چکیده:

جاوا از ابتدا به صورت شی گراست در صورتی که سی پلاس پلاس اینطور نیست.

جاوا از go to استفاده نمی کنند ولی از continue و break در مواقع لزوم استفاده می کنند.

جاوا چون شی گراست ساختارهایی union و struct موجود در c را برداشته است.

جاوا اشاره گر ندارد ولی سی اشاره گر دارد.

کد سی ۱۰بار سریعتر از کد جاوا اجرا می شود .

جاوا از کاراکترهای یونی کد استفاده می کند اما سی نه

نام متغیرها در دو زبان نسبت به حروف حساس است.

نوع بولین در جاوا نمی تواند با عدد مقایسه شود در صورتی که در سی پلاس پلاس مجاز است.

در جاوا دشور typedef نداریم اما درc وجود دارد.

رشته ها در جاوا آرایه ساده ای از کاراکتر ها نیستند.

در جاوا عملوندهای پلاس پلاس و-- می تواند اعشاری باشد بر خلاف C.

در سی پلاس پلاس وراثت چند گانه مستقیما پشتیبانی می شود ولی در جاوا این وجود ندارد.

در سی پلاس پلاس سه نوع وراثت PUBLIC ، PROTECTED وPRIVATEوجود دارد ولی در جاوا وراثت فقط به صورت PUBLIC است.

گرامر جاوا خیلی بزرگتر از سی پلاس پلاس است.

جاوا بازتابش(ازاصول شی گرایی) را رعایت نکرده است (باز تابش : استفاده مجدد از کدها و نوشتن برنامه های الحاقی)

مقدمه:

جاوا یک زبان برنامه نویسی است که در اوایل دهه ۹۰ توسط Java Soft ، بخش نرم افزاری شرکت Sun توسعه داده شد . هدف آن بود که جاوا زبانی ساده ، قوی و همه منظوره باشد . جاوا تمام جنبه های مثبت C و C++ را در خود دارد ، و آن چیزهایی که برنامه نویسان C++ از آن نفرت داشته اند (مانند وراثت چند گانه ، تحریف اپراتورها و اشاره گرها) را به کناری گذاشته است .

مهمترین ویژگیهای جاوا این است که اساساً شیء گرا است . اولین ادعای OOP توانایی استفاده مجدد از کد است : چیزی که C++ با تمام ادعاهایش هرگز نتوانست بدان دست یابد . اما در اولین قدم خواهید دید جاوا در این زمینه تا چه حد اندازه صحت دارد . تصورش را بکنید که با صرف کمی وقت بتوانید برنامه ای بنویسید که در سیستم های ویندوز ، یونیکس

و مکینتاش براحتی اجرا شود . همین که یک شرکت نرم افزاری بتواند برای تمام پلاتفرم های موجود در آن واحد پروژه ای را تولید کند (و مقادیر عظیمی پول صرفه جویی کند) خود می تواند بهترین دلیل اقبال جاوا باشد و امروز دیگر همه (و نه فقط شرکتهای نرم افزاری) به سمت جاوا کشیده شده اند . با این ویژگی (استقلال از پلاتفرم) یک برنامه نویس می تواند برای سیستمی برنامه بنویسد که هرگز با آن کار نکرده است . این ویژگی اصلی ترین علت توفیق جاوا در اینترنت است . اینترنت شبکه پیچیده ای است از میلیونها کامپیوتر مختلف در سراسر دنیا ، و مقاومت در مقابل این وسوسه که بتواند برنامه ای بنویسد که روی تمام این سیستم های متفاوت و نا متجانس اجرا شود چندان ساده نیست .

جاوا یک زبان بسیار ساده است چون شما را وادار نمی کند تا در محیط جدید (و نا آشنایی) کار کنید و این برای کسانی که اطلاعات فنی ناچیزی درباره کامپیوتر دارند بسیار مهم است . ساختار زبان جاوا در نگاه اول

بسیار شبیه C و C++ است و این به هیچ وجه تصادفی نیست . C زبانی است ساخت یافته و C++ زبانیست شیء گرا و مهمتر از همه قسمت اعظم برنامه نویسان دنیا از آنها استفاده می کنند از سوی دیگر این شباهت حرکت به طرف جاوا را برای این قبیل افراد ساده خواهد کرد بنابراین طراحان جاوا برای اجتناب از دوباره کاری از زبانهای C و C++ بعنوان مدل استفاده کردند .

جاوا با دور انداختن اشاره گرها و بر دوش کشیدن بار مدیریت حافظه ، برنامه نویسان C وC++ را برای همیشه از این کابوس ها رهایی بخشیده است . علاه بر آن چون جاوا زبانی برای اینترنت است ، از ایمنی و حفاظت ذاتی بالایی برخوردار است . طراحان جاوا از ابتدا یک محیط برنامه نویسی امن را مد نظر داشته اند . مسئله حفاظت سیستم ها رابطه تنگاتنگی با اشاره گرها دارد . اکثر مهاجمان برای ورود غیر قانونی به سیستم های دیگران از این اشاره گرها استفاده می کنند و جاوا با حذف اشاره گرها این راه را سد کرده است .

۱-زبان سی

سی زبانی همه منظوره، ساخت یافته ،سطح بالا و انعطاف پذیر است که برخی از خصوصیات زبانهای سطح پایین را نیز که معمولا در اسمبلی یا زبان ماشین موجود است داراست. در عین حال این زبان برای کاربردهای ویژه طراحی نشده و می توان از آن در همه ی زمینه ها، بخصوص به دلیل نزدیکی آن به زبان ماشین در برنامه نویسی سیستم، استفاده کرد. بنابراین سی بین زبان های سطح بالا و سطح پایین قرار دارد و در نتیجه اجازه می دهد که برنامه نویس خصوصیات هر دو گروه زبان را به کار برد. از این رو در بسیاری از کابردهای مهندسی به طور انحصاری زبان سی به کار می برند. (زبان های سطح بالا، دستور العمل هایی شبیه زبان انسان و پردازش فکری او داردند، همچنین یک دستور العمل زبان سطح بالا معادل چند دستور العمل به زبان ماشین است).

برنامه های نوشته شده به زبان سی به طور کلی مستقل از ماشین یا نوع کامپیوتر است و تقریباً تحت کنترل هر سیستم عاملی اجرا می شود.

کامپایلرهای سی معمولا فشرده و کم حجم اند و برنامه های هدف ایجاد شده با آنها در مقایسه با سایر زبانهای برنامه سازی سطح بالا، خیلی کوچک و کار آمدند.

برنامه های سی در مقایسه با سایر زبانهای برنامه سازی سطح بالا، به راحتی قابل انتقال اند، دلیل آن این است که سی خیلی از ویژگی های وابسته به نوع کامپیوتر را در توابع کتابخانه ای خود منظور داشته است. بنابراین هر نسخه ازسی با مجموعه ای از توابع کتابخانه ای مخصوص به خود همراه است که بر اساس ویژگی های کامپیوتر میزبان مربوط نوشته شده است. این توابع کتابخانه ای تا حدودی استاندارد است و معمولا هر تابع کتابخانه ای در نسخه های متعدد سی به شکل یکسان در دسترس است.

سی روش برنامه نویسی ماژولار را پشتیبانی می کند. همچنین از نظر عملگر ها نیز زبانی قوی است که

عملگرهای گوناگونی برای دستکاری روی داده ها در سطح بیت داراست.

به طور کلی جامعیت ، عمومیت، خوانایی، سادگی ، کارآیی، و پیمانه ای بودن که همگی از مشخصات برنامه ای ایده آل اند در زبان سی پیاده سازی می شوند.

۲-زبان سی پلاس پلاس

زبان برنامه‌نویسی سی پلاس پلاس یک زبان برنامه‌نویسی کامپیوتری همه منظوره، شی گرا، سطح بالا و چندرگه (که از برنامه‌نویسی رویه‌ای، تجرید داده‌ها و برنامه‌نویسی شئ گرا پشتیبانی می‌کند)، عمومی و با قابلیت‌های سطح بالا و سطح پایین می‌باشد. این زبان دارای قابلیت‌های انواع داده ایستا، نوشتار آزاد، چندمدلی، معمولاً زبان ترجمه شده با پشتیبانی از برنامه‌نویسی ساخت‌یافته، برنامه‌نویسی شی‌گرا، برنامه نویسی جنریک است. سی پلاس پلاس به همراه جد خود سی از پرطرفدار ترین زبانهای برنامه‌نویسی تجاری هستند.

این زبان از تمام روش های مرسوم برنامه نویسی از قبیل برنامه‌نویسی شی‌گرا، برنامه نویسی ساخت یافته و برنامه نویسی پایه پشتیبانی می کند.

زبان سی پلاس پلاس در سال ۱۹۸۳ در آزمایشگاه های بل (Bell Labs) و بر مبنای زبان سی با افزودن امکانات برنامه نویسی شی گرا ساخته شد. این زبان از حوالی سال ۱۹۹۰ میلادی یکی از پرکاربردترین زبان‌های برنامه نویسی به شمار می رود که برای برنامه نویسی سیستمی و برنامه نویسی ویژوال (بر مبنای GUI) در محیط های مختلف از قبیل ویندوز و لینوکس به صورت وسیع به کار می رود.

زبان سی پلاس پلاس یک زبان سطح میانی در نظر گرفته می‌شود. این زبان دارای قابلیت زبان‌های سطح بالا و پایین بصورت همزمان است .توسعه این زبان با اضافه نمودن کلاس‌ها و ویژگی‌های دیگری مانند توابع مجازی، سربارگزاری عملگرها، وراثت چندگانه، قالب توابع، و پردازش استثنا انجام شد.

تاریخچه زبان سی پلاس پلاس

سی یک زبان عمومی، سریع، قابل حمل، و بصورت گسترده در حال استفاده بود. علاوه بر سی و سیمولا زبان‌های دیگری مانندALGOL 68 ،ADA ، CLU، MLنیز بر ساختار این زبان جدید(سی پلاس پلاس) اثر گذاشت. در ابتداویژگی‌های کلاس، کلاس‌های مشتق شده، کنترل نوع قوی، توابع درون خطی، و آرگومان پیش‌فرض از طریق Cfront به سی اضافه شد. اولین نسخه تجاری در سال ۱۹۸۵ ارائه شد.

ویژگی‌های دیگر شامل توابع مجازی، سربارگزاری عملگر و نام تابع، ارجاعات، ثوابت، کنترل حافظه توسط کاربر بصورت آزاد، کنترل نوع بهتر، و توضیحات یک خطی به صورت BCPL با استفاده از «//» نیز به آن اضافه شد. در سال ۱۹۸۹ ویژگی‌های جدیدی مانند ارث‌بری چندگانه، کلاس‌های انتزاعی، اعضای ایستای توابع، اعضای ثابت تابع، و اعضای حفاظت شده به آن اضافه شد .آخرین ویژگی‌های اضافه شده شامل موارد زیر بودند: قالب توابع، استثناها، فضاهای نام، تبدیلات جدید، و یک نوع داده منطقی.

در حین تکامل سی پلاس پلاس کتابخانه استاندارد نیز بوجود آمد. اولین نسخه کتاب استاندارد شامل کتابخانه جریانات I/O بود که جایگزین printf و scanf شد. در ادامه مهم‌ترین ویژگی اضافه شده Standard Template Libraryبوده‌است.

توسعه آینده:

سی پلاس پلاس همچنان در حال تکامل است تا نیازهای آینده را پاسخگو باشد.توسعه‌های مهم ،پشتیبانی از

چندرشته‌ای و مفاهیمی برای راحت نمودن کار با قالب‌هاست .اضافه نمودن ویژگی جمع‌آوری زباله به آن به شدت مورد بحث است.

۲-۱-دلایل طراحی سی پلاس پلاس

۱.سی پلاس پلاس طراحی شده‌است تا یک زبان عمومی با کنترل نوع ایستا و همانند سی قابل حمل و پربازده باشد .

۲.سی پلاس پلاس طراحی شده‌است تا مستقیما و بصورت جامع از چندین شیوه برنامه‌نویسی (برنامه‌نویسی ساخت‌یافته، برنامه‌نویسی شی‌گرا، انتزاع داده، و برنامه‌نویسی جنریک)پشتیبانی کند.

۳.سی پلاس پلاس طراحی شده‌است تا حداکثر تطابق با سی وجود داشته باشد و یک انتقال راحت از سی را ممکن سازد .

۴.سی پلاس پلاس از بکاربردن ویژگی‌های خاص که مانع از عمومی شدن است خودداری می‌نماید .

۵.سی پلاس پلاس طراحی شده‌است تا بدون یک محیط پیچیده عمل نماید.

کتابخانه استاندارد

کتابخانه استاندارد سی پلاس پلاس شامل کتابخانه استاندارد C با یک سری تغییرات برای بهبود عملکرد است. بخش بزرگ بعدی این کتابخانه STL است .STL شامل ابزار بسیار قدرتمندی مانند نگه‌دارنده‌ها (مانند vector وlist)، تکرارکننده‌ها (اشاره‌گرهای عمومی شده) برای شبیه‌سازی دسترسی مانند آرایه الگوریتم‌هایی برای جستجو و مرتب‌سازی در آنها وجود دارند. نقشه‌ها(نقشه‌های چندگانه) ،آرایه شرکت‌پذیر و مجموعه‌ها (مجموعه‌های چندگانه) واسط‌های عمومی فراهم می‌سازند. در نتیجه با استفاده از

قالب تابع، الگوریتم‌های جنریک با هر نگه‌دارنده و دارای تکرارکننده عمل نماید. همانند C ویژگی‌های کتابخانه را می‌توان با استفاده از شبه دستور #include شامل یک سرآیند استاندارد اضافه نمود.سی دارای ۶۹ کتابخانه استاندارد است که ۱۹ تا از آنها نامناسب تشخیص داده شده‌اند.

استفاده از کتابخانه استاندارد – مانند std::vector یا std::string به جای آرایه‌های C- موجب ایجاد برنامه‌های مطمئن‌تر شده‌است.

۲-۲-پیش پردازنده

سی پلاس پلاس بطور عمومی در سه فاز ترجمه می‌گردد: پیش‌پردازنده، ترجمه به کدobject ، پیوند (که دو مرحله آخر به عنوان عمل کامپایل شناخته می‌شود.) در اولین مرحله در پیش‌پردازنده، شبه‌دستورات پیش‌پردازنده تغییرات لغوی بر روی کد منبع ایجاد می‌نمایند و آن را به به مراحل دیگر تحویل می‌دهند. شبه دستورات پیش‌پردازنده با استفاده از کاراکتر # قبل از هر گونه فضای خالی آغاز گشته و رشته‌هایی را در کد منبع با فایل یا رشته‌های دیگر با توجه به قوانین تعریف گشته توسط برنامه‌نویس جایگزین می‌نماید. این دستورات معمولاً اعمال زیر را انجام می‌دهند: جایگزینی ماکروها، شمول فایل‌های دیگر (برخلاف ویژگی سطح بالاتر مانند شمول ماجول‌ها،پکیج‌ها،یونیت‌ها،کامپوننت‌ها)، کامپایل شرطی یا شمول شرطی. به عنوان مثال:

```
#include <iostream>
```

که این دستور تمام سمبل‌ها در فایل سرایند کتابخانه استاندارد iostream را در فایل منبع وارد می‌سازد.

کاربرد معمول دیگر به عنوان ماکرو خوانده می‌شود:

```
#define MY_ASSERT(x) assert(x)
```

که کدMY_ASSERT(x) را با assert(x) در فایل منبع جایگزین می‌نماید. که این جایگزینی امکان کنترل استفاده از این تابع را در اختیار برنامه‌نویس قرار می‌دهد.

استفاده از ماکروها در عمل چندان توصیه نمی‌گردد چرا که امکان کنترل نوع آرگومان‌ها را از بین برده در نتیجه ممکن است اشتباهاتی را وارد کد منبع نماید. طریقه دیگر برای انجام این کار استفاده از توابع درون‌خطی است.

علاوه بر شبه‌دستورات معمول تعدادی شبه دستور برای کنترل جریان کامپایل وجود دارد که امکان شمول یا عدم‌شمول قطعه‌ای کد یا سایر ویژگی‌های کامپایل را در اختیار ما قرار می‌دهد.

دستورات پیش‌پردازنده برای کاربردهای عددی نیز به کار می‌رود که هم‌اکنون استفاده از const به جای #defineترجیح داده می‌شود. این کار علاوه بر ایجاد کنترل نوع قوی مانع از گمراهی در فضاهای نام می‌گردد.

هدف کمیته استانداردسازی از بین بردن پیش‌پردازنده‌است اما با توجه به خصوصیت مدولار سی پلاس پلاس بعید به نظر می‌آید که این حذف امکان‌پذیر باشد.

۳-وراثت

وراثت این امکان را ایجاد می‌کند که یک نوع ویژگی دیگر انواع را داشته باشد. وراثت از یک کلاس پایه می‌تواند عمومی، خصوصی یا حفاظت شده باشد. این تعیین سطح دسترسی مشخص می‌سازد آیا کلاس‌های نامربوط و یا مشتق شده می‌توانند به اعضای عمومی یا حفاظت شده کلاس پایه دسترسی داشته باشند. تنها وراثت عمومی به معنای وراثت به کار رفته بصورت عموم است. دو نوع دیگر وراثت به ندرت

مورد استفاده قرار می‌گیرند. اگر تعیین‌کننده سطح دسترسی حذف شود سطح دسترسی برای کلاس خصوصی و برای ساختمان به صورت عمومی تعریف می‌گردد. کلاس‌های پایه ممکن است بصورت مجازی تعریف شوند که به آن وراثت مجازی گویند. وراثت مجازی تضمین می‌کند که فقط یک نمونه از کلاس پایه وجود داشته باشد و مشکلاتی همانند مشکلات وراثت چندگانه بوجود نیاید.

وراثت چندگانه یکی از ویژگی‌های مورد بحث در سی پلاس پلاس است. وراثت چندگانه امکان اشتقاق از چند کلاس پایه را فراهم می‌نماید که موجب بوجود آمدن گراف رابطه وراثت بسیار پیچیده‌است. به عنوان مثال «گربه پرنده» می‌تواند از کلاس «گربه» و کلاس «پستانداران پرنده» ارث برد. در زبان‌های دیگر مانند سی‌شارپ و جاوا به صورت دیگری ویژگی مشابه را پیاده‌سازی می‌نمایند هر کلاس می‌تواند از چندین واسط اشتقاق یابد اما فقط یک کلاس پایه برای اشتقاق وجود دارد(واسط‌ها برخلاف کلاس پایه فقط تعریف هستند و هیچ‌گونه پیاده‌سازی را شامل نمی‌گردند.

۴-زبان سی شارپ

طراحان زبان #Cبا تاکید و الگوبرداری مناسب از مزایای زبانهایی نظیر C++، C و جاوا و نادیده گرفتن برخی از امکانات تامل برانگیز و کم استفاده شده در هر یک از زبانهای فوق یک زبان برنامه نویسی مدرن شی گرا را طراحی کرده اند.

در مواردی برخی از ویژگی های استفاده نشده و درست درک نشده در هر یک از زبانهای گفته شده حذف و یا با اعمال کنترل های لازم بر روی آنها زمینه ایجاد یک زبان آسان و ایمن برای اغلب پیاده کنندگان نرم افزار بوجود آمده است. مثلا C و++ C می توانند مستقیما با استفاده از اشاره گرها عملیات دلخواه خود را در حافظه

انجام دهند. وجود توانایی فوق برای نوشتن برنامه های کامپیوتری با کارایی بالا ضرورت اساسی دارد. اما در صورتیکه عملیاتی اینچنین بدرستی کنترل و هدایت نگردند خود می تواند باعث بروز مسایل (BUGS) بیشماری گردد.

طراحان زبان C# با درک اهمیت موضوع فوق این ویژگی را کماکان در آن گنجانده ولی بمنظور ممانعت از استفاده نادرست و ایجاد اطمینان های لازم مساله حفاظت نیز مورد توجه قرار گرفته است. جهت استفاده از ویژگی فوق برنامه نویسان می بایست با صراحت و به روشنی خواسته خود را از طریق استفاده از KEYWORDهای مربوطه اعلان نمایند(فراخوانی یک توانایی و استفاده از آن) C#بعنوان یک زبان شی گرا عالی است.

این زبان FIRST- CLASS را برای مفهوم PROPERTY DATA MEMBER بهمراه سایر خصایص عمومی برنامه نویسی شی گرا حمایت می کند. در C و++C و جاوا یک متد GET/SET اغلب برای دستیابی به ویژگی های هر PROPERTY استفاده می گردد. CLI همچنان تعریف PROPERTYرا به متدهای GET/SET ترجمه کرده تا بدین طریق بتواند دارای حداکثر ارتباط متقابل با سایر زبانهای برنامه نویسی باشد. #Cبصورت فطری OVERLOADING REFERENCE TYPE ,DECLARED VALUE ,EVENTS OPERATOR را نیز حمایت می کند.

۴-۱-کد مدیریت یافته

با استفاده از نسخه پیاده سازی شده #Cتوسط مایکروسافت می توان همواره کد مدیریت یافته ای را تولید کرد. یک برنامه #Cپس از کامپایل بصورت برنامه ای در خواهد آمد که شامل دستورالعمل های تلفیق شده CLI: (COMMON LANGUAGE INTERMEDIATE) است (درست بر خلاف دستورالعمل های مختص یک

ماشین خاص). CLI گاهاً با نام (MICROSOFT INTERMEDIATE LANGUAGE MSIL) با به اختصار IL نیز نامیده می شود) در مفهوم مشابه بایت کدهای جاوا بوده و شامل مجموعه ای از CLI دستورالعمل های سطح پایین قابل فهم توسط تکنولوژی مبتنی بر CLI نظیر CLR مایکروسافت خواهد بود. این برنامه ها بدین دلیل کد مدیریت یافته نامیده می شوند که مسوولیت تبدیل این دستورالعمل ها به کدهای قابل اجرا برروی ماشین و ارائه اغلب سرویس های اساسی برای کدینگ نظیر: GARBAGE ,COLLECTION مدیریت HEAP و عمر مفید یک OBJECT و یا VERIFICATION TYPE را فراهم می کند.

۴-۲-روش یادگیری C#

یادگیری این زبان برای افرادیکه دارای سابقه آشنایی با یکی از زبانهای برنامه نویسی C++، C و یا جاوا باشند کار مشکلی نخواهد بود حتی افرادیکه دارای آشنایی اولیه با جاوااسکریپت و یا دیگر زبانهای برنامه نویسی نظیر ویژوال بیسک می باشند امکان پذیر و راحت خواهد بود. برخی از برنامه نویسان حرفه ای بر این باور هستند کهC# نسبت به VB.NET با اقبال بیشتر و سریعتری مواجه خواهد شد چراکه #Cنسبت به ویژوال بیسک UNSIGNED خلاصه تر است.

حتی برنامه های بزرگ و پیچیده ای که توسط #Cنوشته می گردند خواناتر، کوتاه و زیبا خواهند بود. برخی از ویژگی های ارایه شده در #CنظیرOPERATOR OVERLOADING INTEGER و امنیت بیشتر TYPEها در VB.NET وجود نداشته و این امر می تواند دلیلی بر فراگیرتر شدن #Cنسبت به VB.NET نزد برنامه نویسان با تجربه باشد.

برای یادگیری هر یک از زبانهای حمایت شده در دات نت می بایست از BCL(BASIC CLASS LIBRARY) مربوط به NET FRAMEWORK. شروع کرد. #Cخود صرفا دارای کلمه کلیدی یا KEYWORD بوده که برای اکثر برنامه نویسان غریب نخواهند بود. در مقابل BCL دارای کلاس و تعداد بیشماری متد و PROPERTY است که برنامه نویسان #Cمی توانند از آنها برای انجام عملیات دلخواه خود استفاده نمایند.

شاید یکی از مسایل قابل توجه جهت یادگیری این زبان برای برخی از برنامه نویسان حرفه ای عدم وجود برخی از ویژگی ها و امکاناتی باشد که در گذشته و از طریق سایر زبانها استفاده شده بخدمت گرفته می شدند. مثلا عدم وجود امکاناتی جهت وارثت چندگانه (LM) سلسله مراتبی یک شئ.

بدون شک فراگیری و تسلط بر زبان #Cبه منزله کسب یک پتانسیل با ارزش بوده که ثمرات آن برای برنامه نویسان در حال و آینده ای نه چندان دور بیشتر هویدا خواهد شد. استاندارد بودن و وجود کتابخانه ای مملو از کلاس این اطمینان را بوجود خواهد آورد که با فراگیری زبان فوق و کسب مهارت های لازم به یک توانایی فرا محیطی جدید دست پیدا خواهیم کرد که امکان استفاده از آن بر روی محیط های متفاوت وجود خواهد داشت. ویژگی ها و قابلیت های بیشمار این زبان از جمله دلایل قانع کننده دیگری است که فراگیری آن را توجیه پذیر و منطقی می کند.

#C از دو زبان وسی پلاس پلاس وJAVA متولد شده است! حاوی بسیاری از جنبه های سی پلاس پلاس می باشد اما ویژگی های شی گرایی خودش را از جاوا به ارث برده است.#C اگرچه از سی پلاس پلاس گرفته شده است اما یک زبان «خالص» شیء گرا (OBJECT ORIENTED) می باشد. هر دو زبان یاد شده جزو زبانهای

هیبرید محسوب می شوند اما طراحان #Cاین مورد را به اندازه سی پلاس پلاس مهم تلقی نکرده اند. یک زبان هیبرید اجازه برنامه نویسی با شیوه های مختلف را میسر می کند.

دلیل اینکه سی پلاس پلاس هیبرید است این است که قرار بوده تا با زبان C سازگار باشد و همین امر سبب گردیده تا بعضی از جنبه های سی پلاس پلاس بسیار پیچیده شوند. فرض زبان سی شارپ بر این است که شما می خواهید تنها برنامه نویسی شیء گرا انجام دهید و همانند سی پلاس پلاس مخلوطی از برنامه نویسی رویه ای (PROCEDURAL) و شی گرا را نمی خواهید به پایان برسانید. بنابراین باید طرز فکر خودتان را با دنیای شیء گرایی تطبیق دهید. در ادامه خواهید دید که در سی شارپ هر چیزی شیء است حتی یک برنامه سی شارپ.

بدون شک فراگیری و تسلط بر زبان #Cبمنزله کسب یک پتانسیل با ارزش بوده که ثمرات آن برای برنامه نویسان در حال و آینده ای نه چندان دور بیشتر هویدا خواهد شد. استاندارد بودن و وجود کتابخانه ای مملو از کلاس این اطمینان را بوجود خواهد آورد که با فراگیری زبان فوق و کسب، مهارت های لازم، به یک توانائی فرا محیطی جدید دست پیدا خواهیم کرد که امکان استفاده از آن بر روی محیط های متفاوت وجود خواهد داشت. ویژگی ها و قابلیت های بیشمار این زبان از جمله دلایل قانع کننده دیگری است که فراگیری آن را توجیه پذیر و منطقی می کند.

این زبان همچون زبان برنامه‌نویسی جاوا زبانی‌ست شئ‌گرا و بسیار سطح بالا (high level). محصول شرکت Microsoft و بر پایه .NET. از آنجایی که شئ‌گرائی و سطح بالا بودن از ابزارهای مدیریت مؤثر و کارآمد پیچیدگی در فضای پیچیده ی اینترنت مدرن می باشند، در واقع می‌شود جاوا و سی‌شارپ را از جمله

زبان های اصلی برای ایجاد و انجام برنامه‌های کاربردی تحت وب (web applications) و خدمات وب دانست.

بر اساس ادعای شرکت مایکروسافت، این زبان در سال ۲۰۰۰ توسط تیمی به سرکردگی آندرس هلزبرگ و نیز سکات ویلتاموث ساخته شد. سی‌شارپ که فقط برای دات نت است در مجموعه .NET Platform SDK ارائه گردید که در محیط های برنامه نویسی استودیوی بصری دات نت (Visual Studio .NET)، در نسخه‌های ۲۰۰۳ و ۲۰۰۵ آن موجود است. دستورات زبان سی‌شارپ مانند جاوا سطح بالا تر از C و سی پلاس پلاس است و برای مثال برنامه نویس مستقیما به اشاره گرها یا منابع سیستم دسترسی ندارد.

امروزه برنامه نویسی و تسلط کامل بر آن به عنوان یکی از پارامترهای مهم برای مهندسی کامپیوتر مطرح میباشد و در این راستا مایکروسافت در مصاف با جاوا به دنبال ارایه یک زبان کامل بود که سایه جاوا را در میادین برنامه نویسی کم رنگ تر نماید. شاید به همین دلیل باشد که C#را ایجاد کرد.

شباهت های بین دو زبان جاواوسی شارپ بسیار چشمگیر است. مایکروسافت در رابطه با میزان استفاده و گسترش زبان فوق بسیار خوشبین بوده و امیدوار است بسرعت زبان فوق ،گستردگی و مقبولیتی به مراتب بیشتر از جاوا را نزد پیاده کنندگان نرم افزار پیدا کند.

با توجه به نقش محوری این زبان از آن بعنوان مادر زبانهای برنامه نویسی در دات نت نام برده می شود. مورد فوق به تنهایی می تواند دلیل قانع کننده ای برای یادگیری این زبان باشد ولی دلایل متعدد دیگری نیز وجود دارد که در ادامه به برخی از آنها اشاره می گردد.

مطرح شدن به عنوان یک استاندارد صنعتی:

انجمن تولیدکنندگان کامپیوتر اروپا (ECMA) زبان C#را در سوم اکتبر سال ۲۰۰۱ به عنوان یک استاندارد پذیرفته و بدنبال آن تلاش های وسیعی برای کسب گواهی ISO نیز انجام شده است.

زبان فوق در ابتدا توسط شرکت مایکروسافت و بعنوان بخشی از دات نت پیاده سازی و بلافاصله پس از آن توسط شرکت های اینتل، هیولیت پاکارد و مایکروسافت مشترکاً جهت استاندارسازی پیشنهاد گردید. زبان C#بگونه ای طراحی شده است که نه تنها وابستگی به یک PLATFORM خاص را ندارد بلکه در اغلب موارد وابستگی RUNTIME نیز ندارد.

کامپایلر C#می تواند بر روی هر نوع معماری سخت افزاری طراحی و اجرا گردد. در برخی از نسخه های اولیه کامپایلر زبان فوق که توسط برخی از شرکت های جانبی ارایه شده است کدهای C#را به بایت کدهای جاوا کمپایل می کنند. یکی از چنین کامپایلرهایی را می توان در سایت HALCYONSOFT.COMمشاهده نمود. بنابراین کدهای C#براحتی قابلیت حمل بر روی محیط های متفاوت را دارا خواهند بود.

با توجه به موارد گفته شده مشخص می گردد که این زبان بسرعت به سمت استاندارد شدن حرکت و با تایید استانداردهای مربوطه از طرف انجمن های معتبر بین المللی و حمایت فراگیر شرکت های معتبر کامپیوتری در دنیا مسیر خود را به سمت جهانی شدن بخوبی طی می نماید.

۴-۳-سی شارپ دربرابرزبان های دیگر

زبان C# تقلید از جاوا نیست.

هلسبرگ:

واقعیت این است که C# تقلید از جاوا نیست. ما برای طراحی C# به زبانهای زیادی از جمله سی پلاس پلاس ، جاوا ، Modula2 و حتی SmallTalk نگاه کردیم. زبانهای زیادی هم وجود دارند که با وجود هسته یکسان ایده های جالبی ارائه کرده اند که ما شیفته برخی از آنها نظیر امکانات برنامه نویسی شی گرا و ساده سازی اشیا شده ایم.

یکی از تفاوتهای اساسی C# و سایر زبانها مخصوصاً جاوا این است که ما سعی کردیم خیلی نزدیک به سی پلاس پلاس عمل کنیم. بیشتر کلمات کلیدی و عبارات C# مستقیما از سی پلاس پلاس گرفته شده است. همچنین برخی ویژگیها که جاوا آنها را زاید شمرده بود را حفظ کردیم. منظورم این است که چرا امکان استفاده از enum در جاوا وجود ندارد. حال آنکه همین enum ها یکی از قدرتهای برجسته سی پلاس پلاس هستند. ما enum را در C# در نظر گرفتیم و نوع داده آن را به بهترین شکل ممکن ایمن ساختیم.

در C#عبارات enum تنها اعداد صحیح نیستند. بلکه نوع داده قدرتمندی میباشند که از کتابخانه کلاسهای پایه System.Enum در Net. مشتق شده اند. به این ترتیب یک enum از نوع foo با enum از نوع bar قابل برابری نیست.

مثال دیگر این که ما بارگذاری بیش ار اندازه عملگرها را در نظر گرفتیم و ساختار حوزه های نامگذاریمان را به سی پلاس پلاس نزدیک کردیم.

صرف نظر از این امکانات سنتی و استاندارد زبانهای دیگر، یکی از اهداف اصلی ما این بود که C#را به صورت یک زبان جزء گرا (Component Oriented) طراحی کنیم تا خودش بتواند ویژگیهایی را که برای نوشتن اجزای برنامه لازم دارد را فراهم نماید.

ویژگیهایی مثل خصوصیات (Properties) ، متدها ، event ها ، صفات (Attributes) و اسناد (Documents) هستند که همگی ساختارهای اولیه و درجه یک زبان میباشند.

C#نخستین زبانی است که میتواند تگ های xml را پردازش کند و کدهای خود را توسط کامپایلر مستقیما از Source Code به عبارات قابل فهم ماشین تبدیل نماید.

یکی دیگر از این ویژگیها چیزی است که من آن را خاصیت تک مرحله سازی (One-Stop Programming) مینامم. وقتی شما کد مورد نظر خود را در C# می نویسید، همه چیز را یکجا درست میکنید. نیازی به فایلهای Heather ، فایلهای IDL و خطوط توضیحی نیست. یکبار که کدی را مینویسید خودش میتواند خودش را توضیح دهد. با این کار میتوانید برنامه خود را به راحتی مستقل و قابل انتقال کنید. چون هر واحد آن برای اجرا به چیز دیگری غیر از خودش نیاز ندارد.

برنامه ای که بدین ترتیب نوشته شده را میتوان در صفحات ASP جای داد و در محیطهای مختلف میزبانی کرد که قبلا به هیچ وجه امکان پذیر نبود.

با نگاهی به گذشته میفهمیم که بر سر اینکه آیا زبانها باید بر اساس Properties نوشته شوند یا بر اساس event ها دعوای زیادی است. مسلما میتوانستیم فرضیاتمان را با متدها بیان کنیم. میتوانستیم با استفاده از بلوکهای معروف get و set و استفاده همزمان از خصوصیات اشیا (Properties) برنامه را به بهترین حالت خود برسانیم.

حتی ایجاد محیطهایی که اشیا با یکدیگر در ارتباط باشند هم به راحتی امکانپذیر بود تنها به شرطی که C امکان برنامه نویسی شی گرا را فراهم می ساخت. البته کار مشکل تر می شد و رسیدگی به برنامه زمان بیشتری می طلبید. اما در مجموع نتیجه ای بهتر از مدل برنامه نویسی رویه ای نصیب برنامه نویس می گشت و مشتری هم از آن راضی بود.

این روزها برنامه نویسان بجای آنکه بنشینند و برنامه ای کامل و یکپارچه بنویسند، آن را به واحدهای مجزا تقسیم میکنند و کار را بین خود پخش مینمایند و نهایتا این واحدها را با یکدیگر در ارتباط قرار میدهند.

لازمه این کار این است که تمام برنامه نویسان از یک شیوه استاندارد برنامه نویسی پیروی کنند و واحدهای خود را با کلاسهای مشتق شده از کلاسهای پایه طراحی نمایند. این هدیه ای است که برنامه نویسی شی گرا به ما داده و تاکنون تقریبا همین روند ادامه یافته است.

۵-زبان جاوا

امروزه یکی از متداول ترین زبان های برنامه نویسی جهان است. این زبان از لحاظ ظاهری شباهت های زیادی به سی پلاس پلاس دارد ولی در اصل می توان تفاوت های بنیادین زیادی را برای آنها بر شمرد. بر خلاف بسیاری دیگر از زبان های کامپایلری که سورس کد آنها پس از کامپایل شدن به باینری یک ماشین حقیقی ترجمه می شود، برنامه های جاوا پس از کامپایل شدن به باینری ماشینی به نام ماشین مجازی جاوا یا JVM ترجمه می شود. این کد باینری را بایت کد می گویند. ماشین مجازی جاوا عمدتاً بصورت نرم افزاری پیاده سازی می شود ولی پیاده سازی های سخت افزاری یا ترکیبی (بیشتر به صورت کمک پردازنده) از آن نیز

وجود دارد. استفاده از ماشین مجازی یک روش مدرن در زبان های برنامه نویسی محسوب می شود و مزایای زیادی دارد. ماشین مجازی جاوا می تواند امنیت اجرای برنامه ها را تضمین کند و حق دسترسی های مختلفی برای برنامه ها در نظر بگیرد. همچنین ماشین مجازی جاوا سازگاری اجرای برنامه های جاوا را تحت سیستم عامل های مختلف حفظ می کند. یک باور غلط در بین بسیاری از مردم این است که برنامه های نوشته شده به زبان جاوا کندتر از برنامه های نوشته شده به زبان های کامپایلری اجرا می شود، در حالی که سرعت اجرای یک برنامه جاوا کاملا بستگی به نحوه پیاده سازی ماشین مجازی دارد. ماشین های مجازی جدید جاوا معمولا از تکنیکی به نام کامپایل در زمان لازم یا JIT استفاده می کنند. در این روش در هنگام اجرای برنامه بایت کد ابتدا به کد باینری ماشینی حقیقی که بر روی آن اجرا می شود ترجمه می شود و سپس باینری ترجمه شده بر روی ماشین اصلی اجرا می شود. در این فرآیند ماشین مجازی می تواند بهینه سازی های خاص ماشین حقیقی را بر روی کد انجام دهد که در برخی موارد باعث می شود برنامه های جاوا حتی سریع تر دیگر زبان های کامپایلری اجرا شود.

هنگامی که مهندسین شرکت Sun Microsystemsسرگرم انجام پروژه ای موسوم به پروژه گرین بودندتابرای لوازم الکترونیکی تولیداین شرکت نرم افزارپیشرفته بسازنددریافتند که کامپایلرهای C,C++برای اینکار نارسایی دارندوازاین روبه فکرخلق زبان جدیدی افتادندکه درابتداOakنام گرفت وپس از مدتی به جاوا تغییرنام داد.مدیرپروژه جاواJames Goslingبودکه نهایتاً آن رادراوایل دهه نودباموفقیت به اتمام رساند.جاوایک زبان برنامه نویسی شی گرااست که میتوان گفت از C,C++مشتق شده است واهدافی مثل ((عدم وابستگی به محیط اجرا)) را که در C++ درعمل نتوانست به آن دست پیداکند،به زیبایی محقق کرده است. ویژگی ((عدم وابستگی به محیط اجرا)) یکی ازنیازهای اولیه وب

است.جاوازبانی است شی گراکه با تمام مولفه های یک برنامه(مثل متغیرها،توابع ،سابروتینهاویاحتی کل برنامه) به چشم شی می نگرد.درضمن زبانی ساده،قابل حمل وتوانمنددرحمایت از برنامه های چندریسمانی وبامعماری خنثی است.

جاوا در سال ۱۹۹۵ به عنوان مولفه اصلی java platform منتشر شد. هدف آن بود که جاوا زبانی ساده ، قوی و همه منظوره باشد . جاوا تمام جنبه های مثبت C و سی پلاس پلاس را در خود دارد ، و آن چیزهایی که برنامه نویسان سی پلاس پلاس از آن نفرت داشته اند (مانند وراثت چند گانه ، تحریف اپراتورها و اشاره گرها) را به کناری گذاشته است .

این زبان قسمت های بسیاری از گرامر خود را از سی و سی پلاس پلاس گرفته اما دارای مدل شی‌گرایی ساده‌ای است و امکانات سطح پایین کمی دارد. کاربرد جاوا در کامپایل به صورت بایت کد است که قابلیت اجرا روی تمامی ماشین‌های شبیه‌سازی جاوا را داشته باشد صرف نظر از معماری و خصوصیات آن کامپیوتر.

۵-۱-اهداف اولیه ساخت و طراحی جاوا:

۱. این زبان باید ساده، شی‌گرا و مشهور باشد.

۲. مطمئن و بدون خطا باشد.

۳. وابسته به معماری کامپیوتر نبوده و قابل انتقال باشد.

۴. باید با کارایی بالا اجرا شود.

۵. باید به صورت پویا و نخ‌کشی‌شده باشد.

۵-۲-برنامه‌های جاوا و اپلت‌ها

جاوا برای نوشتن انواع برنامه‌های کاربردی مناسب است. با جاوا می‌توان انواع برنامه‌های زیر را نوشت:

- برنامه‌های تحت وب
- برنامه‌نویسی سیستم‌های کوچک مانند موبایل، پاکت پی‌سی و ...
- برنامه‌های کاربردی بزرگ (Enterprise)
- برنامه‌های رومیزی (Desktop)

قابلیت خاصی بنام اپلت در جاوا وجود دارد. اپلت‌ها امکانات فراوانی برای نوشتن برنامه‌های تحت وب در اختیار برنامه‌نویسان قرار می‌دهند که دیگر زبان‌های برنامه‌نویسی فاقد آن هستند. البته وجود ماشین مجازی جاوا برای اجرای اپلت لازم است. اپلت‌ها نظیر فناوری Activex شرکت مایکروسافت هستند که برنامه‌نویسان را قادر می‌سازد تا امکاناتی را به مرورگر کاربر بیافزایند.البته تفاوت این دو(اپلت وفناوریActivex) در امنیت می‌باشد به گونه‌ای که اپلت‌ها بدلیل اینکه در محیطی به نام جعبه شنی اجرا می‌شوند امن هستند ولی Activex ها فاقد چنین امنیتی هستند.

۵-۳- خط مشی جاوا

یکی از ویژگی‌های جاوا قابل حمل بودن آن است. یعنی برنامه ی نوشته شده به زبان جاوا باید به طور مشابهی در کامپیوترهای مختلف با سخت‌افزارهای متفاوت اجرا شود. و باید این توانایی را داشته باشد که برنامه یک بار نوشته شود، یک بار کامپایل شود و در همه کامپیوترها اجرا گردد. به این صورت که کد کامپایل شده ی جاوا را

ذخیره می‌کند، اما نه به‌صورت کد ماشین بلکه به‌صورت بایت کد جاوا. دستورالعمل‌ها شبیه کد ماشین هستند، اما با ماشین‌های مجازی که به طور خاص برای سخت‌افزارهای مختلف نوشته شده‌اند، اجرا می‌شوند. در نهایت کاربر از JRE نصب شده روی ماشین خود یا جستجوگر وب استفاده می‌کند. کتابخانه‌های استاندارد یک راه عمومی برای دسترسی به ویژگی‌های خاص(مانند گرافیک، نخ‌کشی و شبکه) فراهم می‌کنند. در بعضی از نسخه‌های JVM بایت کدها می‌توانند قبل و در زمان اجرای برنامه به کدهای محلی کامپایل شوند. فایدهٔ اصلی استفاده از بایت کد، قسمت کردن است. اما ترجمهٔ کلی یعنی برنامه‌های ترجمه شده تقریباً همیشه کندتر از برنامه‌های کامپایل شدهٔ محلی اجرا می‌شوند. این شکاف می‌تواند با چند تکنیک خوش‌بینانه که در کاربردهای JVM قبلی معرفی شد، کم شود. یکی از این تکنیک‌ها JIT است که بایت کد جاوا را به کد محلی ترجمه کرده و سپس آن را پنهان می‌کند. در نتیجه برنامه خیلی سریع‌تر نسبت به کدهای ترجمه شدهٔ خالص شروع و اجرا می‌شود. بیشتر JVM های پیشرفته، به‌صورت کامپایل مجدد پویا، در آنالیز JVM رفتار برنامهٔ اجرا شده و کامپایل مجدد انتخاب شده و بهینه‌سازی قسمت‌های برنامه، استفاده می‌شوند. کامپایل مجدد پویا می‌تواند کامپایل ایستا را بهینه‌سازی کند. زیرا می‌تواند قسمت hot spot برنامه و گاهی حلقه‌های داخلی که ممکن است زمان اجرای برنامه را افزایش دهند را تشخیص دهد. کامپایل JIT و کامپایل مجدد پویا به برنامه‌های جاوا اجازه می‌دهد که سرعت اجرای کدهای محلی بدون از دست دادن قابلیت انتقال افزایش پیدا کند.

تکنیک بعدی به عنوان کامپایل ایستا شناخته شده است، که کامپایل مستقیم به کدهای محلی است مانند بسیاری از کامپایلرهای قدیمی. کامپایلر ایستای جاوا، بایت کدها را به کدهای شی محلی ترجمه می‌کند.

کارایی جاوا نسبت به نسخه‌های اولیه بیشتر شد. در تعدادی از تست‌ها نشان داده شد که کارایی کامپایلر JIT کاملاً مشابه کامپایلر محلی شد.عملکرد کامپایلرها لزوماً کارایی کدهای کامپایل شده را نشان نمی‌دهند.یکی از پیشرفت های بی نظیر در در زمان اجرای ماشین این بود که خطا ها ماشین را دچار اشکال نمی‌کردند.علاوه بر این در زمان اجرای ماشینی مانند جاوا وسایلی وجود دارد که به زمان اجرای ماشین متصل شده و هر زمانی که یک استثنا رخ می‌دهد،اطلاعات اشکال زدایی که در حافظه وجود دارد ،ثبت می‌کنند.

۵-۴- پیاده سازی

شرکت سان میکروسیستم مجوز رسمی برای پلت فرم استاندارد جاوا را به Microsoft Windows, Linux, و Solaris داده است. همچنین محیط‌های دیگری برای دیگر پلت فرم ها فراهم آورده است. علامت تجاری مجوز شرکت سان میکروسیستم طوری بود که با همهٔ پیاده سازی ها سازگار باشد.به علت اختلاف قانونی که با ماکروسافت پیدا کرد، زمانی که شرکت سان ادعا کرد که پیاده سازی ماکروسافت از RMI یا JNI پشتیبانی نکرده و ویژگی های خاصی را برای خودش اضافه کرده‌است. شرکت سان در سال ۱۹۹۷ پیگیری قانونی کرد و در سال ۲۰۰۱ در توافقی ۲۰ میلیون دلاری برنده شد. در نتیجه کمی بعدماکروسافت جاوا را به ویندوز فرستاد. در نسخهٔ اخیر ویندوز، جستجوگر اینترنت نمی‌تواند از جاوا پلت فرم پشتیبانی کند. شرکت سان و دیگرانیک سیستم اجرای جاوای رایگان برای آنها و نسخه‌های دیگر ویندوز فراهم آوردند.

۵-۵- اداره خودکار حافظه

جاوا از حافظهٔ بازیافتی خودکار برای ادارهٔ حافظه در چرخهٔ زندگی یک شی استفاده می‌کند. برنامه‌نویس زمانی که اشیا به وجود می‌آیند، این حافظه را تعیین می‌کند. و در زمان اجرا نیز، زمانی که این اشیا در استفادهٔ زیاد طولانی نباشند، برنامه نویس مسئول بازگرداندن این حافظه است. زمانی که مرجعی برای شی های باقیمانده نیست، شی‌های غیر قابل دسترس برای آزاد شدن به صورت خودکار توسط بازیافت حافظه،انتخاب می‌شوند. اگر برنامه‌نویس مقداری از حافظه را برای شی‌هایی که زیاد طولانی نیستند، نگه دارد، چیزهایی شبیه سوراخ حافظه اتفاق می‌افتند.

یکی از عقایدی که پشت سر مدل ادارهٔ حافظهٔ خودکار جاوا وجود دارد، این است که برنامه‌نویس هزینهٔ اجرای ادارهٔ دستی حافظه را نادیده می‌گیرد. در بعضی از زبان‌ها حافظه لازم برای ایجاد یک شی، به صورت ضمنی و بدون شرط، به پشته تخصیص داده می‌شود. و یا به‌طور صریح اختصاص داده شده و از heap بازگردانده می‌شود. در هر کدام از این راه‌ها، مسئولیت ادارهٔ اقامت حافظه با برنامه‌نویس است. اگر برنامه شی را برنگرداند، سوراخ حافظه اتفاق می‌افتد. اگر برنامه تلاش کند به حافظه‌ای را که هم‌اکنون بازگردانده شده، دستیابی پیدا کند یا برگرداند، نتیجه تعریف شده نیست و ممکن است برنامه بی‌ثبات شده و یا تخریب شود. این ممکن است با استفاده از اشاره‌گر مدتی باقی بماند، اما سرباری و پیچیدگی برنامه زیاد می‌شود. بازیافت حافظه اجازه دارد در هر زمانی اتفاق بیفتد. به‌طوری که این زمانی اتفاق می‌افتد که برنامه بی‌کار باشد. اگر حافظهٔ خالی کافی برای تخصیص شی جدید در heap وجود نداشته باشد، ممکن است برنامه برای چند دقیقه متوقف شود. در جایی که زمان پاسخ یا اجرا مهم باشد، ادارهٔ حافظه و منابع اشیا استفاده می‌شوند.

جاوا از نوع اشاره‌گر ریاضی سی و سی پلاس پلاس پشتیبانی نمی‌کند. در جایی که آدرس اشیا و اعداد صحیح می‌توانند به جای هم استفاده شوند. همانند سی پلاس پلاس و بعضی زبان‌های شی‌گرای دیگر، متغیرهای نوع‌های اولیهٔ جاوا شی‌گرا نبودند. مقدار نوع‌های اولیه، مستقیماً در فیلدها ذخیره می‌شوند. در فیلدها (برای اشیا) و در پشته (برای توابع)، بیشتر از heap استفاده می‌شود. این یک تصمیم هوشیارانه توسط طراح جاوا برای اجرا است. به همین دلیل جاوا یک زبان شی‌گرای خالص به حساب نمی‌آید.

۵-۶-فایل ها وکلاس ها در جاوا

فایل هادرجاوا بعد از کلاس های عمومی نام گذاری می‌شوند.سپس باید پسوند java را به این صورت اضافه کرد: Hello world.java .این فایل اول باید با استفاده از کامپایلر جاوا به بایت کد کامپایل شود.در نتیجه فایل Hello world.class ایجاد می‌شود.این فایل قابل اجرا است.فایل جاوا ممکن است فقط یک کلاس عمومی داشته باشد.اما می‌تواند شامل چندین کلاس با دستیابی عمومی کمتر باشد.کلاسی که به صورت خصوصی تعریف می‌شود ممکن است در فایل .java ذخیره شود.کامپایلر برای هر کلاسی که در فایل اصلی تعریف می‌شود یک کلاس فایل تولید می‌کندکه نام این کلاس فایل همنام کلاس است با پسوند class.

کلمهٔ کلیدی static (ایستا) در جلوی یک تابع ،یک تابع ایستا را که فقط وابسته به کلاس است و نه قابل استفاده برای نمونه‌هایی از کلاس،نشان می‌دهد.فقط تابع های ایستا می‌توانند توسط اشیا بدون مرجع صدا زده شوند.داده‌های ایستا به متغیر هایی که ایستا نیستند،نمی توانند دسترسی داشته باشند.

کلمهٔ کلیدی void (تهی) نشان می‌دهد که تابع main هیچ مقداری را بر نمی گرداند.اگر برنامهٔ جاوا بخواهد با خطا از برنامه خارج شود،باید system.exit صدا زده شود. کلمهٔ main یک کلمهٔ کلیدی در زبان جاوا نیست.این نام واقعی تابعی است که جاوا برای فرستادن کنترل به برنامه ،صدا می‌زند.برنامه جاوا ممکن است شامل چندین کلاس باشد که هر کدام دارای تابع main هستند.

چاپ کردن،قسمتی از کتابخانهٔ استاندارد جاوا است.کلاس سیستم یک فیلد استاتیک عمومی به نام out تعریف کرده است.شی out یک نمونه از کلاس printstream است و شامل تعداد زیادی تابع برای چاپ کردن اطلاعات در خروجی استاندارد است.

۵-۷-توزیع‌های جاوا

منظور از توزیع جاوا پیاده‌سازی‌های مختلفی است که برای کامپایلر جاوا و همچنین مجموعه کتابخانه‌های استاندارد زبان جاوا (JDK) وجود دارد. در حال حاضر چهار توزیع‌کنندهٔ عمده جاوا وجود دارند:

- **سان میکروسیستمز:** توزیع کننده اصلی جاوا و مبدع آن می‌باشد. در اکثر موارد هنگامی که گفته می‌شود جاوا منظور توزیع سان می‌باشد.
- **GNU Classpath**: این توزیع از سوی موسسه نرم‌افزارهای آزاد منتشر شده و تقریبا تمامی کتابخانه استاندارد زبان جاوا در آن بدون بهره‌گیری از توزیع شرکت سان از اول پیاده‌سازی شده است.

یک کامپایلر به نام GNU Compiler for Java نیز برای کامپایل کردن کدهای جاوا توسط این موسسه ایجاد شده‌است. فلسفه انتخاب نام Classpath برای این پروژه رها کردن تکنولوژی جاوا از وابستگی به علامت تجاری جاوا است بطوریکه هیچ وابستگی یا محدودیتی برای استفاده آن از لحاظ قوانین حقوقی ایجاد نشود و از طرفی به خاطر وجود متغیر محیطی classpath در تمامی محیط های احرایی برنامه‌های جاوا، این نام به نوعی تکنولوژی جاوا را برای خواننده القا می‌کند. کامپایلر GNU توانایی ایجاد کد اجرایی (در مقابل بایت کد توزیع سان) را داراست. لازم به ذکر است که در حال حاضر شرکت سان تقریبا تمامی کدهای JDK را تحت مجوز نرم‌افزارهای آزاد به صورت متن باز منتشر کرده است و قول انتشار قسمت بسیار کوچکی از این مجموعه را که به‌دلیل استفاده از کدهای شرکت‌های ثانویه نتوانسته به صورت متن باز منتشر نماید در آینده نزدیک با بازنویسی این کدها داده است.

- **مایکروسافت #J**: این در حقیقت یک توزیع جاوا نیست. بلکه زبانی مشابه می‌باشد که توسط مایکروسافت و در چارچوب**net.** ارائه شده‌است. انتظار اینکه در سیستم عاملی غیر از ویندوز هم اجرا شود را نداشته باشید.

- **AspectJ**: این نیز یک زبان مجزا نیست. بلکه یک برنامه الحاقی می‌باشد که امکان برنامه نویسی Aspect Oriented را به جاوا می‌افزاید. این برنامه توسط بنیاد برنامه‌نویسی جلوه‌گرا و به صورت کدباز ارائه شده‌است.

۵-۸-کلاس های خاص جاوا

۵-۸-۱- Applet (برنامه‌های کاربردی کوچک)

اپلت جاوا برنامه‌هایی هستند که برای کاربردهایی نظیر نمایش صفحات وب در جستجوگر وب، ایجاد شده‌اند. واژهٔ import باعث می‌شود کامپایلر جاوا کلاس‌های javaapplet.Applet و java.awt.Graphics را به کامپایل برنامه اضافه کند. کلاس Hello کلاس Applet را توسعه می‌دهد. کلاس اپلت چارچوبی برای کاربردهای گروهی برای نمایش و کنترل چرخهٔ زندگی اپلت، درست می‌کند. کلاس اپلت یک تابع پنجره‌ای مجرد است که برنامه‌های کوچکی با قابلیت نشان دادن واسط گرافیکی برای کاربر را فراهم می‌کند. کلاس Hello تابع موروثی print(Graphics)را از سوپر کلاس container باطل می‌کند، برای اینکه کدی که اپلت را نمایش می‌دهد، فراهم کند. تابع paint شی‌های گرافیکی را که شامل زمینه‌های گرافیکی هستند را می فرستد تا برای نمایش اپلت‌ها استفاده شوند. تابع paint برای نمایش "Hello world!" تابع(drawstring(string,int,int را صدا می‌زند.

۵-۸-۲- Servlet

تکنولوژی servlet جاوا گسترس وب را به آسانی فراهم می‌کند.و شامل مکانیزم هایی برای توسعهٔ تابعی سرور وب و برایدسترسی به سیستم های تجاری موجود است.servlet قسمتی از javaEE است که به درخواست های مشتری پاسخ می‌دهد.

واژهٔ import کامپایلر جاوا را هدایت می‌کند که تمام کلاس های عمومی و واسط ها را از بسته‌های java.io و java.servlet را در کامپایل وارد کند.

۵-۸-۳- کتابخانه‌های کلاس

کتابخانه‌های جاوا که به صورت بایت کد از کد اصلی کامپایل شده اند،برای پشتیبانی از بعضی از کاربرد های جاوا،توسط JRE منتشر شده است. مثال هایی از این کتابخانه‌ها عبارتند از:

- کتابخانه‌های مرکزی که شامل:
- کتابخانه‌هایی که برای ساختار داده کاربرد دارند.مثل لیست ها،درخت ها،مجموعه ها،مترجم ها.
- کتابخانهٔ پرداز XML (تجزیه،تغییر شکل،اعتبار)
- کتابخانه‌های موضعی و بین المللی
- کتابخانه‌های انتگرال گیری که امکان تایپ کردن توسط سیستم های بیرونی را می‌دهند.
- JDBC برای دستیابی به داده ها
- JNDI برای مراجعه و کشف کردن
- RMI & CORBA برای توسعهٔ کاربرد توزیع کردن
- کتابخانه‌های واسط کاربر
- AWT (توابع پنجره‌ای مجرد)که قسمت هایی از GUI را فراهم می‌کنند.

- کتابخانه‌های swing که در AWT ساخته شده اند اما کاربرد هایی از AWT widgetry را فراهم می‌کنند.
- APL ها برای ضبط صدا،پردازش و بازنواختی
- کاربردهای وابستهٔ پلت فرم ماشین های مجازی جاوا
- Plugins که توانایی اجرا شدن در جستجوگر های وب را به اپلت می‌دهد.
- java web start
- دادن مجوز و مستند سازی

۵-۹- نقاط ضعف زبان جاوا

مهم‌ترین ایرادی که برنامه نویسان سایر زبان‌ها به زبان جاوا می‌گیرند سرعت اجرایی بسیار پایین جاوا است. یک برنامه جاوا به صورت بایت کد می‌باشد و باید در ماشین مجازی جاوا اجرا گردد. به همین دلیل سرعت اجرای پایینی را در مقابل زبان‌های قدرتمندی همچون سی پلاس پلاس دارد. به صورت دیگر یک برنامه سی به طور متوسط تا ۱۰ برابر سریعتر از برنامه مشابه جاوا اجرا می‌گردد. جاوا علی رغم شی گرا بودن در بخشی از قسمت‌ها برای ایجاد انعطاف بیشتر یا بازاریابی بهتر برخی اصول شی گرایی را نادیده گرفته‌است. از جمله این قسمت‌ها قابلیت بازتابش (Reflection)می‌باشد. هدف اصلی بازتابش این است که استفاده مجدد از کدها و گسترش کدهای موجود و مهم‌تر از همه نوشتن برنامه‌های الحاقی آسان گردد ولی این مهم با زیر پا گذاشتن

بعضی اصول ممکن شده‌است. برای نمونه با کمک بازتابش به راحتی می‌توان متدهای خصوصی دیگر کلاس‌ها را فراخوانی کرد!

۵-۱۰-پاسخ برنامه‌نویسان جاوا به ایرادات

سرعت پایین برنامه‌های جاوا در محیطی که اجرا می‌شوند ملاک کارایی نبوده زیرا در محیط وب مسئله‌ای که سرعت را کند می‌سازد، شبکه بوده و ابتدا باید سربار شبکه را از روی برنامه‌ها برداشت. از طرف دیگر در برنامه‌های رومیزی هم در JDK 5.0 و ۶.۰، بهینه سازی بسیاری بوجود آمده که این مسئله باعث شده که در آخرین تست کارایی که انجام شده یک برنامه جاوا در محدودهٔ ۰.۸ تا ۱.۳ همان برنامه در سی پلاس پلاس کارایی داشته باشد که ۱.۳ آن مربوط به بخش واسط کاربری و سرعت ۰.۸ آن مربوط به بسته تخلیه حافظه می‌شده که هیچ الگوریتمی نتوانست از الگوریتم Garbage Collector جاوا پیشی بگیرد. همچنین سال ۱۹۹۹ در مقاله‌ای آقای Lutz Prechelt این مسئله را ثابت کردند که تجربه برنامه‌نویسی که برنامه‌ای را می‌نویسد از انتخاب زبانی که برنامه برروی آن نوشته می‌شود در کارایی تأثیر بیشتری دارد و این بدان معناست که کارایی یک برنامه را برنامه‌نویس مشخص می‌کند و نه زبان برنامه‌نویسی(ایشان در همان مقاله از زبان جاوا استفاده نمودند تا ذهنیت بد را از بین ببرند)

۵-۱۱-دلایل حذف اشاره گرها درجاوا

حذف اشاره‌گرها در جاوا به دلیل مشکلاتی بوده که آنها در طول تاریخشان بوجود آورده‌اند، اگرچه این موارد در برنامه‌های سیستمی لازم به‌نظر می‌رسد ولی در محیط‌های تحت‌وب که بستر اصلی جاوا هستند می‌توانند اثراتی به مراتب شدیدتر نسبت به آنچه در برنامه‌های سیستمی دارند داشته باشند و باعث می‌شود که توجه برنامه‌نویسان از مسائلی چون کارایی، قابلیت اطمینان و مقیاس‌پذیری برنامه به تنظیم اشاره‌گرها معطوف گردد.

۵-۱۲-یک اشتباه متداول

برخی مردم به علت شباهت اسمی، جاوا و جاوااسکریپت را با هم اشتباه می‌گیرند. در حالیکه این دو زبان گرچه در ظاهر و کلمات شبیه اند ولی بطور ساختاری با یکدیگر متفاوتند. جاوا اسکریپت محصول شرکت نت اسکیپ است.

۶-اسکریپت چیست ؟

۶-۱- اسکریپت

زبان های اسکریپتی برای ارائه تحولات و ایجاد پویایی در صفحات وب ایجاد شدند . این زبان ها از روی زبان های برنامه نویسی ساخته شدند و بهمین دلیل دارای تشابه بسیاری با هم هستند . این زبان ها در اصل نمونه کوچک شده زبان های مادر خود هستند . تعدادی از فرمان ها و امکانات زبان های بزرگ در این زبان ها حذف

شده اند . مثلا امکان نوشتن فایل یا پاک کردن فایل ها برروی سیستم کاربر مانند زبان های برنامه نویسی وجود ندارد . البته این زبان ها برای استفاده در زمینه کاری شبکه طراحی شده اند و حذف این دستورات علل خاصی (از جمله بالا رفتن امنیت و...) داشته است .

۶-۲-جاوا اسکریپت

این زبان ساختاری شبیه زبان سی دارد و بیشتر برای ایجاد افکت بر روی کامپیوتر کاربر استفاده میشود . (Client-side)احتمالا تا کنون سایت هایی را دیده اید که در آن کلمه خاصی دنبال موس میدود . یا هنگام وارد شدن به آن سایت مرورگر شما در صفحه ویندوزتان می لرزد . این قبیل کد ها که فقط روی سیستم کاربر اجرا می شوند و نیازی به پردازش توسط سرور ندارند را کد های سمت کاربر (کلاینت ساید) می گویند . البته این زبان نیز قابلیت های استفاده به صورت server-side را داراست . اما چون استفاده از زبان VBS آسانتر است معمولا از VBS برای نوشتن برنامه های سرور-ساید استفاده میشود . یکی از تفاوت های این دو زبان در طرز نوشتن حروف است . در VBS تفاوتی ندارد که دستورات را با حروف کوچک یا بزرگ بنویسید ، اما در JavaScriptاگر دستوری که با حروف کوچک است با حروف بزرگ بنویسید با Error در صفحات خود مواجه میشوید.برای دیدن قدرت زبان JavaScript میتوانید از سایت AnfyTeam دیدن کنید . این سایت همچنین امکان دانلود برنامه ای برای ساخت افکت های جاوا اسکریپت خود را در اختیارتان میگذارد .

۷-وراثت (Inheritance)

یکی از مهمترین مباحثی که در برنامه نویسی زبانهای شی گرا (مانند سی شارپ و وی بی دات نت) وجود دارد، وراثت یا Inheritance می باشد. وقتی که ما یک پروژه ویندوزی مانند برنامه Hello World رو شروع می کنیم، محیط برنامه نویسی ویژوال استدیو (Visual Studio .NET Integrated Development Environment) یک تعریف به شکل زیر رو ایجاد می کند:

```
System.Windows.Forms.Form : public class Form1
```

این تعریف به معنای این است که فرم ایجاد شده فرزند کلاسی به نام Form است. که دارای مفاهیم بسیار قوی است. ما می توانیم اشیایی رو بسازیم و خصوصیات آنها رو تغییر بدهیم. در این صورت رفتار هر شی بصورت خاص همان شی ولی در کل مشابه کلاس اصلی خواهد بود.

۷-۱-استفاده از وراثت

وراثت در سی شارپ این قابلیت را به ما می دهد که یک کلاس از کلاس آماده دیگر مشتق بگیریم. در کلاس جدید مشتق شده ما فقط می توانیم متدهای جدید را اضافه نماییم یا متدهای قبلی را تغییر دهیم. بقیه موارد دیگر بصورت خودکار از کلاس پایه به کلاس جدید به ارث خواهد رسید.

۸-سازنده ها (Constructors)

همه کلاسها دارای یک سازنده یا Constructor هستند که در زمان ایجاد یک نمونه از کلاس فراخوانی می شوند. سازنده همیشه همنام با نام کلاس است. زمانی که یک کلاس را تعریف می کنید، باید یک سازنده برای مقداردهی اولیه آن ایجاد نمایید. البته می توانید پارامترهایی رو به آن ارجاع دهید تا مقداردهی آرگومانهای کلاس مطابق آنچه شما مد نظر دارید مقداردهی شوند. اگر شما برای کلاسی که می نویسید سازنده قرار ندهید، یک سازنده بدون آرگومان در حالت مخفی ساخته می شود.

۹-برنامه‌نویسی شی‌گرا

برنامه نویسی شی‌گرا (به انگلیسی Oriented Programming Object مخفف OOP) شیوه‌ای ازتحلیل و طراحی نرم افزار است که بر تجزیه‌ی مسئله به اشیاء تاکید دارد. اشیاء صور انتزاعی از ماهیت‌های مطرح در مسئله هستند که دو جنبه دارند. اشیاء دارای حالت یا داده بوده و همچنین دارای عملیات بر روی داده‌ها می‌باشند. این نوع نگرش دارای مزایای بسیاری از جمله مدیریت پیچیدگی وهزینه نگهداری کمتری است.

در برنامه نویسی شی‌گرا (Object Oriented) همه چیز یک شی (Object)است. هر شی ویژگی‌ها(Properties) و توابع مربوط به خودش را دارد.

زبانهای برنامه نویسی شی گرا، زبانهایی هستند که در آن برنامه نویس می‌تواند اشیا مختلفی را تعریف نماید و از اشیاء تولید شده استفاده نماید. هر شی یک سری خصوصیت و قابلیت دارد، که اصطلاحاً Properties و

Methods خوانده می‌شوند. در این روش از برنامه نویسی دید برنامه نویس به سیستم دید شخصی است که سعی می نماید به پیدا کردن اشیاء مختلف در سیستم و برقراری ارتباط بین آنها سیستم را تولید نماید.

۱۰-جاوا در مقابل سی

جاوا یک زبان برنامه نویسی است که در اوایل دهه ۹۰ توسط Java Soft ، بخش نرم افزاری شرکت Sun توسعه داده شد . هدف آن بود که جاوا زبانی ساده ، قوی و همه منظوره باشد . جاوا تمام جنبه های مثبت C و سی پلاس پلاس را در خود دارد ، و آن چیزهایی که برنامه نویسان سی پلاس پلاس از آن نفرت داشته اند (مانند وراثت چند گانه ، تحریف اپراتورها و اشاره گرها) را به کناری گذاشته است .

مهمترین ویژگیهای جاوا این است که اساساً شیء گرا است . اولین ادعای OOP توانایی استفاده مجدد از کد است : چیزی که سی پلاس پلاس با تمام ادعاهایش هرگز نتوانست بدان دست یابد . اما در اولین قدم خواهید دیدادعا جاوا در این زمینه تا چه اندازه صحت دارد . تصورش را بکنید که با صرف کمی وقت بتوانید برنامه ای بنویسید که در سیستم های ویندوز ، یونیکس

و مکینتاش براحتی اجرا شود . همین که یک شرکت نرم افزاری بتواند برای تمام پلاتفرم های موجود در آن واحد پروژه ای را تولید کند (و مقادیر عظیمی پول صرفه جویی کند) خود می تواند بهترین دلیل اقبال جاوا باشد و امروز دیگر همه (و نه فقط شرکتهای نرم افزاری) به سمت جاوا کشیده شده اند . با این ویژگی (استقلال از پلاتفرم) یک برنامه نویس می تواند برای سیستمی برنامه بنویسد که هرگز با آن کار نکرده است . این ویژگی اصلی ترین علت توفیق جاوا در اینترنت است . اینترنت شبکه پیچیده ای است از میلیونها کامپیوتر

مختلف در سراسر دنیا ، و مقاومت در مقابل این وسوسه که بتواند برنامه ای بنویسد که روی تمام این سیستم های متفاوت و نا متجانس اجرا شود چندان ساده نیست .

جاوا یک زبان بسیار ساده است چون شما را وادار نمی کند تا در محیط جدید (و نا آشنایی) کار کنید و این برای کسانی که اطلاعات فنی ناچیزی درباره کامپیوتر دارند بسیار مهم است . ساختار زبان جاوا در نگاه اول بسیار شبیه C و سی پلاس پلاس است و این به هیچ وجه تصادفی نیست . C زبانی است ساخت یافته و سی پلاس پلاس زبانیست شیء گرا و مهمتر از همه قسمت اعظم برنامه نویسان دنیا از آنها استفاده می کنند از سوی دیگر این شباهت حرکت به طرف جاوا را برای این قبیل افراد ساده خواهد کرد بنابراین طراحان جاوا برای اجتناب از دوباره کاری از زبانهای C و سی پلاس پلاس بعنوان مدل استفاده کردند .

جاوا با دور انداختن اشاره گرها و بر دوش کشیدن بار مدیریت حافظه ، برنامه نویسان C وسی پلاس پلاس را برای همیشه از این کابوس ها رهایی بخشیده است . علاه بر آن چون جاوا زبانی برای اینترنت است ، از ایمنی و حفاظت ذاتی بالایی برخوردار است . طراحان جاوا از ابتدا یک محیط برنامه نویسی امن را مد نظر داشته اند . مسئله حفاظت سیستم ها رابطه تنگاتنگی با اشاره گرها دارد . اکثر مهاجمان برای ورود غیر قانونی به سیستم های دیگران از این اشاره گرها استفاده می کنند و جاوا با حذف اشاره گرها این راه را سد کرده است .

۱۱-جاوادرمقابل سی پلاس پلاس

جاوا از ابتدا به صورت شی گراست در صورتی که سی پلاس پلاس اینطور نیست.

جاوا از go to استفاده نمی کند ولی از continue و break در مواقع لزوم استفاده می کند.

جاوا چون شی گراست ساختارهای union و struct موجود در c را برداشته است.

جاوا اشاره گر ندارد ولی سی اشاره گر دارد.

کد سی ۱۰بار سریعتر از کد جاوا اجرا می شود .

جاوا از کاراکترهای یونی کد استفاده می کند اما سی نه

نام متغیرها در دو زبان نسبت به حروف حساس است.

نوع بولین در جاوا نمی تواند با عدد مقایسه شود در صورتی که در سی پلاس پلاس مجاز است.

در جاوا دستور typedef نداریم اما درc وجود دارد.

رشته ها در جاوا آرایه ساده ای از کاراکتر ها نیستند.

در جاوا عملوندهای ++ و-- می توانند اعشاری باشند بر خلاف C.

در سی پلاس پلاس وراثت چند گانه مستقیماً پشتیبانی می شود ولی در جاوا این وجود ندارد.

در سی پلاس پلاس سه نوع وراثت PUBLIC ، PROTECTED وPRIVATEوجود دارد ولی در جاوا وراثت فقط به صورت PUBLIC است.

گرامر جاوا خیلی بزرگتر از سی پلاس پلاس است.

جاوا بازتابش(ازاصول شی گرایی) را رعایت نکرده است (باز تابش : استفاده مجدد از کدها و نوشتن برنامه های الحاقی)

۱۱-۱-تفاوتها و شباهتهایc++,java

۱۱-۱-۱-معرفی جاوا

جاوا یک زبان ساده ، شئ‌گرا ، توزیع شده ، تفسیر شده ، قدرتمند ، مستقل از پلتفرم ، ایمن ، چند رشته ای، با معماری خنثی ، قابل حمل ، با عملکرد سطح بالا چند نخ کشی شده و پویا است که شرکت سان مایکروسیستمز آن را ابداع کرده‌است. زبان جاوا شبیه به سی پلاس پلاس است اما مدل شیءگرایی آسان تری دارد.یکی از قابلیت‌های اصلی جاوا این است که مدیریت حافظه را بطور خودکار انجام می‌دهد. ضریب اطمینان عملکرد برنامه‌های نوشته‌شده به این زبان بالا است و وابسته به سیستم عامل خاصی نیست، به عبارت دیگر می‌توان آن را روی هررایانه با هر نوع سیستم عاملی اجرا کرد.

۱۱-۱-۲-شی گرا(Object Oriented)

جاوا یک زبان برنامه نویسی شی گرا است و همه چیز در جاوا به صورت کلاس ،تابع یا آبجکت می باشد در یک سیستم شی گرا، یک کلاس مجموعه‌ای از داده‌ها و روش‌هایی است که روی آن داده عمل می‌کنند. همراه بودن داده‌ها و متدها رفتار و حالت یک شی را بیان می دارد بنابراین یک زیر کلاس می تواند از تمام داده ها و روشهای کلاس بالاتر ارث ببرد. برای یک برنامه نویس این به این معنا است که به جای فکر کردن به قسمت های رویه برنامه ، باید به کاربرد داده ها و روش هایی که روی آن داده ها عمل میکنند ، توجه شود.

اگر شما به برنامه نویسی با اعلان رویه در C عادت کرده اید ،ممکن است دریابید که به هنگام استفاده از جاوا مجبور به تغییر در روش و چگونگی برنامه تان هستید . هنگامی که فهمیدید این الگوی جدید چقدر قدرتمند است ، به سرعت با آن هماهنگ میشوید.

در یک سیستم شی گرا ، یک کلاس مجموعه ای از داده ها و روش هایی است که روی آن داده عمل میکنند. همراه بودن داده ها و متد ها رفتار و حالت یک شی را بیان می دارد . کلاس ها به صورت سلسله مراتبی مرتب شده اند ، بنابر این یک زیر کلاس میتواند رفتار هایی را از کلاس بالاتر به ارث ببرد . یک کلاس سلسله مراتبی همیشه یک کلاس ریشه دارد که کلاسی با رفتار های کاملا عمومی است.

جاوا به همراه دسته ی گسترده ای از کلاس هایی است که در بسته هایی مرتب شده اند و شما می توانید از آنها در برنامه ی خود استفاده کنید . برای مثال جاوا کلاس هایی را ایجاد میکند که:

بخش های رابط گرافیکی را میسازند (the java.awt package) ،کلاس هایی که عملیات

ورودی و خروجی را به عهده دارند (the java.io package)و کلاس هایی که از شبکه پشتیبانی میکنند(the java.net package)

یک شی کلاس (in the java.lang package)به عنوان ریشه کلاس سلسله مراتبی جاوا انجام وظیفه میکند.

جاوا بر خلاف C++ طوری طراحی شده است که از همان ابتدا به صورت شی گرا باشد . اکثر چیز ها در جاوا اشیا هستند . ارقام ابتدایی ، کاراکترها و مدل های منطقی تنها استثناء ها هستند . حتی رشته ها هم در جاوا به وسیله اشیا حاضر میشوند ،همان طور که ساختمان های مهم دیگر این زبان ، مثل نخ ها احضار میشوند . یک کلاس ، یک واحد پایه برای کامپایل و اجرا شدن در جاوا است . تمام برنامه های جاوا متشکل از کلاس ها است.

درست است که جاوا طوری طراحی شده است که مثل C++ باشد و خاصیت های آن را داشته باشد ، اما هنگامی که با آن کار کنید خواهید فهمید که بسیاری از پیچیده گی های آن زبان را از بین برده است.

اگر شما یک برنامه نویس C++ هستید حتما لازم است که ساختار های شی گرایی در جاوا را به دقت مطالعه کنید . اگرچه ترکیب و نحوه دستورات آن تقریبا شبیه C++ است ،اما رفتار های آن خیلی مشابه نیست.

۱۱-۱-۳-ساده (simple)

جاوازبان بسیار راحتی است به این دلیل که میشود براحتی نوشت و خواند و همین دلیل کافی بود تا طراحان جاوا به فکربیفتند تا زبانی بوجود آورند که برنامه نویسیان بتوانند به سرعت آنرا یاد بگیرند و با امکان مدیریت اتوماتیک حافظه ای که دارد باعث شد که برنامه نویسیان دیگر کمتر به مدیریت حافظه فکر کنند . جاوا یک زبان ساده است . طراحان جاوا سعی در این داشتند تا زبانی بوجود بیاورند که برنامه نویسان بتوانند به سرعت آن را یاد بگیرند . بنابراین تعداد ساختار های این زبان تقریبا کم است . هدف دیگر طراحی این زبان این بود که به منظور راحتی انتقال آن ، آن را طوری طراحی کنند که برای عده ی زیادی از برنامه نویسان آشنا باشد . اگر شما یک برنامه نویس C یا C++ هستید ، خواهید فهمید که جاوا از بسیاری از ساختار های C و C++ استفاده میکند .

برای اینکه این زبان را هم به طور ساده و هم آشنا و ملموس و هم کوچک نگه دارند بسیاری از خصوصیات C و C++ را در آن حذف کردند . اینها خصوصیاتی بودند که باعث می شدند برنامه نویسی ضعیفی صورت بگیرد یا آنهایی بودند که به ندرت در برنامه استفاده می شدند . برای مثال جاوا از دستور goto استفاده نمی کند ، در عوض از دستورهای break , continue در مواقع نیاز استفاده می کند .

جاوا از سر فایل ها (header files)استفاده نمی کند و پردازشگر C را هم حذف کرده است . به این دلیل که جاوا یک زبان شی گرا است ، ساختار های C مثل struct , union از آن برداشته شده است . جاوا حتی بارگذاری مجدد و خواص چندگانه ارث بری از C++ را هم حذف کرده است . شاید مهمترین پارامتر ساده بودن جاوا عدم استفاده این زبان از اشاره گر ها باشد . اشاره گر ها یکی از بیشترین موجودیت های دردسرساز در C , C++ هستند . چون جاوا ساختمان ندارد و آرایه ها و رشته ها اشیاء آن هستند ، بنابراین احتیاجی به

اشاره گر نیست .جاوا به طور خودکار آدرس دهی و دستذسی به محتوای موجود در یک آدرس را برای شما انجام میدهد .

جاوا حتی زباله های حافظه ای را هم به طور خودکار جمع آوری میکند (Garbage Collectin). جمع آوری آشغال فرایندی است برای ترمیم خودکار حافظه انباشته شده . بلوک هایی از حافظه که زمانی به فایل ها اختصاص داشتند اما مدتی است که از آنها استفاده نمی شود و بلوک هایی از حافظه که هنوز مورد استفاده قرار میگیرند ممکن است حرکت داده شوند تا از به هم پیوستن فضاهای خالی حافظه بلوک های خالی بزرگتری بدست آید .

بنابراین لازم نیست که نسبت به موضوع مدیریت حافظه نگران باشید ، همه اینها شما را از نگرانی در مورد اشاره گر های بی ارزش ، خطرناک و هرز های حافظه رها میکنند . بنابر این شما میتوانید وقت خود را صرف بهبود برنامه تان کنید.

۱۱-۱-۴-مستقل از پلتفرم (platform independent)

جاوا به گونه ای طراحی شده است که به سیستم عامل وابسته نیست و این برای برنامه های Enterprise یک ویژگی بسیار مهم است که این امکان را فراهم می کند که با یکبار برنامه نویسی آنرا روی هر سیستم عاملی اجرا کرد و این امکان توسط ماشن مجازی ، که قسمتی از جاوا می باشد فراهم شده است .

۱۱-۱-۵-چندرشته ای (multi-thread)

جاوا یک زبان چند رشته ایست است، که از چندین رشته اجرایی پشتیبانی می‌کند و می‌تواند چندین کار را به طورهمزمان انجام دهد که این امکان برای بساری از برنامه های کابردی از جمله بازی ها و انیمیشن ها بسیار مهم می باشد.

۱۱-۱-۶-تفسیر شده(Interpreted)

جاوا یک زبان تفسیر شده است . کامپایلر جاوا به جای ایجاد کد محلی ماشین ، کد بایتی برای ماشین مجازی جاوا ایجاد میکند . برای اجرای دقیق برنامه ، از مفسر جاوا برای اجرای کد های بایتی کامپایل شده استفاده میشود . به دلیل اینکه کد های بایتی جاوا به نوع کامپیو تر بستگی ند ارند ، برنامه ها ی جا وا میتو ا نند روی هر نوع کامپیوتری که JVM (Java Virtual Machine)را دارد ، اجرا شوند.

در محیط تفسیر شده ، مرحله لینک استاندارد توسعه برنامه از دید کاربر پنهان است . اگر جاوا تنها یک مرحله لینک داشت ، فقط بارگذاری کلاس جدید به محیط پردازش میشد که یک پردازش نموی سبک وزن است که در زمان اجرا مشاهده میشود . که این خصوصیت با چرخه کامپایل-لینک-اجرا ی آرام و طاقت فرسای زبان هایی مانند C یا C++ در تضاد است.

۱۱-۱-۷-معماری خنثی و قابل حمل(Architecture Neutral and Portable)

به دلیل اینکه برنامه های جاوا در فرمت کد بایتی با معماری خنثی کامپایل شده اند ، برنامه کاربردی جاوا میتواند در هر سیستمی اجرا شود.

البته با این شرط که آن سیستم توانایی پیاده سازی ماشین مجازی جاوا را داشته باشد . این مسئله تقریبا برای کاربرد های توزیع شده روی اینترنت و یا دیگر شبکه های ناهمگن مهم است . اما روش معماری خنثی برای کاربرد های بر مبنای شبکه مفید است.

به عنوان یک توسعه دهنده برنامه های کاربردی در بازار نرم افزاری امروز ممکن است بخواهید مدل های کاربردی خود را توسعه دهید ، به طوری که بتواند روی Pc ، مکینتاش و سیستم عامل Unix اجرا شود .با وجود گونه های مختلف Unix ، Windows روی Pc و مکینتاش قوی جدید ، رفته رفته تولید نرم افزار برای همه انواع این کامپیوتر ها سخت می شود . اگر شما برنامه تان را در جاوا بنویسید میتواند روی همه ی این کامپیوترها اجرا شود .

در حقیقت تفسیر شده بودن جاوا وتعریف یک استاندارد ، معماری خنثی داشتن و فرمت کد بایتی آن از بزرگترین دلایل قابل حمل بودن آن به شمار می آیند.

اما جاوا باز از این هم بیشتر گام برمیدارد ، با اطمینان حاصل کردن از اینکه هیچیک از جنبه های وابستگی اجرایی زبان را ندارد . برای مثال جاوا به طور صریح اندازه هریک از انواع داده را تعریف میکند که این با C تفاوت دارد ، برای مثال هریک از انواع صحیح می تواند بسته به نوع کامپیوتر۱۶-۳۲ یا ۶۴ بیت طول داشته باشد.

هنگامی که به صورت تکنیکی امکان نوشتن برنامه های غیر قابل حمل در جاوا فراهم شد ، جلوگیری از چند

خاصیت وابسته به نوع کامپیوتر که توسط جاوا API تولید شده و به طور قطع قابل حمل نوشته شده است ، آسان است.

یک برنامه جاوا به تولید کنندگان نرم افزار کمک میکند تا از قابل حمل بودن کد هایشان اطمینان حاصل کنند . برنامه نویسان فقط برای پرهیز از دام غیر قابل حمل بودن برنامه احتیاج به یک تلاش ساده دارند که شعار تجارتی شرکت Sunرا زنده نگهدارند و آن شعار این است:

" یک بار بنویس ، همه جا اجرا کن ".

۱۱-۱-۸-پویا و توزیع شده (Dynamic and Distributed)

جاوا یک زبان پویا است . هر کلاس جاوا میتواند در هر زمانی روی مفسر جاوا بارگذاری شود . سپس این کلاس های بارگذاری شده ی پویا میتوانند به صورت پویا معرفی شوند . حتی کتابخانه کد های محلی میتواند به طور پویا بارگذاری شود .کلاس ها در جاوا با Classفراخوانی میشوند ؛ شما میتوانید به طور پویا در مورد یک کلاس در زمان اجرا اطلاعاتی بدست بیاورید . این خصوصیت در جاوا ۱-۱ به طور درستی موجود است . با وجود بازتاب API اضافه شده (Application Program Interface) که به برنامه ساز امکان میدهد که با برنامه از طریق یک برنامه کاربردی دیگر ارتباط برقرار کند.

جاوا حتی با نام زبان توزیع شده نیز خوانده میشود . به طور ساده این به این معنا است که این زبان پشتیبانی سطح بالایی برای شبکه به وجود می آورد . برای مثال کلاس URL و کلاس های مرتبط با آن در بسته ی Java.net خواندن فایل های دوردست را به همان سادگی خواندن فایل های محلی کرده است . به طور مشابه در جاوا ۱-۱ ، احضار روش کنترلی RMI (Remote Method Invocation) ، API به یک

برنامه جاوا اجازه میدهد که روش هایی از اشیاء دور دست جاوا را به همان صورتی که اگر آن اشیاء محلی بدند آنها را میخواند ، بخواند .(جاوا حتی از سیستم شبکه ای سطح پایین که شامل آدرس مقصد و مسیر جریانی که توسط سوکت ها متصل شده است ، نیز پشتیبانی میکند).

طبیعت توزیع شده ی جاوا زمانیکه با امکانات پویای بارگذاری کلاس همراه میشود ، واقعا درخشنده است . این خصوصیات با هم این امکان را برای مفسر جاوا به وجود می آورند که کد ها را از اینترنت بارگذاری و اجرا کند . (همان طور که بعدا خواهیم دید جاوا باعث میشود که با وجود ابزار قدرتمند و ایمن این کار به طور مطمئن انجام شود .) این چیزی است که در هنگام بارگذاری و اجرای یک برنامه کاربردی از اینترنت توسط مرورگر وب ، اتفاق می افتد . اما داستان پیچیده تر از این هم میتواند باشد . تصور کنید یک پردازشگر کلمه چند رسانه ای در جاوا نوشته شده است . وقتی از این برنامه پرسیده میشود که چند نوع از داده هایی را که قبلا هرگز وارد نشده را نمایش دهد ، ممکن است به طور دینامیکی یک کلاس را که میتواند داده را شناسایی کند ، از شبکه بارگذاری کند و بعد کلاس دیگری را که بتواند داده را از درون یک پوشه ترکیبی بخواند ، باز به طور دینامیکی بارگذاری میکند . برنامه ای مانند این از منابع توزیع شده در شبکه برای رشد و سازگاری خودکار کاربران استفاده میکند.

۱۱-۱-۹-ساختارجاواوسی پلاس پلاس

جاوا یک زبان ساده است . طراحان جاوا سعی در این داشتند تا زبانی بوجود بیاورند که برنامه نویسان بتوانند به سرعت آن را یاد بگیرند . بنابراین تعداد ساختار های این زبان تقریبا کم است . هدف دیگر طراحی این زبان

این بود که به منظور راحتی انتقال آن ، آن را طوری طراحی کنند که برای عده ی زیادی از برنامه نویسان آشنا باشد . اگر شما یک برنامه نویس C یا C++ هستید ، خواهید فهمید که جاوا از بسیاری از ساختار های C و C++استفاده میکند.

برای اینکه این زبان را هم به طور ساده و هم آشنا و ملموس و هم کوچک نگه دارند بسیاری از خصوصیات C و C++ را در آن حذف کردند . اینها خصوصیاتی بودند که باعث می شدند برنامه نویسی ضعیفی صورت بگیرد یا آنهایی بودند که به ندرت در برنامه استفاده می شدند . برای مثال جاوا از دستور goto استفاده نمی کند ، در عوض از دستورهای break , continueدر مواقع نیاز استفاده می کند.

جاوا از سر فایل ها (header files)استفاده نمی کند و پردازشگر C را هم حذف کرده است . به این دلیل که جاوا یک زبان شی گرا است ، ساختار های C مثل struct , union از آن برداشته شده است . جاوا حتی بارگذاری مجدد و خواص چندگانه ارث بری از C++ را هم حذف کرده است . شاید مهمترین پارامتر ساده بودن جاوا عدم استفاده این زبان از اشاره گر ها باشد . اشاره گر ها یکی از بیشترین موجودیت های دردسرساز در C , C++ هستند . چون جاوا ساختمان ندارد و آرایه ها و رشته ها اشیاء آن هستند ، بنابراین احتیاجی به اشاره گر نیست. جاوا به طور خودکار آدرس دهی و دسترسی به محتوای موجود در یک آدرس را برای شما انجام میدهد .

جاوا حتی زباله های حافظه ای را هم به طور خودکار جمع آوری میکند (Garbage Collectin). جمع آوری آشغال فرایندی برای ترمیم خودکار حافظه انباشته شده است . بلوک هایی از حافظه که زمانی به فایل ها اختصاص داشتند اما مدتی است که از آنها استفاده نمی شود و بلوک هایی از حافظه که هنوز مورد استفاده قرار میگیرند ممکن است حرکت داده شوند تا از به هم پیوستن فضاهای خالی حافظه بلوک های خالی بزرگتری

بدست آید،بنابراین لازم نیست که نسبت به موضوع مدیریت حافظه نگران باشید ، همه اینها شما را از نگرانی در مورد اشاره گر های بی ارزش ، خطرناک و هرز های حافظه رها میکنند . بنابر این شما میتوانید وقت خود را صرف بهبود برنامه تان کنید.

۱۱-۱-۱۰-قدرتمند (Robust)

جاوا برای نوشتن نرم افزارهای قدرتمند و بسیار ایمن ساخته شده است . جاوا هنوز هم به طور قطع نرم افزار ها را تضمین نمیکند . تقریبا هنوز هم امکان نوشتن برنامه های مشکل ساز در جاوا وجود دارد ، هرچند که جاوا برخی از انواع مشخص خطاهای برنامه نویسی را حذف کرده که به طرز چشمگیری نوشتن نرم افزار های ایمن را آسان تر کرده است.

جاوا یک زبان تایپ شده قدرتمند است ، که اجازه چک شدن مشکلات و خطاهای تایپی را در زمان کامپایل می دهد . جاوا بسیار قویتر از C++ تایپ شده است که بسیاری از خصوصیات انعطاف پذیر در زمان کامپایل را از C به ارث برده است (مخصوصا هنگام اعلان توابع).جاوا به مدل اعلان صریح احتیاج دارد ، زیرا که از مدل اعلان صریح C پشتیبانی نمیکند . این مسئله مارا از اینکه کامپایلر میتواند خطاهای زمان اعلان را بدست آورد ، مطمئن میکند . مسئله ای که منجر به ایجاد برنامه های ایمن تری میشود.

یکی از چیزهایی که باعث شده که جاوا ساده باشد عدم وجود اشاره گر ها ومحاسبات بر روی آنها است . این ویژگی حتی قدرت جاوا را هم با از میان بردن یک کلاس سراسری اشاره گر افزایش میدهد.

به طور مشابه تمام دسترسی به آرایه ها و رشته ها در زمان اجرا چک می شوند تا از قطعی بودن آنها اطمینان حاصل شود .با از بین بردن امکان دوباره نویسی حافظه و داده های هرزه ، تعویض نقش اشیاء از نوعی به نوع

دیگر هم در زمان اجرا کنترل میشود تا از مجاز بودن آن اطمینان حاصل شود .

سرانجام زباله جمع کن خودکار جاوا بسیاری از عملیات پاکسازی مرتبط با معماری حافظه را راه اندازی میکند .چیزی که از خطاهای خطر ساز مرتبط با تخصیص و آزاد سازی حافظه جلوگیری میکند.

۱۱-۱-۱۱-ایمن(Secure)

یکی از دلایل پرطرفدار بودن جاوا این است که یک زبان ایمن است . این ویژگی مخصوصا به خاطر طبیعت توزیع شده ی آن بسیار مهم است و به صورتی طراحی شده است که دارای چندین لایه کنترل امنیت می باشد که شما را در برابر کد های خطرساز محافظت می‌کنند و به کاربراجازه می‌دهد که برنامه‌های ناشناخته را با خیال راحت اجرا کند .بدون وجود امنیت شما قطعا نمیخواهید که یک کد را از یک سایت تصادفی اینترنت بارگذاری کنید و به آن اجازه اجرا شدن روی کامپیوتر خودتان را هم بدهید . این دقیقا همان چیزی است که مردم هرروز با یک کد جاوا انجام میدهند . جاوا به صورت ایمن طراحی شده و چندین لایه کنترل امنیت به وجود می آورد که شما را در برابر کد های خطرساز محافظت می کنند و به کاربر اجازه میدهد که برنامه های ناشناخته را با خیال راحت اجرا کند.

همان طور که دیدیم ، برنامه جاوا نمیتواند اشاره گر ها را به حافظه یا آرایه های سرریز یا حافظه خواندنی خارج از محدوده یک آرایه یا رشته اشاره ، اشاره دهد. این خصوصیت یکی از اصلی ترین وسایل دفاع جاوا در برابر کدهای خطرساز است . دومین راه دفاع در برابر کد های خطرساز ، پردازش کدهای بایتی به صورت قابل تصدیق و تایید است که مفسر جاوا به روی هر کدی که در حال بار گذاری باشد اعمال میکند . این مراحل تایید از اینکه کد به صورت درستی ساخته شده اطمینان حاصل میکنند ، که برای مثال پشته سرریزی یا

زیرریزی نداشته باشد ، یا شامل کد های بایتی غیر مجاز نباشد.

کدهای بایتی خراب یا خطرساز ممکن است از ضعف های اجرایی در مفسر جاوا سوء استفاده کنند . لایه ای که در اینجا ما را به طور ایمن محافظت میکند ، مدل جعبه شنی (Sand box) است، کدهای ناشناخته در یک جعبه شنی قرار میگیرند جایی که میتوانند به صورت ایمن اجرا شوند ، بدون اینکه هیچ صدمه ای به بقیه اجزاء یا محیط جاوا بزنند.

وقتی یک برنامه کاربردی یا دیگر کدهای ناشناخته در جعبه شنی در حال اجرا است ، چند محدودیت در مورد کاری که میتواند انجام دهد ، وجود دارد . واضح ترین این محدودیت ها این است که هیچ دسترسی به هیچ یک از فایل های محلی سیستم وجود ندارد . در جعبه شنی محدودیت دیگری هم وجود دارد که به وسیله کلاس مدیریت امنیت اعمال میشود . این مدل در ابتدا از اینکه سیستم های امنیتی را نصب کرده اید یا نه ، مطمئن میشود ، چرا که همه کلاس های جاوا نیاز به عملیات حساسی مانند دسترسی به سیستم فایل را دارند . اگر فراخوانی به وسیله یک کد ناشناخته به صورت مستقیم یا غیر مستقیم انجام شد ، مدیر امنیت مورد استثناء را می فرستد و عملیات صورت نمی گیرد.

و سرانجام در جاوا ۱-۱ یک راه حل ممکن دیگر برای مشکلات امنیتی وجود دارد ، به وسیله ضمیمه کردن یک امضاء دیجیتالی به کد جاوا که اصل آن کد میتواند به صورت پنهانی و نهفته ساخته شود . اگر شما اعتماد خود را به یک شخص یا یک سازمان مشخص کرده باشید ، کدی که امضاء آن هویت مورد اعتماد روی آن قرار دارد ، ایمن و مطمئن است . حتی زمانیکه در حال بارگذاری شدن در شبکه است و ممکن است حتی بدون جلوگیری توسط جعبه شنی اجرا شود.

۱۱-۱-۱۲-عملکرد سطح بالا(High Performance)

جاوا یک زبان تفسیر شده است ، بنابر این هرگز به سرعت زبان کامپایل شده ای مثل C نخواهد بود . کد C کامپایل شده ۱۰ برابر سریع تر از کد های تفسیر شده ی جاوا اجرا میشود . اما قبل از اینکه به خاطر این موضوع مایوس شوید ، آگاه باشید که این سرعت بیشتر از آن چیزی است که برای برنامه های پرسرعت ، کاربردهای (GUI(Graphical User Interface ، برنامه های برمبنای شبکه ، جایی که برنامه کاربردی معمولا آماده برای اجرا شدن است ، انتظار برای کاربر که یک دستوری اعمال کند و یا انتظار برای دریافت از شبکه ، لازم است.

به علاوه قسمت هایی که به سرعت های بالا نیاز دارند ، که کارهایی از قبیل الحاق رشته ها و مقایسه را انجام میدهند ، با کد محلی جاوا اجرا میشوند.

علاوه بر این کارایی ، بسیاری از مفسر های جاوا اکنون شامل کامپایلر های فقط در زمان " just in time "نیز هستند که میتواند کدهای بایتی جاوا را برای هر نوع CPUدر زمان اجرا به کد ماشین ترجمه کند . فرمت کد بایتی جاوا با این کامپایلر های در زمان در مرکز کافی و مناسب است و انصافا کدهای خوبی تولید میکند . در حقیقت Sun ادعا میکند که کارایی کد های بایتی که به کد ماشین تبدیل شده اند ، تقریبا به خوبی کارآیی آن در C و C++ است.

اگر شما خواهان این هستید که قابل حمل بودن کد ها را قربانی بهبود در سرعت آن کنید ، میتوانید بخش قابل توجهی از برنامه خود را در C و C++ بنویسید و از روش های مخصوص جاوا برای مشترک کردن با این کد محلی جاوا استفاده کنید.

۱۱-۱-۱۳-چند نخ کشی شده(Multitbreaded)

در یک برنامه کاربردی بر مبنای GUI شبکه ای ، مثل مرورگر وب ، تصور اینکه چند چیز بتوانند به طور همزمان اجرا شوند ، آسان است . یک کاربر میتواند همزمان با اینکه دارد یک صفحه وب را میخواند به یک کلیپ صوتی گوش دهد و همزمان در پس زمینه مرورگر یک عکس را بارگذاری کند . جاوا یک زبان چندنخ کشی شده است ، که از چندین رشته اجرایی (گاهی پردازش سبک وزن خوانده میشود) پشتیبانی میکند و میتواند چندین کار را انجام دهد . یکی از مزیت های چندنخ کشی شده این است که عملکرد سطح بالایی برای کاربردهای گرافیکی برای کاربر فراهم میکند.اگر شما سعی کرده اید که با نخ ها در C و C++ کار کنید ، میدانید که کمی مشکل است . جاوا برنامه نویسی با نخ ها را بسیار آسان تر کرده است ، با به وجود آوردن زبان درون ساخته شده ای که از نخ ها پشتیبانی میکند . بسته jana.lang یک کلاس بوجود آورده است که از روش هایی برای شروع و پایان یک نخ ، و مرتب کردن ترتیب گره ها در میان چیز های دیگر ، پشتیبانی میکند .

حتی دستورات زبان جاوا از نخ ها پشتیبانی میکنند ، که با استفاده از کلمات کلیدی مطابق شده . این کلمات کلیدی علامت گذاری بخش های کد یا تمامی روش هایی را که باید فقط با یک نخ در یک زمان اجرا شوند را به شدت آسان کرده است.

به دلیل اینکه جاوا استفاده از نخ ها را بسیار ساده میکند ، کلاس جاوا در شماری از جاها از این نخ ها استفاده میکند . برای مثال هر برنامه کاربردی که انیمیشن اجرا میکند ، از نخ ها استفاده کرده است.

به طور مشابه جاوا از برنامه های نا همگام ، ورودی ها و خروجی های بلاک نشده با اخطاری به وسیله سیگنال

ها یا وقفه ها پشتیبانی نمیکند ، در این صورت شما باید یک نخ بسازید که روی هر کانال ورودی خروجی که با آن کار میکنید بلاک شده باشد.

۱۲-بنیادهای کلاس java

چیزی که باید درباره یک کلاس بدانید این است که کلاس یک نوع جدید داده را تعریف می کند . هربار که این نوع تعریف شود ، می توان از آن برای ایجاد اشیائی ازهمان نوع استفاده نمود . بنابراین ، یک کلاس قالبی (template)برای یک شی ئ است و یک شی نمونه ای (instance) از یک کلاس است . چون شی یک نمونه از یک کلاس است.غالبا" کلمات شی (object)و نمونه (instance) را بصورت مترادف بکار می بریم .

۱۲-۱-شکل عمومی یک کلاس

هنگامیکه یک کلاس را تعریف می کنید ، در حقیقت شکل و طبیعت دقیق آن کلاس را اعلان می کنید . ابتکار را با توصیف داده های موجود در آن کلاس و کدهایی که روی آن داده ها عمل می کنند ، انجام می دهید . در حالیکه کلاسها ممکن است خیلی ساده فقط شامل داده یا فقط کد باشند، اکثر کلاسهای واقعی هردو موضوع را دربرمیگیرند .

بعدا" خواهید دید که کد یک کلاس ، رابط آن به داده های همان کلاس را توصیف میکند .

یک کلاس را با واژه کلیدی class اعلان می کنند داده یا متغیرهایی که داخل یک کلاس تعریف شده اند را متغیرهای نمونه(instance variables) می نامند . کدها ، داخل روشها (methods) قرار می گیرند .

روشها و متغیرهای تعریف شده داخل یک کلاس را اعضائ (members) یک کلاس می نامند . در اکثر کلاسها ، متغیرهای نمونه یا روی روشهای تعریف شده برای آن کلاس عمل کرده یا توسط این روشها مورد دسترسی قرار می گیرند . بنابراین ، روشها تعیین کننده چگونگی استفاده از داده های یک کلاس هستند .

متغیرهای تعریف شده داخل یک کلاس ، متغیرهای نمونه خوانده شده زیرا هر نمونه از کلاس (یعنی هر شی ئ یک کلاس) شامل کپی خاص خودش از این متغیرهاست . بنابراین داده مربوط به یک شی ، جدا و منحصر بفرد از داده مربوط به شی دیگری است .

کلیه روشها نظیر ()mainهمان شکل عمومی را دارند که تاکنون استفاده کرده ایم . اما ، اکثر روشها را بعنوان publicیاstatic ا توصیف نمی کنند . توجه داشته

باشید که شکل عمومی یک کلاس ، یک روش ()main را توصیف نمی کند . کلاسهای جاوا لزومی ندارد که یک روش ()main داشته باشند . فقط اگر کلاس ، نقطه شروع برنامه شما باشد ، باید یک روش ()main را توصیف نمایید . علاوه بر این ، ریز برنامه ها (applets) اصولا "نیازی به روش ()main ندارند .

نکته : برنامه نویسان C++ آگاه باشند که اعلان کلاس و پیاده سازی روشها در یک مکان ذخیره شده و بصورت جداگانه تعریف نمی شوند. این حالت گاهی فایلهای

خیلی بزرگ java ایجاد می کند ، زیرا هر کلاس باید کاملا در یک فایل منبع

تکی تعریف شود . این طرح در جاوا رعایت شد زیرا احساس می شد که در بلند

مدت ، در اختیار داشتن مشخصات ، اعلانها و پیاده سازی در یک مکان ، امکان

دسترسی آسانتر کد را بوجود می آورد.

۱۳-آرایه ها در جاوا

یک آرایه گروهی از متغیرهای یک نوع است که با یک نام مشترک به آنها ارجاع می شود . می توان آرایه ها را برای هر یک از انواع ایجاد نمود و ممکن است این آرایه ها دارای یک یا چندین بعد باشند . برای دسترسی به یک عضو آرایه از نمایه (index)آن آرایه استفاده می شود . آرایه ها یک وسیله مناسب برای گروه بندی اطلاعات مرتبط با هم هستند.

نکته : اگر باC و++C و آشنایی دارید ، آگاه باشید . آرایه ها در جاوا بطور متفاوتی نسبت به زبانهای دیگر کار می کنند .

۱۳-۱-آرایه های یک بعدی

آرایه یک بعدی بطور ضروری فهرستی از متغیرهای یکنوع است . برای ایجاد یک آرایه ، باید یک متغیر آرایه از نوع مورد نظرتان ایجاد کنید .

۱۳-۲-آرایه های چند بعدی

در جاوا آرایه های چند بعدی در واقع آرایه ای از آرایه ها هستند . این قضیه همانطوریکه انتظار دارید ظاهر و عملکردی مشابه آرایه های چندبعدی منظم (regular)

دارد . اما خواهید دید که تاوتهای ظریفی هم وجود دارند . برای اعلان یک متغیر آرایه چند بعدی ، با استفاده از مجموعه دیگری از کروشه ها هر یک از نمایه های اضافی را مشخص می کنید.

۱۴-عملگرها ی ریاضی و بیتی

جاوا یک محیط عملگر غنی را فراهم کرده است . اکثر عملگرهای آن را می توان در چهار گروه طبقه بندی نمود : حسابی (arithmetic)،رفتار بیتی (bitwise)،رابطه ای(relational)، و منطقی(logical)جاوا همچنین برخی عملگرهای اضافی برای اداره حالتهای خاص و مشخص تعریف کرده است .

نکته : اگر با C++/C آشنایی دارید ، حتما خوشحال می شوید که بدانید کارکرد عملگرها در جاوا دقیقامشابه با C++/C است . اما همچنان تفاوتهای ظریفی وجود دارد .

انواع ساده The simple Types

جاوا هشت نوع ساده (یاابتدایی)از داده را تعریف می کند:

boolean، double، float، char، long،short bbyte int این انواع را می توان در چهار گروه بشرح زیر دسته بندی نمود :

۱) integers (اعداد صحیح) : این گروه دربرگیرندهbyte، short، int،و long میباشد که مختص ارقام علامتدار مقدار کل (whole-valued signed numbers) میباشد .

۲) floating-point numbers (اعداد اعشاری) : این گروه دربرگیرنده float وdouble

است که معرف اعداد با دقت زیاد است.

۳) characters (کاراکترها): این گروه فقط شامل char بوده که نشانه هایی نظیر حروف و ارقام را در یک مجموعهخاص از کاراکترها معرفی می کند .

۴) Boolean (بولی) : این گروه فقط شامل boolean است . که نوع خاصی ازمعرفی و بیان مقادیر صحیح / ناصحیح می باشد .

شما می توانید از این انواع همانطوریکه هستند استفاده کرده ، یا آرایه ها وانواع کلاسهای خود را بسازید .

انواع اتمی معرف مقادیر تکی و نه اشیا پیچیده هستند . اگر چه جاوا همواره شی ئ گرا است ، اما انواع ساده اینطورنیستند . این انواع ، مشابه انواع ساده ای هستند که در اکثر زبانهای غیر شی ئگرا مشاهده می شود . دلیل این امرکارایی است . ساختن انواع ساده در اشیا سبب افت بیش از حد کارایی و عملکردمی شود .

انواع ساده بگونه ای تعریف شده اندتا یک دامنه روشن و رفتاری ریاضی داشته باشند . و زبانهایی نظیرC و C++ امکان می دهند تا اندازه یک عدد صحیح براساس ملاحظات مربوط به محیط اجرایی تغییر یابد . اما جاوا متفاوت عمل می کند . بدلیل نیازهای موجود برای قابلیت حمل جاوا ،کلیه انواع داده در این زبان دارای یک دامنه کاملا تعریف شده هستند . بعنوان مثال یک int همیشه ۳۲ بیتی است ، صرفنظر از اینکه زیر بنای خاص محیطی آن چگونه باشد . این حالت به برنامه های نوشته شده اجازه می دهد تا با اطمینان و بدون درنظر گرفتن معماری خاص یک ماشین اجراشوند.

۱۵-ویژگیهای زبان برنامه نویسی**Java**

جاوا ویژگیهای متعددی دارد که آن را منحصربه فرد کرده است . جاوا هم کامپایلر دارد و اینترپرتر . توضیح : (**کامپایلر** برنامه ای است که متن برنامه را گرفته و در پایان یک فایل exe تولید می کند **.**بعد از کامپایل شدن یک برنامه ، دیگر به وجود کامپایلر نیازی نیست و می توان برنامه exe را روی هر کامپیوتر سازگاری اجرا کرد . اما **اینترپرتر** هیچ برنامه exe ای تولید نمی کند و برنامه را خط به خط اجرا می کند ، برای اجرای برنامه حتما باید اینترپرتر هم روی کامپیوتر مورد نظر موجود باشد).

هر کامپایلر فقط برای یک سیستم خاص (مانند اینتل ، اپل یا آلفا) می تواند کد اجرایی تولید کند اما کامپایلر جاوا کد اجرایی Exe تولید نمی کند و در عوض یک فایل بینابینی می سازد که بایت کد Byte code نام دارد و بایت کد چیزی شبیه زبان اسمبلی است ، اما این زبان مختص هیچ پروسسور خاصی نیست بلکه زبان اسمبلی یک ماشین ویژه بنام ماشین مجازی جاوا (Java Virtual Mashing) دارد که روی ماشین مجازی جاوا اجرا می شود ، دستورات فایل بایت کد را به دستورات قابل فهم برای پروسسوری که روی آن اجرا می شود تبدیل خواهد کرد **.**

۱۶-برنامه نویسی با جاوا

برنامه ای می نویسیم یک برنامه متکی به خود است بنام **.** Hello World تفاوت یک برنامه متکی به خود و یک اپلت آن است که در برنامه متکی به خود (که از این به بعد به آن فقط برنامه خواهیم گفت) از متدی بنام main()استفاده می شود در حالیکه اپلت چنین متدی ندارد **.**

برنامه Hello World بسیار ساده است و فقط جمله "Hello World" را نمایش می دهد با این حال می

توان از آن به عنوان سنگ بنای برنامه های پیشرفته تر استفاده کرد چون تمام برنامه های جاوا ساختار کلی مشابهی دارند.

برای کامپایل کردن برنامه باید از کامپایلر جاوا (javac) استفاده کرد روش کار چنین است :

Java HelloWorld. java

کامپایلر بعد از پایان کار یک فایل کلاس بنام HelloWorld. Class تولید خواهد کرد . اصولاً کامپایلر برای هر کلاس برنامه یک فایل کلاس جداگانه تولید خواهد کرد . فایل تولید شده یک فایل اجرایی مستقل نیست . برای اجرای این فایل باید از اینترپرتر جاوا استفاده کرد . اینترپرتر جاوا، Java نام دارد . برای اجرای فایل کلاس تولید شده چنین باید کرد :

Java HelloWorld

کامپایلر جاوا و اینترپرتر جاوا دو چیز متفاوتند . کامپایلر از فایل متن برنامه یک فایل کلاس می سازد و اینترپرتر فایل کلاس را اجرا می کند.

۱۷-دستورات و عبارات

یک دستور ساده ترین کاری است که در جاوا می توان انجام داد ، هر دستور یک عمل انجام می دهد . در ذیل چند دستور ساده جاوا را مشاهده می کنید.

گاهی یک دستور مقدار برگشتی دارد مثل جمع دو عدد . به این نوع دستورات عبارت گفته می شود. مهمترین چیزی که در مورد دستورات لازم جاوا باید به خاطر داشته باشید این است که در پایان هر دستور یک سمی کولون (لازم است . در غیر اینصورت برنامه بدرستی کامپایل نخواهد شد.

هر جا که بتوان از یک دستور استفاده کرد از یک دستور مرکب ، یا بلوک ، هم می توان استفاده کرد . دستورات یک بلوک درون یک جفت آکولاد ({}) قرار می گیرند.

از این لحاظ جاوا و سی پلاس پلاس با هم مشابهند.

۱۸-متغیرها و انواع داده

یک متغیر (Variable) مکانی است در حافظه که می توان مقادیری را در آن ذخیره کرد.. هر متغیر دارای سه چیز است : نام ، نوع ، مقدار . قبل از استفاده از یک متغیر باید آنرا تعریف (Declare) کنید .

در جاوا سه نوع متغیر وجود دارد : متغیر وهله ، متغیر کلاس ، متغیر محلی.

متغیرهای وهله خواص یک شیء خاص را در خود نگه می دارند .

متغیرهای کلاس مانند متغیرهای وهله هستند با این تفاوت که به وهله های یک کلاس مربوط می شوند .

متغیرهای محلی اغلب در درون متدها مورد استفاده قرار می گیرند و برای نگهداری مقادیر درون متد هستند ، در بلوک ها هم می توان از متغیرهای محلی استفاده کرد.

همین کد اجرای متد (یا بلوک) به پایان رسید ، متغیرهای محلی دورن آن هم از بین می روند .

با آن که نحوه تعریف این سه نوع متغیر یکسان است ، اما نحوه دسترسی به آنها کمی متفاوت است .

نکته:

بر خلاف سی پلاس پلاس ، جاوا متغیر عمومی (همگانی) ندارد . برای ارتباط بین اشیاء از متغیرهای وهله و

کلاس می توان استفاده کرد . به یاد داشته باشید که جاوا یک زبان شی گرا است و شما به هم کنش اشیاء فکر کنید.

۱۸-۱-تعریف متغیرها

برای استفاده از یک متغیر در برنامه های جاوا و سی پلاس پلاس ، ابتدا باید آن را تعریف کنید. تعریف متغیر از یک نوع و یک نام تشکیل می شود.

تعریف یک متغیر می تواند در هر کجای یک متد انجام شود ، ولی بهتر است در ابتدای متد باشد.

چند متغیر از یک نوع را می توان در یک جا تعریف کرد و حتی می توان در هنگام تعریف متغیر به آن مقدار داد .

به متغیرهای محلی قبل از استفاده حتماً باید مقدار داد، در غیر اینصورت برنامه بدرستی کامپایل نخواهد شد .

مقدار دادن اولیه به متغیرهای وهله و کلاس الزامی نیست . چون این متغیرها هنگام تعریف دارای مقدار پیش فرض هستند.

۱۸-۲-نکاتی درباره نام متغیرها

نام متغیرها در جاوا و سی پلاس پلاس می تواند با یک حرف ، زیر خط (_) یا علامت دلار ($) شروع شود ولی نباید با یک عدد آغاز شود . بعد از حرف اول می توان از تمام حروف دیگر استفاده کرد ولی هنگام استفاده از ٪ ، * ، @ و مانند آنها (که در اپراتورها هستند) به مشکلاتی که می تواند بروز کند دقت کنید .

جاوا بر خلاف سی پلاس پلاس از کاراکترهای یونی کد (Unicode) استفاده می کند ، یونی کد استانداردی است که بر خلاف اسکی برای هر کاراکتر از دو بایت استفاده می کند و می تواند در آن واحد تا 65000 کاراکتر را پشتیبانی کند . البته تمام کاراکترهای ذیل 00CO رزرو شده اند و شما می توانید از کاراکترهای بالای این حد آزادانه استفاده کنید .

نام متغیرها در هر دو زبان نسبت به نوع حروف حساس است و این دقت زیادی را در هنگام نوشتن برنامه ها می طلبد . به همین دلیل رعایت یک قرارداد هنگام نامگذاری متغیرها می تواند کمک بزرگی در مقابله با مشکلات احتمالی باشد .

قراردادهای استفاده شده در نامگذاری متغیرهابدین صورت است: نام های با معنی ، ترکیب چند کلمه ، کلمه اول با حرف کوچک شروع می شود ، کلمات بعدی با حرف بزرگ شروع می شوند .

۱۸-۳-انواع متغیرها

هنگام تعریف هر متغیر علاوه بر نام آن باید نوع آن هم مشخص شود . نوع متغیر تعیین می کند که یک متغیر چه مقادیری را می تواند بگیرد . هر متغیر می تواند یکی از سه نوع ذیل باشد :

- یکی از هشت نوع داده اولیه
- نام یک کلاس یا واسط
- یک آرایه

هشت نوع داده اولیه جاوا برای کار با اعداد صحیح ، اعداد اعشاری ، کاراکترها و مقادیر منطقی (درست یا نادرست) هستند ، به آنها انواع اولیه گفته می شود. در جاوا چهار نوع عدد صحیح (Integer) وجود دارد.

انواع داده ای علامت دار می توانند اعداد مثبت و منفی را در خود ذخیره کنند . نوع متغیر انتخاب شده به عددی که می خواهید ذخیره کنید بستگی دارد . اگر عدد بزرگتر از متغیر باشد ، مقداراضافه حذف خواهد شد. برای ذخیره کردن اعدا ددارای ممیز از نوع اعشاری (با ممیز شناور floating point–) استفاده می شود . اعداد اعشاری در جاوا از استاندارد IEEE 754 تبعیت می کنند .

دو نوع عدد اعشاری وجود دارد :float 32bit دقت ساده و Double 64bit دقت مضاعف .

نوع داده کاراکتر (char) برای ذخیره کردن یک کاراکتر است . چون جاوا از یونی کد استفاده می کند هر متغیر Char دارای ۱۶ بیت (بدون علامت) خواهد بود .

آخرین نوع داده اولیه در جاوا نوع منطقی (Boolean) است که می تواند دو مقدار True یا False بگیرد . بر خلاف C++ ، نوع منطقی یک عدد نیست و نباید آن را با اعداد مقایسه کرد . علاوه بر این انواع ، متغیرهای جاوا می توانند از نوع کلاس هم باشند:

این متغیرها یک وهله از کلاس مربوطه هستند .

در جاوا (بر خلاف C++)، دستور typedef وجود ندارد . برای تعریف انواع جدید در جاوا ، ابتدا یک کلاس جدید ایجاد کنید ، و سپس متغیری از نوع این کلاس تعریف کنید.

۱۸-۴-مقدار دادن به متغیرها

بعد از تعریف متغیرها ، با استفاده از عملگر = می توان به آنها مقدار داد:

توضیحات : (Comments)

سه نوع توضیح در این زبانها وجود دارد . در نوع اول در جاوا و C++ از /* برای شروع و از */ برای ختم آن استفاده می شود . کامپایلر هر چه را بین این دو علامت بیابد نادیده خواهد گرفت.

*/

این نوع توضیح می تواند چند خطی باشد.

در نوع دوم برای توضیحات تک خطی از // می توان استفاده کرد:

در نوع سوم توضیح که برنامه javadoc از آن استفاده می کند با /** شروع و با */ پایان می یابد . این نوع توضیح از همه نظر شبیه نوع اول است

۱۹-واژه ها

برای نمایش مقادیر مشخص و ساده از واژه ها (Literal) استفاده می شود . این واژه ها می تواند عدد ، کاراکتر ، رشته یا مقادیر منطقی باشند .

۱۹-۱-واژه های عددی

چندین واژه صحیح وجود دارد . مثلاً ، ۴ یک واژه صحیح از نوع int است . اگر عدد واژه از یک int بزرگتر باشد بطور خودکار به نوع long تبدیل خواهد شد . می توانید حتی یک عدد کوچک از نوع long داشته باشید ، برای اینکار در جاوا باید جلوی عدد از حرف l یا L استفاده کنید . مثلاً ، ۴ Lعدد صحیح ۴ را در یک واژه long ذخیره می کند .

واژه های اعشاری معمولاً دو قسمت دارند . یک قسمت صحیح و یک قسمت اعشاری . تمام واژه های

اعشاری صرفنظر از دقت عدد از نوع double خواهند بود مگر اینکه با قید حرف f یاF در جلوی آن تصریح شود که عدد مزبور از نوع float باید باشد . واژه های اعشاری را با استفاده از حرف E یا e می توان به صورت نمایش هم نوشت 10e45 – یا -۳.۶ . E-2

۱۹-۲-واژه های منطقی

در جاوا یک واژه منطقی فقط می تواند معادل کلمات کلیدی True یا False باشد .

۱۹-۳-واژه های کاراکتری

یک واژه کاراکتری عبارت است از یک حرف که با علامت نقل محصور شده باشد 'a' ، '#' و غیره .در جاوا واژه های کاراکتری به صورت یونی کد (۱۶ بیتی) ذخیره می شوند در حالیکه در سی پلاس پلاس به صورت اسکی ذخیره می شوند.

۱۹-۴-واژه های رشته ای

یک رشته (String) عبارتست از مجموعه چند کاراکتر . هر رشته در جاوا وهله ایست از کلاس String .بر خلاف C++ ، رشته ها در جاوا آرایه ساده کاراکترها نیستند(اگر چه بسیاری از خواص آرایه ها را ندارند) . چون رشته های جاوا اشیاء حقیقی هستند ، متدهایی دارند که کار با آنها را بسیار زنده می سازند .

یک واژه رشته ای عبارت است از چند کاراکتر که در علامت نقل دو گانه محصور شده باشند:

اینکه شما می توانید در رشته های جاوا از کاراکترهای یونی استفاده کنید بدان معنا نیست که می توانید آن کاراکترها را ببینید . برای دیدن اینگونه کاراکترها کامپیوتر یا سیستم عامل شما باید از یونی کد پشتیبانی کند و فونت بکار رفته را هم داشته باشد .

-تفاوت واژه های رشته ای با دیگر انواع واژه های رشته ای (بر خلاف دیگر واژه ها) اشیاء واقعی (وهله های کلاس String)هستند .

۲۰-عبارات و عملگرها

عبارت (expression) ساده ترین واحد عملیاتی است .

عبارت دستوری است که یک مقدار بر می گرداند . در عبارات از علایم خاصی استفاده می شود که به آنها عملگر (Operator) گفته می شود . ساده ترین نوع عبارات به مقایسه مقادیر و محاسبه می پردازد . عبارات را می توان به یک متغیر نسبت داد چون دارای مقدار برگشتی است.

عملگرهای جاوا عبارتند از عملگرهای محاسباتی ، انواع مختلف انتساب مقدار ، افزایش و کاهش ، و عملیات منطقی.

۲۰-۱-محاسبات

هر دو زبان(C++وjava) دارای پنج عملگر محاسباتی است .

هر عملگر دو عملوند (Operand) لازم دارد . از عملگر تفریق (-) برای منفی کردن اعداد هم می توان استفاده کرد . تقسیم اعداد صحیح دارای خارج قسمت صحیح خواهد بود و مقدار اعشار آن نادیده گرفته خواهد شد . مثلاً حاصل تقسیم ۳۱/۹ معادل ۳ خواهد بود . عملگر ٪ باقیمانده تقسیم را بر می گرداند . برای مثال حاصل عبارت ۳۱٪۹ معادل ۴ خواهد شد .

حاصل عملیات دو عدد صحیح همواره یک عدد صحیح خواهد بود . نوع داده مقدار برگشتی با نوع داده عملوندی که جای بیشتری اشغال می کند معادل خواهد بود .

۲۰-۲-انتساب مقادیر

نسبت دادن مقدار (assignment) به یک متغیر نوعی عبارت است . در حقیقت ، چون هر عبارت یک مقدار برگشتی دارد می توان چند عبارت انتسابی را به هم پیوند داد :

```
X=y=z=0;
```

همیشه ابتدا مقدار عبارت سمت راست محاسبه شده و به عبارت سمت چپ نسبت داده می شود . این بدان معناست که عبارت x=x+2 یک عبارت صحیح است ؛ به x دو واحد اضافه شده و حاصل در x قرار داده می شود . عملیاتی از این دست چنان در برنامه نویسی رایج است که جاوا برای آن عملگر ویژه ای دارد (این ویژگی را جاوا از C++ بعاریت گرفته است).

۲۰-۳-افزایش و کاهش

مانند C++ در جاوا هم برای اضافه یا کم کردن ۱ از عملگرهای ++ یا – –استفاده می شود . مثلاً عبارت X++به X یکی اضافه می کند و معادل x=x+1 است . عبارت X-- هم یکی از x کم می کند . بر خلاف C++،درجاوا عملوند در عبارات ++ یا – – می توانند اعشاری هم باشد .

عملگرهای ++ و -- می توانند قبل یا بعد از متغیر قرار گیرند . در عبارات ساده این موضوع چندان اهمیتی ندارد ، ولی در عبارات پیچیده می تواند باعث بروز تفاوتهایی شود . برای مثال ، به دو عبارت ذیل دقت کنید :

```
y = x++;
y = ++ x;
```

نتیجه این دو عبارت بسیار متفاوت خواهد بود . در عملگر پسوند ++ یا --x ، y مقدار x را قبل از تغییر آن می گیرد ؛ در عملگر پیشوند (-- یا x--)مقدار x بعد از تغییر به y داده می شود .

۲۱- مقایسه ها

هر دو زبان برای مقایسه مقادیر عبارات متعددی دارد . تمام این عبارات یک مقدار(Boolean یعنی True یا False)بر می گرداند.

۲۱-۱-عملگرهای منطقی

عباراتی که مقدار برگشتی آنها Boolean است را می توان با اپراتورهای منطقی AND ، OR ، XOR با هم ترکیب کرد .

برای AND کردن دو عبارت باید از & یا && استفاده کرد . حاصل عبارت زمانی درست خواهد بود که هر دو قسمت درست باشند ، در غیر اینصورت حاصل عبارت نادرست خواهد شد . تفاوت این دو عملگر در نحوه ارزیابی عبارت است . با عملگر &، هر دو قسمت عبارت ارزیابی می شوند . اما با عملگر && اگر سمت چپ عبارت نادرست باشد ، برای تمام عبارت مقدار False برگشت داده می شود و سمت راست عبارت ارزیابی نخواهد شد .

برای OR کردن دو عبارت از | یا || استفاده می شود . حاصل عبارت زمانی درست خواهد بود که یکی یا هر

دو قسمت آن درست باشند ، فقط وقتی عبارت نادرست است که هر دو قسمت آن نادرست باشند . با عملگر | هر دو قسمت عبارت ارزیابی می شوند . ولی عملگر || اگر قسمت اول درست باشد ، برای تمام عبارت مقدار Trueبرگشت داده می شود و سمت راست عبارت فقط ارزیابی نخواهد شد .

هنگام XOR کردن دو عبارت (که عملگر آن ^ است) فقط زمانی حاصل عبارت درست خواهد بود که در قسمت آن ارزش متضاد داشته باشد و اگر هر دو قسمت عبارت هم ارزش باشد ، حاصل عبارت نادرست خواهد شد . در کل ، عملگرهای || و && برای عملیات منطقی و عملگرهای | ، & و ^ برای عملیات منطقی بیت گرا (Bitwise) مورد استفاده قرار می گیرند .

عملگر (!) NOT فقط روی یک آرگومان عمل کرده و ارزش آن را معکوس می کند . مثلاً ، اگر x درست باشد ، ! X نادرست خواهد بود.

بر خلاف C++ ، درجاوامتغیرهای Boolean فقط مقادیر True یا False می گیرند و نمی توانند عدد بگیرند .

۲۱-۲-عملگرهای بیت گرا

عملگرهای بیت گرا روی بیت های عملوند ها عمل می کنند . چون عملیات بیت گرا جزء مباحث پیشرفته برنامه نویسی است .

۲۲-تفاوتهای دوزبان

جاوا به زبان C++(نتیجه مستقیم زبان C)وابسته است . بسیاری از خصلتهای جاوا بطور مستقیم از این

دو زبان گرفته شده است . دستور زبان جاوا منتج ازدستور زبان C است . بسیاری از جنبه های oop زبان جاوا از C++بعاریت گرفته شده است . در حقیقت بسیاری از خصلتهای زبان جاوا از این دو زبان مشتق شده یا باآنها مرتبط است . علاوه بر این ، تولید جاوا بطور عمیقی متاثراز روال پالایش وتطبیقی است که طی سه دهه گذشته برای زبانهای برنامه نویسی موجود پیش آمده است .

۲۲-۱-تفاوت وراثت در جاوا و سی پلاس پلاس

۱. در سی پلاس پلاس، وراثت چندگانه مستقیماً پشتیبانی می شود ولی در جاوا این مسئله وجودندارد ولی می شود از چند Interface چندگانه وراثت کرد.

۲. در سی پلاس پلاس، چیزی به نام Interface که کاملا Abstract باشد نداریم ولی در جاواوجود دارد.

۳. سی پلاس پلاس انواع وراثت را دارد یعنی وراثت می تواندpublic ، privateو یا protected باشه ولی در جاوا وراثت فقط به صورت public امکانپذیر هست.

این یکی از معجزات جاوا هست که وراثت چند گانه رو نداردبلکه interface دارد.

۴. فرق چندانی به لحاظ مفهومی (OOP) ندارند.فرق در میزان قابلیت نگهداری کد ها و میزان پیچیدگی مورد نیاز برنامه نویس است.در واقع هر دومفاهیم مختلف OOP را با ابزار مختلفی پشتیبانی می کنند.

۲۲-۲-تفاوت گرامر در جاواوسی پلاس پلاس

گرامر جاوا خیلی بزرگتر از سی پلاس پلاس است. مثل سی پلاس پلاس که ترکیب ساختارها و برنامه‌های شی‌گرا می‌باشد، نیست. بلکه زبان جاوا یک زبان شی‌گرای خالص است. همهٔ کدهایی که داخل کلاس نوشته می‌شود و همهٔ چیزهایی که داخل شی است، با استثنائات نوع دادهٔ اصلی، که به صورت کلاس نیستند، برای اجرا. جاوا بسیاری از ویژگی‌ها را پشتیبانی می‌کند. از کلاس‌ها برای ساده‌تر کردن زبان و جلوگیری از رخداد خطا.

دلایل استفاده ازOOP :

۱. سادگی و سرعت در ایجاد برنامه های بزرگ

۲. پشتیبانی ساده تر و ارزان تر

۳. مناسب برای پروژه های گروهی

۴. مناسب برای طراحی و پیاده سازی

۲۳-یادگیری کدام زبان راحت تر است ؟

زبان جاوا در مقابل زبانی مثل سی پلاس پلاس ساده تر و یادگیری آن آسانتر است. این آسانتر بودن به سادگی به دست نیامده‌است بلکه با حذف بسیاری از موارد که باعث قدرتمند تر بودن زبان سی پلاس پلاس بوده‌اند ایجاد شده‌است. مهم‌ترین این موارد اشاره گرها و وراثت چندگانه بوده‌اند که در زبان جاوا یافت نمی‌شوند.از

آنجایی که جاوا زبانی با عدم وابستگی به بستر می‌باشد پس استفاده از توابع سیستم عامل را در برنامه نمی‌پذیرد. به همین صورت نمی‌توان از واسط های برنامه نویسی غیر از جاوا در آن استفاده نمود.

۲۴-برتری‌های جاوا نسبت به زبان سی پلاس پلاس

جاوا یک زبان برنامه نویسی است که در اوایل دهه ۹۰ توسط Java Soft ، بخش نرم افزاری شرکت Sun توسعه داده شد . هدف آن بود که جاوا زبانی ساده ، قوی و همه منظوره باشد . جاوا تمام جنبه های مثبت C++را در خود دارد ، و آن چیزهایی که برنامه نویسان C++ از آن نفرت داشته اند (مانند وراثت چند گانه ، تحریف اپراتورها و اشاره گرها) را به کناری گذاشته است .

مهمترین ویژگیهای جاوا این است که اساساً شیء گرا است . اولین ادعای OOP توانایی استفاده مجدد از کد است ، چیزی که C++ با تمام ادعاهایش هرگز نتوانست بدان دست یابد . تصورش را بکنید که با صرف کمی وقت بتوانید برنامه ای بنویسید که در سیستم های ویندوز ، یونیکس و مکینتاش براحتی اجرا شود . همین که یک شرکت نرم افزاری بتواند برای تمام پلت فرم های موجود در آن واحد پروژه ای را تولید کند (و مقادیر عظیمی پول صرفه جویی کند) خود می تواند بهترین دلیل اقبال جاوا باشد و امروز دیگر همه (و نه فقط شرکتهای نرم افزاری) به سمت جاوا کشیده شده اند . با این ویژگی (استقلال از پلت فرم) یک برنامه نویس می تواند برای سیستمی برنامه بنویسد که هرگز با آن کار نکرده است . این ویژگی اصلی ترین علت توفیق جاوا در اینترنت است . اینترنت شبکه پیچیده ای است از میلیونها کامپیوتر مختلف در سراسر دنیا ، و مقاومت در مقابل این وسوسه که بتواند برنامه ای بنویسد که روی تمام این سیستم های متفاوت و نامتجانس اجرا شود

چندان ساده نیست .

جاوا یک زبان بسیار ساده است چون شما را وادار نمی کند تا در محیط جدید (و نا آشنایی) کار کنید و این برای کسانی که اطلاعات فنی ناچیزی درباره کامپیوتر دارند بسیار مهم است . ساختار زبان جاوا در نگاه اول بسیار شبیه C و C++ است و این به هیچ وجه تصادفی نیست. C زبانی است ساخت یافته و C++ زبانیست شیء گرا و مهمتر از همه قسمت اعظم برنامه نویسان دنیا از آنها استفاده می کنند از سوی دیگر این شباهت حرکت به طرف جاوا را برای این قبیل افراد ساده خواهد کرد. بنابراین طراحان جاوا برای اجتناب از دوباره کاری از زبانهای C و C++ بعنوان مدل استفاده کردند .

جاوا با دور انداختن اشاره گرها و بر دوش کشیدن بار مدیریت حافظه ، برنامه نویسان C و C++را برای همیشه از این کابوس ها رهایی بخشیده است . علاه بر آن چون جاوا زبانی برای اینترنت است، از ایمنی و حفاظت ذاتی بالایی برخوردار است . طراحان جاوا از ابتدا یک محیط برنامه نویسی امن را مد نظر داشته اند . مسئله حفاظت سیستم ها رابطه تنگاتنگی با اشاره گرها دارد . اکثر مهاجمان برای ورود غیر قانونی به سیستم های دیگران از این اشاره گرها استفاده می کنند و جاوا با حذف اشاره گرها این راه را سد کرده است . جاوا مکانیزم های حفاظتی دیگری هم دارد.

جاوا شباهت‌هایی به سی‌پلاس‌پلاس دارد، ولی قابلیت انتقال آن بهتر است و استفاده از آن ساده تر از C++ است. همچنین مدیریت حافظه نیز توسط خود ماشین مجازی جاوا انجام می‌شود. طراحی این زبان به گونه ایست که دارای اطمینان بسیاری بوده و وابسته به سیستم عامل خاصی نیست. و دلیل این موضوع این است که جاوا یک ماشین مجازی در سیستم شما راه می‌اندازد و برنامه‌ها را در آن ماشین مجازی اجرا می‌کند. این ماشین مجازی «ماشین مجازی جاوا» یا به اختصار JVM نامیده می‌شود.

۲۵-مقایسه سرعت اجرای دوزبان:

راه انداختن ماشین مجازی جاوا باعث دو مشکل می‌شود. هنگامی که نرم‌افزار شما در یک ماشین مجازی اجرا می‌شود سرعت کمتری خواهد داشت، همچنین شما نیاز دارید قبل از اجرای برنامه‌های جاوا یکبار سیستم مجازی جاوا را که حجم نسبتاً بالایی دارد، از اینترنت بارگذاری و یا از جای دیگری تهیه و نصب کنید ولی مزیت آن عدم وابستگی به سیستم عامل مقصد است.

مهم‌ترین ایرادی که برنامه نویسان سایر زبان‌ها به زبان جاوا می گیرند سرعت اجرایی بسیار پایین جاوا است. یک برنامه جاوا به صورت بایت کد می باشد و باید در ماشین مجازی جاوا اجرا گردد. به همین دلیل سرعت اجرای پایینی را در مقابل زبان‌های قدرتمندی همچون C++ دارد. به صورت دیگر یک برنامه C++ به طور متوسط تا ۱۰ برابر سریعتر از برنامه مشابه جاوا اجرا می‌گردد.

۲۶-مقایسه دوزبان ازلحاظ شی گرایی و وراثت

جاوا علی رقم شی گرا بودن در بخشی از قسمت‌ها برای ایجاد انعطاف بیشتر یا بازاریابی بهتر برخی اصول شی گرایی را نادیده گرفته است. از جمله این قسمت‌ها قابلیت بازتابش Reflection می باشد. هدف اصلی بازتابش این است که استفاده مجدد از کدها و گسترش کدهای موجود و مهم‌تر از همه نوشتن برنامه‌های الحاقی آسان گردد ولی این مهم با زیر پا گذاشتن بعضی اصول ممکن شده است. برای نمونه با کمک بازتابش به راحتی می‌توان متدهای خصوصی دیگر کلاس‌ها را فراخوانی کرد!

زبان جاوا در مقابل زبانی مثل C++ ساده تر و یادگیری آن آسانتر است. این آسانتر بودن به سادگی به دست

نیامده است بلکه با حذف بسیاری از موارد که باعث قدرتمند تر بودن زبان C++ بوده‌اند ایجاد شده است. مهم‌ترین این موارد اشاره گرها و وراثت چندگانه بوده‌اند که در زبان جاوا یافت نمی شوند. از آنجایی که جاوا زبانی با عدم وابستگی به بستر می باشد پس استفاده از توابع سیستم عامل را در برنامه نمی پذیرد .به همین صورت نمی‌توان از واسط‌های برنامه نویسی غیر از جاوا در آن استفاده نمود.

جاوا از وراثت منفرد Single استفاده می کند . وراثت منفرد یعنی هر کلاس جاوا می تواند فقط یک فوق کلاس داشته باشد . اما عکس آن درست نیست ، یعنی یک کلاس می تواند چندین زیر کلاس داشته باشد . در زبانهای شیء گرای دیگر ، مانندC++ ، یک کلاس می تواند از چند فوق کلاس به ارث ببرد . به این وضعیت وراثت چند گانه (Multiple) گفته می شود . با وراثت چند گانه می توان کلاس های فوق العاده جالبی بوجود آورد ، ولی کد نویسی آنها بسیار دشوار است.

دیدید که در جاوا هر کلاس فقط از یک فوق کلاس ارث می برد . با اینکه وراثت منفرد برنامه نویسی را ساده تر می کند ولی کمی محدودتر هم هست . مثلاً ، اگر در شاخه های مختلف یک سلسله مراتب متدهای مشابهی داشته باشید ، باید تمام آنها را جداگانه پیاده سازی کنید . جاوا با استفاده از مفهومی بنام واسط مشکل به اشتراک گذاشتن متدها را حل کرده است .

واسط عبارت است از مجموعه ی نام چند متد ، بدون تعریف آنها ، که واسط آنها در اختیار کلاس استفاده کننده می گذارد .

یک کلاس جاوا می تواند در آن واحد از چندین واسط استفاده کند ، و با این کار کلاس های بسیار متفاوت می توانند رفتارهای مشابهی داشته باشند .

در جاوا کلاس و واسط های مرتبط با هم در یک بسته گرد آورده می شوند . کلاس های اصلی جاوا در بسته

ای بنام java گرد آورده شده اند و فقط محتویات این بسته است که در تمام نسخه های جاوا ثابت می ماند . البته در بسته Java بسته های دیگری وجود دارند ولی بستهJava.lang به طور پیش فرض در اختیار تمام برنامه هاست . برای استفاده از بسته های دیگر باید آنها را به طور صریح تعریف کرد . نام بسته ها و کلاس ها در هنگام تعریف با نقطه (.) از هم جدا می شوند . مثلاً برای استفاده از کلاس Color که در بسته awt (که خود در داخل بسته Java می باشد) قرار دارد ، باید چنین نوشت.java.awt.Color :

۲۶-۱-حل مشکل وراثت چندگانه درسی پلاس پلاس بوسیله جاوا

وجود وراثت چندگانه در زبانی مانند سی پلاس پلاس، باعث ایجاد مشکلات اساسی‌ای می‌گردید که اکثر برنامه‌نویسان سی پلاس پلاس از آن دوری می‌کرده و هنوز هم می‌کنند. ولی قابلیت چندریخته شدن یک کلاس از لحاظ شی گرایی بسیار مهم بوده و بنابراین توجیهی برای وجود وراثت چندگانه را فراهم می‌نمود. در جاوا با وارد شدن مفهومی به نام واسط برنامه‌سازی (Interface)، دیگر نیازی به وجود وراثت چندگانه احساس نشد و بنابراین وراثت چندگانه اززبان جاوا حذف گردید. در حال حاضر اکثر طراحان برنامه‌ها حتی به این نتیجه رسیده‌اند که وراثت تکی هم باعث ایجاد مشکل بوده و تا آنجایی که می‌شود باید از Composition استفاده نمود .

از ابتدای بوجود آمدن جاوا، کتابخانهJNI (Java Native Interface) در آن وجود داشته که قابلیت فراخوانی و دستکاری برنامه‌هایی در سی پلاس پلاس و ... را می‌داده که از نمونه‌های آن می‌توان به Jtwain که یک بسته‌ایست که از کتابخانه‌های ویندوز برای اسکن عکس استفاده می‌کند، یا SWT که یک بسته

نرم‌افزاریست که از کتابخانه‌های ویندوز و لینوکس (برحسب سیستم عامل) برای ساخت واسط کاربری (UI) استفاده می‌کند، نام برد.

جاوا شبیه به سی پلاس پلاس ،اما کوچک‌تر، قابلیت انتقال آن بهتر و استفاده از آن ساده تر از سی پلاس پلاس است. زیرا دارای قابلیت‌های فراوان بوده و مدیریت حافظه را خود انجام می‌دهد.طراحی این زبان به گونه ایست که دارای اطمینان بسیار بوده و و وابسته به سیستم عامل خاصی نیست .(به عبارت دیگر می‌توان آن را روی هر کامپیوتر با هر نوع سیستم عاملی اجرا کرد.) و دلیل آن هم این است که برنامه‌های جاوا به صورت کدهای بیتی همگردانی (کامپایل) می‌شوند. که مانند کد ماشین بوده و به ویژه وابسته به سیستم عامل خاصی نیست. به این ترتیب جاوا برای نوشتن برنامه‌های کاربردی وب مناسب است، زیرا کاربر از طریق انواع مختلف رایانه‌ها می‌تواند برنامه‌های وبی جاوا را اجرا کند.

جاوا تمام جنبه های مثبت C و سی پلاس پلاس را در خود دارد ، و آن چیزهایی که برنامه نویسان سی پلاس پلاس از آن نفرت داشته اند (مانند وراثت چند گانه ، تحریف اپراتورها و اشاره گرها) را به کناری گذاشته است .

مهمترین ویژگیهای جاوا این است که اساساً شیء گرا است . اولین ادعای OOP توانایی استفاده مجدد از کد است : چیزی که سی پلاس پلاس با تمام ادعاهایش هرگز نتوانست بدان دست یابد . اما در اولین قدم خواهید دید جاوا در این زمینه تا چه حد اندازه صحت دارد . تصورش را بکنید که با صرف کمی وقت بتوانید برنامه ای بنویسید که در سیستم های ویندوز ، یونیکس و مکینتاش براحتی اجرا شود . همین که یک شرکت نرم افزاری بتواند برای تمام پلاتفرم های موجود در آن واحد پروژه ای را تولید کند (و مقادیر عظیمی پول صرفه جویی

کند) خود می تواند بهترین دلیل اقبال جاوا باشد و امروز دیگر همه (و نه فقط شرکتهای نرم افزاری) به سمت جاوا کشیده شده اند . با این ویژگی (استقلال از پلاتفرم) یک برنامه نویس می تواند برای سیستمی برنامه بنویسد که هرگز با آن کار نکرده است . این ویژگی اصلی ترین علت توفیق جاوا در اینترنت است . اینترنت شبکه پیچیده ای است از میلیونها کامپیوتر مختلف در سراسر دنیا ، و مقاومت در مقابل این وسوسه که بتواند برنامه ای بنویسد که روی تمام این سیستم های متفاوت و نا متجانس اجرا شود چندان ساده نیست .

ساختار زبان جاوا در نگاه اول بسیار شبیه C و سی پلاس پلاس است و این به هیچ وجه تصادفی نیست . C زبانی است ساخت یافته و سی پلاس پلاس زبانیست شیء گرا و مهمتر از همه قسمت اعظم برنامه نویسان دنیا از آنها استفاده می کنند از سوی دیگر این شباهت حرکت به طرف جاوا را برای این قبیل افراد ساده خواهد کرد بنابراین طراحان جاوا برای اجتناب از دوباره کاری از زبانهای C و سی پلاس پلاس بعنوان مدل استفاده کردند .

جاوا با دور انداختن اشاره گرها و بر دوش کشیدن بار مدیریت حافظه ، برنامه نویسان C وسی پلاس پلاس را برای همیشه از این کابوس ها رهایی بخشیده است . علاه بر آن چون جاوا زبانی برای اینترنت است ، از ایمنی و حفاظت ذاتی بالایی برخوردار است . طراحان جاوا از ابتدا یک محیط برنامه نویسی امن را مد نظر داشته اند . مسئله حفاظت سیستم ها رابطه تنگاتنگی با اشاره گرها دارد . اکثر مهاجمان برای ورود غیر قانونی به سیستم های دیگران از این اشاره گرها استفاده می کنند و جاوا با حذف اشاره گرها این راه را سد کرده است .

۲۷-جاوادرمقابل سی شارپ

سی شارپ همانند جاوا زبانی ساده ، شی گرا ، تفسیری ، قدرتمند ، امن ، قابل حمل ، قابلیت بالا و قابلیت اجرای برنامه های چند نخی را دارا می باشد . سی شارپ برنامه نویسی مبتنی بر کامپوننت را حمایت کرده و گزینه ای مناسب برای نوشتن برنامه های توزیع شده وب می باشد . نمی توان گفت ویندوز تنها سیستم عاملی است که سی شارپ با آن در ارتباط است بلکه پر استفاده ترین آنها می باشد . سی شارپ و جاوا هر دو از زبان سی و سی پلاس پلاس الهام گرفته و مشتق شده اند و نمی توان به سی شارپ به عنوان نسل بعدی جاوا نگریست .سی شارپ بر پایه .NET گسترش داده شده است و یکی از انتخابهای اصلی برای توسعه برنامه های کاربردی .NET می باشد .به طور خلاصه c# زبانی از نظر مفهومی شبیه C ، از نظر قدرت مانند C++ و یک نرم افزار قوی همچون جاوا می باشد .

۲۷-۱-محیط اجرایی

در برنامه هایی که با .Net نوشته می شوند مدیر اصلی برای فعال سازی اشیا در حافظه ، تخصیص حافظه مورد نیاز به آنها ، جمع آوری حافظه های مازاد و CLR(common language runtime) می باشد . وقتی کد سی شارپ که نوشته اید را کامپایل می کنید مستقیما به کد زبان ماشین تبدیل نمی شود و به کد زبان میانی مایکروسافت MSIL(microsft intermediate Language) تبدیل می شود و هنگامی که قصد اجرای برنامه خود را داشته باشید بخش دیگری از Clrبه نام JIT Compiler (just in time) کد msil رو به کد زبان ماشین بهینه برای آن ماشینی که برنامه بر روی آن اجرا می شود تبدیل می کند و این طور نیست که تمام کد را یکجا به زبان ماشین تبدیل کند و هز زمان که به یک قسمت کد مراجعه شود jitکار خود را

انجام می دهد و آنقدر هوشمند میباشد که بداند آن قسمت را تا به حال ترجمه کرده است یا خیر .Msil همانند بایت کد های جاوا مستقل از پلتفرم می باشد که مجموعه ای از دستورات اسمبلی سطح پایین می باشد .CLRبا (JVM(java virtual machine مقایسه می شود و قابلیت اجرای کدهای مدیریت شده و غیر مدیریت شده را دارا می باشد.

۲۷-۲-ارث بری چندگانه

در سی شارپ میتوانید از overload کردن عملگر ها استفاده کنید.

ارث بری سلسله مراتبی یک سری معایب و یک سری مزایا دارد. باارث بری سلسله مراتبی برنامه نویس کمتر دچار سردرگمی میشود. ولی راه های زیادی وجود دارد که طرح خودتان را به صورت ارث بری چند گانه شبیه سازی کنید .

هیچ زبانی مثل جاوا در بحث شی گرایی و قواعد آن پیشتاز نیست پس قواعد شی گرایی ایجاب میکند که شما ارث بری چندگانه نداشته باشید.

۲۷-۳-کامپایل و اجرای برنامه ها از طریق خط فرمان

در سی شارپ:

کد:

```
> csc <program_name>.cs
> <program_name>
```

در جاوا:

کد:

```
> javac <program_name>.java
> java <program_name>
```

فضاهای نام گذاری برای دسته بندی توابع مختلف که در یک زمینه خاص وجود دارد استفاده می شوند . به عنوان مثال FCL در .Net مجموعه ای از ۲۵۰۰ کلاس می باشد .خوب به طور قطع انتخاب ۲۵۰۰ نام مختلف که قابل فهم نیز باشد برای توابعی که در این محدوده ها قرار می گیرند کار مشکلی می باشد . بنابر این با وجود فضاهای نام گذاری امکان استفاده از نام های یکسان در فضاهای متفاوت بوجود آمده است.

در جاوا:

کد:

```
package mySpace1.mySpace2.mySpace3;
class FirstClass
{
}
```

در سی شارپ :

کد:

```
namespace mySpace1

{   namespace mySpace2
  {
    namespace mySpace3
     {
          class FirstClass
          {
           }
     }
  }
}
```

در جاوا و سی شارپ کلاسها توسط کلمه کلیدی New ایجاد می شوند .سازنده یک شی را به محض ایجاد مقدار دهی اولیه می کند . فرمتی کاملا مشابه بین دو زبان استفاده می شود

کد:

```
<access modifier>  <classname>
{
    /* constructor code */
}
```

برای جمع آوری حافظه های مازاد یعنی حافظه هایی که به اشیا اختصاص داده شده و دیگرهیچ رفرنسی به آنها در حافظه وجود ندارد در بازه های زمانی تصادفی حافظه بررسی شده و از این نوع حافظه های بی مصرف پاک شده و به حافظه سیستم برگردانده می شود و همه این اعمال بدون اطلاع برنامه نویس انجام می شود و یا اینکه می توان مانند زبان c++ به صورت صریح از مخرب برای برگرداندن حافظه به سیستم استفاده کرد.

کد:

```
~<classname>
{
    /* constructor code */
}
```

۲۷-۴-برتری‌های جاوا نسبت به زبان سی‌شارپ

از برتری‌های جاوا نسبت به زبانهای مایکروسافتی مثل سی‌شارپ که بسیار شبیه این زبان است می‌توان موارد زیر را نام برد:

- **سیستم عامل**

هر چقدر زبانهای .NET قوی باشند تنها بر روی پلت فرم ویندوز اجرا می‌شوند و برخی ویندوز را سیستم عامل غیر قابل اعتمادی در برنامه نویسی Enterprise می دانند. ولی جاوا از این نظر انتخابی خوب است.

- **قابلیت حمل**

جاوا بر روی پلتفرم‌های گوناگونی قابل اجرا است، از ATM و ماشین رختشویی گرفته تا سرورهای سولاریس با قابلیت پشتیبانی از ۱۰۲۴ cpu برای پردازش.

- **جاوا بیشتر از یک زبان است:**

جاوا فقط یک زبان نیست و انجمن‌هایی متشکل از بزرگان صنایع و برنامه‌نویسان زیادی مشغول به توسعه و ایجاد استانداردهای جدید و به روز هستند.

جاوا فرا تر از یک زبان برنامه نویسی است زیرا برای هر کاری solution دارد. از جاوا داخل وسایل میکرو مثل موبایل یا ماشین لباسشویی تا سرورهای قول پیکر شرکت سان استفاده میشود.

۲۸-تکنولوژی‌های اصلی جاوا در حال حاضر

برنامه‌نویسی برای سیستم‌های رومیزی (J2SE)

برنامه‌نویسی سمت سرور (J2EE) که به تازگی به JAVA EE 5 تغییر نام داده است.

برنامه‌نویسی برای سیستم‌های موبایل و رایانه‌های کوچک (J2ME)

منابع مورد استفاده:

- **کتاب C و C++ مهندس عین اله جعفر نژاد قمی**
- **آدرس های اینترنتی زیر:**

http://ayamidanestid.blogfa.com/post-28.aspx

http://amozesh-all.blogsky.com/1386/12/16/post-43/

http://lahijan-university.blogfa.com/page/javascript.aspx

http://lianstudents.blogfa.com/post-110.aspx

http://www.barnamenevis.biz/forum/forumdisplay.php?f=5

http://forum.mihandownload.com/

http://www.iranweb3.com/weblog/?u=roonin

http://books.bloghaa.com/

http://forum.majidonline.com/forumdisplay.php?f=111

http://mil1368.blogfa.com/post-18.aspx

http://www.farhaddoost.blogfa.com/post-38.aspx

http://forum.online-dl.com/showthread.php?t=15241

http://4downloads.ir/2009/02/php-blumentals-rapid-php-2008-777.php

http://forum.codecorona.com/forumdisplay.php?fid=54

http://www.wsdevelop.ir/?p=56