده خواهید کرد.

حال چنانچه نسخه JRE مد نظر شما در لیست موجود نباشد می توانید روی دکمه Add کلیک نموده و آن را به لیست فوق اضافه نمایید. در پنجره ای که باز خواهد شد روی گزینه Standard VM کلیک نمایید و سپس دکمه Next را کلیک نمایید. حال در بخش JER home مکانی که در آن جاوا را روی سیستم عامل خود نصب نموده اید را مشخص نمایید. نسخه های JRE که روی ویندوز نصب می شوند در مسیر دایرکتوری C > Program Files > Java قرار می گیرند. در صورتی که اکلیپس به صورت خودکار بخش JRE name را پر نکرد می توانید این کار را خود با نامی مرتبط با JRE مد نظر انجام دهید. حال می توانید گزینه Finish را انتخاب نمایید تا تغییرات شما اعمال شود و به صفحه ای بازگردید که در آن نسخه های نصب شده JRE سیستم شما قابل مشاهده هستند. نسخه نصب شده جدید خود را پیدا نموده و آن را انتخاب نمایید. در این مرحله کار شما به پایان رسیده است.

در ادامه، در همان منوی اصلی جاوا که از آن Installed JREs را انتخاب نمودید، زیرشاخه Compiler را پیدا نموده و به هیچ وجه آن را باز ننمایید بلکه فقط با کلیک کردن ماوس روی آن، آن را انتخاب نمایید. حال در منوی سمت راست مرتبط با زیرشاخه Complier از بخش Complier compliance level لیستی از نسخه های نصب شده JRE بر روی سیستم عامل را خواهید دید. به طور مثال برای جاوای نسخه 7 مورد 7.1 را از لیست انتخاب نمایید. اکنون می توانید دکمه OK صفحه Preferences را بزنید و به صفحه اصلی نرم افزار اکلیپس باز گردید. در این مرحله، کلیه ی ابزارهای مورد نیاز برای نوشتن اولین برنامه جاوای خود را به صورت رایگان در اختیار دارید.

**آشنایی با ماشین مجازی جاوا**

در این آموزش قصد داریم تا به تفصیل مفهوم JVM یا همان ماشین مجازی جاوا را مورد بررسی قرار دهیم. در واقع از بدو ایجاد زبان برنامه نویسی جاوا شعار طراحان این زبان محبوب این بوده است که Write One, Run Anywhere! به این معنی که "**یک بار برنامه جاوای خود را بنویسید سپس روی هر پلتفرمی که خواستید آن را اجرا کنید!**" که چنین قابلیتی جز با به کاری گیری مفهومی تحت عنوان "ماشین مجازی جاوا" امکان پذیر نخواهد بود. به طور کلی ماشین مجازی جاوا را می توان در قالب تصویر زیر نشان داد:



همانطور که در تصویر بالا مشخص است، برنامه جاوایی که ما می نویسیم یا همان سورس کد برنامه ابتدا توسط جاوا کامپایل می شود. منظور از اصطلاح Compile تبدیل کدهای جاوا به بایت کد است. در واقع ما یک بار بیشتر برنامه خود را کامپایل نخواهیم کرد و این برنامه Compiled شده برای هر یک از پلتفرم های موجود از قبیل ویندوز، مکینتاش، لینوکس و ... یکسان خواهد بود (لازم به ذکر است که اصطلاح Compiled یک صفت در انگلیسی محسوب می شود نه گذشته فعل Compile)

از این پس این وظیفه ماشین های مجازی است که برنامه Compiled شده را به شکلی که برای پلتفرم مد نظر قابل فهم است تبدیل نماید. به عبارت دیگر ماشین های مجازی تمام تلاش خود را به کار خواهند بست تا از روی کد یکسانی نتایج نسبتاً یکسانی را در پلتفرم های مختلف ایجاد کنند. اکنون پس از آشنایی با نحوه عملکرد ماشین مجازی، قصد داریم تا بیشتر درباره ماهیت خود این ماشین مجازی نکاتی را بیان کنیم:

  
همانطور که در تصویر فوق مشخص است ماشین مجازی جاوا را می توانیم به عنوان یک لایه مجازی مابین برنامه جاوای نوشته شده و سیستم عاملی که قرار است آن برنامه روی آن اجرا شود تلقی کرد. از سوی دیگر این ماشین مجازی هیچ کاری با سورس برنامه ما نخواهد داشت چرا که برنامه ما ابتدا Compile خواهد شد که در این پروسه همان طور که قبلاً گفته شد، به زبان قابل فهم برای سیستم تبدیل می شود که اصطلاحاً Bytecode نامیده می شود. سپس این زبان قابل فهم برای سیستم مورد استفاده ماشین مجازی قرار می گیرد تا برای سیستم عامل های مختلف بهینه شود.

یکی از مزایای به کارگیری از Bytecode این است که از این طریق می توان دیگر زبان های برنامه نویسی را نیز داخل جاوا به کار برد. به عبارت دیگر، چنانچه زبان برنامه نویسی یی وجود داشته باشد که بتواند خود را به Bytecode قابل فهم برای جاوا Compile کند، ماشین مجازی جاوا خواهد توانست  آن بخش از کدی را که در زبان دیگری نوشته شده است را نیز در کنار کدهای جاوا اجرا سازد.

**آشنایی با مفهوم Garbage Collection**  
منظور از واژه Garbage در زبان انگلیسی "**زباله**" است و واژه Collection نیز به معنی "**جمع آوری**" می باشد. زمانی که در زبان برنامه نویسی جاوا از این اصطلاح استفاده می کنیم منظور این است که ما به عنوان یک برنامه نویس جاوا نیازی نیست تا خود را درگیر مسائل مربوط به Memory یا "**حافظه**" کنیم چرا که این وظیفه ای است که بر عهده خود ماشین مجازی جاوا گذاشته شده است. برای روشن شدن نحوه عملکرد Garbage Collection مثالی می زنیم. حافظه در داخل برنامه‌های جاوا به دو قسمت تقسیم می‌شود، حافظه‌ی Heap و حافظه‌ی Stack. وقتی در یک برنامه آبجکتی را ایجاد می کنیم، مثلا شیئی از روی کلاس String ایجاد می کنیم:

String str = new String();

آبجکت تولید شده در حافظه‌ی Heap ذخیره می‌شود. توجه داشته باشید که str شیء نیست، بلکه ارجاعی به شیء ساخته شده در حافظه‌ی هیپ است. بنابراین متغیر str در حافظه‌ی Stack ذخیره شده است. در جاوا مفهوم اشارگر وجود ندارد اما در این قسمت می‌توان گفت که str یک نوع اشارگر است زیرا به شیئی که ساخته‌ایم در حافظه‌ی هیپ اشاره می‌کند.

در بیشتر منابع آموزشی گفته می‌شود که متغیر‌هایی مانند str شیء هستند که صحیح نیست! اما تقریبا تمام افراد متغیر‌هایی مانند str را با نام شیء صدا می‌کنند (حتی کسانی که از این موضوع با خبر هستند چرا که در گفتن راحت‌تر است.) بعد از این که آبجکت را ایجاد کردیم، با اجرای برنامه و در جایی از برنامه، آبجکت مورد نظر استفاده می‌شود و بعد از آن اگر دیگر از آن آبجکت استفاده نشود، آن آبجکت یک شیء «مُرده» به حساب می‌آید.

در زبان‌هایی مانند سی پلاس پلاس حذف کردن آبجکت‌های مُرده توسط برنامه نویس انجام می‌شود که این کار در برنامه‌های بزرگ و پیچیده به علت ارجاعات متعددی که آبجکت‌ها با یکدیگر دارند، فرآیندی بسیار سخت، پیچیده و پر خطا است اما در جاوا یک قابلیتی وجود دارد به نام Garbage Collector که وظیفه‌ی حذف آبجکت‌های مُرده در حافظه‌ی هیپ را بر عهده دارد. Garbage Collector بخشی از JVM است و الگوریتم‌های بسیار پیچیده‌ای دارد و فقط این را بدانید که هر از چند گاهی گاربج کالکتور به حافظه‌ی هیپ سر می‌زند و آبجکت‌های مُرده را پاک می‌کند تا حافظه آزاد شود. دقت داشته باشید که گاربج کالکتور فقط حافظه‌ی هیپ را پاک می‌کند. البته برنامه نویس هم می‌تواند گاربج کالکتور را فراخوانی کند، اما این کار اصلا توصیه نمی‌شود.