بنام یگانه خالق هستی بخش

فصل اول –
آشنایی با Activex

تکنولوژی Activex یکی از قویترین ابزارهاییست که برنامه نویسان ویندوز در اختیار دارند . تقریبا هر برنامه ای که بنویسند از Activex استفاده خواهند کرد . در این فصل با سر فصل ذیل آشنا خواهیم شد :

- اهمیت مدول های نرم افزاری
- تاریخچه و اصطلاحات Activex
- درک مفهوم جاوا و کنترل Activex
- مطالبی درباره امنیت و توزیع کنترل های Activex

انقلاب مدول های نرم افزاری :

همانند هر صنعت دیگری ، کارآیی یکی از دغدغه های اصلی برنامه نویسان است و یکی از جنبه های مهم کارآیی اجتناب از دوباره کاری است . اگر کدی نوشته اید که کار خاصی را انجام می دهد ، چرا باید دفعه بعد همان کد را دوباره بنویسید ؟
اولین گام در راه استفاده مجدد از کدها ، مفهوم زیر مجموعه یا روال Procedure است . کدی را یکبار می نویسید و از آن به بعد در هر جای برنامه که لازم بود فقط کافی است آنرا احضار Call کنید . قرار دادن روال های کلی در یک مدول برنامه نویسی قدمی به پیش بود . با این تمهید می توان از آن روال ها در برنامه های مختلف استفاده کرد .

برنامه نویسی شی گرا :

به موازات رشد برنامه نویسی مدولار ، تکنولوژی دیگری در زمینه برنامه نویسی اختراع شد و توسعه یافت .
برنامه نویسی شی گرا یا OOP
(Object – Oriented Programming) محرک توسعه این تکنولوژی رشد و پیچیدگی روز افزون برنامه ها و مشکلاتی بود که به تیغ آن برنامه نویسان را درگیر خود کرده بود . مهمترین منبع این مشکلات بر هم کنش غیر قابل پیش بینی قسمتهای مختلف یک برنامه با یکدیگر بود . چون این قسمتها مانند دانه های یک زنجیر در هم بافته شده بوند و هر تغییری در یک قسمت به راحتی سایر قسمتها را متاثر می کرد .
را ه حل این مشکل آن بود که هر قسمت برنامه در یک بسته بنام شی Object ، کپسوله یا Encapsulation شود . ساز و کار درونی هر شی مطلقا از دید دنیای خارج مخفی است و آنها نمی توانند تاثیری بر عملکرد وی بگذارند . البته یک شی نمی تواند بکلی از دنیای اطراف خود ایزوله شود چون بدین ترتیب دیگر چیز بی فایده ای بیش نخواهد بود ! به همین دلیل برای ارتباط با دیگر قسمتهای برنامه ، هر

شی از وسیله ای بنام واسط یا Interface استفاده می کند . واسط هر شی دو بخش دارد : خواص (داده ها) و متدها (کدها) ی آن .

تاریخچه کوتاهی از Axtivex :

برنامه نویسی مدولار سالها برنامه نویسان را به خود مشغول کرده بود و در واقع یکی از محرکهای اصلی توسعه سیستم عامل ویندوز هم همین ایده کدهای قابل اشتراک و قابل استفاده مجدد بوده است . اولین گام در راه پیاده سازی عناصر مدولار تکنولوژی OLE یا Object Linking and Embedding بود . هدف اولیه OLE ایجاد سندهای مرکب Compound Documents با استفاده از برنامه های مختلف بود .

سندی که مقداری متن و مقداری نمودار دارد ، و هر کدام آنها با نرم افزار خاص خود ایجاد شده اند ، نمونه ای از یک سند مرکب است . وقتی با متن کار می کنید نرم افزارها واژه پرداز کنترل را بدست می گیرد و وقتی با نمودارها کار می کنید نرم افزار ترسیمی مسئولیت را به عهده می گیرد . OLE با وجود کندی و مشکلات دیگر مسلماً قدمی به پیش بود .

تکنولوژی OLE خود بر یک استاندارد کلی تر بنام COM یا Component Object Model استوار است . بزودی COM از سندهای مرکب فراتر رفت و OLE را هم بدنبال خود کشاند و از آن به بعد OLE اصطلاحی شد برای هر چیزی که از تکنولوژی COM استفاده می کرد . سالها بعد که میکروسافت بطور جدی درگیر اینترنت شد ، اصطلاح Activex هم وارد ادبیات کامپیوتری شد . ابتدا این اصطلاح فقط در رابطه با اینترنت و وب بود اما اوضاع بدین منوال باقی نماند و اکنون Activex به آن بخش از تکنولوژیهای COM گفته می شود که در آنها یک قطعه نرم افزاری امکانات خود را در اختیار برنامه های دیگر می گذارد . یکی از ادعاهای Activex (که بویژه به اینترنت مربوط می شود) پشتیبانی از نرم افزارهای توزیع شده Distributed است ، و این یعنی ، کنترل های Activex به شما سرویس خواهند داد ، حتی اگر در کامپیوتری دیگر (و هزاران کیلومتر دورتر) باشند . اما چه بر سر OLE آمد ؟ این تکنولوژی دوباره به وضعیت اولیه اش برگشت و اکنون فقط با سندهای مرکب سروکار دارد .

نقاط ضعف و قوت Activex :

تکنولوژی Activex بهترین وسیله برای ایجاد محتویات فعال در وب است . البته برخی با این عقیده مخالفند و باید گفت که این تکنولوژی مسلما تنها ابزار محتویات فعال وب نیست . پس اجازه دهید نگاهی به جنبه های مثبت و منفی این تکنولوژی بیندازیم .

نقاط قوت :

یکی از مهمترین نقاط قوت Activex قدرت آن است . یک کنترل Activex تقریباً از عهده هر کاری که یک برنامه معمولی بتواند انجام دهد ، برمی آید . از دیدگاه یک برنامه نویس وب این بهترین جنبه یک ابزار خلق محتویات دینامیک است . دیگر ابزارهای محتویات فعال ، مانند جاوا و CGI ، در این زمینه بسیار محدودتر از Activex هستند . نکته مثبت دیگر ، حداقل برای بسیاری از افراد ، امکان استفاده از مهارتهای عادی برنامه نویسی در خلق کنترل های Activex است .

روش انجام بارگیری های وب هم یکی از مزایای Activex است . وقتی به یک صفحه وب که عناصر Activex دارد می روید ، وقایع ذیل اتفاق می افتد :

۱- اطلاعات مختصری درباره کنترل و شماره ویرایش آن بار می شود .

۲- کاوشگر بررسی می کند که آیا این نرم افزار در کامپیوتر شما نصب شده یا خیر .

۳- اگر این نرم افزار در سیستم شما وجود نداشت (یا اینکه ویرایش آن قدیمی تر بود) ، کاوشگر آنرا بار کرده و سپس نصب و اجرا خواهد کرد .

۴- اگر نرم افزار در سیستم شما وجود داشت ، کاوشگر آنرا از همان جا اجرا خواهد کرد . حتما متوجه مزیت این روش شده اید : هر قطعه نرم افزاری فقط یکبار باید بار شود و دفعات بعد دیگر نیازی به بار شدن آن نخواهد بود ، بنابراین صرفه جویی زیادی در زمان نمایش آن صفحه وب خواهد شد . در ابزارهایی مانند جاوا ، این قبیل نرم افزارها باید هر دفعه مجددا بار شوند و این اجرای آنها را به مراتب کندتر خواهد کرد .

نقاط ضعف – هیچ چیز کامل نیست !

شاید مهمترین جنبه منفی Activex ضعف امنیتی آن باشد . این ضعف ظاهرا یکی از تبعات اجتناب ناپذیر قدرت و انعطاف تکنولوژی Activex است . یک برنامه نویس شرور می تواند براحتی با این تکنولوژی نرم افزاری بسازد که به کامپیوتر دیگران صدمه بزند . با توجه به این وضعیت ، آیا می توان هنگام برخورد با کنترل های Activex روی اینترنت احساس راحتی کرد ؟ پاسخ این سئوال مثبت است و در ادامه توضیح خواهیم داد که چگونه میکروسافت اقدامات امنیتی خاص را برای این تکنولوژی پیاده سازی کرده است . با این وجود همواره این مطلب را در نظر داشته باشید که هیچ سد امنیتی ذاتاً نفوذ ناپذیر نیست .

یکی دیگر از نقاط ضعف تکنولوژی Activex آن است که فقط برخی از کاوشگرهای امروزی از آن پشتیبانی می کنند . اگر در یک صفحه وب نرم افزارهای Activex وجود داشته باشد و فردی با یک کاوشگر که از Activex پشتیبانی نمی کند این صفحه را باز کند قادر به استفاده از مزایای آن نخواهد بود . خوشبختانه این مشکل آنچنان که به نظر می آید حاد نیست ، چون دو تا از مهمترین کاوشگرهای وب

(Netscape Navigator , Internet Explorer) از این تکنولوژی پشتیبانی می کنند (اولی مستقیم و دومی از طریق یک افزودنی قابل نصب) به هر حال ، وقتی با برنامه نویسی اینترنت سروکار داریم ، این مشکل اساساً وجود ندارد چون این ما هستیم که شبکه را کنترل می کنیم .
مشکل دیگر رفتار کنترل های Activex آن است که تمام آنها روی سیستم شما جا خوش کرده و فضای هارددیسک را اشغال خواهند کرد . اما بنظر من نگرانی در این مورد هم بیهوده است چون عناصر Activex معمولا کوچکند و چند مگابایت جایی که احتمالا (صدها کنترل Activex) اشغال خواهند کرد در هارد دیسک های بسیار بزرگ امروزی نمی توانند مشکلی ایجاد کند .

وضعیت جاوا چگونه است ؟

جاوا یکی دیگر از تکنولوژیهای عمده ایجاد محتویات فعال در اینترنت است .
اپلت های جاوا ، که توسط کاوشگر بار و اجرا می شوند ، دارای بسیاری از قابلیت های Activex هستند ، اپلت های جاوا برای جلوگیری از اقدامات خرابکارانه ، فاقد توانایی های خاص (از قبیل دسترسی به سیستم فایل Systemfile هستند و این آنها را بسیار محدود کرده است . از نظر تئوری ، اپلت های جاوا ایمن هستند چون اساساً نمی توانند کارهای خطرناک انجام دهند !

درک مفهوم جاوا و کنترل های Activex :

جاوا یک زبان برنامه نویسی کامپیوتر است که توسط شرکت Sun به بازار عرضه شده است تا به وسیله آن برنامه نویسان قادر باشند برنامه هایی را برای مرورگرهای وب بنویسند که کار با آنها برای استفاده کنندگان بسیار راحت باشد ابزارهای استاندارد شبکه وب به فعالیت ها و عملکردهایی که از صفحه وب قابل دسترسی هستند محدود می شوند . با یک نرم افزار کامل برنامه نویسی مانند جاوا محدودیت های کمی برای طراحی عملکردهای دورن یک صفحه وب وجود دارد . بعنوان مثال شما می توانید با مراجعه به سایت اینترنتی www.java.sun.com با برخی از نمونه های کاری جاوا آشنا شوید . به شکل زیر توجه کنید :

- وسایل بالا و پایین برنده صفحه به همراه صدا
- طراحی تصاویر سه بعدی از اشیاء مختلف نظیر مولکول
- نشانگرهای اعلان یک مطلب که بطور خودکار عوض می شود

برنامه های جاوا را با نام Applet نامگذاری کرده اند . Applet جاوا توسط صفحات وب بر روی یک سرویس دهنده وب نگهداری می شود . وقتی که شما یک صفحه وب را که دارای قسمت هایی است که بوسیله جاوا برنامه ریزی شده نگاه می کنید برنامه های جاوا بصورت خودکار در هنگام دیدن آن صفحه وب از روی اینترنت گرفته و بر روی کامپیوتر شما قرار می گیرند . برای انجام این کار ، احتیاجی نیست شما کاری انجام دهید .
نرم افزار IE خود می داند که چگونه برنامه های جاوا را اجرا کند . یک برنامه جاوا به همان سرعت که به دستگاه شما می رسد بر روی آن نیز اجرا می شود .
برای اینکه کامپیوتر خود را بصورت مطمئن نگهداری کنید نرم افزار IE مدلی از امنیت جاوا را به شما ارائه می کند که در آن اجازه اجرای کلیه اپلت های جاوا داده شده است . بدون اینکه در مورد آسیب دیدن برنامه های کامپیوتر تان یا نفوذ به اطلاعات شخصی خودتان که بر روی دستگاه شما موجود است نگران باشید .
Javascript ساده ترین زبان برنامه نویسی کامپیوتر است که برای طراحی اجزاء صفحه وب شما بکار می رود .
Javascript بوسیله شرکت Netscape به بازار عرضه شده است . برخلاف جاوا این نرم افزار نمی تواند برای نوشتن برنامه ها یا Applet بکار رود . Javascript فقط برای اعمال ساده نظیر روشن کردن یک کلمه وقتی که شما با ماوس بر روی آن می روید و یا تغییر شکل یک نشانه به شکلی که شما تصور کنید آن را فشار داده اید بکار می رود .
Internet Explorer قادر است که Javascript را اجرا کند و صفحاتی را که در آنها از اجزایی استفاده شده که بوسیله Javascript برای همگان قابل استفاده شده است ممکن است در نوشتن آنها مشکلاتی پیش آید و در نتیجه امکان دارد شما در هنگام دیدن صفحات وب با اشکالاتی در رابطه با دستورات Javascript مواجه شوید و عملکردهای Javascript بر روی دستگاه شما به شکل درستی اجرا نشود .
شرکت Microsoft دوست دارد که Activex را یک مجموعه از امکانات جدید برای ساختن صفحات وب فعالتر معرفی کند بر خلاف جاوا Microsoft Activex یک زبان برنامه نویسی نیست بلکه یک مجموعه از قطعات نرم افزاری است که توسط دیگر نرم افزارهای برنامه نویسی نظیر جاوا می تواند استفاده شود . Activex مانند add-ons و Plug – ins قابلیت نرم افزار IE را گسترش داده است . همچنین Activex قابلیت های نرم افزار جاوا را نیز بهبود بخشیده است . برنامه های Activex کنترل نامیده می شوند . مانند اپلت های جاوا ، کنترل ها از روی اینترنت گرفته می شوند و بر روی دستگاه شما اجرا می گردد این عمل هنگامی صورت می گیرد که شما صفحات وبی را مشاهده می کنید که دارای کنترل های Activex هستند در گوشه پایین سمت چپ پنجره IE شما می توانید جمله (نصب قطعات نرم

افزاری) وقتی که کنترلهای Activex در حال انتقال به دستگاه شما هستند را مشاهده می کنید .
مواقعی ممکن است شما با پنجره هایی در صفحه مانیتور کامپیوترتان مواجه شوید که بوسیله آنها از شما سئوال می شود که آیا می خواهید کنترل های Activex به دستگاه شما منتقل شوند یا نه ؟ تکنولوژی Activex در حقیقت قسمتی از مدلهای COM (com مخفف مدلهای شی ء گرای برنامه نویسی میکروسافت می باشد) میکروسافت می باشد .
این مدل به برنامه نویسان اجازه می دهد تا نرم افزارهایی بصورت مستقل برای صفحات اینترنت خود ایجاد کنند و یا برنامه هایی را طراحی کنند که عملیات خاصی را انجام دهد . وقتی که این برنامه ها نوشته و طراحی می شوند می توان از آنها در جاهای دیگر هم دوباره استفاده کرد . در قسمت پایین تعدادی از کنترل های Activex که در داخل IE بکار می روند آورده شده است :

- نمایشگر فایل های Power Point
- رابطه هایی برای بکارگیری بانک های اطلاعاتی
- ساعت های بین المللی
- نقشه راه ها که کار با آنها ساده است .

یکی از بزرگترین امتیازات جاوا و Activex که در Plug – ins ، add – ons و نمایشگر ها وجود ندارد این است که آنها بصورت خودکار کار می کنند و شما احتیاجی ندارید که به پایگاه های اینترنت مراجعه کرده و آنها را به کامپیوتر خود منتقل کنید و مدتی وقت صرف کنید تا فرم های ثبت نام شما را پر نمایید . پس از انتقال آنها به دستگاه مدتی را برای نصب آنها وقت صرف کنید . از زمانی که نرم افزار IE توانست اپلت های جاوا Java Applet و کنترل های Activex را اجرا کند کاربران اینترنت دیگر احتیاجی ندارند که به پایگاههای خاص برای گرفتن اطلاعات مراجعه کنند بلکه اپلت ها و کنترل ها به همان سرعت که به روی دستگاه شما متصل ی شوند در همان فاصله نیز به اجرا در خواهند آمد . پایگاه اینترنت مربوط به

اطلاعات Activex

برای دسترسی به این سایت می توانید از آدرس زیر استفاده کنید :
http://www.developer.com

بکار بردن کنترل های Activex :

کنترل های Activex صفحات اینترنتی شما را بصورت زنده در می آورند بطوریکه شما قادر خواهید بود فایل های صوتی زنده را اجرا کرده و یا نشانگرهای متغییر را ببینید و بسیاری کارهای مشابه دیگر .

۱- به یک صفحه وب که دارای کنترل Activex است متصل شوید . به عنوان مثال به آدرس فوق مراجعه کنید : http://carpoint.msn.com
۲- گزینه مربوط به اجرای یک فایل صوتی یا تصویری را کلیلک کنید .
۳- در صورت نیاز گزینه Yes را کلیک کنید تا نصب کنترل Activex بر روی دستگاه شما منتقل شده و اجرا می شود .

اگر شما به یک پایگاه اینترنت که دارای یک کنترل Activex هست مراجعه کنید نرم افزار IE چک می کند که کدام کنترل ها بصورت دیجیتالی تایید شده اند . یک کنترل تایید شده دیجیتالی برنامه ای است که بصورت مستقل تایید شده است که دارای ویروس های کامپیوتری نیست و تاثیرات منفی بر روی دستگاه شما ندارد . شما می توانید پنجره ای بر روی صفحه مانیتور خود ببینید که به شما اطلاع می دهد که آیا نصب کردن این نرم افزار بر روی دستگاه شما به امنیت آن صدمه ای نمی زند و یا اخطار می دهد که در صورت نصب به دستگاه شما بصورت نرم افزاری آسیب می رساند .

نکته :

جریان داده : در اینترنت به جای اینکه یک فایل صوتی بزرگ را قبل از اینکه بشنوید به کامپیوترتان منتقل نکنید از جریان داده استفاده می شود . پایگاههای اینترنتی

اطلاعات خود را بصورت یک جریان داده می فرستند ، کامپیوتر شما پس از چند ثانیه از شروع جریان داده آن را با خبر کرده و شروع به پخش آن می کند و به همین ترتیب آن فایل صوتی یا تصویری را تا انتها اجرا می کند .

اجرای Java Applet و نمایشگرها :

نرم افزار IE یک Java Applet را وقتی اجرا می کند که صفحه وب حاوی آن در حال شروع به نمایش بر روی صفحه کامپیوتر شما می باشد . هزاران Java Applet بر روی وب وجود دارد نظیر نشانگرهای متغییر بالا و پایین برنده صفحه که از خود صدا تولید می کند ، ماشین حسابها و بسیاری از عملکردهای دیگر .

اجرای یک Java Applet :

۱- به یک صفحه وب که دارای Java Applet هست متصل شوید . به عنوان مثال به آدرس های زیر مراجعه کنید :

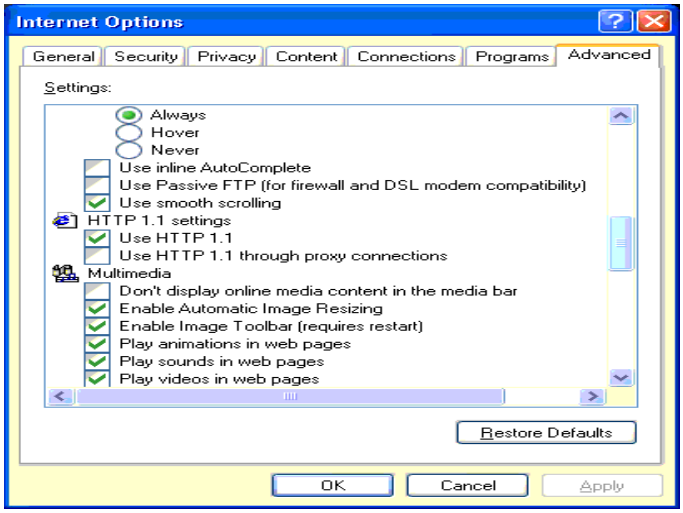
http://www.gamelan.com

و یا

http://www.javasoft.com

۲- منوی Category را انتخاب کنید و سپس گزینه مربوط به نمایش صفحه وب همراه با Java Applet را کلیک نمایید . Java Applet به کامپیوتر شما منتقل خواهد شد و به اجرا در خواهد آمد .

۳- اگر لازم شد اطلاعات درخواستی اپلت های جاوا و یا منوهای مناسب آن را کلیک کنید تا اجرای آن کامل شود .



خاموش کردن اجرا کننده JIT :

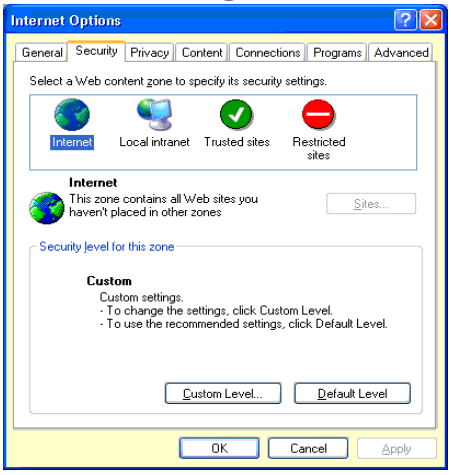
۱- منوی ابزار را کلیک کرده و سپس Internet Option را کلیک کنید .

۲- جعبه Advance را کلیک کنید .صفحه را به پایین آورید تا به قسمت Java MV برسید .
۳- جعبه مربوط به گزینه Java JIT Complier
۴- کلید OK را کلیک کنید .

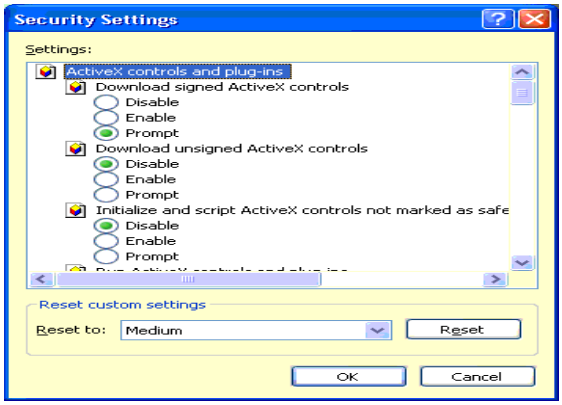
نرم افزار IE دارای یک اجرا کننده نرم افزار است که دقیقا در همان زمان گرفتن برنامه آن را اجرا می کند (JIT) بصورت پیش فرض روشن است . وقتی که شما با مشکلی در اجرای اپلت های جاوا (Java Applet) برخورد کردید ، می توانید اجرا کننده JIT خود را در منوی Advance در Internet Option خاموش کنید . بدین ترتیب اپلت های جاوا بر روی دستگاه شما دیگر اجرا نخواهند شد .

کنترل کردن Activex و برنامه های Java :

توسط IE شما می توانید دقیقا تعیین کنید کنترل های Activex و برنامه های Java Applet تا چه حد بر روی برنامه های دستگاه شما تاثیر بگذارند با استفاده از Security Zones استفاده کنندگان و مدیران شبکه می توانند درباره سطح دسترسی کنترل های Activex و برنامه های Java Applet تصمیم بگیرند . به عنوان مثال شما می توانید به برنامه های Java Applet که از پایگاههای معتبر بر روی اینترنت دریافت می کنید اجازه دسترسی بیشتری به کامپیوترتان را بدهید و یا اینکه جاوا اپلت های گرفته شده از پایگاههای غیر معتبر را از لحاظ دسترسی به فایل های دستگاهتان



محدود کنید تا نتوانند به آنها آسیب برسانند .



تغییر دادن درجه امنیت برنامه های Activex

۱- منوی Tools را کلیک کنید و سپس گزینه Internet Option را کلیک کنید .
۲- گزینه Security را در بالای پنجره کلیک کنید .
۳- بر روی گزینه مناسب کلیک کنید .
۴- آنگاه می توانید با بالا بردن نشانه ، تدابیر امنیتی شدیدتری اتخاذ کنید و یا بالعکس با پایین آوردن نشانه درجه امنیت را پایین تر بیاورید و در اصطلاح به Java Applets اطمینان کنید .
۵- برای تغییر مشخصات حرفه ای تر می توانید گزینه Custom Level را کلیک کنید .
۶- برای تغییر هر کدام از مشخصات زیر یکی از ۳ گزینه (قابل اجرا) Enable ، آماده Prompt و (غیر قابل اجرا) را انتخاب کنید .

- مشخصه Script Activex در حقیقت امن بودن اسکریپت ها را چک می کند .
- مشخصه بعدی امنیت اجرای کنترل های Activex و برنامه های Plug – ins چک می کند .
- گزینه بعدی مربوط به گرفتن کنترل های Activex تایید شده توسط علامت دیجیتالی است (Signed)
- گزینه بعدی مربوط به گرفتن کنترل های Activex تائید نشده توسط علامت های دیجیتالی است (Unsigned)
- گزینه بعدی مربوط به مقدار دهی و اجرای کنترل های Activex است که امنیت آنها تایید نشده است .
- گزینه Ok را کلیک کنید .

Activex و امنیت:

مطمئناً تا به حال درباره ویروس های کامپیوتری (برنامه هایی که با آلوده کردن سیستم ها صدماتی به آنها می زنند) چیزهایی شنیده اید . ویروس های مختلفی از

انواع بی آزار تا بسیار مخرب وجود دارند و تلاش های زیادی صورت می گیرد تا جلوی این آلودگی و انتشار ویروس ها گرفته شود . اما آیا این احتمال وجود ندارد که ویروسها (یا دیگر نرم افزارهای مخرب) از طریق اینترنت پراکنده شوند ؟
در روزهای اول وب این خطر چندان جدی نبود چون سندهای HTML و فایلهای گرافیکی و تصویری تنها انواع فایل هایی بود که روی اینترنت جابجا می شد . این قبیل فایلها می توانستند خراب باشند اما در ضمن هیچ خطری برای سیستمی که آنها را بار می کرد نداشتند .
اما با روی کار آمدن محتویات فعال اوضاع دگرگون شد ، چون در این حالت برنامه ها هم جزیی از وب شده بودند و یک برنامه هم قاعدتاً هر کاری می تواند انجام می دهد (از پاک کردن فایل های کامپیوتری مقصد گرفته تا دزدیدن اطلاعات آن و انتقال آنها به جاهای نامعلوم) . در اینجا بود که شرکتهای تولید کننده ابزارهای برنامه نویسی وب به ضرورت تمهیدات امنیتی پی بردند . وقتی در وب گشت و گذار می کنید . مایلید مطمئن باشید کدهایی که بار می کنید سیستم تان را بهم نخواهند ریخت ! برای مقابله با این مشکل بالقوه دو روش اساسی ابداع شد .
روش اول آن بود که اساساً اجازه اعمال بالقوه خطرناک به برنامه های وب داده نشود . این راهی بود که جاوا در پیش گرفت . اپلت های جاوا قادر به انجام افعال بالقوه مخرب (مانند دسترسی به سیستم فایل) نیستند و می توانید مطمئن باشید که آنها بی خطر هستند درست همانگونه که یک چاقوی پلاستیکی بی خطر است !
در روش دوم ، که میکروسافت برای عناصر Activex در پیش گرفت ، به جای محدود کردن ذاتی برنامه ها از یک تکنولوژی بنام کد تعیین اعتبار یا Authenticode برای تعیین صحت و سقم نرم افزارها استفاده می شود . ایده نهفته در این روش این است که اگر شما بدانید که یک عنصر Activex (یا هر نرم افزار دیگری) از کجا آمده (خالق آن کیست) و تغییری هم در آن صورت نگرفته ، دیگر می توانید با خیال راحت از آن استفاده کنید . به این روش امضای دیجیتالی یا Digital Signing هم گفته می شود .

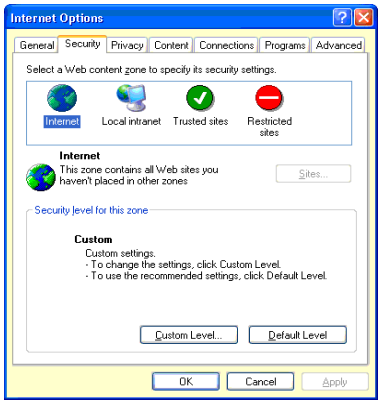
امضاء دیجیتالی از دیدگاه کاربران :

وقتی از یک کاوشگر با قابلیت Activex استفاده می کنید در واقع سطوح امنیتی مختلفی را در اختیار دارید ، که می توانید از آنها بسته به نیازتان استفاده کنید . این سطوح عبارتند از : ضعیف ، متوسط ، شدید . برای انتخاب سطح امنیتی دلخواه در کاوشگر Internet Explorer ، آیتم Internet Options را از منوی View انتخاب کرده و سپس به برگه Security بروید .
در قسمت پایین دیالوگ می توانید سایتهای هر منطقه را انتخاب کرده و سپس در قسمت پایین دیالوگ سطح امنیتی هر منطقه را تعیین کنید . سایتهای قابل اطمینان و شناخته شده را می توانید در منطقه مورد اعتماد (Trusted) قرار دهید . این منطقه

دارای سطح امنیتی ضعیف است . سایتهای ناشناخته را در منطقه محدود (Restricted) که سطح امنیتی بالایی دارد ، قرار دهید .
سطوح امنیتی Internet Explorer عبارتند از :

- شدید (High) : برنامه های بالقوه خطرناک هرگز بار نمی شوند . هر اقدامی برای بار کردن محتویات فعال به شما اطلاع داده خواهد شد .
- متوسط (Medium) : هنگام بارکردن محتویات فعال به شما اخطار داده می شود ولی این امکان را خواهید داشت که (بر اساس امضاء دیجیتالی) آنرا بار کنید .
- ضعیف (Low) : تمام محتویات فعال بدون هیچ اخطاری بار خواهند شد .
- قابل تنظیم (Custom) : می توانید برای هر نوع محتویات فعال سطح امنیتی مناسب (هرگز بار نشود ، با دادن اخطار بار شود ، همیشه بار شود) را تعیین کنید . توصیه می شود تا زمانیکه با تمام انواع محتویات موجود در وب آشنا نشده اید از این گزینه استفاده نکنید .

روش کار این سیستم چنین است : وقتی به صفحه وبی می روید که می خواهد محتویات فعال خود را برای شما بفرستد ، کاوشگر ابتدا امضای دیجیتالی تمام آیتم های نرم افزاری آنرا چک می کند . اگر امضای دیجیتالی وجود نداشته باشد ، محتویات فقط در سطح امنیتی ضعیف اجازه بارشدن خواهند داشت . در غیر این صورت ، با توجه به اطلاعات موجود در امضای دیجیتالی ، سالم بودن آیتم ها (عدم تغییر در محتویات آنها) بررسی می شود . ضرورت این کار از آنجا ناشی می شود که افراد شرور می توانند برای یک نرم افزار بی ضرر امضای دیجیتالی و تائیدیه بگیرند و سپس آنرا تغییر داده و بصورت خرابکار و خطرناک در آورند .
در سطح امنیتی متوسط نام تولید کننده نرم افزار بایستی از منابع تایید شده از سوی شما مطابقت داده خواهد شد . اگر این نام در لیست مورد اعتماد شما نباشد ، دیالوگی موضوع را گوشزد کرده و اجازه بارکردن آنرا به شما خواهد داد . در همین دیالوگ می توانید تولید کننده مزبور را هم به لیست معتمدین خود اضافه کنید .



سایتها در Internet Explorer .

امضای دیجیتالی از دید برنامه نویسی :

در یک امضای دیجیتالی سه دسته اطلاعات وجود دارد : هویت تولید کننده نرم افزار، هویت منبع تایید کننده (سازمانی که امضاء را صادر کرده) و یک عدد رمز برای تایید این مطلب که محتویات نرم افزار دستکاری نشده است .

اگر می خواهید برای وب محتویات فعال بنویسید باید یک گواهینامه کد تعیین اعتبار برای خود دست و پا کنید تا بتوانید برای نرم افزار های خود امضای دیجیتالی بگیرید . اگر فقط برای اینترنت برنامه می نویسید نیازی به این مراحل ندارید چون سطح امنیتی در آنها معمولا پایین است و نیازی به امضای دیجیتالی وجود ندارد .

اگر صرفاً برای شرکت خود نرم افزار می نویسید می توانید از گواهینامه آن استفاده کنید . اما توصیه می شود خودتان هم این گواهینامه را بگیرید . با آن که شرکتهای متعددی برای صدور گواهینامه کد تعیین اعتبار وجود دارند ، میکروسافت شرکت Verisign را توصیه می کند . برای کسب اطلاعات بیشتر می توانید به سایت وب این شرکت که در زیر آمده است مراجعه کنید :

http://www.verisign.com/developers/index.html

هزینه دریافت این گواهینامه ۲۰ دلار در سال و مراحل انجام آن بسیار ساده است :

۱. در سایت مزبور ، یک فرم پر کنید و در آن اطلاعات خواسته شده (از جمله اطلاعات مربوط به کارت اعتباری) را وارد کنید .
۲. شرکت Verisign کد شناسایی شما را با پست الکترونیک برایتان ارسال خواهد کرد .
۳. به صفحه نصب گواهینامه رفته و کد شناسایی خود را وارد کنید . این کار باید در همان کامپیوتری که توسط آن کد شناسایی را گرفته اید ، انجام شود .
۴. گواهینامه به کامپیوتر شما فرستاده خواهد شد .

هنگام ثبت گواهینامه دو گزینه در اختیار دارید : ذخیره کردن آن در یک فایل یا در رجیستری ویندوز . توصیه می شود گواهینامه خود را در یک فایل و روی دیسک ذخیره کنید تا بتوانید آن را از گزند نامحرمان حفظ کنید . در حقیقت ، دو فایل به کامپیوتر شما فرستاده می شود : یکی حاوی خود گواهینامه (با پسوند SPC) و دیگری حاوی کلید رمزبندی (با پسوند PVK) .

نرم افزارهای مورد نیاز برای امضای دیجیتالی :

بعد از دریافت کد تعیین اعتبار و کلید رمز بندی ، آماده اید تا امضای خود را به نرم افزارهایی که نوشته اید اضافه کنید . این برنامه ها جزیی از کیت توسعه نرم افزار Activex هستند و می توانید آنها را مجانی از سایت میکروسافت بردارید . فایل هایی که نیاز دارید از این قرارند :

- Make Cert: یک گواهینامه X.509 برای تست می سازد.
- Cert2 SPC: یک گواهینامه توزیع نرم افزار آزمایشی می سازد.
- Sing Code: یک فایل محتویات فعال را امضا می کند.
- Chk Trust : اعتبار فایل را بررسی می کند (مشابه همان کاری که کاوشگر می کند)
- Make CTL: لیستی از گواهینامه های قابل اعتماد می سازد.
- Cert Mgr: برنامه مدیریت گواهینامه ها ، لیست های گواهینامه های قابل اعتماد و باطل شده.
- Set Reg: برنامه ثبت گواهینامه در رجیستری.

باید توجه داشت که تمام این ابزارها برنامه های DOS هستند و باید پنجره MS-DOS اجرا شوند . برای باز کردن نرم افزارهای فوق می توانید به آدرس زیر مراجعه کنید :

http://www.msdn.microsoft.com

جاوا چیست ؟

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 .

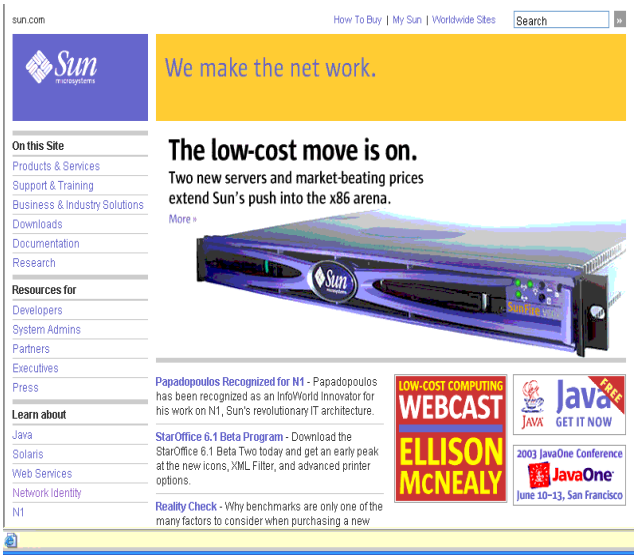
جاوا یک زبان بسیار ساده است چون شما را وادار نمی کند تا در محیط جدید (و نا آشنایی) کار کنید و این برای کسانی که اطلاعات فنی ناچیزی درباره کامپیوتر دارند

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 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 که در جای خود به آنها اشاره خواهیم کرد .

تکامل جاوا :

جاوا هم مانند اکثر اختراعات مهم حاصل تلاش گروهی دانشمندان پیشتاز است . اما نکته جالب درباره جاوا آن است که جاوا از ویرانه های یک پروژه شکست خورده سر بر می آورد .
اگر پروژه گرین (Green) که شرکت Sun در سال ۱۹۹۱ آن را شروع کرد به موفقیت می انجامید امروز نه سان دیگر یک شرکت کامپیوتری می بود و نه جاوا در سال ۱۹۹۱ ، شرکت Sun Microsystems در آغاز راه تبدیل شدن به بزرگترین تولید کننده ایستگاههای کاری Unix بود و در طول فقط پنج سال درآمدش را از ۲۱۰ میلیون دلار به ۲/۵ میلیارد دلار رسانده بود و به نظر می رسد هیچ چیز نمی تواند جلودار آن باشد . عامل عمده موفقیت Sun کارهای پیشتازانه در ایجاد شبکه های باز بود و در این زمینه هیچ کس به پای آن نمی رسید .
مدیران Sun به این فکر افتادند تا کاری کنند که این رشد برای مدتی طولانی ادامه یابد یکی از زمینه های مورد علاقه Sun بازار لوازم الکترونیکی بود ، پروژه Green بوجود آمد تا برای این بازار نرم افزاری پیشرفته خلق کند .
مهندسان Sun توجه خود را به میکروپروسسورهایی معطوف کردند تا دارای قابلیت استفاده در این گونه سیستم ها باشند . اگر پروژه Green می خواست موفق باشد باید کاری می کرد تا سیستم مزبور بتواند به راحتی به بسترهای سخت افزاری مختلف منتقل شود برای این منظور ابتدا از کامپایلر C++ استفاده شد ولی بزودی نارسایی های C++ در این زمینه خود را نشان دادند . مهندسان Sun خیلی سریع دریافتند که برای ادامه کار باید چیزی جدید و قوی خلق کنند . در همین زمان بود که James

Gosling کار بر روی زبان جدید اوک Oak را شروع کرد تقدیر آن بود که این زبان جاوا نامیده شود ، چون اداره ثبت نام های تجاری Oak را رد کرد .
بتدریج مهندسین یک دستگاه شبه PAD (Personal Digital Assistant) ساختند که در آن از سیستم عامل ، زبان برنامه نویسی و واسط کاربر جاوا استفاده شده بود و با وجود ابتدایی بودن توانست توجه مدیران Sun را به خود جلب کند .
بعدها گروهی که در پروژه Green کار کرده بودند شرکت Java Soft را بنیان نهادند .
شکل زیر پایگاه وب این شرکت را نشان می دهد :



برای رفتن به این پایگاه اینترنتی می بایست به آدرس زیر مراجعه کنید :
http://www.sun.com

متاسفانه بخت با پروژه Green یار نبود و بعد از چند شکست متوالی Sun آن را کنار گذاشت . در همین زمان بود که عده ای از محققان NCSA شرکت Net Scape را تاسیس کردند و این اتفاقات با آتشی که به سرعت دنیا را در می نوردید و شبکه تار عنکبوتی جهانی WWW (که مخفف World Width Web می باشد) نام داشت همزمان شد .
بنابراین اتفاقی نبود که مهندسان پروژه Green به سمت اینترنت و وب به عنوان بستر پروژه Sun روانه شوند و بعد از آن دیگر جزء تاریخ است .
چهار سال بعد پروژه Green به بار نشست و جاوا تبدیل به داغترین موضوع اینترنت شد ، همه از برنامه نویسان و طراحان صفحات وب گرفته تا کاربران اینترنت می خواهند از جاوا استفاده کنند . Sun بلافاصله برای جاوا یک کاوشگر هم تولید

کرد .این اولین کاوشگری بود که می توانست اپلت های جاوا را اجرا کند و نام آن هم Hot Java بود .

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 .

<table>
<tr><td colspan="3">برنامه های کاربردی جاوا</td></tr>
<tr><td colspan="3">اشیاء جاوا</td></tr>
<tr><td colspan="3">ماشین مجازی جاوا</td></tr>
<tr><td>مکینتاش</td><td>ویندوز یا Os2</td><td>یونیکس</td></tr>
<tr><td colspan="3">سیستم عامل</td></tr>
</table>

اما شاید از خود سئوال کنید که چرا جاوا این همه کار را بر خود (و دیگران) سخت گرفته است ؟ این همه لایه و ماشین مجازی برای چیست ؟ یکی از ادعاهای جاوا ، آزاد بودن از بستر سخت افزاری است اما این یعنی چه ؟ اگر جاوا بخواهد در اینترنت موفق باشد برنامه های آن باید بتوانند بدون هیچ اشکالی روی تمام کامپیوترهای متصل به اینترنت اجرا شوند . اما شما بهتر می دانید که اینترنت شبکه عظیمی است از کامپیوترهای نامتجانس ، از سوپر کامپیوترهای عظیم گرفته تا PC های مبتنی بر

روسسور های اینتل و پاور پی سی تا ماشین های یونیکس مبتنی بر پروسسورهای آلفا و ... !
چگونه یک برنامه اجرایی می تواند روی این طیف وسیع کامپیوترها اجرا شود ؟
ایده اولیه ساده است : برنامه ها برای ماشین مجازی کامپایل شوند و این ماشین مجازی روی تمام کامپیوترهای متصل به اینترنت نصب شود .
این دقیقاً همان روشی است که جاوا در پیش گرفته است . برنامه تان را با یک کامپایلر جاوا کامپایل کنید ، آن را در پایگاه وب خود قرار دهید و به دیگران اطلاع دهید تا با یک کاوشگر مجهز به جاوا (Java – Enabled Brower) این برنامه را که ممکن است در دل یک صفحه وب قرار داشته باشد اجرا کنند . شاید هرگز تصور نمی کردید که برنامه ای را که با یک PC اینتل نوشته اید روی یک ماشین مکینتاش یا یونیکس اجرا شود ولی اکنون به لطف جاوا این دیگر آرزویی دست نیافتنی نیست .
شیء گرایی در جاوا به غایت خود رسیده است . جاوا در استفاده از اشیاء بسیار مفید و سخت گیر است و تخطی از اصول را نمی پذیرد . در C++ شما می توانید از اشیاء به موازات برنامه نویسی به سبک قدیم استفاده کنید و اکثراً در پایان کار چنان ملغمه ای بوجود می آید که مدیریت آن با روش های برنامه نویسی شی ء گرا امکان دارد نه با روش های قدیمی . جاوا دیگر به شما اجازه نمی دهد تا خود را در چنین مخمصه ای گرفتار کنید !

شروع برنامه نویسی با جاوا :

اولین برنامه ای که 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 . بطور مثال به لیست زیر نگاه کنید :

```
1: class Hello World {
2: Public static void main (string args []){
3: system. Out. Println ("Hello World"),
4 :}
5 :}
```

این برنامه پنج خطی دارای تمام ویژگیهای یک برنامه کامل و مفصل شی ء گراست در ضمن اعداد ابتدای هر خط را نباید وارد کرد چون جزئی از برنامه نیستند اینها فقط برای ارجاع به برنامه هنگام تشریح آن است . برنامه را با نام HelloWorld. java ذخیره کنید . نام برنامه باید همان نام کلاس برنامه باشد .

تحلیل برنامه : این برنامه دو قسمت مهم دارد .
۱- تمام برنامه در واقع یک تعریف کلاس است .
۲- کل برنامه در متد ( )Main قرار دارد .
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

اگر همه چیز درست پیش رفته باشد ، باید جمله "Hello World!" را روی صفحه کامپیوتر خود مشاهده کنید .
نکته : دقت کنید که 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 .

ایجاد یک اپلت جاوا :

ایجاد اپلت با ایجاد برنامه فرق دارد و قواعد متفاوتی بر آن حمکفرماست . یک اپلت برای اجرا در صفحات وب نوشته می شود ، بنابراین کمی پیچیده تر از یک برنامه است . در واقع یک اپلت تا زمانی که در یک کاوشگر اجرا نشود خروجی خود را نمایش نخواهد داد .
باید سعی شود برنامه ها ، اپلت ها و صفحات وب را در دایرکتوریهای جداگانه ذخیره کرد .
اپلتی که در این قسمت می نویسیم Hello World Applet نام دارد . به لیست زیر نگاه کنید :

```
1: import java. Awt. Graphics;
2:
3: public class Hello World Applet Extends java. Applet. Applet {
4:
5: Public Void Paint (Graphics G) {
6: g. drawstring ("Hello World!",5 , 25 );
7: }
8: }
```

اپلت را با نام HelloWorldApplet. Java ذخیره کنید. در مورد یک اپلت باید به چند نکته توجه کنید :
دستور import در خط ۱ شبیه دستور #include در زبان C است با این دستور اپلت می تواند از کلاسهای JDK برای کارهای گرافیکی استفاده کند .
متد ( ) Paint برای نمایش محتویات اپلت است در این جا عبارت Hello World ! روی صفحه نمایش داده خواهد شد اپلت ها متد ( )main ندارند و به جای آن از متدهای ( )init و ( )Start یا ( ) Paint استفاده می کنند .
برای کامپایل کردن این اپلت چنین باید کرد java Hello World Applet. java در این حالت هم کامپایلر یک فایل کلاس به نام Hello World Applet. Class ایجاد خواهد کرد اما برای اجرای یک اپلت یک قدم دیگر هم باید برداشت و آن ایجاد یک فایل HTML است که اپلت در آن اجرا خواهد شد می توانید یک فایل HTML را برای این منظور مشاهده کنید .

یک HTML برای اجرای اپلت Hello World :

```
1: <html>
2: <head>
3: <title>Hello to Everyone</title>
4: </head><body>
5: <p> My Java Applet says:
6: <applet code=”Hello World Applet. Class” width=150 height=25>
7: </applet>
8: </body>
9: </html>
```

تحلیل برنامه :
برای اجرای یک اپلت در فایل HTML باید از برچسب < applet > استفاده کرد برای مشخص کردن نام کلاسی که اپلت در آن است از صفت CODE استفاده کنید . برای مشخص کردن ابعاد پنجره ای که اپلت در آن اجرا خواهد شد از صفت های Width و Height استفاده کنید . اعداد مشخص شده بر حسب پیکسل هستند این فایل را با نام HelloWorldApplet.html ذخیره کنید حالا آماده اید تا اپلت را اجرا کنید این کار به دو روش امکانپذیر است :

۱- استفاده از کاوشگرهای مجهز به جاوا مانند Net Scape
۲- استفاده از برنامه Applet Viewer که با JDK می آید . این برنامه کل صفحه وب را نمی تواند نمایش دهد و فقط اپلت را نمایش خواهد داد .

برای دیدن اپلت در یک کاوشگر ، فایل html فوق را باز کنید ، با این کار صفحه وب و اپلت درون آن به نمایش در خواهد آمد . روش استفاده از applet viewer چنین است .
نکته : با وجود اینکه می توان برنامه applet viewer را از محل فایل html اجرا کرد ولی در این حالت بدون بستن آن نمی توان اپلت را مجدداً بار کرد . در حالیکه اگر مشاهده گر اپلت از محل دیگری اجرا شود (مانند دستور فوق) می توان یک اپلت را تغییر داده ، مجدداً کامپایل کرده و سپس دوباره در مشاهده گر بار کرد .
بطور کلی جاوا یک زبان برنامه نویسی است که با آن می توان برنامه های متنوعی نوشت بیشترین کاربرد جاوا در اینجا اپلتهایی است که در کاوشگرهای وب قابل مشاهده هستند .
اپلتها برنامه هایی هستند که جزیی از صفحات وب محسوب می شوند . تقریباً هر کاری با اپلت ها امکانپذیر است .
قدرت جاوا در سادگی و استقلال از آن بستر سخت افزاری (چه در فایل منبع برنامه و چه در کد باینری آن) است . با جاوا می توان برنامه هایی هم نوشت که بدون کاوشگرهای وب قابل اجرا باشند . اپلت های جاوا را فقط در فایلهای HTML می توان مورد استفاده قرار داد .
برنامه نویسی شی ء گرا و جاوا (Object Oriented Programming – OOP) :
یکی از بزرگترین ایده های برنامه نویسی در دو دهه اخیر است که نیاز به تسلط کامل دارد .

مفاهیم برنامه نویسی شی ء گرا :

برنامه نویسی شی ء گرا قصد دارد مدلی از دنیای واقعی را وارد برنامه نویسی کامپیوتر کند دنیای اطراف ما از اشیاء ساخته شده است .

اشیاء (Objects) و کلاسها (Classes) :

شیی ء کلیدی ترین مفهوم برنامه نویسی شی گرا است هر شی ء یک حالت و رفتار دارد و برنامه عبارت است از بر هم کنش بین اشیاء . حالت یک شی ء عبارت است از متغییرهای عناصر داده ای شییء و مقدار آنها . رفتار یک شی ء را متدهای آن تعیین می کند .
شیء در واقع مقداری کد است که کار خاصیتی انجام می دهد .
هر شیء کپسولی است از مقداری متغییر و کد که کار نگهداری و به روز در آوردن آنها را ساده می کند . معمولاً درون یک شیء از دسترس دنیای خارج به دور است و برای کار با آن باید از ارسال پیام استفاده کرد . مزیتش آن است که کاربر برای کار با آن هیچ نیازی به دانستن مکانیزم های درونی آن ندارد و فقط باید ساختار پیام ها را بداند . معمولاً ساختار پیام ها در اشیاء مختلف یکسان است . در دنیای واقعی هم وضع به همین منوال است مثلاً برای استفاده از یک تلویزیون شما هیچ نیازی به

آشنایی با پیچیدگیهای درون آن ندارید فقط کافی است بدانید که باید آن را به برق زده و روشن کنید و یک کانال را انتخاب کنید . کلاس در واقع الگویی است برای ایجاد شیء . در واسط برنامه نویسی جاوا (java API) چندین کلاس مختلف وجود دارد . چندین کلاس با هم یک کتابخانه کلاس Class Library می سازند .
برنامه نویسی جاوا اصولاً چیزی نیست جز طراحی و پیاده سازی کلاس ها .

برنامه نویسی شیء گرا در جاوا :

برنامه نویسی شیء (OOP) یکی از بزرگترین ایده های برنامه نویسی در دو دهه اخیر است ، اما تسلط کامل بر این ایده به سالها زمان و ممارست نیاز دارد . برنامه نویسی شیء در واقع پیوند دنیای واقعی با برنامه نویسی کامپیوتر است . در اینجا با مفاهیم برنامه نویسی شیء گرا در جاوا ، و ارتباط این مفاهیم با ساختار برنامه ها آشنا می شویم :

- شیء و کلاس چیست و رابطه آنها چگونه است .
- یک شیء یا کلاس دو جزء مهم دارد : رفتار و حالت .
- وراثت و نقش آن در طراحی برنامه ها .
- بسته ها و واسط ها در جاوا .

ایجاد یک کلاس :

در این قسمت یک مثال عملی خواهیم آورد . در این مثال یک موتور سیکلت را شبیه سازی می کنیم . یک موتور سیکلت واقعی دارای حالت و رفتار خاص خود است . حالت موتور سیکلت در واقع همان خواص آن (مانند رنگ ، مدل ، نوع و ...) است . مجموع این خواص یک موتور سیکلت خاص را مشخص خواهند کرد که از موتور سیکلت های دیگر قابل تمیز است . هر شیء وهله ایست از یک کلاس . کلاس موجودی قابل لمس نیست ، در حالیکه شیء موجودی است قابل لمس که به کلاس خود موجودیت داده است . مثلاً ، انسان یک کلاس است در حالیکه شما نمونه ای هستید از کلاس انسان ، شما موجودیت دارید در حالیکه چیزی بنام انسان وجود خارجی ندارد، بلکه در اشیاء خاص (من ، شما ، دیگران) موجودیت می یابد .
موتور سیکلت ما دارای رفتارهایی هم هست ، رفتارهایی مانند روشن شدن ، خاموش شدن ، تعویض دنده ، سرعت گرفتن ، ترمز کردن و غیره .
ادیتور خود را اجرا کنید و کلاس موتور سیکلت را مانند ذیل تعریف کنید :

```
Class Motorcycle {
}
```

البته این کلاس هنوز کار چندانی انجام نمی دهد (در واقع هیچ کاری انجام نمی دهد !) اجازه دهید کارمان را با تعریف چند متغییر حالت ادامه دهیم ، این تعریف ها را بعد از } وارد کنید :

```
String make;
String color;
(Class library)Boolean engineState;
```

نکته :
بر خلاف C ، متغییرهای Boolean فقط مقادیر True یا False می گیرند و نمی توانند عدد بگیرند .
حال چند رفتار (متد) به این کلاس اضافه می کنیم . گفتیم که یک موتور سیکلت می تواند رفتارهای متعددی داشته باشد ولی ما در اینجا برای سادگی کار فقط متد روشن شدن موتور را پیاده سازی خواهیم کرد .
کد ذیل را بعد از تعریف متغییرهای حالت اضافه کنید :

```
Void startEngine()
{
      If (engineState = = True)
      System.out.println("The engine is already on.");
Else {
      engineState = True;
      System.out.println ("The engine is now on.");
      }
}
```

این متد ابتدا تست می کند که آیا موتور روشن است (engineState= =True) ، اگر چنین باشد فقط با پیامی این موضوع را گوشزد می کند . اگر موتور روشن نباشد، آن را روشن کرده و پیام می دهد که کارش را به پایان رسانده است .
کلاس را با نام Motorcycle.java ذخیره کنید . این کلاس تا اینجا باید چنین باشد :

```
Class motorcycle
{
       String make;
       String color;
       Boolean engineState;
```

```
    Void StartEngine()
    {
        If (engineState= = true)
                System. out println ( “The engine is already on.”);
        Else {
                engineState=true;
                System.out.println(“The engine is now on.”)
              }
    }
}
```

قبل از کامپایل کردن این کلاس ، اجازه دهید یک متد دیگر به آن اضافه کنیم . این متد ShowArts ، مقدار فعلی متغییرهای حالت یک وهله خاص از این کلاس را نمایش خواهند داد . این متد چنین است :

```
Void showArts()
{
    System.out.println(“This motorcycle is a “+color+””+make);
    If (engineState= = true)
        System.out.println(“The engine is on.”);
    Else System.out.println(“The engine is off.”);
}
```

این متد دو خط اطلاعات روی صفحه نمایش خواهد داد . خط اول رنگ و نوع موتور سیکلت ، و خط دوم حالت موتور آن را نمایش می دهد .
فایل را ذخیره کرده و سپس آن را با javac کامپایل کنید :

```
Javac Motorcycle.java
```

اگر سعی شود این کلاس را با اینترپرتر جاوا اجرا کنید ، با خطا مواجه خواهید شد چون اینترپرتر دنبال متدی بنام main() می گردد و چون آن را نمی یابد پیام خطای ذیل را نمایش خواهد داد :

```
In class Motorcycle:void mail (string args[])is not defined
```

برای آنکه این کلاس را عملیاتی کنیم باید یک برنامه جاوا بنویسیم و در آن از این کلاس استفاده کنیم . اما روش ساده تر آن است که به این کلاس یک متد main() اضافه کنیم . لیست زیر این متد را نشان می دهد .

متد main() برای Motorcycle.java

```
1: Public static void main (string args[]){
2: Motorcycle m=new motorcycle();
3: m.make = “Yamaha RZ350”;
4: m.color=”yellow”;
5: System.out.println(“Calling ShowArts…”);
6: m.show Arts();
7: System.out.println (“……”);
8: System.out.println(“Starting engine…”);
9: m.startEngine();
10: System.out.println(“…..”);
11: System.out.println(“Calling showArts…”);
12: m.showArts();
13: System.out.println(“…….”);
14: System.out.println(“Starting engine…”);
15: m.startEngine();
16:}
```

با اضافه کردن این متد ، کلاس Motorcycle اکنون یک برنامه است و می توان آنرا کامپایل و اجرا کرد . خروجی برنامه شبیه ذیل خواهد بود :

```
Calling ShowArts …
This motorcycle is a yellow Yamaha RZ350
The engine is off.
……..
Starting engine …
The engine is now on.
…….
Calling showArts …
This motorcycle is a yellow Yamaha RZ350
The engine is on.
…….
Starting engine ….
The engine is already on.
```

تحلیل برنامه :

چیزهایی که در متد main() می بینید خط ۱ تعریف متد main() است و شما آن را همواره به همین شکل خواهید دید . در خط ۲ یک شیء از کلاس Motorcycle بنام m تعریف شده است . توجه کنید که با کلاس ها به طور منظم کاری صورت نمی گیرد بلکه یک وهله از آنها (یک شیء) مورد استفاده قرار می گیرد . در خطهای ۳ و ۴ متغییرهای حالت این شییء ست می شوند : Yamaha RZ350 و رنگ زرد . در خط های ۵ و ۶ متد ShowArts() فراخوانی می شود (در حقیقت فقط خط ۶ فراخوانی این متد است و خط ۵ فقط پیامی است مبنی بر فراخوان آن) . این متد نوع ، رنگ و روشن یا خاموش بودن شییء موتور سیکلت را نمایش خواهد داد . توجه کنید که در ابتدا موتور خاموش است .

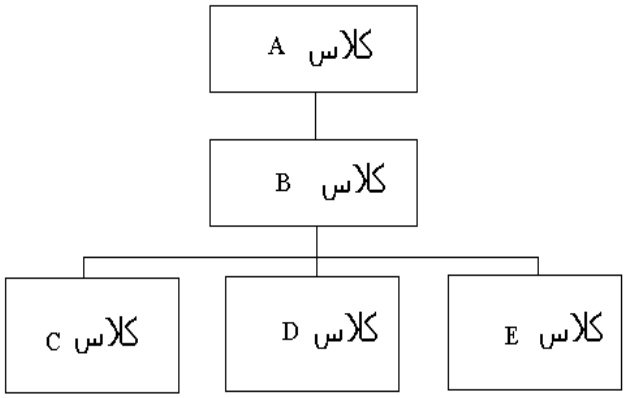
خط ۷ یک خط رسم می کند که قسمتهای مختلف خروجی را از هم جدا می کند و فقط برای شکیل تر شدن برنامه است . در خط ۹ با فراخوانی متد startEngine موتور روشن می شود . خط ۱۲ دوباره متد showArts() را فراخوانی کرده و مقدار متغییرهای حالت را نمایش می دهد . در این لحظه موتور روشن است .

خط ۱۵ سعی می کند که موتور را دوباره روشن کند ، ولی موتور قبلاً روشن شده است و پیام داده شده هم موید همین مطلب است .

وراثت (Inheritance) :

وراثت یکی از کلیدی ترین مفاهیم برنامه نویسی شیء گراست و تاثیر مستقیمی روی نحوه طراحی و نوشتن کلاس های جاوا دارد . وراثت مکانیزمی است برای تغییر شکل دادن به کلاس ها و استفاده مجدد از آنها ، با مکانیزم وراثت می توان به طور خودکار از اطلاعات کلاس های دیگر استفاده کرد .

اصطلاح جدید (وراثت تمام کلاسها را در یک سلسله مراتب گرد آورده و مرتب می کند .) به شکل زیر نگاه کنید .



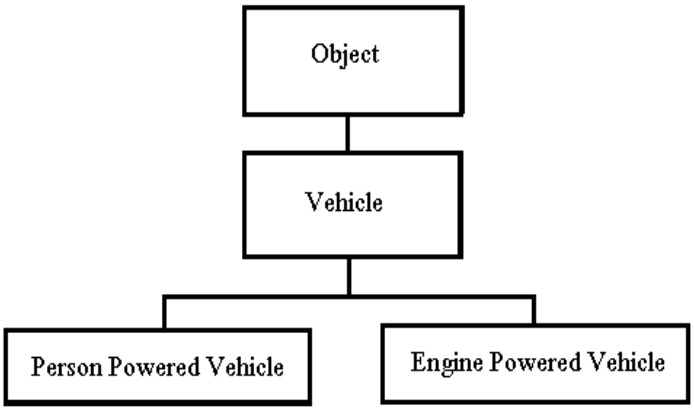
هر کلاس یک فوق کلاس (Super class) ، و یک یا چند زیر کلاس (Sub Class) دارد .
هر کلاس از کلاسهای بالاتر خود در این سلسله مراتب ارث می برد و به کلاس پایین تر از خود ارث می دهد . هر زیر کلاس تمام متدها و متغییرهای فوق کلاس خود را به ارث می برد و دیگر نیازی به تعریف مجدد آنها وجود ندارد .
بالاترین کلاس در سلسله مراتب کلاس های جاوا کلاس Object است و تمام کلاسهای دیگر از این کلاس مشتق می شوند . این کلاس تعیین کننده کلی ترین متدها و خواص تمام کلاس های ذیل خود است . معمولاً کلاس ها ضمن ارث بردن از کلاسهای بالاتر از خود ، چیزهایی را به آن اضافه و پیاده سازی می کنند .

ایجاد سلسله مراتب کلاس :

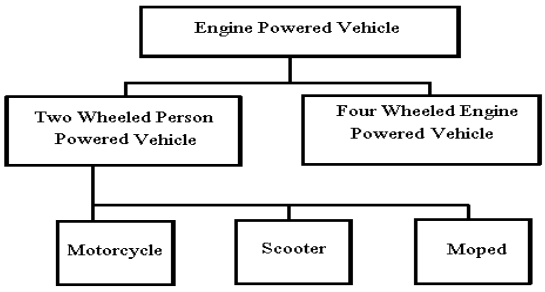
وقتی تعداد کلاس ها زیاد است ، منطقی است که آنها را در یک سلسله مراتب گرد آوریم تا بتوانیم نحوه به ارث رسیدن متدها و خواص را بهتر کنترل کنیم . بدین ترتیب نیاز به دوباره نویسی کدها به حداقل رسیده و هنگام نیاز به تغییر کدها نیز اصلاحات فقط در یک نقطه انجام خواهد شد و از آنجا به تمام نقاط دیگر دسترسی خواهند یافت .
اجازه دهید با استفاده از مثال Motorcycle موضوع را روشنتر کنیم . دیدید که کلاس Motorcycle چگونه طراحی شد (و بخوبی کار کرد). حال فرض کنید بخواهیم کلاس جدیدی بنام Car (اتومبیل) بسازیم . موتور سیکلت و اتومبیل شباهتهای زیادی با هم دارند . در نظر اول بنظر می رسد که بهتر است کلاس Motorcycle را باز کنیم و اطلاعات درون آن را به داخل کلاس جدیدی بنام Car کپی کنیم . اما ایده بهتری هم وجود دارد و آن انتقال اطلاعات مشترک این دو کلاس به یک کلاس کلی تر است . با اینکه این روش کار بیشتری می طلبد ولی توسعه آن و ایجاد کلاسهای جدیدتر (مانند دوچرخه ، کامیون و غیره) را ساده تر خواهد کرد .
اجازه دهید با شروع از بالاترین کلاس جاوا در سلسله مراتب کلاس ها ، یعنی کلاس Object ، یک سلسله مراتب جدید بسازیم . کلی ترین کلاسی که می تواند این دو وسیله نقلیه را در خود جای دهد را کلاس Vehicle می نامیم . یک وسیله نقلیه (Vehicle) چیزی است که می تواند فردی را از نقطه ای به نقطه دیگر منتقل کند . در کلاس Vehicle فقط رفتار انتقال از نقطه a به نقطه b را تعریف خواهیم کرد و نه هیچ چیز بیشتر .
در ذیل کلاس Vehicle چه چیزی باید قرار دهیم ؟ مثلاً ماشین متحرک با نیروی انسانی و ماشین متحرک با موتور چطور است ؟ ماشین متحرک با نیروی انسان مثلاً می تواند پدال داشته باشد (و یا هر مکانیزمی برای انتقال حرکت انسان به

ماشین). ماشین متحرک با موتور هم (به احتمال زیاد!) یک موتور دارد ، به سوخت نیاز دارد و به وسیله ای برای تنظیم سرعت .
در ذیل کلاس Engine Powered Vehicle می توانیم چند کلاس مثل Car ، Motorcycle و Truck قرار دهیم . ولی می توانیم با ایجاد یک کلاس بینایی باز هم رفتارها را تقسیم بندی کنیم .
بالاخره کلاس Motorcycle را در ذیل کلاس Two Wheeled (دو چرخ) قرار داده ایم . اما شاید بپرسید که خواصی مثل رنگ و نوع را کجا باید قرار داد ؟ می توان این خواص را در کلاس Vehicle قرار داد تا تمام زیر کلاسها آنها را به ارث ببرند . تنها چیزی که باید به خاطر داشته باشید این است که وقتی یک حالت (خاصیت) یا رفتار را تعریف کردید ، دیگر به طور خودکار در سلسله مراتب به تمام زیر کلاسها انتقال خواهد یافت (به ارث خواهد رسید).

سلسله مراتب کلاس Vehicle

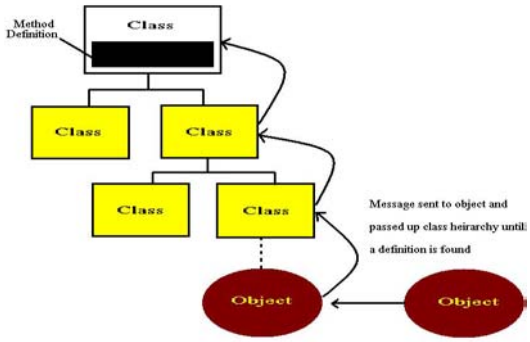


وسایل نقلیه دوچرخ و چهار چرخ



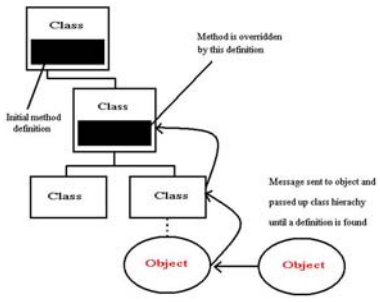
وراثت چگونه عمل می کند ؟

وراثت چگونه عمل می کند ؟ چگونه یک شیء متغییرها و متدهای کلاس بالاتر را به طور خودکار بدست می آورد ؟ در مورد متغییرها ، وقتی شییء بوجود می آید یک مسیر دسترسی به تمام متغییرهای کلاس مربوطه را بدست می آورد . در مورد متدها هم وضع به همین منوال است و تعریف یک متد در اختیار تمام زیر کلاسها قرار خواهند گرفت . وقتی در یک شیء متدی فراخوانی می شود ، جاوا ابتدا در همان کلاسی که متد تعریف شده به دنبال کد آن می گردد . اگر کد آن را نیافت ، در سلسله مراتب یک پله بالاتر رفته و در فوق کلاس بدنبال آن خواهد گشت تا زمانیکه کد متد را بیابند . به شکل زیر نگاه کنید .



نحوه تعیین محل متدها

وقتی در یک زیر کلاس متدی با نام مشابه متدی در یک کلاس بالاتر تعریف شود، کارها کمی پیچیده خواهد شد . در این حالت متدی که زودتر (در حرکت از پایین) یافت شود اجرا خواهد شد . بدین ترتیب متد تعریف شده در زیر کلاس متد فوق کلاس را مخفی می کند . به این وضعیت تحریف (Override) متد گفته می شود . شکل زیر را ببینید .



تحریف متدها

اصطلاح جدید :

به تعریف یک متد با نام مشابه متدی در یک فوق کلاس ، تحریف می شود . متد زیر کلاس متد فوق کلاس را مخفی می کند .

وراثت منفرد و چند گانه :

جاوا به گونه ای که دیدید 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 ، مانند C++ ، 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 .

واسط ها (Interfaces) و بسته ها (Packages) :

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 .

توضیح :

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 بسته Java.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.

ایجاد یک زیر کلاس :

چگونه می توان یک زیر کلاس ساخت و چند متد را در آن تحریف کرد . در این مثال با بسته ها هم بیشتر آشنا خواهید شد .

فرض کنید می خواهید یک اپلت بوجود آورید . تمام اپلت ها در جاوا زیر کلاس کلاسی بنام Applet هستند (که در بسته java.applet قرار دارد) . با ایجاد یک زیر کلاس از کلاس Applet می توانیم تمام رفتارهای آن (از قبیل کار با پنجره ها ، ارتباط با سیستم و پاسخ به رویدادهای Keyboard و Mouse را به ارث ببریم . در این مثال یک اپلت شبیه اپلت Hello World خواهیم ساخت که این عبارات را با رنگ و اندازه دیگری نمایش خواهد داد . ابتدا یک کلاس تعریف می کنیم :

```
Public class HelloAgainApplet extends java.applet.applet{
}
```

در اینجا کلاسی بنام HelloAgainApplet تعریف شده است ، به قسمت extends... دقت کنید ، در حقیقت این عبارت است که می گوید اپلت شما یک زیر کلاس از کلاس Applet خواهد بود . به مسیر کامل کلاس Applet دقت کنید ، چون این بسته بطور پیش فرض برای جاوا تعریف نشده ، باید مسیر کامل آن ذکر شود .

کلمه Public می گوید که این کلاس در اختیار تمام سیستم قرار خواهد داشت . تا اینجا کلاس ما تمام متغییرها و متدهایش را از کلاس بالاتر (Applet) به ارث برده و کاملاً شبیه آن است . اجازه دهید فونت آن را عوض کنیم :

```
Font f=new font("TimesRoman",Font.BOLD,36);
```

در اینجا f متغییری است از کلاس Font که جزء بسته java.awt است . با تعریف فوق فونت سیستم از نوع TimesRoman ضخیم و با اندازه ۳۶ خواهد شد .

با استفاده از شیيء فونت می توان فونت اپلت را تغییر داد .

حال باید متدی تعریف کرد تا از این فونت استفاده کند . متدی که عبارات را روی صفحه نمایش می دهد متد ()paint است که در اینجا آن را تع

ریف خواهیم کرد تا عبارت Hello World با فونت جدید رسم شود . تعریف جدید متد مزبور چنین است :

```
Public void paint (Graphics g)
{ ,
      g.setFont(f);
      g.setColor(Color.red);
      g.drwString("Hello again!" ,5 , 25);
}
```

در اینجا به دو نکته باید توجه کنید . اول اینکه این متد Public است چون متدی که هم نام آن است خود Public است . وقتی می خواهید متدی را تحریف کنید باید میدان دید (Scope) آن مانند متد فوق کلاس متناظرش باشد ، دوم اینکه ، متد ()Paint یک

آرگومان ورودی ، که یک شیء از کلاس Graphics است ، دارد . این کلاس یک روش مستقل از سخت افزار برای عملیات گرافیکی است .
در متد ()Paint سه کار انجام داده ایم :

- گفته شده که شیء گرافیکی از فونت f استفاده کند .
- گفته شده که رنگ رسم عبارت رنگ قرمز (Color.red) است .
- در پایان هر عبارت "Hello Again" در نقطه (5,25) رسم می شود .

کلاس جدید به همین راحتی تعریف شد ! اپلت ما تا این جا باید مانند ذیل باشد :

```
Public class HellAgainApplet extends java.applet.Applet
{
      Font f=new Font("TimesRoman",Font.BOLD,36);


      Public void paint (Graphics g)
      {
          g.setFont(f);
          g.setColor(Color.red);
          g.drawString("Hello again!" ,5,40);
      }
}
```

اما مثال ما یک اشتباه دارد . اگر نمی توانید حدس بزنید که این اشتباه چیست ، اپلت را کامپایل کنید . کامپایلر با پیغام ذیل کارش را متوقف خواهد کرد :

```
HelloAgainApplet.java:7:Class Graphics not found in type declaration
```

این پیغام خطا برای چیست ؟ به یاد دارید که بسته پیش فرض java بسته java.lang است ولی شما در خط اول تعریف کلاس از بسته java.applet استفاده کرده اید . اما می بینید که کامپایلر از این خط خطا نگرفته است چون مسیر کامل بسته را قید کرده ایم . یک راه حل برای رهایی از این قبیل مشکلات آن است که مسیر کامل تمام کلاسها را قید کنیم . اما اگر در یک اپلت بدفعات از یک یا چند کلاس استفاده کنیم ، هر بار نوشتن نام کامل مسیر کلاس ها بسیار خسته کننده و وقت گیر خواهد بود . در این موارد بهتر است از دستور import استفاده کنیم . در اپلت فوق از سه کلاس (Color , Font , Graphics) استفاده کرده ایم که همگی در بسته java.awt قرار دارند . بنابراین ، دستورات ذیل را به اول برنامه ، قبل از تعریف کلاس ، اضافه کنید :

```
Import java.awt.Graphics;
Import java.awt.Font;
Import java.awt.Color;
```

نکته :
با استفاده از یک ستاره (*) می توان تمام کلاس های Public یک بسته را مورد استفاده قرار داد ، مانند ذیل :

```
Import java.awt.*;
```

حال که کلاس را تصحیح کردیم ، می توانیم اپلت Hello Again Applet را کامپایل کنیم . برای تست این اپلت ، فایل HTML جدیدی مانند ذیل بسازید :

```
<HTML>
<HEAD>
<TITLE>Another Applet</TITEL>
</HEAD>
<BODY>
<P>My second java applet says:
<BR><APPLET CODE=”Hello Again Applet.class” WIDTH=200
HEIGHT=50>
</APPLET>
</BODY>
</HTML>
```

این فایل را با نام HelloAgainApplet.html ذخیره کرده و آن را در یک کاوشگر مجهز به جاوا اجرا کنید .

یکی از سخت ترین جنبه های تسلط بر برنامه نویسی شیيء گرا درک همین مفاهیم (عجیب و غریب) آن است . در اینجا مرور مجددی بر این مفاهیم خواهیم داشت .

- کلاس : الگویی برای اشیاء که در آن رفتار و خواص شیيء تعریف نشده اند .
- شیيء: یک نمونه قابل لمس از یک کلاس از یک روی کلاس می توان چند شیيء بوجود آورد که هر کدام رفتار و خواص خود را دارند .
- وهله : هر شیيء یک وهله است از یک کلاس .
- فوق کلاس : کلاسی که به زیر کلاس هایش ارث می دهد .
- زیر کلاس : کلاسی که از والدینش ارث می برد .
- متد کلاس : متدی که در یک کلاس تعریف می شود .

کلیات جاوا :

حال نگاهی به دستورات جاوا می اندازیم و با آنها آشنا می شویم . از این دستورات معمولاً در متد main() استفاده می شود .

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 .

```
Int i=1;
Import java.awt.font;
System.out.println("This motorcycles is a"+color + ""+make");
m.engineState=true;
```

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 .

متغییرها و انواع داده :

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

متغییرهای وهله خواص یک شیء خاص را در خود نگه می دارند . متغییرهای کلاس مانند متغییرهای وهله هستند با این تفاوت که به وهله های یک کلاس مربوط می شوند . متغییرهای محلی اغلب در درون متدها مورد استفاده قرار می گیرند و برای نگهداری مقادیر درون متد هستند ، در بلوک ها هم می توان از متغییرهای محلی استفاده کرد.همین کد اجرای متد (یا بلوک) به پایان رسید ، متغییرهای محلی دورن آن هم از بین می روند .

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

نکته :

بر خلاف زبانهای دیگر ، جاوا متغییر عمومی (همگانی) ندارد . برای ارتباط بین اشیاء از متغییرهای وهله و کلاس می توان استفاده کرد . به یاد داشته باشید که جاوا یک زبان شییء گرا است و شما به هم کنش اشیاء فکر کنید و نه چیز دیگر .

تعریف متغییرها :

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

```
In myAge;
String myName;
Boolean isTired;
```

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

```
Public static woid main (String arg[])
{
        int count;
        String title;
        Boolean isAsleep;
…
}
```

چند متغییر از یک نوع را می توان در یک جا تعریف کرد :

```
In x,y,z;
String firstName,LastName;
```

و حتی می توان در هنگام تعریف متغییر به آن مقدار داد :

```
Int myAge, mySize, numShoes=28;
String myName=”Laura”;
Boolean is Tired=true;
Int a=4,b=5,c=6
```

به متغییرهای محلی قبل از استفاده حتماً باید مقدار داد، در غیر اینصورت برنامه بدرستی کامپایل نخواهد شد . مقدار دادن اولیه به متغییرهای وهله و کلاس الزامی نیست . چون این متغییرها هنگام تعریف دارای مقدار پیش فرض هستند .

نکاتی درباره نام متغییرها :

نام متغییرها در جاوا 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 اپراتورهای جاوا هستند) به مشکلاتی که می تواند بروز کند دقت کنید . جاوا 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 ۶۵۰۰۰ 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 متغییرها در جاوا نسبت به نوع حروف حساس است و این دقت زیادی را در در هنگام نوشتن برنامه ها می طلبد . به همین دلیل رعایت یک قرارداد هنگام نامگذاری متغییرها می تواند کمک بزرگی در مقابله با مشکلات احتمالی باشد . قراردادهای استفاده شده چنین اند : نام های با معنی ، ترکیب چند کلمه ، کلمه اول با حرف کوچک شروع می شود ، کلمات بعدی با حرف بزرگ شروع می شوند . به این مثال دقت کنید :

```
Button theButton;
Long reallyBigNumber;
Boolean current WeatherStateOfPlanteXshortVersion
```

انواع متغییرها :
هنگام تعریف هر متغییر علاوه بر نام آن باید نوع آن هم مشخص شود . نوع متغییر تعیین می کند که یک متغییر چه مقادیری را می تواند بگیرد . هر متغییر 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 . به جدول زیر نگاه کنید .

نوع	اندازه	محدوده
Byte	۸ بیت	۱۲۸ تا ۱۲۷
Short	۱۶ بیت	۳۲/۷۶۸- تا ۳۲/۷۶۷
Int	۳۲ بیت	۲/۱۴۷/۴۸۳/۶۴۸- تا ۲/۱۴۷/۴۸۳/۶۴۷
Long	۶۴ بیت	۹/۲۳۳/۳۷۲/۰۳۶/۸۵۴/۷۷۵/۸۰۷- تا ۹/۲۲۳/۳۷۲/۰۳۶/۸۵۴/۷۷۵/۸۰۷

این انواع همگی علامت دار هستند ، یعنی می توانند اعداد مثبت و منفی را در خود ذخیره کنند . نوع متغییر انتخاب شده به عددی که می خواهید ذخیره کنید بستگی دارد . اگر عدد بزرگتر از متغییر باشد ، بی سرو صدا قیچی خواهد شد !

برای ذخیره کردن اعدا دارای ممیز از نوع اعشاری (با ممیز شناور – floating point) استفاده می شود . اعداد اعشاری در جاوا از استاندارد IEEE 754 تبعیت می کنند . در جاوا دو نوع عدد اعشاری وجود دارد : float (۳۲ بیت ، دقت ساده) و Double (۶۴ بیت ، دقت مضاعف) .
نوع داده کاراکتر (char) برای ذخیره کردن یک کاراکتر است . چون جاوا از یونی کد استفاده می کند هر متغییر 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 ، متغییرهای جاوا می توانند از نوع کلاس هم باشند :

```
String lastName;
Font basicFont:
OvalShape myOval;
```

این متغییرها یک وهله از کلاس مربوطه هستند .

نکته :

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

مقدار دادن به متغییرها :

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

```
Size=14;
TooMuchCaffiene=true;
```

توضیحات (Comments) :

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

```
/* I don’t know how I wrote this next part; I was working
   Realy late one night and it just sort of appeared. I
   Suspect the code elves did it for me.it might be wise
   Not to try and change it.
/*
```

این نوع صحیح می تواند چند خطی باشد .
برای توضیحات تک خطی از // می توان استفاده کرد:

Int vices=7; //are there really only 7 vices?
نوع سوم توضیح که برنامه javadoc از آن استفاده می کند یا **/ شروع و با */ پایان می یابد . این نوع توضیح از همه نظر شبیه نوع اول است .

واژه ها :

در جاوا 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 .

واژه های عددی :

در جاوا 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 باید جلوی عدد از حرف l یا L استفاده کنید . مثلاً ، 4L عدد صحیح 4 را در یک واژه long ذخیره می کند .

اعداد صحیح را به صورت اکتال (در مبنای هشت) و هگزا دسیمال (در مبنای شانزده) هم می توان ذخیره کرد . یک ۰ در جلوی عدد نمایش دهنده اکتال بودن آن است – مانند ، ۰۷۷۷ یا ۰۰۰۴ . اگر جلوی واژه 0X نوشته شود آن واژه به صورت هگزا دسیمال ذخیره خواهد شد (0XFF یا 0XAF45) . در اعداد هگز علاوه بر رقم های ۰ تا ۹ از حروف A (ده) تا F (پانزده) هم می توان استفاده کرد .

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 (یا F) 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 3.6E-2- .

واژه های منطقی :

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

واژه های کاراکتری :

یک واژه کاراکتری عبارت است از یک حرف که با علامت نقل محصور شده باشد ’a‘ ، ’#‘ ، ’3‘ و غیره . واژه های کاراکتری به صورت یونی کد (۱۶ بیتی) ذخیره می شوند . در جدول زیر چند کد غیر چاپی خاص و کاراکترهای یونی کد را مشاهده می کنید . (در این جدول d نماینده یک رقم است) .

کد	مفهوم
/n	خط جدید (سر خط)
/t	فاصله جدولی (tab

/b	یک کاراکتر به عقب
/r	برگشت سر خط
/f	یک خط به پایین
\\	اسلاش وارونه
/-	نقل تکی
/"	نقل دوتایی
/ddd	عدد اکتال
\xdd	عدد هگزا دسیمال
/udddd	کاراکتر یونی کد

واژه های رشته ای :

یک رشته (String) عبارتست از مجموعه چند کاراکتر . هر رشته در جاوا وهله ایست از کلاس String . بر خلاف C یا 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 :

“Hi , Im a striong”

“”//an empty string

رشته ها می توانند شامل کدهای خاص جدول زیر نیز می باشند :

“A string with a \t tab in it”

“Netsted String are/ “string inside of \ “ other string”

“This String broth to you by java\u2122”

در مثال آخر ، \u2122 کاراکتر یونی کد علامت تجاری ™ است .

نکته :

- 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 .

عبارات و عملگرها :

عبارت (expression) ساده ترین واحد عملیاتی در جاواست .

اصطلاح جدید :

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 متغییر نسبت داد چون دارای مقدار برگشتی است .

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

محاسبات :

جاوا دارای پنج عملگر محاسباتی است . (به جدول زیر نگاه کنید)

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 31%9 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 .

عملگر	مفهموم	مثال
+	جمع	۴+۳
-	تفریق	۷-۵
*	ضرب	۵*۵
/	تقسیم	۱۴/۷
%	باقیمانده	۲۰%۷

در لیست زیر چند محاسبه ساده ریاضی را مشاهده می کنید .

```
1: class Arithmetic Test{
2: public static void main (string args[]){
3: short x=6;
4: int y = 4;
5: float a=12.5f;
6: float b=7f;
```

```
7:
8: system.out.println(“x is”+x+”,y is”+y);
9: system.out.println(“x+y=”+(x+y));
10: system.out.println(“x-y=”+(x-y));
11: system.out.println(”x % y=”+(x/y));
12: system.out.println(“x % y=”+(x%y));
13:
14: system.out.println(“a is “+a+,”b is”+b);
15: system.out.println(“a / b”+(a/b));
16: }
17: }
```

خروجی :

```
X is 6 , y is 4
X + y = 10
X – y =2
X / y=1
X % y =2
a is 12.5 m b is 7
a/b = 1.7871
```

تحلیل برنامه :

در این برنامه ساده جاوا (به متد Main() توجه کنید) ابتدا متغییرها تعریف شده اند (خطوط ۳ تا ۶) به نحوه تعریف متغییرها دقت کنید . باقیمانده برنامه صرفاً چند عمل ساده ریاضی روی این متغییرها و نمایش حاصل این محاسبات است .

دیگر نکته قابل توجه این برنامه متد system.out.println() است . این متد صرفاً یک پیام را روی خروجی استاندارد (که معمولاً مانیتور است) چاپ می کند . متد system.out.println() فقط یک آرگومان ورودی دارد ولی می توان با عملگر + چند رشته را ترکیب کرد و به آن فرستاد .

توضیح بیشتری درباره انتساب مقادیر :

نسبت دادن مقدار (assignment) به یک متغییر 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) . جدول عملگر های ویژه را نشان داده است .

عبارت	مفهوم
X+=y	X=x+y
x-=y	X=x-y
X*=y	X=x*y
x/=y	X=x/y

نکته :
در عبارت پیچیده ممکن است نتیجه این عملگرهای ویژه با معادلهایشان یکسان نباشد . این موضوع کاملاً به نوع عبارت ، پیچیدگی آن و ترتیب محاسبه بستگی دارد .

افزایش و کاهش :
مانند C و 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 --Xهم یکی از x کم می کند . بر خلاف C و C++ ، عملوند در عبارات ++ یا -- می توانند اعشاری هم باشند .
عملگرهای ++ و -- می توانند قبل یا بعد از متغییر 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 . در لیست زیر مثالی که این مطلب را نشان می دهد آورده شده است .

مثال : آزمایش عملگرهای پسوند و پیشوندی افزایش

```
1: class PrePostFixTest{
2:public static void main(String args[]){
3:int x=0;
4:int y=0;
```

```
5:
6: system.out.println(“x and y are”+x+”and”+y);
7: x++;
8: system.out.println(“x++results in” +x);
9: ++x;
10: system.out.println(“++ x results in “+x);
11: system.out.println(“Resetting x back to 0.”);
12: x=0;
13: system.out.println(“----------“);
14: y=x++;
15: system.out.println(“y=x++(Postfix) results in:”);
16: system.out.println(“x is”+x);
17: system.out.println(“y is”+ y);
18: system.out.println(“----------“);
19:
20: y=++x;
21: system.out.println(“y=++x(perfix) results in:”);
22: system.out.println(“x is”+x);
23: system.out.println(“y is”+y);
24: system.out.println (“----------“);
25:
26:}
27:
28:}
```

خروجی :

```
X and y are 0 and 0
X ++ results in 1
++ x results in 2
Resetting x back to 0
----------
Y = x++(postfix) resultls in :
X is 1
Y is 0
----------
Y =++x(prefix) results in :
```

X is 2
y is 2

تحلیل برنامه :

در قسمت اول این برنامه ، X با عملگرهای پیشوندی و پسوندی افزایش داده شده است . در هر دو مورد به X یکی اضافه شده است . در این مثال ساده نتیجه کار در هر دو مورد یکسان است .

در قسمت دوم ، از عبارت y=x++ استفاده کرده ایم . ابتدا مقدار مورد X به Y داده شده و سپس x یکی اضافه می شود ؛ در نتیجه مقدار Y برابر با 0 (مقدار اولیه x) خواهد بود .

در قسمت سوم ، از عبارت پیشوندی Y=++X استفاده کرده ایم . اینجا ، اتفاقی که می افتد بر عکس است یعنی ابتدا مقدار X افزایش داده شده و سپس به Y نسبت داده می شود . در این مورد مقدار y برابر با 2 (مقدار x بعد از افزایش) خواهد بود .

مقایسه ها :

جاوا برای مقایسه مقادیر عبارات متعددی دارد . تمام این عبارات یک مقدار Boolean (یعنی True یا False) بر می گرداند . جدول زیر عملگرهای مقایسه ای را نشان می دهد :

عملگر	مفهوم	مثال
= =	تساوی	X = = 3
= !	نا مساوی	X!=3
<	کوچکتر از	X<3
>	بزرگتر از	X >3
<=	کوچکتر یا مساوی	X<=3
>=	بزرگتر یا مساوی	X >=3

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 True برگشت 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 .

عملگرهای بیت گرا :

عملگرهای بیت گرا روی بیت های عملوند ها عمل می کنند . چون عملیات بیت گرا جزء مباحث پیشرفته برنامه نویسی است در جدول زیر این عملگرها را مشاهده می کنید :

عملگر	مفهوم
&	AND بیت گرا
|	OR بیت گرا
^	XOR بیت گرا
<<	انتقال به چپ
>>	انتقال به راست
>>>	انتقال به راست و پر کردن با صفر
‘	مکمل بیت گرا
=<<	انتساب مقدار بعد از انتقال به چپ (x=x<<y)
=>>	انتساب مقدار بعد از انتقال به راست (x=x>>y)
=>>>	انتساب مقدار بعد از انتقال به راست و پر کردن با صفر (x=x>>>y)
X&=y	AND و انتساب مقدار (x=x&y)
X|=y	OR و انتساب مقدار (x = x|y)
X ^ =y	XOR و انتساب مقدار (x=x^y)

تقدم عملگرها :

هنگام ارزیابی یک عبارت توسط کامپایلر ، تقدم عملگرها نتیجه را مشخص می کند . در اغلب موارد ، این موضوع بر مقدار برگشتی عبارت تاثیر خواهد گذارد . به عبارت ذیل توجه کنید :

Y = 6+4/2

اگر در این عبارت ابتدا 6+4 محاسبه شده و سپس بر ۲ تقسیم شود حاصل ۵ خواهد بود ولی اگر ۴,۲ محاسبه شده و سپس با ۶ جمع شود حاصل ۸ خواهد شد . تقدم عملگرها نحوه محاسبه عبارت را تعیین می کند ، از این رو می توان نتیجه کار پیش بینی کرد . در کل ، افزایش و کاهش قبل از محاسبات حسابی انجام می شوند ؛ عملیات حسابی قبل از عملیات مقایسه انجام می شود ؛ و مقایسه ها قبل از عملیات منطقی صورت خواهد گرفت . در آخر مرحله انتساب مقدار انجام خواهد شد .

جدول زیر تقدم عملگرها را در جاوا نشان می دهد . تقدم عملگرها از بالا و پایین کم می شود ؛ عملگرهای هر سطر دارای تقدم یکسانند و از چپ به راست ارزیابی خواهند شد . با توجه به جدول ، در عبارت y=6+4.2 ابتدا تقسیم و سپس جمع انجام خواهد شد بنابراین حاصل عبارت ۸ خواهد بود .

عملگر	توضیح
.[]()	پرانتز () برای دسته بندی عبارات است ؛ برای دسترسی به متدها و متغییرهای یک شیء یا کلاس از نقطه (.) استفاده می شود ؛ کروشه [] در آرایه ها مورد استفاده قرار می گیرد .
++ - - ! instanceof	اگر یک شیء وهله ای از یک کلاس باشد ، عملگر instanceof درست خواهد بود با new یک وهله از کلاس ها ایجاد می شود ، () برای تغییر نوع داده متغییرها به کار می رود .
*/%	ضرب ، تقسیم ، باقیمانده
+-	جمع ، تفریق
>>><<>>	انتقال های بیت گرا
<> <=>=	مقایسه ها
== !=	تساوی
&	AND
^	XOR
|	OR
&&	AND منطقی

||	OR منطقی
?:	نوع خلاصه شده if … then … else
=/=*=-=+= <<=>>>=%=^=&=|=	انواع انتساب مقدارها

با استفاده از پرانتز همیشه می توان ترتیب ارزیابی عبارات را تغییر داد . پرانتزها می توانند در درون هم قرار داشته باشند (پرانتزهای تو در تو) . ارزیابی از داخلی ترین پرانتز شروع می شود . اگر در مثال قبل از پرانتز مانند ذیل استفاده کنیم :

Y=(6+2)/2

ابتدا ۴+۶ محاسبه شده (۱۰)و سپس بر ۲ تقسیم خواهد شد (y=5) . هر گاه در حدس زدن نحوه ارزیابی عبارات توسط کامپایلر دچار شک و تردید شدید ، از پرانتز برای گروه بندی عبارت استفاده کنید .

عملیات رشته ای :

برای وصل کردن دو رشته به یکدیگر در جاوا از عملگر جمع (+) استفاده می شود . قبلاً دیدید چگونه از این عملگر استفاده کردیم . عملگر + از چند رشته یک رشته جدید می سازد . اگر هر یک عملوندهای این عملگر رشته نباشد به طور خودکار به رشته تبدیل خواهد شد .

عملگر =+(که در قسمت قبل با آن آشنا شدید) روی رشته ها هم عمل می کند . مثلاً ، عبارت

My Name+-“Jr.”;

معادل عبارت ذیل است :

myName=myName+”jr.”;

یک برنامه جاوا از کلاس ها و اشیاء تشکیل می شود . هر کلاس و شییء دارای متدهایی است و یک متد از دستورات و عبارات ساخته می شود .

عملگرهای جاوا :

عملگر	مفهوم
+	جمع
-	تفریق
*	ضرب
/	تقسیم
%	باقیمانده

<	کوچکتر است از
>	بزرگتر است از
=<	کوچکتر یا مساوی
>=	بزرگتر یا مساوی
= =	تساوی
=!	نامساوی
&&	AND منطقی
||	OR منطقی
!	NOT منطقی
&	AND
|	OR
^	XOR
<<	انتقال به چپ
>>	انتقال به راست
>>>	انتقال به راست و پر کردن با صفر
‘	مکمل
=	انتساب
++	افزایش
- -	کاهش
=+	جمع و انتساب
=-	تفریق و انتساب
=*	ضرب و انتساب
=/	تقسیم و انتساب
=%	باقیمانده و انتساب
=&	AND و انتساب
=|	OR و انتساب
= <<	انتقال به چپ و انتساب
=>>	انتقال به راست و انتساب
=^	XOR و انتساب
=>>>	انتقال به راست و پر کردن با صفر و انتساب