تاريخچه جاوا :

شايد بزرگترين خدمت انقلاب ميكروپروسسورها، اين بود كه امكان توسعه كامپيوترهاي شخصي را به بشر داد. همانطور كه ديديد كامپيوترهاي شخصي تاثير شگرفي بر زندگي انسانها گذاشت و كليه شئون زندگي بشري را وابسته به خود ساخته است.
بسياري از مردم باور دارند يكي ديگر از تبعات بسيار مهم انقلاب ميكروپروسسورها، مصرف كننده هوشمند در ابزارهاي الكترونيكي مي باشد . به همين منظور در سال 1991 شركت معظم Sun Microsystems گروه تحقيقاتي تحت عنوان Green دايز نمود. هدف اين پروژه خلق يك زبان برنامه نويسي پيشرفته براساس زبانهاي C و C++ بود كه در ابتدا توسط جيمز كاسلينگ مدير اين پروژه به ياد درخت بلوط مستقر در باغ شركت سان Oak ناميده شد ولي تقدير اين بود كه اين پروژه Oak نام نگيرد! زيرا بعداً مديران سان متوجه شدند زبان برنامه نويسي ديگري به نام Oak قبلاً در اداره اختراعات ثبت شده است. سرانجام تقدير اين بود كه در جلسه مديران سان در كافي شاپ محل شركت به پيشنهاد يكي از اعضا اين زبان برنامه نويسي، جاوا ناميده شود.
پروژه گرين چنانچه مورد نظر مسئولين سان بود، ادامه نيافت. زيرا بازار ابزار مصرف كنده هاي هوشمند به سرعتي كه مورد نظر بود جلو نمي رفت و خطر فروش پروژه به شركتهاي ديگر، پروژه گرين را تهديد مي كرد. ولي با انقلاب وب و WWW در سال 1993 اميد تازه اي براي سان به وجود آمد. آنها متوجه شدند ابزار بسيار مناسبي براي ساخت صفحات وب پويا (dynamic web pages)مي باشد.
جاوا رسماً در ماه مي 1995 به بازار ارائه شد و توانست به سرعت نظر مخاطبين را جلب كند. امروزه جاوا كاربران بسيار زيادي با زمينه هاي كاري مختلف دارد؛ از ساخت صفحات وب پويا تا برنامه هاي مورد نياز برروي تلفنهاي همراه و كامپيوترهاي جيبي (Personal Digital Assistants – PDA) و اين راه هنوز ادامه دارد.

كتابخانه كلاسهاي جاوا :

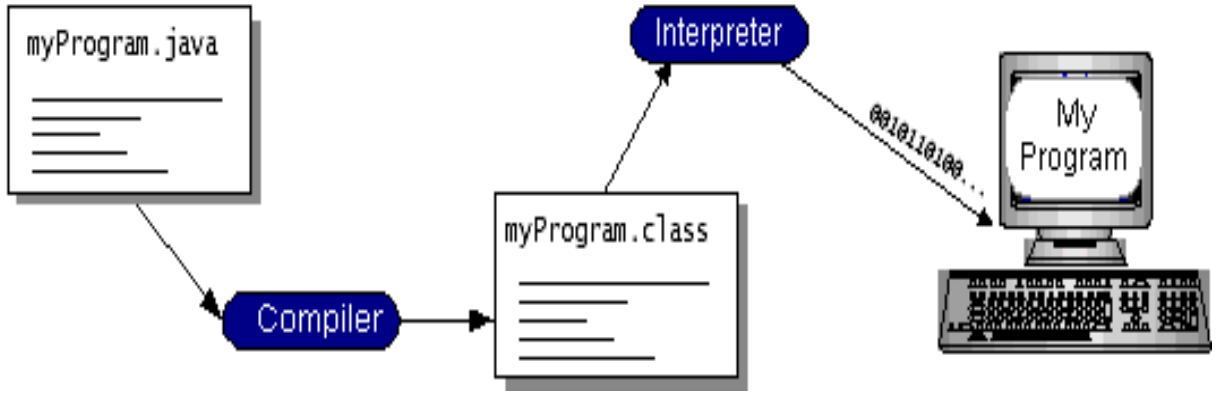
برنامه هاي جاوا شامل كلاسها هستند. كلاسها شامل متدها مي شوند.معمولاً برنامه نويسان به جاي نوشتن تك تك كلاسها، از گنجينه عظيم كتابخانه كلاس جاوا (Java Class Libraries) استفاده مي كنند كه به نام JAVA API (Java Application Programming Interface) نيز معروف است. پس دو راه براي نوشتن يك برنامه به وسيله جاوا وجود دارد اول نوشتن تك تك كلاسها توسط خود برنامه نويس، و راه دوم آموزش و يادگيري چگونگي استفاده از كتابخانه هاي كلاس جاوا.
معمولاً كتابخانه هاي مورد استفاده توسط كامپايلرها و محيط برنامه سازي ارائه مي شود. همين طور در جهان پي كران اينترنت شما به راحتي مي توانيد كلاسهاي مورد نظر خود را رايگان يا با بهايي مختصر بيابيد.
امتياز و مزيت نوشتن كلاس و متد توسط خود برنامه نويس اين است كه، در صورت نوشتن كلاس به كليه مراحل آن كنترل كامل دارد و چگونگي اجراي آن را كاملاً كنترل مي كند. از نكات منفي نوشتن كلاس توسط برنامه نويس مي توان به هدر رفتن زمان اشاره كرد.

زبان برنامه نويسي جاوا :

زبان برنامه نويسي جاوا، يك زبان سطح بالاست كه با كلمات زير مي توان آن را توصيف كرد:

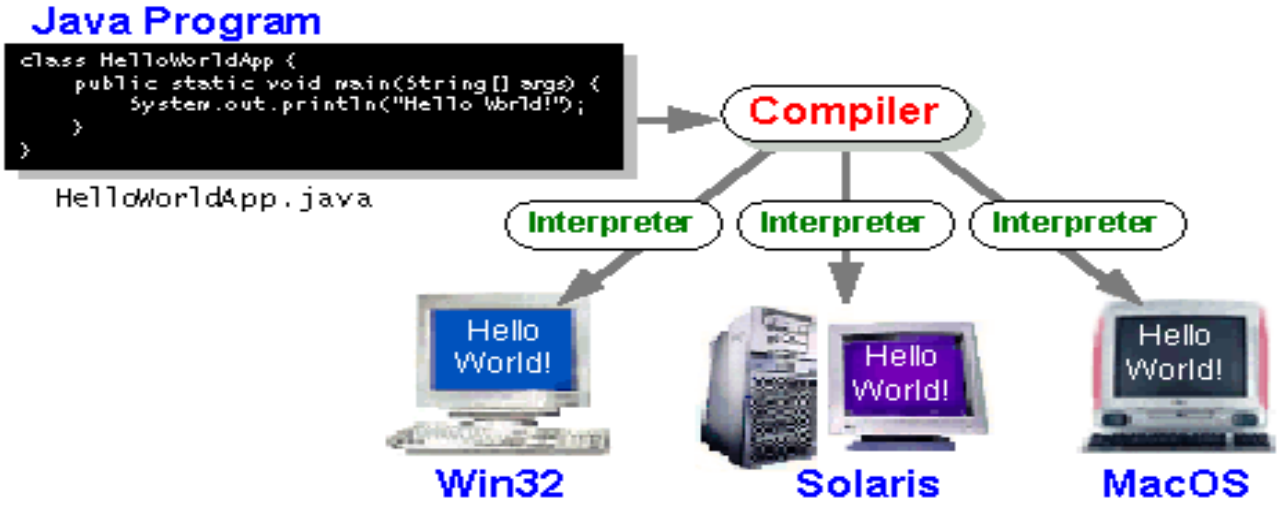
- سادگي
- معطري بي طرفانه
- شيئ گرا
- قابل حمل
- توزيع پذير
- كارا
- مفسري
- پويا
- مطمئن

در اغلب زبانهاي برنامه نويسي، برنامه را طوري ترجمه يا تفسير مي كنيد كه قابل اجرا روي كامپيوتر شما باشد. اما در مقابل زبان برنامه نويسي جاوا، برنامه را هم ترجمه و هم تفسير مي كند. با مترجم، ابتدا برنامه را به زبان ميانه اي ترجمه‌مي كيد كه بايت كد جاوا ناميده مي شود. كد مستقل از محيطي است كه توسط مفسر در محيط جاوا تفسير مي شود . مفسر هر بخشي از دستورات بايت كد را خوانده و در كامپيوتر مقصد اجرا مي كند. ترجمه فقط يكبار اجرا مي شود؛ در حالي كه تفسير در هربار اجراي برنامه انجام مي گيرد. به شكل زير توجه كنيد:



بايت كد جاوا را مي توانيد دستورات زبان ماشين براي ماشين مجازي جاوا (JVM)در نظر بگيريد. هر مفسر جاوا، چه ابزار توسعه جاوا باشد يا مرورگر صفحات وب كه قادر به اجراي اپلت جاوا باشد، يك ابزار ماشين مجازي جاوا است.

بايت كدهاي جاوا "يكبار بنويس، همه جا اجرا كن" را ممكن مي سازد .مي توانيد بايت كد برنامه را در هر محيطي كه مترجم جاوا وجود دارد، بسازيد. بايت كدها مي توانند در هر ماشين مجازي جاوا اجرا بشوند. به اين معني كه هر كامپيوتري كه ماشين مجازي جاوا را داشته باشد، همان برنامه اي كه با زبان برنامه نويسي جاوا نوشته شده باشد مي تواند روي ويندوز 2000، ايستگاه كاري سولاريس يا مكينتاش اجرا گردد.



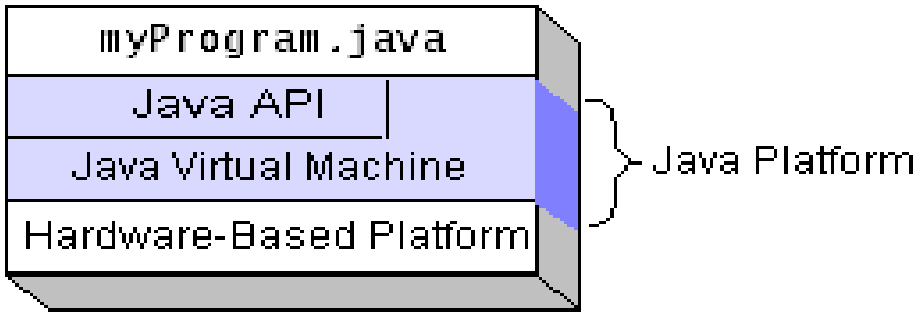
محيط جاوا :

محيط محدوده‌سخت افزاري يا نرم افزاري است كه برنامه در آن اجرا خواهد شد. قبلا برخي از مشهورترين محيط ها را مثل ويندوز 2000، لينوكس و سولاريس ذكر كرديم. اغلب محيط ها مي توانند به عنوان تركيبي از سيستم عامل و سخت افزار تعريف شوند. محيط جاوا با اغلب محيط هاي ديگر مي تواند متفاوت باشد به اين معني كه جاوا محيطصرفا نرم افزاري است كه روي اغلب محيط هاي سخت افزاري ديگر اجرا مي شود.

محيط جاوا از دو جزء تشكيل شده است.

- ماشين مجازي جاوا (JVM)
- رابط برنامه نويسي جاوا (Java API)

تا الان با ماشين مجازي جاوا آشنا شده ايد. اين پايه محيط جاوا است و به محيط هاي سخت افزاري مختلفي برده شده است.
واسط برنامه نويسي جاوا مجموعه وسيعي از اجزا نرم افزاري آماده به كار است كه قابليتهاي بسياري مثل رابط كاربري گرافيكي (GUI)را مهيا مي كند. رابط برنامه نويسي جاوا به كتابخانه هايي متصل و رابط ها دسته بندي شده است؛ اين كتابخانه ها به عنوان *بسته‌ها* شناخته شده اند.
تصوير زير نشان مي دهد كه برنامه اي چگونه در محيط جاوا اجرا خواهد شد. همانطور كه تصوير نشان مي دهد، رابط برنامه نويسي جاوا و ماشين مجازي مانند عايقي برنامه را از سخت افزار جدا مي كنند.



كد طبيعي، كدي است كه بعد از ترجمه ايجاد مي شود، كد ترجمه شده در محيط سخت افزاري خاص اجرا مي شود . در محدوده مستقل از محيط، محيط جاوا مي تواند از كد طبيعي كمي كندتر باشد . هر چند، مترجم هاي هوشمند، مفسرهاي ميزان شده مي توانند كارايي را در حد كدهاي طبيعي برسانند بدون آن كه آسيبي به قابل حمل بودن برنامه برساند.

تكنولوژي جاوا چه كاري مي تواند انجام دهد؟

اغلب برنامه هايي كه با زبان برنامه نويسي جاوا تهيه مي شوند اپلتها هستند يا برنامه ها. اگر در اينترنت چرخ زده باشيد، ممكن است كه با اپلتها آشنا باشيد. اپلت برنامه اي است كه به اصولي پايبند است كه به آن اجازه مي دهد در مرورگرهاي وب با قابليت اجراي جاوا، كار كند.
وليكن، زبان برنامه نويسي جاوا تنها براي اپلتهاي جذاب و سرگرمي نيست. زبان برنامه نويسي همه-منظوره و سطح بالاي جاوا محيط نرم افزاري قويي را نيز مهيا كرده است. با استفاده از رابطهاي برنامه نويسي فراوان، مي توانيد انواع برنامه هاي متفاوت را بنويسيد.
برنامه كاربردي، برنامه مستقلي است كه مستقيما در محيط جاوا اجرا مي شود. نوع خاصي از برنامه هاي كاربردي به صورت سرويس دهنده شناخته شده اند و وظيفه آن سرويس و پشتيباني از استفاده كنندگان شبكه است. براي مثال مي توان از سرويس دهنده وب، سرويس دهنده پراكسي، سرويس دهنده نامه و سرويس دهنده چاپ نام برد. برنامه خاص ديگر سرولت است. سرولت را مي توان اپلتي در نظر گرفت كه در سمت سرويس دهنده اجرا مي گردد. سرولتهاي جاوا انتخابي عامه پسند براي ساختن برنامه هاي تعاملي تحت وب است، كه جايگزين CGIاست. سرولتها مشابه اپلتهات هستند، بجز آن كه به جاي آن كه در مرورگرها كار كنند، در سرويس دهنده وب جاوا اجرا مي گردند.
رابط برنامه چگونه تمامي انواع برنامه ها را مي تواند پشتيباني كند؟ اين عمل توسط بسته اجزاء نرم افزاري انجام مي گيرد كه كارايي زيادي را فراهم مي كند. هر ابزاري كه محيط كامل جاوا را در اختيار مي گذارد، مزاياي زير را در اختيار مي گذارد:

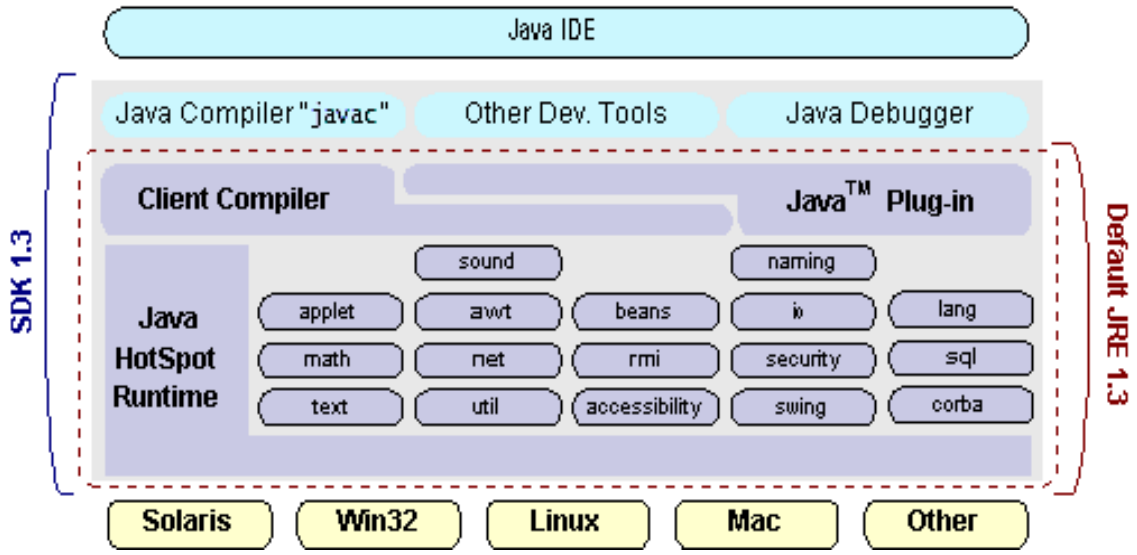
- **اساسها:** اشياء، رشته ها، اعداد، ورودي و خروجي، ساختمان داده، ويژگي هاي سيستم، تاريخ و ساعت و غيره.
- **اپلتها:**مجموعه اي از قراردادها كه توسط اپلتها استفاده مي گردد.
- **شبكه:** URL، TCP، UDP، IP
- **بين المللی سازی:** كمك به برنامه نويسان در جهت توليد نرم افزارهايي كه توسط كاربران در سراسر دنيا مورد استفاده قرار گيرد. برنامه ها مي توانند به طور خودكار با موقعيت محل مورد استفاده خود را وفق دهند و نمايش آنها به زبان محل مورد استفاده تغيير كند.
- **امنيت:** در هر دو سطح پايين و بالا، شامل امضا الكترونيك، مديريت كليد عمومي و خاص، كنترل دسترسي و اسناد.

· **اجزا نرم افزاري:** كه با جاوابين (JavaBean) شناخته شده است و مي تواند به معماري موجود متصل شود.

· **سري سازي اشياء:** اصرار بر كم وزن بودن را اجازه مي دهد و ارتباط از طريق وا اندازي روش راه دور (RMI).

· **ارتباط پايگاه داده ها (JDBC):** دسترسي يكنواختي به محدوده وسيعي از پايگاه داده هاي مرتبط را ميسر مي سازد.

همچنين محيط جاوا رابط هاي برنامه نويسي براي گرافيكهاي 2 و 3 بعدي، دسترسي، سرويس هنده ها، همكاري، تلفني، گفتار، متحرك سازي و ... دارد. تصوير زير نشان مي دهد كه چه چيزهايي در جاوا 2 وجود دارد.



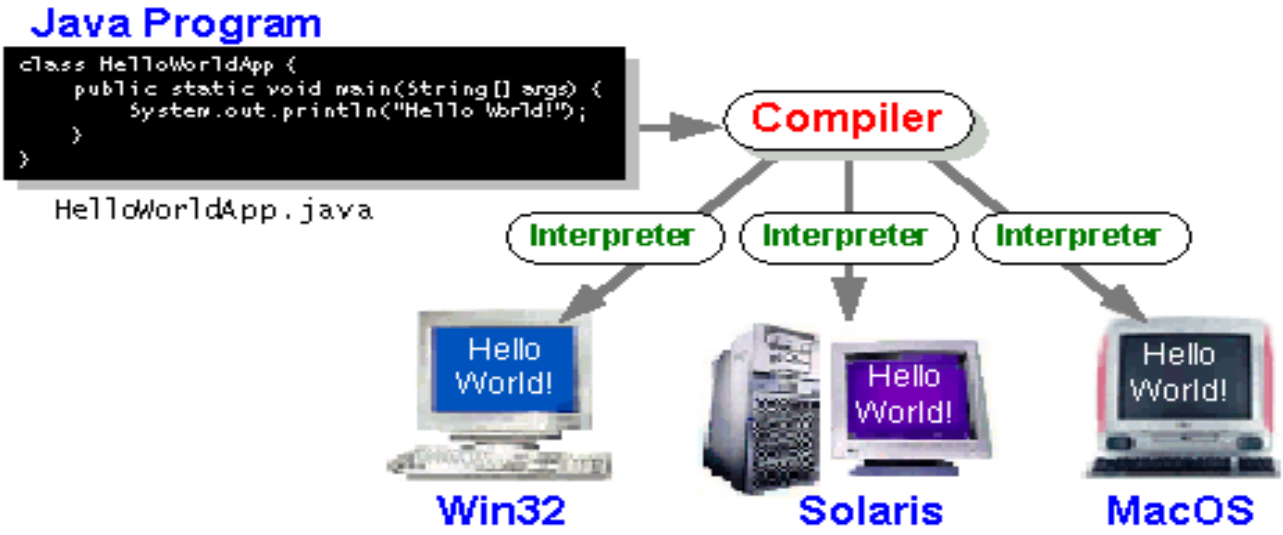
ايجاد اولين برنامه :

اولين برنامه، HelloWorldApp، فقط پيغام خوشامد گويي "Hello World!" را بر روي صفحه نمايش، نشان خواهد داد. براي انجام اين برنامه كارهاي زير را بايد انجام دهيد:

· **ايجاد فايل منبع جاوا.** فايل منبع، شامل متني است كه به زبان برنامه نويسي جاوا نوشته شده است. شما و ديگر برنامه نويسان قادر به درك آن خواهند بود. از هر ويرايشگر متن مي توانيد براي ايجاد فايل اصلي منبع استفاده كنيد.

· **ترجمه فايل به فايل بايت كد.** مترجم جاوا، javac، فايل **منبع** شما را گرفته و متن آن را به دستوراتي ترجمه مي كند كه *ماشين مجازي جاوا* مي تواند بفهمد. مترجم جاوا اين دستورات را در فايل بايت كد قرار مي دهد.

· **اجرا برنامه اي كه در فايل بايت كد قرار داده شده است.** ماشين مجازي جاوا با مفسر جاوا، java، كامل شده است. مفسر فايل بايت كد را گرفته و با ترجمه آن به دستوراتي كه كامپيوتر قادر به درك آن است، خواسته شما را انجام مي دهد.



ایجاد فایل منبع جاوا :

NotePadرا اجرا کنید و کد زیر را در آن تایپ کنید (براي صرفه جويي در وقت مي توانيد اين متن را در NotePad کپي کنيد)

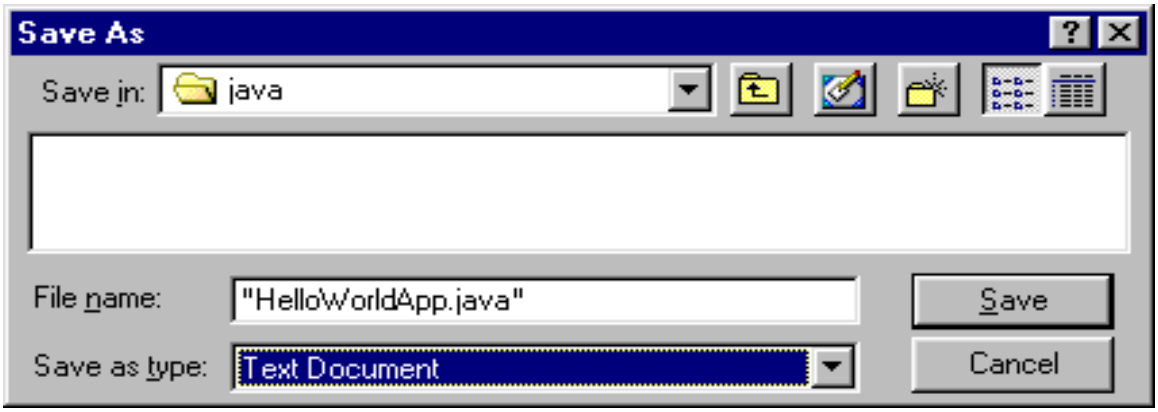
```
/**
* The HelloWorldApp class implements an application that
* simply displays "Hello World!" to the standard output.
*/
class HelloWorldApp {
public static void main(String[] args) {
// Display "Hello World!"
System.out.println("Hello World!");
}
```

موقع تايپ به اين نکته دقت کنيد

HelloWorldApp ≠ helloworldapp

اين کد را فايلي ذخيره کنيد. براي اين کار از منوي اصلي File > Save As را انتخاب کنيد. در پنجره نمايش داده شده کارهاي زير را انجام دهيد:

- در محلي که با Savein مشخص شده است، شاخه (پوشه)اي را که مي خواهيد برنامه در آن ذخيره گردد، مشخص کنيد. در اين مثال درايو C و شاخه java انتخاب شده است.
- در محلي که با Filename مشخص شده است، "HelloWorldApp.java" را تايپ کنيد. علامتهاي " را نيز تايپ کنيد.
- از Save as type عبارت Text Document را انتخاب کنيد.



حالا روي Save كليك كنيد و از NotePad خارج شويد.

ترجمه فايل منبع جاوا

از منوي Start، MS-Dos Prompt (در ويندوز 95 و 98) يا Command Prompt (در ويندوز ان تي) را انتخاب كنيد. وقتي برنامه اجرا شود پنجره اي مشابه اين شكل خواهيد ديد:

محلي كه با Prompt (اعلان) مشخص شده است، شاخه فعلي را نشان مي دهد. در ويندوز 95 و 98 معمولا اين شاخه Widows در درايو C است، همانطور كه در شكل نشان داده شده است، يا Winnt در ويندوز ان تي مي باشد. براي ترجمه فايل منبع به شاخه اي كه فايل شما در آن قرار دارد مراجعه كنيد. مثلا، اگر فايل شما در شاخه java در درايو C قرار داده شده است، دستور زير را در اعلان تايپ كنيد و كليد Enter را بزنيد:

```
cd c:\java
```

نكته: براي تغيير به شاخه اي در درايوي ديگر، دستور اضافه اي لازم است تايپ كنيد.

```
C:\WINDOWS>cd d:\java
C:\WINDOWS>d:
D:\java>
```

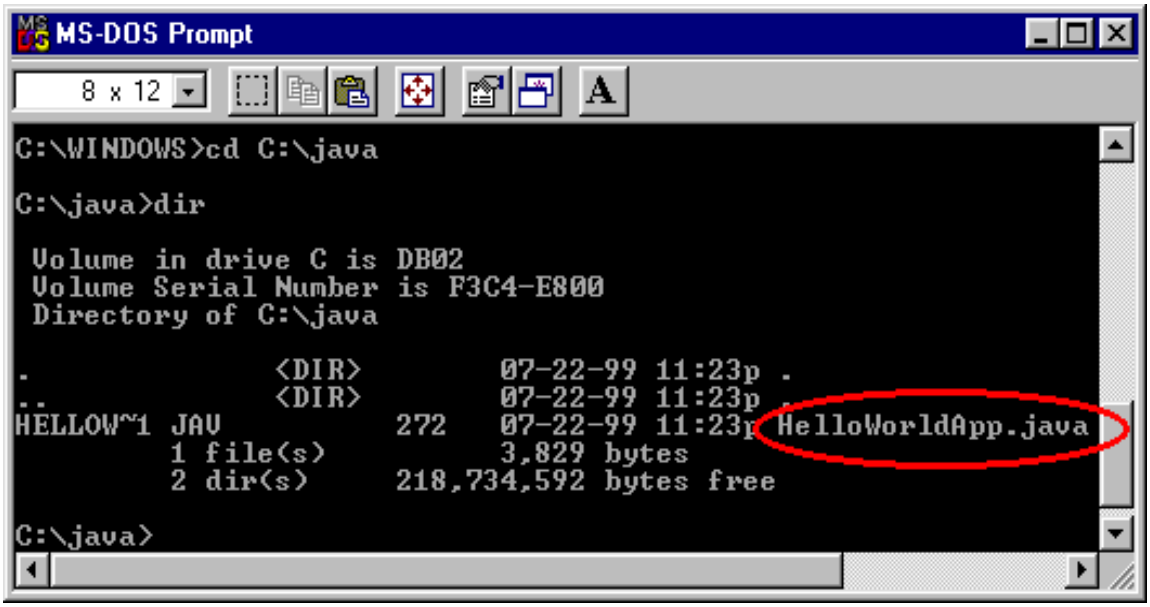
همينطور كه در اين شكل نشان داده شده است، براي تغيير به شاخه java در درايو D بايد d را دوباره تايپ كنيد.

تمامي كد، دستورات و نامهاي فايل بايد دقيقا مشابه آن چه كه نمايش داده شده است، تايپ شود. مترجم و مفسر جاوا نسبت به حالت حروف (كوچكي و بزرگي حروف) حساس هستند.

بعد از انجام اين كارها، پنجره اي مشابه شكل زير خواهيد ديد:

حال بايد اعلان به صورت c:\java ديده شود.

اگر در اعلان dir را تايپ كنيد، مي بايست فايل خود را ببينيد:



الان مي توانيد برنامه را ترجمه كنيد، دستور زير را تايپ كنيد و Enterرا بزنيد.

```
Javac HelloWorldApp.java
```

اگر اعلان بدون پيغام خطايي ظاهر شد، تويك مي گوييم. برنامه شما ترجمه شده است.

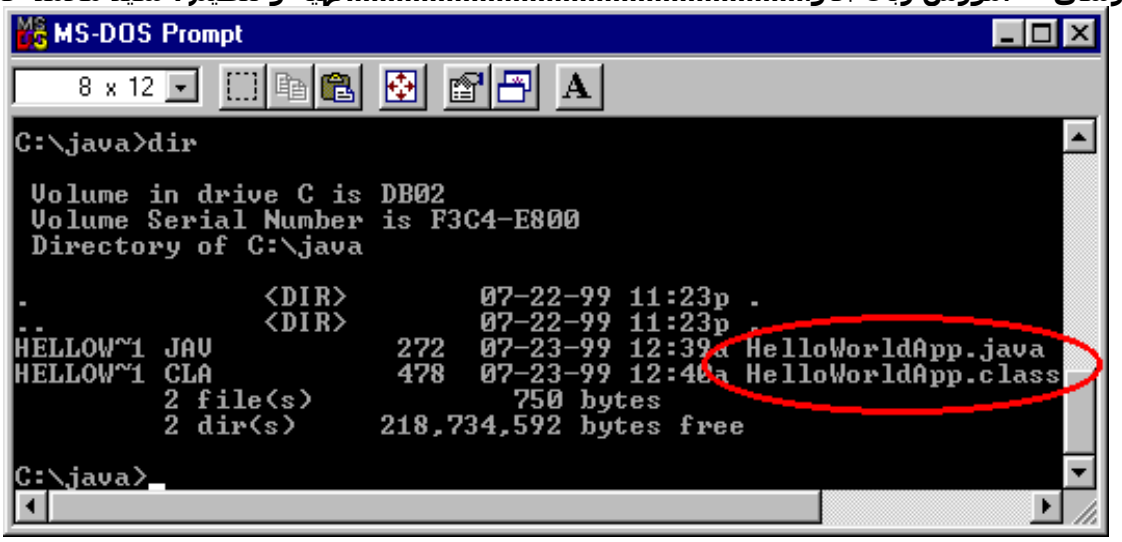
تشريح خطا

Bad command or file name *(Windows 95/98)*
The name specified is not recognized as an internal or external command, operable program or batch file*(Windows NT)*
اگر اين پيغام را مشاهده كرديد، ويندوز محل مترجم جاوا، javac،را پيدا نكرده است.
در اينجا راهي ارائه شده است كه مي توانيد به ويندوز بگوييد از كجا javac را پيدا كند. فرض كيد نرم افزار جاوا 2 را در c:\jdk1.2.2نصب كرده ايد. در اعلان دستور زير را تايپ كنيد و **Enter** را بزنيد:

C:\jdk1.2.2\bin\javac HelloWorldApp.java

نكته: اگر شما اين روش را انتخاب كرده ايد، در هر زماني كه بخواهيد برنامه اي را اجرا يا ترجمه نماييد، بايد دستور javaيا javacرا بعد از C:\jdk1.2.2\binوارد كنيد. براي جلوگيري از تايپ اضافه به بخش *Update the PATH variable*مراجعه كنيد.

مترجم فايل بيت كد جاوا، HelloWorldApp.class، را توليد كرده است. در اعلان dir را تايپ كنيد تا فايل جديدي كه ساخته شده است را ببنيد.



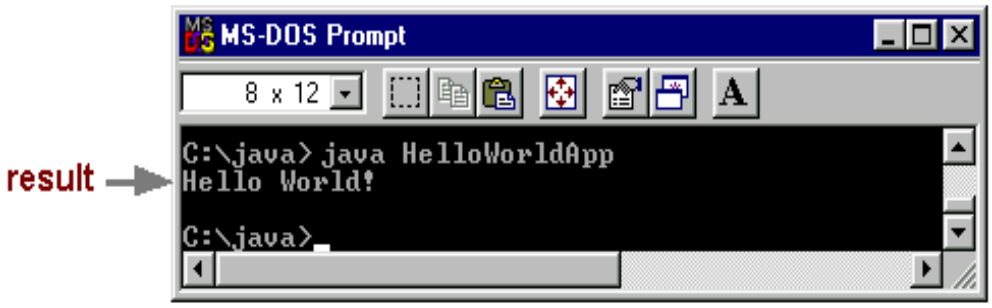
حالا که فایل .class را دارید، مي توانيد برنامه را اجرا کنید.

اجراي برنامه

در همان شاخه دستور زیر را وارد کنید:

```
Java
HelloWorldApp
```

باید شکل زیر را مشاهده کنید:



تبريك! برنامه شط کار مي کند.

به تشریح خطا در ادامه توجه کنید :

تشریح خطا

Exception in thread "main" java.lang.NoClassDefFoundError: HelloWorldApp

اگر پیغام فوق را مشاهده کردید، java نتوانسته فایل بایت کد HelloWorldApp.class را پیدا کند.

یکي از جاهايي که java فايل بايت کد را جستجو مي کند، شاخه فعلي است. پس اگر فايل بايت کد در شاخه c:\java قرار گرفته است، بايد شاخه فعلي را به آن تغيير دهيد. براي تغيير شاخه فعلي دستور زير را در اعلان وارد کنيد:

cd c:\java

اعلان بايد به c:\java> تغيير پيدا کند. اگر dir در اعلان را وارد کنيد، بايد فايلهاي .java و .classرا ببنيد.
حالا دوباره java HelloWorldApp را وارد کنيد.
اگر مشکل همچنان وجود داشت، بايد متغيير CLASSPATH را تغيير دهيد. براي مشاهده اين که آيا اين کار لازم است، سعي کنيد اين متغيير را از کار بياندازيد. دستور زير را وارد کنيد:

```
=set CLASSPATH
```

حالا java HelloWorldApp را دوباره وارد کنید. اگر برنامه این بار درست کار کرد، باید متغییر CLASSPATH تغییر بدهید. براي اطلاعات بیشتر بخش *Check the CLASSPATH Variable* را ببنید.

ایجاد اولین اپلت

HelloWorldApp مثالي از *برنامه* جاوا است، که به تنهايي قابل اجرا است. حالا اپلتي با جاوا خواهيم ساخت که HelloWorld نامیده مي شود و باز هم پیغام "Hello World!" را نمایش خواهد داد . هرچند برخلاف HelloWorldApp اپلت در مرورگر وبي که قابلیت-جاوا داشته باشد، از قبیل NetScape Navigator، HotJava یا Microsoft Internet Explorer اجرا خواهد شد.
براي ایجاد این اپلت همان گامهاي اساسي مرحله پیش را انجام مي دهید: ایجاد فایل اصلي جاوا، ترجمه فایل اصلي؛ و اجراي برنامه.

ایجاد فایل منبع جاوا

NotePad را اجرا کنید و کد زیر را وارد کنید:

```
import java.applet.*;
import java.awt.*;

/**
 * The HelloWorld class implements an applet
that
 * simply displays "Hello World!".
 */
public class HelloWorld extends Applet {
 public void paint(Graphics g) {
 // Display "Hello World!"
 g.drawString("Hello world!", 50, 25);
 }
}
```

در فایلي به نام HelloWorld.java ذخیره اش کنید.
همچنین به فایل HTML احتیاج خواهید داشت که برنامه شما را همراهي کند. کد زیر را در فایل جدید در NotePad تایپ کنید.

```
<HTML>
<HEAD>
```

```
<TITLE>A Simple Program</TITLE>
</HEAD>
<BODY>
Here is the output of my program:
<APPLET CODE="HelloWorld.class"
WIDTH=150 HEIGHT=25>
</APPLET>
</BODY>
</HTML>
```

این کد را در فایلي به نام Hello.html ذخیره کنید.

ترجمه فایل منبع

در اعلان، دستور زیر را تایپ کنید و Enter را بزنید:

```
Javac
HelloWorld.java
```

مترجم جاوا مي بایست فایل بایت کد، HelloWorld.class را تولید کرده باشد.

3-3 اجراي برنامه

هرچند که مي توانيد اپلت جاوا را در مرورگر وب ببنید، شاید ساده تر باشد که اپلت را با برنامه ساده appletviewer آزمایش کنید که در محیط جاوا وجود دارد . براي دیدن اپلت HelloWorld با استفاده از appletviewer در اعلان دستور زیر را تایپ کنید:

```
Appletviewer Hello.html
```

حالا باید پنجره زیر را ببنید:

تبريك! برنامه شط كار مي كند.

متغيرها :

يك شيء خصوصياتش را در متغيرها ذخيره مي كند.

تعريف: متغير جزئي از داده است كه با شناسه اي ناميده مي شود.

براي هر متغيري كه در برنامه استفاده مي كنيد بايد نام و نوع را صريحا آماده كنيد. نام متغير بايد شناسه اي درست باشد –يك سري از كركترهاي يوني كد كه با يك حرف شروع مي شود. با اشاره به نام متغير به داده اي كه در آن ذخيره شده است دست پيدا مي كنيد. نوع متغير تعيين مي كند كه چه عمليات روي آن متغير قابل اجراست. براي دادن نام و نوع به متغير، تعريف متغير را مي نويسيد، كه معمولا شبيه به اين است:

type name

علاوه بر نام و نوعي كه صراحتا براي متغير تعيين مي كنيد، متغير محدود *عمل* (scope) دارد. بخشي از كد كه نام متغير در آن مي تواند مورد استفاده قرار گيرد، محدوه عمل متغير است. محدوده عمل متغير به طور ضمني توسط محلي كه متغير تعريف مي شود، تعيين مي گردد، يعني، جايي كه تعريف صورت مي گيرد در ارتباط با ساير عناصر كد برنامه.

برنامه MaxVariablesDemo ، در زير آمده است، هشت متغير با نوعهاي متفاوت در متد main تعريف مي كند. تعريف متغيرها قرمز است:

```
public class MaxVariablesDemo {
 public static void main(String args[]) {
 // integers
 byte largestByte = Byte.MAX_VALUE;
 short largestShort = Short.MAX_VALUE;
 int largestInteger = Integer.MAX_VALUE;
 long largestLong = Long.MAX_VALUE;
 // real numbers
 float largestFloat = Float.MAX_VALUE;
 double largestDouble = Double.MAX_VALUE;
 // other primitive types
 char aChar = 'S';
 boolean aBoolean = true;
 // display them all
```

```
 System.out.println("The largest byte value is " +
largestByte);
 System.out.println("The largest short value is " +
largestShort);
 System.out.println("The largest integer value is " +
largestInteger);
 System.out.println("The largest long value is " +
largestLong);
 System.out.println("The largest float value is " +
largestFloat);
 System.out.println("The largest double value is " +
largestDouble);
 if (Character.isUpperCase(aChar)) {
 System.out.println("The character " + aChar + " is upper
case.");
 } else {
 System.out.println("The character " + aChar + " is lower
case.");
 }
 System.out.println("The value of aBoolean is " +
aBoolean);
 }
}
```

خروجي برنامه در زير آمده است:

```
The largest byte value is 127
The largest short value is 32767
The largest integer value is 2147483647
The largest long value is 9223372036854775807
The largest float value is 3.40282e+38
The largest double value is 1.79769e+308
The character S is upper case.
The value of aBoolean is true
```

در بخشهاي بعدي جبه هاي ديگر متغيرها با دقت بيشتري توضيح داده شده است كه شامل نوع داده ها، اسامي، محدوده عمل، مقدار اوليه، و متغيرهاي نهايي است. برنامه MaxVariablesDemo از دو موردي استفادهمي كند كه ممكن است شما با آن آشنايي نداشته باشيد و در اين بخش پوشش داده نشود : بسياري از مقادير ثابت كه MAX_VALUE و عبارت if-else . محيط جاوا هر ثابت MAX_VALUE و بيشترين مقدار آن را كه به متغيري مي توان نسبت داد، در يكي از كلاسهاي موجود خود تعريف كرده است.

انواع داده

هر متغيري بايد نوع داده اي داشته باشد. نوع داه ي متغير مشخص مي كند كه چه مقاديري متغير مي تواند قبول كند و چه عملياتي روي آن انجام پذيرد. مثلا در برنامه MaxVariablesDemo، تعريف

int largestInteger تعريف مي كند كه largestInteger نوع داده اي عدد صحيح است (int). متغيري كه از اين نوع تعريف شود، فقط مي تواند شامل اعداد صحيح مثبت و منفي باشد. اعمال حسابي مي توانيد روي آنها انجام دهيد، مثل جمع.
زبان برنامه نويسي جاوا دو دسته نوع داده دارد: *ابتدايي (primitive)* و *ارجاعي (reference)*. متغير ابتدايي شامل يك مقدار ساده است با اندازه و فرمت مناسب است: يك عدد، يك كركتر، يا يك مقدار بولي. مثلا، يك عدد صحيح 32 بيت داده را كه به صورت متمم دو شناخته شده است، مقدار char داده اي 16 بيتي است كه به صورت كركترهاي يونيكد مي باشد.

جدول زير تمام داده هاي ابتدايي را كه توسط جاوا پشتيباني مي گردد، ليست كرده است.

Primitive Data Types

Keyword	Description	Size/Format
	(integers)	
byte	Byte-length integer	8-bit two's complement
short	Short integer	16-bit two's complement
int	Integer	32-bit two's complement
long	Long integer	64-bit two's complement
	(real numbers)	
float	Single-precision floating point	32-bit IEEE 754
double	Double-precision floating point	bit IEEE 754-64
	(other types)	
char	A single character	16-bit Unicode character
boolean	A boolean value (true or false)	true or false

نكته: در زبانهاي ديگر، فرمت و اندازه نوع داده ابتدايي ممكن است به محيط اجرايي كه برنامه در آن اجرا مي شود، بستگي داشته باشد. در عوض، زبان برنامه نويسي جاوا اندازه و فرمت نوع داده را خودش تعيين مي كند. بنابراين، نگران بستگيهاي سيستمي نباشيد.

مي توانيد متغير ابتدايي لفظي (literal) را مستقيما در كدتان وارد كنيد. مثلا، اگر لازم است كه مقدار 4 را به يك متغير صحيح نسبت دهيد، مي توانيد بنويسيد:

int anInt = 4;
رقم 4 مقدار صحيح لفظي است. در اينجا برخي مثالهايي از انواع متغيرهاي ابتدايي داده شده است:

Literal	Data Type
178	Int
8864L	Long
37.266	double
37.266D	double
87.363F	float
26.77e3	double
'c'	char
true	boolean
false	boolean

صحبت كلي، يك سري از ارقام بدون نقطه اعشار از نوع عدد صحيح است.مي توانيد با گذاشتن 'L' يا 'l' بعد از عدد يك عدد صحيح طولاني (long integer) داشته باشيد. 'L'بهتر است زيرا با عدد '1'اشتباه نمي شود. يك سري از اعداد با نقطه اعشار از نوع دبل (double) است. اعداد اعشاري شناور (float) را با قرار دادن 'f' يا 'F' بعد از عدد مي توانيد مشخص كنيد. يك مقدار كركتري هر كركتر يوني كدي است كه بين دو علامت كوت (quote) قرار گرفته باشد.
آرايه ها، كلاسها و رابطها از نوع *ارجاعها*(reference) هستند. مقدار متغيري از نوع ارجاع، در مقابل نوع ابتدايي قرار دارد، ارجاعي است به (آدرسي از) مقدار يا مجموعه مقادير نشان داده شده توسط متغير.
يك ارجاع در زبانهاي ديگر اشاره گر (pointer)يا آدرس حافظه ناميده مي شود. زبان برنامه نويسي جاوا برخلاف ساير زبانهاي برنامه نويسي از آدرس دادن مستقيم استفاده نمي كند. در عوض از نام متغير استفاده مي كنيد.

نام متغيرها

برنامه از طريق نام متغير به مقدار آن دسترسي پيدا مي كند . مثلا، وقتي برنامه مقدار متغير largestByte را نشان مي دهد، برنامه MaxVaraibleDemo از نام largestByte استفاده مي كند. اسمي مثل largestByte را، كه از يك شناسه تركيب شده است، *اسم ساده* ناميده مي شود. اسمهاي ساده در مقابل اسمهاي *توصيفي* قرار دارند، كه كلاس براي ارجاع به اعضا متغير در شيء يك كلاس ديگر استفاده مي كند.
در زبان برنامه نويسي جاوا، قواعد زير براي نام گذاري بايد رعايت شود:

1. شناسه دريت بايد باشد. شناسه يك دنباله از كركترهاي يوني كد است كه با يك حرف شروع مي شود.

2.كلمه كليدي، لفظ بولي (Boolean literal)يعني true يا false، يا كلمه رزورو شده nullنبايد باشد.

3.در محدوده عمل خودش يكتا باشد. متغيري ممكن است هم اسم با متغير ديگري باشد كه در محدوده متفاوتي تعريف شده است. در برخي حالتها، ممكن است كه متغيري نام مشتركي در بلوكهاي تو درتو داشته باشد.

قرارداد: اسم متغيرها با حروف كوچك شروع مي شود و اسم كلاسها با حروف بزرگ. اگر اسم متغيري بيش از يك كلمه باشد، كلمات به يكديگر چسبيده نوشته مي شوند، و هر كلمه ي بعد از كلمه اول با حرف بزرگ شروع مي شود. شبيه به: isVisible. كركتر زير خط (underscore _)در هر قسمتي از اسم قابل قبول است،ولي طبق قرار داد براي جدا كردن مقادير ثابت به كار مي رود (زيرا ثابتها طبق قرارداد همه با حروف بزرگ نوشته مي شوند و بنابراين با وضعيت حروف نمي توان آنها را از هم جدا كرد.)

محدوده عمل (Scope)

محدوده عمل يك متغير منطقه اي از برنامه است كه در آن متغير با نامش مي تواند ارجاع داده شود. ثانيا، محدوده عمل تعيين مي كند كه سيستم حافظه اي براي متغير ايجاد كند و از بين ببرد. محدوده عمل متمايز از ميدان ديد (visibility) است، كه كاربرد آن فقط براي متغيرهاي عضو است و تعيين مي كند آيا متغير مي تواند براي كلاسي كه خارج از آن تعريف شده است استفاده بشود.

محل تعريف متغير در داخل برنامه محدوده عمل را تعيين مي كند و در يكي از اين چهار دسته قرار مي دهد:

- متغير عضو
- متغير موضعي
- پارامتر متد
- پارامتر راهبري-استثناء (Exception-handler)

متغير عضو، عضوي از كلاس يا شيء است. درون كلاس و خارج از هر متد يا سازنده (constructor) تعريف مي شود. محدوده عمل (scope) متغير عضو در تمام محدوده تعريف كلاس مي باشد. هرچند، براي تعريف عضو لازم است كه پيش از آن كه استفاده شود تعريف ظاهر شود.

متغيرهاي موضعي (local) را داخل بلوكي از كد تعريف كنيد. در حالت كلي، محدوده عمل متغير موضعي از جايي كه تعريف مي شود تا انتهاي بلوكي است كه در آن تعريف شده است. در MaxVariablesDemo تمام متغيرهايي كه داخل متد main تعريف شده‌اند، متغيرهاي موضعي هستند. محدوده عمل هر متغير در آن برنامه از تعريف متغير تا انتهاي متد main است – كه با اولين آكولاد } در كد برنامه مشخص مي شود.

پارامترها آرگومانهاي رسمي (formal) براي متدها يا سازنده ها هستند و براي فرستادن مقادير به متدها و سازنده ها استفاده مي شوند. محدوده عمل يك پارامتر كل متد يا سازنده است.

پارامترهاي راهبري-استثناء (Exception-handler) مشابه پارامترها هستند با اين تفاوت كه آرگومانهاي يك راهبري-استثناء هستند به جاي متد يا سازنده. محدوده عمل يك پارامتر راهبري-اعتراض بلوكي از كد است كه بين { و } محصور شده باشد. راهبري خطاها با استثناء درباره استفاده از استثناء (exception) براي راهبري خطاها صحبت مي كند و نشان مي دهد كه چطور كدي براي راهبري استثنائي بنويسيم كه پارامتري هم داشته باشد.

كد مثال زير را در نظر بگيريد:

```
if (...) {
 int i = 17;
 ...
}
```

System.out.println("The value of i = " + i)وك if منتقل گردد يا فراخواني

مقدار اوليه دادن به متغير

در هنگام تعريفِ متغيرهاي محلي و متغيرهاي عضو، اين مي توانند با يك عبارت گمارشي (assignment statement) مقدار دهي بشوند. نوع متغير بايد با مقداري كه به آن اختصاص داده مي شود متناسب باشد. برنامه MaxVariableDemoوقتي كه متغيرهاي محلي را تعريف مي كند براي تمام آنها مقدار اوليه اي در نظر مي گيرد. تعريف متغيرهاي محلي آن برنامه در زير آمده است، قسمتي از كد كه قرمز نوشته شده است مقدار دهي اوليه است:

```
// integers
byte largestByte = Byte.MAX_VALUE;
short largestShort = Short.MAX_VALUE;
int largestInteger = Integer.MAX_VALUE;
long largestLong = Long.MAX_VALUE;



// real numbers
float largestFloat = Float.MAX_VALUE;
double largestDouble = Double.MAX_VALUE;


// other primitive types
char aChar = 'S';
boolean aBoolean = true;
```

پارمترها و راهبرهاي اعتراض از اين راه نمي توانند مقدار دهي اوليه بشوند. مقدار يك پارامتر توسط فراخوان تعيين مي شود.

متغيرهاي نهايي

در هر محدوده ي عملي مي توانيد متغيري تعريف كنيد كه finalباشد. مقدار يك متغير نهايي (final) بعد از آن كه مقداردهي اوليه شد، قابل تغيير نيست. متغيرهاي اين چنين شبيه به مقادير ثابت در ديگر زبانهاي برنامه نويسي مي باشند.
براي تعريف يك متغير نهايي، از كلمه كليدي finalپيش از نوع متغير در تعريف متغير استفاده مي كنيم:

```
final int aFinalVar = 0;
```

عبارت بالا يك متغير نهايي تعريف مي كند و مقدار دهي مي كند، همه اينها يكجا انجام مي شود. هر تلاش بعدي براي مقدار دادن به aFinalVar منجر به پيغام خطاي مترجم خواهد شد.مي توانيد، اگر لازم باشد، محل مقدار اوليه دادن به يك متغير *محلي* نهايي را تغيير داهيد. فقط متغير محلي را تعريف كنيد و مقدار اوليه دادن به آن را بعدا انجام دهيد:

```
final int blankfinal;
. . .
blankfinal = 0;
```

متغير نهايي محلي كه تعريف شده باشد اما مقدار دهي نشده باشد، *نهايي نانوشته* (blank final) ناميده مي شود. بازهم، وقتي كه متغير محلي يك بار مقدار دهي شد، مي توان مجددا به آن مقدار دهي كرد، و هر تلاشي براي مقدار دادن به blankfinal منجر به پيغام خطاي مترجم خواهد شد.

خلاصه متغيرها

وقتي متغيري تعريف مي كنيد، به روشني نوع و نام آن را مشخص مي كنيد. زبان برنامه نويسي جاوا دو مقوله نوع داده دارد: اوليه و ارجاعي. متغيري از نوع اوليه مقداري را در بر دارد. اين جدول تمام نوعهاي اوليه به همراه فرمت و اندازه شان نشان مي دهد:

Keyword	Description	Size/Format
	(integers)	
`byte`	Byte-length integer	8-bit two's complement
`short`	Short integer	16-bit two's complement
`int`	Integer	32-bit two's

		complement
`long`	Long integer	64-bit two's complement
	(real numbers)	
`float`	Single-precision floating point	32-bit IEEE 754
`double`	Double-precision floating point	64-bit IEEE 754
	(other types)	
`char`	A single character	16-bit Unicode character
`boolean`	A boolean value (`true` or `false`)	true or false

محل تعريف يك متغير صريحا تعيين مي كند كه محدوده عمل متغير كجاست، و تعيين مي كند كه در كدام بخش از كد با ذكر فقط نام متغير؛ مي توان به متغير ارجاع داد. چهار مقوله محدوده عمل وجود دارد: محدوده عمل متغير عضو، محدوده عمل متغير محلي، محدوده عمل پارامتر، و محدوده عمل پارامتر راهبر استثناء.
مي توانيد مقدار اوليه براي يك متغير را با استفاده از عملگر گمارش (=) (assignment operator) انجام دهيد.
مي توانيد متغيري را به صورت نهايي (final) تعريف كنيد. مقدار يك متغير نهايي نمي تواند بعد از آن كه مقدار دهي شد، تغيير داده بشود.

عملگرها

عملگرها (operator)تابعي را بر روي يك، دو يا سه عملوند (operand)انجام مي دهند. عملگري كه يك عملوند داشته باشد *عملگر يكاني*(unary operator) ناميده مي شود. مثلا، + + يك عملگر يكاني است كه مقدار عملوند خود را يك واحد افزايش مي دهد. عملگري كه به دو عملوند نياز داشته باشد *عملگر دوتايي*(binary operator)ناميده مي شود. مثلا، = عملگري دوتايي است كه مقدار عملوند سمت راست را به عملوند سمت چپ نسبت مي دهد . و نها يتا، *عملگر سه تا يي* (ternary operator)به سه عملوند نياز دارد. زبان برنامه نويسي جاوا يك عملگر سه تايي دارد، ?:، كه مختصر شده‌ي عبارت if-else است.
عملگر هاي يكاني ا ز علامت گذاري پيشوند(prefix) يا پسوند (postfix)پشتيباني مي كنند. علامت گذاري پيشوند به اين معني است كه عملگر *پيش از* عملوندش مي آيد:

```
operator op          //prefix notation
```

علامت گذاري پسوند به معني آن است كه عملگر *بعد از* عملوندش مي آيد:

```
op operator          //postfix notation
```

تمام عملگرهاي دوتايي از علامت گذاري ميانوند (infix) استفاده مي كنند، كه به معني قرار گرفتن عملگر *ميان* عملوندهايش است:

```
op1 operator op2     //infix notation
```

عملگر سه تايي نيز ميانوند است؛ هر جزء عملگر بين عملوندها ظاهر مي شود:

```
op1 ? op2 : op3      //infix notation
```

به علاوه براي انجام عمليات، يك عملگر مقداري برمي گرداند. مقدار برگشتي و نوعش بستگي به عملگر و نوع عملوندهايش دارد. مثلا، عملگرهاي رياضي، كه عمليات اصلي رياضي مثل جمع و تفريق را انجام مي دهند، اعدادي را برمي گردانند كه نتيجه عمل رياضي است. نوع داده اي كه عمل رياضي برمي گرداند، بستگي به نوع عملوندهايش دارد: اگر دو عدد صحيح را با هم جمع كنيد، نتيجه ي برگشت داده شده عدد صحيح است.
ما عملگرها را به اين مقولات تقسيم مي كنيم:

عملگرهاي رياضي

زبان برنامه نويسي جاوا از عملگرهاي رياضي متنوعي براي تمام اعداد با نقطه شناور (floating-point) و عدد صحيح پشتيباني مي كند. اين عملگرها + (جمع)، - (تفريق)، *(ضرب)، /(تقسيم)، و %(باقيمانده) هستند.
جدول زير عملگرهاي رياضي دوتايي را در زبان برنامه نويسي جاوا خلاصه كرده است.

در اينجا برنامه ي مثالي آورده ايم، ArithmeticDemo، كه دو عدد صحيح تعريف مي كند و دو عدد نقطه شناور با دقت دوبرابر و از پنج عملگر رياضي براي انجام عمليات رياضي متنوع استفاده مي كند. اين برنامه از + براي متصل كردن دو رشته نيز استفاده مي كند. عمليات رياضي با قرمز نشان داده شده است:

```
public class ArithmeticDemo {
   public static void main(String[] args) {


      //a few numbers
      int i = 37;
      int j = 42;
      double x = 27.475;
      double y = 7.22;
      System.out.println("Variable values...");
      System.out.println("    i = " + i);
      System.out.println("    j = " + j);
      System.out.println("    x = " + x);
      System.out.println("    y = " + y);


      //adding numbers
      System.out.println("Adding...");
      System.out.println("    i + j = " + (i + j));
      System.out.println("    x + y = " + (x + y));


      //subtracting numbers
      System.out.println("Subtracting...");
      System.out.println("    i - j = " + (i - j));
      System.out.println("    x - y = " + (x - y));


      //multiplying numbers
      System.out.println("Multiplying...");
      System.out.println("    i * j = " + (i * j));
```

```
        System.out.println("    x * y = " + (x * y));




 //dividing numbers
        System.out.println("Dividing...");
        System.out.println("    i / j = " + (i / j));
        System.out.println("    x / y = " + (x / y));



        //computing the remainder resulting from dividing numbers
        System.out.println("Computing the remainder...");
        System.out.println("    i % j = " + (i % j));
        System.out.println("    x % y = " + (x % y));



        //mixing types
        System.out.println("Mixing types...");
        System.out.println("    j + y = " + (j + y));
        System.out.println("    i * x = " + (i * x));
    }
}
```

خروجي برنامه:

```
Variable values...
    i = 37
    j = 42
    x = 27.475
    y = 7.22
Adding...
    i + j = 79
    x + y = 34.695
Subtracting...
    i - j = -5
    x - y = 20.255
Multiplying...
    i * j = 1554
    x * y = 198.37
Dividing...
    i / j = 0
    x / y = 3.8054
```

```
Computing the remainder...
   i % j = 37
   x % y = 5.815
Mixing types...
   j + y = 49.22
   i * x = 1016.58
```

دقت كنيد كه وقتي يك عدد صحيح و يك عدد نقطه شناور به صورت عملوندهاي يك عمل رياضي به كار مي روند، نتيجه عدد نقطه شناور خواهد بود. پيش از آنكه عملي انجام شود، عدد صحيح به طور ضمني به عدد نقطه شناور تبديل مي شود . جدول زير نوع داده هايي را كه عملگرهاي رياضي مي گردانند خلاصه كرده است. تبديلات لازم پيش از آن كه عملي انجام بشود، اتفاق مي افتد.

علاوه بر شكل دوتايي + و - ، هركدام از اين عملگرها يك شكل يكاني دارند كه عمليات زير را انجام مي دهند:

دو عملگر ميلبُر رياضي وجود دارند: + + ، كه عملگرش را يك واحد افزايش مي دهد، و - - ، كه عملگرش را يك واحد كاهش مي دهد. ++ يا - -مي توانند پيش از *(پيشوند)* از بعد از *(پسوند)* عملوندشان ظاهر شوند. شكل پيشوند، ++op/--op، مقدار پسوند را *بعد از* افزودن/كاستن عملوند ارزيابي مي كند. شكل پيشوند، op++/op--، مقدار عملوند را *پيش از* افزودن/كاستن ارزيابي مي كند.

برنامه زير، SortDemo، دوبار از + + استفاده كرده است و يك بار از - - .

```
public class SortDemo {
    public static void main(String[] args) {
        int[] arrayOfInts = { 32, 87, 3, 589, 12, 1076,
                              2000, 8, 622, 127 };


        for (int i = arrayOfInts.length; --i >= 0; ) {
            for (int j = 0; j < i; j++) {
                if (arrayOfInts[j] > arrayOfInts[j+1]) {
                    int temp = arrayOfInts[j];
                    arrayOfInts[j] = arrayOfInts[j+1];
                    arrayOfInts[j+1] = temp;
                }
            }
        }
```

```
        for (int i = 0; i < arrayOfInts.length; i++) {
            System.out.print(arrayOfInts[i] + " ");
        }
        System.out.println();
    }
}
```

اين برنامه 10 عدد صحيح را در يك آرايه – ساختاري با طول ثابت كه مي توانيد مقاديري با نوع يكسان را نگه دارد- قرار مي دهد و بعد آنها را مرتب مي كند. خطي از كد كه با قرمز است آرايه اي را تعريف مي كند كه با arrayOfInts ارجاع داده مي شود، آرايه را ايجاد مي كند، عدد صحيح را در آن قرار

مي دهد. برنامه از arrayOfInts.length براي گرفتن عناصر آرايه استفاده مي كند . هر عنصر تكي با اين علامت گذاري قابل دست يابي است: arrayOfIndex[index]، وقتي index عدد صحيحي است كه موقعيت عنصر در آرايه را نشان مي دهد. توجه داشته باشيد كه انديس از صفر شروع مي شود.
خروجي برنامه ليستي از اعداد مرتب شده از كوچكترين به بزرگترين است:

2000 1076 622 589 127 87 32 12 8 3

بياييد نگاهي بياندازيم كه SortDemo چطور از حلقه ي خارجي دو حلقه تودرتوي مرتب سازي استفاده مي كند. اينجا عبارتي است كه حلقه بيروني را كنترل مي كند:

```
for (int i = arrayOfInts.length; --i >= 0; ) {
    ...
}
```

عبارت for يك ساختار حلقه است. چيزي كه اينجا مهم است كدي است كه با قرمز نوشته شده است، كه حلقه for را تا زماني كه --i بزرگتر از يا مساوي با صفر باشد، ادامه مي دهد. استفاده از شكل پيشوند -- به اين معني است كه آخرين تكرار حلقه زماني اجرا مي شود كه i برابر با صفر باشد. اگر كد را به صورت پسوند -- تغيير دهيم،آخرين تكرار حلقه زماني اجرا مي گردد كه i برابر با 1- باشد، كه در اين برنامه اشتباه است زيرا كه i براي انديس آرايه استفاده شده است و 1- انديس آرايه صحيحي نمي باشد.
دو حلقه ي ديگر در برنامه از شكل پسوند ++ استفاده مي كنند. در هر دو حالت، شكل استفاده شده واقعا مطرح نيست، زيرا مقدار برگشت داده شده توسط عملگر براي هيچ چيزي استفاده نمي شود.
وقتي كه مقدار برگشتي اين عملگر ميانبُر براي جايي استفاده نمي شود، عرف بر اين است كه از شكل پسوندي استفاده شود.
عملگرهاي ميانبُر افزودن/كاستن در جدول زير خلاصه شده اند.

عملگرهاي شرطي و رابطه اي

يك عملگر رابطه اي دو مقدار را با يكديگر مقايسه كرده و رابطه ي بين آنها را مشخص مي كند. براي مثال، !=اگر دو عملوند آن نامساوي باشند trueبرمي گرداند. جدول زير عملگرهاي رابطه اي را خلاصه كرده است:

مثال زير، RelationalDemo، سه عدد صحيح معرفي مي كند و از يك عملگر رابطه اي براي مقايسه ي آنها استفاده مي كند. عملهاي رابطه اي با قرمز نوشته شده است:

```
public class RelationalDemo {
   public static void main(String[] args) {
      //a few numbers


   int i = 37;
      int j = 42;
      int k = 42;
      System.out.println("Variable values...");
      System.out.println("    i = " + i);
      System.out.println("    j = " + j);
      System.out.println("    k = " + k);
    //greater than
      System.out.println("Greater than...");
      System.out.println("    i > j = " + (i > j));    //false
      System.out.println("    j > i = " + (j > i));    //true
      System.out.println("    k > j = " + (k > j));    //false, they are equal
    //greater than or equal to
      System.out.println("Greater than or equal to...");
      System.out.println("    i >= j = " + (i >= j));   //false
      System.out.println("    j >= i = " + (j >= i));   //true
      System.out.println("    k >= j = " + (k >= j));   //true
    //less than
      System.out.println("Less than...");
      System.out.println("    i < j = " + (i < j));    //true
      System.out.println("    j < i = " + (j < i));    //false
      System.out.println("    k < j = " + (k < j));    //false
    //less than or equal to
      System.out.println("Less than or equal to...");
```

```
        System.out.println("    i <= j = " + (i <= j));   //true
        System.out.println("    j <= i = " + (j <= i));   //false
        System.out.println("    k <= j = " + (k <= j));   //true
      //equal to
        System.out.println("Equal to...");
        System.out.println("    i == j = " + (i == j));   //false
        System.out.println("    k == j = " + (k == j));   //true
      //not equal to
        System.out.println("Not equal to...");
        System.out.println("    i != j = " + (i != j));   //true
        System.out.println("    k != j = " + (k != j));   //false

    }
}
```

خروجي برنامه:

```
Variable values...
    i = 37
    j = 42
    k = 42
Greater than...
    i > j = false
    j > i = true
    k > j = false
Greater than or equal to...
    i >= j = false
    j >= i = true
    k >= j = true
Less than...
    i < j = true
    j < i = false
    k < j = false
Less than or equal to...


    i <= j = true
    j <= i = false
    k <= j = true
Equal to...
    i == j = false
    k == j = true
Not equal to...
    i != j = true
    k != j = false
```

عملگرهاي رابطه اي اغلب با عملگرهاي شرطي استفاده مي شوند تا عبارتهاي تصميم گيري پيچيده تري بسازند. زبان برنامه نويسي جاوا شش عملگر شرطي – پنج عملگر دوتايي ويك عملگر يكاني- را پشتيباني مي كند كه در جدول زير آمده است.
عملگري مثل && عمل *شرطي* AND را انجام مي دهد.مي توانيد از عملگر رابطه اي به همراه && استفاده كنيد تا مشخص كنيد آيا هر دو رابطه درست است يا نه. كدي كه در زير آمده است از اين تكنيك استفاده كرده است تا مشخص كند آيا انديس آرايه بين دو مرز قرار دارد. مشخص مي كند آيا

انديس همزمان هر دو شرط بزرگتر يا مساوي صفر و كوچكتر از NUM_ENTRIESرا ارضا مي كند (NUM_ENTERIES قبلا به مقدار ثابت تعريف شده است).

```
0 <= index && index < NUM_ENTRIES
```

توجه كنيد كه در برخي مثالها، عملوند دوم در يك عملگر شرطي ممكن است محاسبه نشود. اين تكه كد را در نظر بگيريد:

```
(numChars < LIMIT) && (...)
```

عملگر && وقتي trueبرخواهد گرداند فقط اگر هردو عملوند true باشند. لذا، اگر numCharsبزرگتر از يا مساوي با LIMITباشد، عملوند سمت چپ && نادرست (false) خواهد بود، و بنابراين مقدار برگشتي && مي تواند بدون محاسبه عملوند سمت راست مشخص بشود. در بعضي حالتها، مفسر عملوند سمت راست را محاسبه نمي كند.اين مفهوم مهمي دارد اگر عملوند سمت راست اثرات جانبي داشته باشد، مثلا از يك هباله اي مي خواند، يك مقداري را به روز مي كند، يا محاسباتي انجام مي دهد.
وقتي هردو عملوند منطقي باشند (boolean)، عملگر &همان كار && را خواهد كرد. وليكن، & هميشه هردو عملوند را ارزيابي مي كند و trueبرمي گرداند اگر هردو true باشند. مشابها، وقتي عملوندها منطقي باشند، | همان كار || را انجام مي دهد. عملگر | هميشه هردو عملوندش را محاسبه مي كند

و trueبرمي گرداند اگر حداقل يكي از عملوندها true باشد. وقتي عملوندها عدد باشند، & و | عمليات روي بيت انجام مي دهند.
بخش بعدي اطلاعات بيشتري به شما مي دهد.

عملگرهاي منطقي و جابجايي

يك عملگر جابجايي دستكاري روي بيتهاي يك داده اي انجام مي دهد كه بيتهاي عملوندش را به سمت چپ يا راست جابجا مي كند. جدول زير خلاصه اي است از عملگرهاي جابجايي كه در زبان برنامه نويسي جاوا در دسترس است.
هر عملگر بيتهاي عملوند سمت راست به اندازه ي مشخص شده در عملوند سمت راست جابجا مي كند. تغيير جهت در جهتي است كه به وسيله خود عملگر مشخص مي شود. عبارت زير بيتهاي عدد صحيح 13 يك بيت به سمت راست جابجا مي كند:

```
13 >> 1
```

نمايش مبناي دوي عدد 13 به صورت 1101 است. نتيجه عمل جابجايي 1101 به راست به اندازه يك بيت 110 يا 6 در مبناي 10 است. بيتهاي سمت چپ در صورت نياز با صفر پر مي شوند.
جدول زير چهار عملگري را نشان مي دهد كه زبان برنامه نويسي جاوا براي عمليات بيتي روي عملوندهايش تهيه كرده است.
وقتي عملوندهايش عدد هستند، & عمل تابع AND بيتي روي هر جفت بيت موازي در عملوند را انجام مي دهد. تابع AND بيت نتيجه را 1 قرار مي دهد اگر كه بيت متناظر در هردو عملوند 1 باشد. به جدول زير توجه كنيد.
فرض كنيد مقادير 13 و 12 را بخواهيد با هم AND كنيد، شبيه به 12 & 13. نتيجه اين عمل 12است زيرا نمايش مبناي دوي 12 به صورت 1100 است، و نمايش مبناي دوي 13 به صورت 1101 است.

```
    1101    //13
&   1100    //12
    ------
    1100    //12
```

اگر بيتهاي هر دو عملوند 1 باشد، تابع AND بيت نتيجه را 1 قرار خواهد داد؛ در غير اين صورت، بيت نتيجه صفر خواهد بود. لذا، وقتي دو عملوند را به خط مي كنيد و تابع AND را اعمال مي كنيد، مي توانيد ببينيد كه دو بيت مرتبه بالا (high-order) (دوبيتي كه در سمت چپ هر عددي قرار دارد) هر عملوندي 1 است. بنابراين، بيت نتيجه نيز 1 خواهد بود. بيتهاي مرتبه پايين (low-order) به صفر محاسبه شده ان زيرا يكي يا هردوي آنها در عملوند صفر است. وقتي هردوي عملوندها عدد باشند،

عملگر |عمل *ياء منطقي* را انجام‌مي دهد. و ^ عمل *ياء مانعه الجمع* را انجام مي دهد. *ياء منطقي* به معني اين است كه اگر يكي از دو بيت 1 باشد، نتيجه 1 است. جدول زير نتايج عمل ياء منطقي را نشان مي دهد:

ياء مانعه الجمع به اين معني است كه اگر دو عملوند بيتي متفاوت باشند نتيجه 1 است، در غير اين صورت نتيجه صفر است. جدول زير نتايج عمل *ياء منطقي* را خلاصه كرده است.

و نهايتا، عملگر متمم مقدار هر بيتِ عملوند را برعكس مي كند: اگر بيت عملگر 1 باشد نتيجه صفر است و اگر بيت عملوند صفر باشد نتيجه 1 است.

در كنار ساير چيزها، دستكاري بيتها براي مديريت نشانه هاي (flag) منطقي مفيد است. فرض كنيد، برنامه ي شما، تعدادي نشانه ي منطقي دارد كه وضعيت اجزا (component)متفاوتي را در برنامه شما مشخص‌مي كند: آيا قبل ديدن (visible)است، آيا قابل كشيدن (draggable) است، و قس عليهذا. به جاي آن كه متغيرهاي منطقي جداگانه اي براي هر نشانه تعريف كنيد، مي توانيد يك متغير تعريف كنيد، flags، براي تمام آنها. هر بيتِ flags نشاندهنده وضعيت فعلي يكي از نشانه ها خواهد بود. بعد از دستكاري بيتي براي گرفتن و قرار دادن هر نشانه استفاده مي كنيد. اول، مقادير ثابتي تعريف مي كنيد كه نشانه هاي مختلف در برنامه ي شما را مشخص مي كند. براي اطمينان از اين كه هر نشانه فقط از يك بيت استفاده مي كند، هر كدام از اين نشانه ها توانهاي متفاوتي از 2 بايد باشد. يك متغير تعريف مي كنيد، flags، كه بيتهايش متناظر با وضعيت فعلي هر نشانه تنظيم مي شود. كد نمونه ي زير مقدار اوليه به صفر را به flagsمي دهد كه نشان مي دهد تمام نشانه ها false است (هيچ بيتي تنظيم نشده است).

```
static final int VISIBLE = 1;
static final int DRAGGABLE = 2;


static final int SELECTABLE = 4;
static final int EDITABLE = 8;
int flags = 0;
```

براي اين كه نشانه "قابل ديدن" تنظيم شود، وقتي چيزي قابل ديدن شد؛ از عبارت زير استفاده مي كنيد:

```
flags = flags | VISIBLE;
```

براي آزمايش قابل ديدن، مي توانيد بنويسيد:

```
if ((flags & VISIBLE) == VISIBLE) {
   ...
}
```

اينجا كد كامل برنامه آمده است، BitwiseDemo، كه شامل كدهاي فوق است.

```
public class BitwiseDemo {
   static final int VISIBLE = 1;
   static final int DRAGGABLE = 2;
   static final int SELECTABLE = 4;
   static final int EDITABLE = 8;
   public static void main(String[] args)
   {
      int flags = 0;
      flags = flags | VISIBLE;
      flags = flags | DRAGGABLE;
      if ((flags & VISIBLE) == VISIBLE) {
         if ((flags & DRAGGABLE) == DRAGGABLE) {
             System.out.println("Flags are Visible and Draggable.");
         }
      }
      flags = flags | EDITABLE;
      if ((flags & EDITABLE) == EDITABLE) {
        System.out.println("Flags are now also Editable.");
```

```
    }
  }
}
```

خروجي برنامه:

```
Flags are Visible and Draggable.
Flags are now also Editable
```

عملگرهاي نسبت دادن

از عملگر نسبت دادن اصلي، =، براي نسبت دادن يك مقدار به ديگر استفاده مي كنيد. برنامه MaxVariablesDemo از = براي مقدار دهي اوليه به تمام متغيرهاي محلي استفاده مي كند:

```
// integers
byte largestByte = Byte.MAX_VALUE;
short largestShort = Short.MAX_VALUE;
int largestInteger = Integer.MAX_VALUE;
long largestLong = Long.MAX_VALUE;
// real numbers


float largestFloat = Float.MAX_VALUE;
double largestDouble = Double.MAX_VALUE;
// other primitive types
char aChar = 'S';
boolean aBoolean = true;
```

زبان برنامه نويسي جاوا چندين عملگرهاي نسبت دادن ميانبُر آماده كرده است كه اجازه مي دهد يك عمل رياضي، جابجايي، يا عمليات بيتي و عمل نسبت دادن با يك عملگر انجام شود. فرض كنيد مي خواستيد مقداري را با عددي جمع كنيد و نتيجه را به متغير باز گردانيد، شبيه:

```
i = i + 2
```

اين عبارت را مي توانيد با استفاده از ميانبُر +=كوتاه كنيد. شبيه:

```
i += 2
```

دوخط بالا يكسان هستند.
جدول زير ميانبُر عملگرهاي نسبت دادن و معادل طولاني آنها آمده است.

ديگر عملگرها

جدول زير تمام عملگرهاي ديگري كه زبان برنامه نويسي جاوا پشتيباني مي كند، آمده است.

عبارت ميانبُر if-else

عملگر ?: يك عملگر شرطي است كه كوتاه شدهي عبارت if-else است:

```
Op1 ? op2 : op3
```

عملگر ?: اگر op1 درست باشد op2 را باز مي گرداند و در اگر op1 غلط باشد op3 را برمي گرداند.

عملگر []

از كروشه براي تعريف، ايجاد و دسترسي به عنصر خاصي ازآرايه ها استفاده مي كنيد. در اينجا مثالي براي تعريف آرايه آمده است:

```
Float[] arrayOfFloats = new float[10];
```

كدِ قبلْ آرايه اي تعريف مي كند كه مي تواند 10 عدد نقطه شناور را در خود نگه دارد. در اينجا چگونگي دسترسي به عنصر هفتم آمده است:

```
arrraOfFloat[6];
```

توجه كنيد كه انديسهاي آرايه از صفر شروع مي شود.

عملگر .

نقطه (.) به عضو نمونه ي يك شيء يا اعضاي يك كلاس دسترسي مي دهد.

عملگر ()

وقتي متدي تعريف يا فراخواني مي كنيد، آرگومانهاي متد را بين) و (ليست مي كنيد.مي توانيد ليست خالي از آرگومان را با () بدون هيچ چيزي بين آن مشخص كنيد.

عملگر (نوع)

مقداري را به نوع مشخص شده تبديل مي كند.

عملگر new

از عملگر new براي ايجاد يك شيء جديد يا يك آرايه جديد استفادهمي شود. در اينجا مثالي است كه يك شيء جديد Integer از كلاس Integer در بسته java.langمي سازد.

```
Integer anInteger = new Integer(10);
```

عملگر instanceof

عملگر instanceof آزمايش مي كند كه آيا عملوند اول نمونه اي از عملوند دوم هست يا نه.

```
op1 instanceof op2
```

op1 بايد نام يك شيء باشد و op2 بايد نام يك كلاس باشد. يك شيء نمونه اي از يك كلاس در نظر گرفته شده است اگر به طور مستقيم يا غير مستقيم فرزند آن باشد.

تمارين :

1- برنامه ای بنویسید که سه عدد را به عنوان ورودی بگیرد. اگر عدد اول از عدد دوم بزرگتربود عدد سوم را چاپ کند. در غیر این صورت عدد اول را چاپ نماید.

مثال:

مقدار ورودی:

23
14
8

مقدار خروجی:

8

2- برنامه ای بنویسید که از کاربر نام او و سال تولدش را دریافت کند. سپس به او خوشامد بگوید و سن او رامحاسبه نموده و چاپ کند.

مثال:

مقدار ورودی به ترتیب:

Mohammad
1368

مقدار خروجی:

Hi Mohammad
Your age is 19

تهیه و تنظیم :

سید محمد حسینی

دانشجوی مهندسی کامپیوتر (IT)

تابستان 87

Email: m.hosseini@live.com