برنامه جاوا توسط یکی از برنامه نویسان شرکت Sun Microsystems به نام جیمز گوسلینگJames Gosling به عنوان روش بهتر برای ایجاد برنامه های کامپیوتری اختراع گردید . گوسلینگ از نحوه کاربرد زبان C++ روی پروژه ای که مشغول انجام آن بوده رضایت نداشت و از زبان جدیدی ابداع کرد که بهتر از عهده آن کار بر آید .

شروع کار :
اغلب برنامه های کامپیوتری به همان شکلی نوشته می شوند که شما یک نامه را می نویسد . یعنی با تایپ کردن هر جمله در یک ویراشگر متن .
برخی از ابزارهای برنامه نویسی دارای ویرایشگر خاص خود می باشد برخی دیگر با هر نرم افزار ویرایشگر متن کار می کنند .
برای نوشتن برنامه های جاوا نیز باید ابتدا یک ویرایشگر متن مانند Not pad یا Word pad را گشوده برنامه را نوشته آنگاه برنامه را با پسوند Java ذخیره کرد به عنوان مثال Calculator. java نامی برای یک فایل برنامه جاوا است .

پیش از شروع به نوشتن برنامه های ، جاوا لازم است که یکی از نرم افزارهای برنامه نویسی جاوا را تهیه و راه اندازی کنید، کیت توسعه جاوا JDK نامیده می شود که نسخه های متعددی مانند ۱,۲، ۱,٤... دارد .
نسخه های این کیت را می توانید از سایت زیر Download کنید .

Http://www.java.sun.com

در این درس شما اولین برنامه جاوای خود را با وارد کردن آن درون ویرایشگر دلخواه خود، ایجاد خواهد کرد . پس از آن برنامه را ذخیره و کامپایل می کنید و سپس آن را آزمایش خواهید کرد .

آغاز برنامه :
با استفاده از ویرایشگر خود (wordpad,notepad) تمام سطرهای زیر را وارد کرده، دقت کنید که حروف بزرگ و کوچک را دقیقاً به همان صورتی که نشان داده شده وارد کنید.

```
Class exampel۱ {
        Public static void main(String[] arguments) {
            //this is my first java program
        }
}
```

حال برنامه را با نام exampel۱.java ذخیره کنید . باید توجه داشته باشید که کلمه exampel۱ در سطر اول، نشان دهنده نام برنامه است و در برنامه های دیگر تغییر خواهد کرد .
سطر سوم نیز کاملاً واضح است . چرا که جمله ایست به زبان انگلیسی و تنها برای توضیحات بیشتر در برنامه آورده شده است ، به این توضیحات که در ابتدای آنها از // استفاده می شود comments می گویند .
سطر اول برنامه (class exampel۱ {) به ماشین می گوید که نام برنامه را exampel۱ قرار بده یعنی شما از جمله class برای نامگذاری برنامه ی کامپیوتری خود استفاده می کنید البته در درسهای بعدی در رابطه با این جمله توضیحات بیشتری خواهیم داد .
توجه داشته باشید که نام برنامه دقیقاً باید مطابق نام فایل باشد یعنی هر برنامه باید با نامی save شود که دقیقاً بعد از کلمهclass نوشته شده است البته با پسوند java.
سطر بعدی برنامه به این ترتیب است

```
Public static void main(String[ ] arguments) {
```

این سطر به برنامه می گوید که بخش اصلی برنامه از اینجا شروع می شـــود ، در واقع main نقطه شروع برنامه می باشد باید توجه داشت که از آکولاد ها برای گروه بندی بخشهایی از برنامه استفاده می شود ، هر آنچه بین آکولاد باز({) و آکولاد بسته({) قرار دارد بخشهای یک بلوک (block) حساب می شود .
شما باید همواره برای مشخص کردن آغاز و پایان برنامه هایتان بعد از نام برنامه در سطر اول آکولاد را باز کنید و در سطر آخر برنامه آکولاد را ببندید .
در درسهای بعدی بیشتر به توضیح این سطر می پردازیم اینک بیایید برنامه را کامپایل کرده و آنرا اجرا نماییم .
وارد برنامه commant prompt شوید (start/all programas/Accessibility/commant prompt) یا چنانچه علاقه زیادی به کار در محیط dos ندارید از خط فرمانRun در منوی start خط زیر را تایپ کنید:

```
Javac exampel۱.java
```

در صورتی که کامپایل برنامه با موفقیت انجام شود ، فایل جدیدی به نام exampel۱.class در همان پوشه حاوی exampel۱.java ایجاد خواهد شد .

سپس عبارت زیر را در خط فرمان تایپ کنید : Java exampel۱
با اجرای برنامه مطمئناً چیزی مشاهده نمی کنید چرا که هیچ فرمانی مبتنی بر print به برنامه ندادید اما نگران نباشید در آینده ای نزدیک خروجی را نیز مشاهده خواهید کرد

نکته مهمی که در برنامه نویسی جاوا باید به آن توجه کرد آن است که برنامه مورد نظر کجا اجرا می شود.

برنامه های جاوایی که به طور محلی روی کامپیوتر اجرا می شوند،برنامه های کاربردی(Applications) و برنامه هایی که در صفحات وب اجرا می شوند، اپلت (Applet) نامیده می شوند .
اینک توضیحات ما مربوط به برنامه های کاربردی می باشد ، در درسهای آتی به توضیح اپلتها خواهیم پرداخت.

متغییرها:
در برنامه های جاوا متغییرها با دستورالعملی ایجاد می شوند که مشتمل بر دو بخش است :

- نام متغییر
- نوع اطلاعاتی که متغییر در خود ذخیره خواهد کرد .

انواع متغییر
برای نگهداری متغییری از نوع اعداد صحیح (اعداد غیر اعشاری) از نوع int می توان استفاده کرد .int هر عدد صحیح بین ۲/۱٤ – میلیارد تا ۲/۱٤ میلیارد را نگهداری می کند.

مثال: int max; یعنی متغییری با نام max و از نوع اعداد صحیح (int)

و برای ذخیره متغییرهای اعشاری از نوع float استفاده می شود .

مانند: float average; یعنی متغییری با نام average و از نوع اعداد اعشاری (float)

و اما برای ذخیره متغییرهای غیر عددی اگر متغییر از نوع کاراکتر بود از عبارت char استفاده می شود و اگر متغییر رشته بود نوع String به کار برده می شود.

مانند: char key='c'; یعنی متغییری با نام key با مقدار c از نوع کاراکتر
و String name="orbitz"; یعنی متغییری با نام name با مقدار orbitz از نوع رشته

توجه داشته باشید هنگامی که از مقادیر کارکتری استفاده می کنید باید در دو طرف کارکتری که به متغییر نسبت داده می شود ، علامت نقل قول منفرد قرار دهید و در مورد مقادیر رشته ای ، از علامت نقل قول دوتایی استفاده کنید در ضمن یادتان نرود که همواره در تایپ کلمه String ، S را با حرف بزرگ چاپ کنید .

متغییرهایی که تاکنون معرفی کردیم مهمترین انواعی هستند که در اغلب برنامه های جاوا به کار می رود. اما برای اعداد صحیح سه نوع متغییر دیگر نیز به کار می برند ، نوع اول ، byte برای اعداد بین ۱۲۸- تا ۱۲۷ مورد استفاده قرار می گیرد .

مثال byte key=۲۱; این جمله متغییری به نام key با مقدار اولیه ۲۱ ایجاد می کند .

نوع دوم short است که محدوده آن از ۳۲,۷٦۸- تا ۳۲,۷٦۷ می باشد

مانند short number =٤۳; این جمله متغییری به نام number با مقدار اولیه ٤۳ ایجاد می کند .

نوع آخر، یعنی long برای اعداد صحیح بزرگی به کار می رود که نوع int برای نگهداری آنها کفایت نی کند .

و همچنین در زبان جاوا با متغییری برخورد خواهیم کرد به نام Boolean که برای ذخیره مقادیر true (درست) و false (نادرست) به کار برده می شود .
در این زبان برنامه نویسی به دو صورت می توان متغییر ها را مقداردهی کرد :
روش اول : می توان هنگام ایجاد یک متغییر مقداری را به آن تخصیص داد
روش دوم : بعد از تعریف متغییر در هر جایی از برنامه می توان مقداری را در آن قرار داد.
و در صورتی که دو متغییر از یک نوع باشند می توان مقدار یک متغییر را در متغییر دیگر قرار داد ، مثال :

```
Int number=۲٤;
Int key=number;
```

در نتیجه شما دو متغییر با مقدار اولیه ۲٤ تعریف کرده اید .

تقدم عملگرها :
برای ارزیابی عبارات و محاسبه آنها ترتیب زیر باید رعایت شود :

- عملگرهای افزایش (مانند++x ، این عملگر به مقدار x یک واحد می افزاید) و کاهش (مانند--y ، این عملگر از مقدار y یک واحد می کاهد) اولویت اول را دارند .
- ضرب ، تقسیم و تقسیم صحیح در اولویت بعد بعد قرار می گیرند .
- جمع و تفریق در اولویت سوم قرار دارند .

- عملگرهای مقایسه در اولویت بعدی قرار می گیرند .
- علامت مساوی = که برای مقداردهی یک متغییر استفاده می شود دارای کمترین اولویت است.

مثال :

```
Int z=٥;
Int number=x++*٦+٤*١٠/٢;
```

در نتیجه مقدار number مساوی ٥٠ می باشد.

روش نمایش یک رشته در برنامه های جاوا،استفاده از جمله زیر است.

```
System.out.println( );
```

این جمله هر رشته یا متغییر را که داخل پرانتزقرار دارد را چاپ می کند.
مثال:

```
System.out.println("hello");
```

دستور println در واقع مخفف "print this line" (این سطر را در یک خط چاپ کن) می باشد.

چاپ کاراکترهای ویژه:

کاراکترهای ویژه	خروجی
\'	چاپ علامت نقل قول منفرد
\"	چاپ علامت نقل قول دوتایی
\\	backslash
\t	tab
\b	backspace
\r	Carriage return
\f	صفحه جدید
\n	سطر جدید

هنگام استفاده از جمله System.out.println( ); و کار کردن با رشته ها مواردی پیش می آید که می خواهیم دو رشته را به هم بچسبانیم ، برای انجام اینکار از عملگر + استفاده کنیم (این عملگر به جای جمع کردن ، دو رشته را به هم می چسباند).
مثال:

```
System.out.println("\' welcome to"+"\t my weblog\' ");
```

بعد از اجرای دستور بالا مشاهده خواهید کرد عبارت 'welcom to my weblog' چاپ خواهد شد .
و برای چاپ یک متغیر می توان مانند مثال زیر عمل نمود:

```
Int lesson=٦;
System.out.println(" this lesson is"+lesson);
```

در خروجی عبارت this lesson is ٦ نمایان است .

معرفی چند تابع مفید :

برای تعیین طول رشته (برحسب تعداد کاراکتر) از تابع length( ) استفاده می کنند .
مثال :

```
String myname="paradise";
Int key=myname.length( ) ;
```

این مثال key که یک متغییر صحیح است را برابر ١٥ قرار می دهد .

و برای مقایسه دو رشته از تابع equal( ) استفاده می کنند به شکل زیر :

```
String myname="mona";
String myweblog="paradise١٩٧٩١٩٧٩";
System.out.println(" answer:"+ myname.equal(myweblog));
```

با اجرای برنامه در خروجی خواهید داشت :

Answer: false
و برای نمایش دادن متغییرهای رشته ای با حروف بزرگ از تابع ( )toUppercase و نمایش با حروف کوچک از تابع ( )toLowercase در زبان جاوا استفاده می شود .

برای بررسی یک شرط در یک برنامه جاوا ، ساده ترین راه استفاده از جمله if می باشد . عبارت if همواره با یک شرط که قرار است بررسی شود و فقط در صورتی عمل مورد نظر را انجام می دهد که نتیجه شرط true باشد مورد استفاده قرار می گیرد ، مانند جمله زیر :

```
If (average<۱۲)
   System.out.println ("that's very bad");
```

باید توجه داشته باشید در زبان برنامه نویسی جاوا ،عملگر >به معنی کوچکتر از ، عملگر< به معنی بزرگتر از ، عملگر =>به معنی کوچکتر مساوی ، عملگر=< به معنی بزرگترمساوی و عملگر== به معنی مساوی می باشد .

در بسیاری از موارد لازم است که در پاسخ به یک جمله if ، بیش از یک عمل انجام بگیرد . برای انجام این کار از علائم ({،}) برای ایجاد یک جمله بلوک (جملاتی هستند که به صورت یک گروه سازماندهی شده اند .) استفاده می شود .

مثال :

```
if (playerScore>۹۹۹۹) {
        playerLives++;
         system.out.println(" Extra life!");
         level=level+۵;
  }
```

و اما مواقعی پیش می آید که بخواهید عملیاتی را در صورت درست بودن و عملیات دیگری را در صورت نادرست بودن شرط انجام دهید .
این کار با استفاده از جمله else به همراه جمله if انجام می شود.

مثال :

```
 If (grade=='A')
                        System.out.println("You got an A. Great job!");
 Else if (grade=='B')
        System.out.println("You got an B. Good work!");
Else if (grade=='C')
         System.out.println("You got an C. You'll never get into a good!");
else
         System.out.println("You got an F. You'll do well in congress!");
```

جملات if و else در مواقعی مفید واقع می شوند که دارای دو حالت ممکن می باشند ، اما گاهی ناچار به تصمیم گیری درباره چندین انتخاب ممکن هستیم. در این موارد از دستور switch استفاده می شود .
حال برای اینکه با طرز استفاده از دستور switch آشنا شوید مثال بالا را از طریق دستور switch می نویسیم :

```
Switch (grade) {
      Case 'A':
                        System.out.println("You got an A. Great job!");
         Break;
       Case 'B':
         System.out.println("You got an B. Good work!");
          Break;
       Case 'C':
          System.out.println("You got an C. You'll never get into a good!");
          Break;
       Deafalt:
          System.out.println("You got an F. You'll do well in congress!");
}
```

عملگر سه گانه (ternary) ، پیچیده ترین جمله شرطی است که شما در زبان برنامه نویسی جاوا با آن برخورد خواهید کرد.

این عملگر در مواقعی استفاده می شود که شما بخواهید براساس نتیجه یک شرط ، مقداری را تخصیص داده . این عملگر به صورت زیر به کار برده می شود :

- شرط مورد بررسی که باید در درون پرانتز قرار گیرد . مثال (age>۱۰)
- یک علامت سووال .(?)
- مقداری که در صورت درست بودن شرط ، مورد استفاده قرار می گیرد.
- علامت دو نقطه(:)
- مقداری که در صورت نادرست بودن شرط ، مورد استفاده قرار می گیرد.

مثال:

```
;Numberclass=(age۱۲:۲۰?(۱۰<
```

البته نگران نباشید اگر کار با این عملگر برایتان سخت است می توانید از همان جملاتif وelse استفاده کنید.

مثال :

```
If (age>۱۰)
  Numberclass=۱۲;
Else
  Numberclass=۲۰;
```

<u>حلقه ها</u> :

حلقه یک جمله یا یک مجموعه ای از جملات است که در برنامه تکرار خواهند شد ، برخی از حلقه ها چنان تنظیم می شوند که به دفعات معینی اجرا گردند و دفعات تکرار برخی دیگر می تواند نامعین باشد.

سه نوع جمله حلقه در جاوا وجود دارد : while,do,for . این جملات ، معمولا قابل تعویض با یکدیگر هستند .

حلقه های for :

حلقه for پیچیده ترین نوع از جملات حلقه می باشد. این حلقه اغلب در مواردی به کار می رود که بخواهیم قسمتی از برنامه به تعداد معین تکرار شود .

حلقه for از سه بخش متفاوت به صورت زیر تعریف می شود :

- بخش مقدار دهی اولیه : در این بخش متغییر با یک مقدار اولیه ، مقدار دهی می شود .
- بخش شرط : در این بخش ، از یک جمله شرطی ، مشابه آنچه در جملات if دیدیم ، استفاده می شود .
- بخش تغییر : قسمت سوم جمله ای است که مقدار متغییر را با از عملگرها تغییر می دهد .

مثال :

```
For (int number=۰; number<۱۰۰۰; number++) {
        If (number % ۵ ==۰)
            System.out.println("#:" +number);
}
```

حلقه های while :

حلقه while مانند حلقه for دارای بخشهای مختلف نمی باشند و تنها چیزی که برای آن ضروری است یک جمله شرطی است که همراه جمله while مورد استفاده قرار می گیرد .

مثال :

```
While ( key<=۲۰) {
      Key= key-۲;
      If (key==۱۰)
        Break;
      System.out.println("the key is :" +key);
}
```

توجه داشته باشید که دستور break کار خروج از حلقه را انجام می دهد و دستور continue سبب تکرار در دستورات می شود.

حلقه های do...while :

حلقه do...while از لحاظ عملکرد مشابه حلقه while می باشد ، با این تفاوت که دستورات یکبار اجرا می شوند سپس شرط حلقه بررسی می شود و در صورت درست بودن شرط بار دیگر دستورات حلقه اجرا می شود .

مثال :

```
 Do{
     Number=numer+۲;
     Sum=number+sum;
     Average=num/sum;
} while (number>=۲۰);
```

در نظر داشته باشید که حلقه ها نیز مانند سایر جملات برنامه جاوا ، می توانند در درون یکدیگر واقع شوندو همچین هرگاه حلقه ای دارای نام باشد، می توانید این نام را بعد از جمله break وcontinue به کار ببرید.

مثال:

```
Loop۱:
While (sum<۱۰۰){
        For (int count=۰; count<۱۰; count++) {
              Sum=sum+count;
        If (sum>۲۵۰)
               Break loop۱;
        }
}
```

ذخیره اطلاعات توسط آرایه :

آرایه گروهی از متغییرهای مرتبط با هم می باشند که دارای نوع یکسانی هستند و متغییرهایی هستند که تحت یک نام مشترک گروه بندی شده اند .
آرایه ها نیز همانند متغییرها با مشخص کردن نوع اطلاعاتی که درون آنها ذخیره خواهد شد و نیز نام آرایه ایجاد می شوند. تنها اختلاف عبارت است از افزوده شدن علائم کروشه ، یعنی] و [.

مثال :

```
String[] key;
```

جمله بالا یک آرایه از متغییرهایی با نوع رشته ای ایجاد می کند.
و برای مقداردهی اولیه آن یا باید همواره با نوع متغییر ، از جمله new استفاده کنید و یا اینکه مقادیر اولیه را بین علامتهای { و } درون آرایه ذخیره کنید .

مثال :

```
Int[] number=new int[۲۵۰];
```

مثال فوق، آرایه ای از اعداد صحیح به نام number با ۲۵۰ عنصر ایجاد می کند.

مثال :

```
String[] month={" farvardin" ,"ordibehesht" , "khordad" }
```

<u>استفاده از آرایه:</u>

نحوه کاربرد آرایه در برنامه ، مشابه چگونگی کاربرد سایر متغییرهاست و تنها تفاوت عبارت است از شماره عنصر که بعد از نام آرایه و درون علامت کروشه قرار می گیرد. پس از مشخص کردن شماره عنصر می توان این عنصر آرایه را در هر موقعیتی که امکان استفاده از یک متغییر وجود دارد .
و باید توجه داشت که نخستین عنصر آرایه به جای ۱ باشماره ۰ مشخص می شود.

مثال :

```
Number[۱۲] +=۱;
Key[٦٧٥٦]="max";
```

<u>آرایه های چند بعدی :</u>

برای تعریف و استفاده از آرایه های دوبعدی، باید از یک کروشه اضافی استفاده کنید ، مانند مثال زیر :

```
Boolean[][] selectedkey=new Boolean[٥٠][٥٠];
Selectedkey[٧][٤]=true;
Selectedkey[٣][٨]=true;
```

این مثال ، آرایه ای از مقادیر Boolean به نام selectedkey ایجاد می کند ، این آرایه در بعد اول دارای ۵۰ عنصر و در بعد دوم دارای ۵۰ عنصر است و لذا دارای ۲۵۰۰ عنصرمنفرد برای ذخیره سازی مقادبر می باشد.

ایجاد نخستین شیء

در برنامه نویسی شیءگرا ، هر شیء مشتمل بر دو جیز است :خصوصیات و رفتار .
خصوصیات (attributes) چیزهایی هستند که یک شی ء را توصیف و تفاوت آن با دیگر اشیا را مشخص می کنند.
رفتار (behavior) عبارتست از کاری که یک شی ء انجام می دهد .
در زبان جاوا ،اشیا با استفاده از یک کلاس (class) که حکم الگو یا قالب (template) را دارد ، ایجاد می شوند.

مثال:

```
Public class Dog {
 String name;

  Public void speak() {
     System,out.println("Hop! Hop١");
  }
}
```

این برنامه شباهت زیادی به برنامه هایی که تا کنون نوشته ایم دارد،این کلاس مانند آن برنامه ها ، با جمله class آغاز می شود، با این تفاوت که یک جمله public نیز به آن افزوده شده است . جملهpublic به این معناست که عموم برنامه ها می توانند از اشیای Dog استفاده کنند.
اگر بخواهید از شی ءDog در یک برنامه استفاده کنید، می توانید شی ء مورد نظر را مانند یک متغییر ایجاد نمایید.

مثال:

```
Dog firstDog=new Dog( );
firstDog.name="checkers";
firstDog.speak(
```

درک مفهوم ارث بری

ارث بری (inheritance) مکانیزمی است که از طریق آن ، یک شی ء رفتارها و خصوصیات اشیای مشابه دیگر را به ارث می برد.
هنگامی که شروع به ایجاد اشیا برای استفاده در برنامه های مختلف بکنید ، به زودی در می یابید که برخی اشیای جدید که مورد نیاز شما می باشند،شباهت زیادی دارند به اشیای دیگری که قبلاً ایجاد کرده اید.پس از طریق ارث بری، برنامه نویس می تواند کلاس جدیدی تعریف کرده و فقط تفاوتهای آن را با یکی از کلاسهای موجود مشخص سازد.
ارث بری یک کلاس از کلاس دیگر، توسط جمله extends انجام می گیرد. در زیر چارچوب کلی کلاس topname را که از کلاسname ارث می برد، مشاهده می کنید :

```
Class topname extends name {
        //program goes here
}
```

تبدیل اشیا و متغییرهای ساده

یکی از عملیات رایج در زبان جاوا ، تبدیل اطلاعات از شکلی به شکل دیگر است . این تبدیلات ، انواع مختلفی دارند :

- تبدیل یک شئ به شئ دیگر
- تبدیل یک متغیر ساده به متغیری از نوع دیگر
- استفاده از یک شئ برای ایجاد یک متغیر ساده
- استفاده از یک متغیر ساده برای ایجاد یک شئ

متغیر های ساده ، همان انواع داده های پایه ای هستند . این انواع ، شامل int ، char، long، double می باشند .
هنگام استفاده از یک متد یا عبارت در برنامه ، باید از انواع صحیح اطلاعات که مورد نظر آن متد یا عبارت می باشد ، استفاده کنید . برای مثال ، اگر متدی انتظار دریافت یک شئ از نوع رشته را داشته باشد . باید شئ از نوع رشته را در اختیار آن قرار دهید . اگر به متدی که تنها یک آرگومان از نوع عدد صحیح می گیرد،یک عدد اعشاری ارسال کنید ، هنگام کامپایل کردن برنامه ، با پیام خطا مواجه خواهید شد .

تبدیل اطلاعات از یک شکل به شکلی دیگر را در یک برنامه ، قالب ریزی (casting) می نامیم . قالب ریزی ، مقدار جدیدی ایجاد می کند که نوع آن با نوع متغیر یا شئ اولیه متفاوت است . هنگام قالب ریزی ، مقدار متغیر یا شئ اولیه تغیر نمی یابد .بلکه متغیر یا شئ جدیدی با قالب مورد نیاز ایجاد می شود .
جاوا هنگام بررسی نوع داده ها بسیار سختگیر است . تنها استثنا بر این قاعده رشته ها می باشند . مثلا با اینکه متدی مانند ( ) System.out.println نیاز به آرگومانی از نوع رشته دارد ، شما می توانید انواع مختلفی از اطلاعات را با عملگر + ترکیب و به این متد ارسال کنید . تا وقتی که یکی از عناصر شرکت کننده در این ترکیب ، از نوع رشته باشد ، کل ارگومان نیز به یک رشته تبدیل خواهد شد و لذا جملهٔ زیر ، به عنوان مثال ، جمله ای است معتبر :

```
Float sentence = ١٧,٥ F;
System . out .printin("My sentence has been reduced to"
        +sentence + "years.");
```

هنگام بحث دربارهٔ مفهوم قالب ریزی ، از اصطلاحات مبدأ (source) و مقصد (destination) استفاده می کنیم . مبدأ عبارت است از گونه ای از اطلاعات (متغیر یا شئ) در شکل اولیهٔ آن و مقصد ، نسخهٔ تبدیل یافتهٔ مبدأ می باشد .

قالب ریزی متغییرهای ساده

در رابطه با متغیرهای ساده ، قالب ریزی معمولا بین متغیر های عددی (مانند اعداد صحیح و اعداد اعشاری) صورت می گیرد . نوعی از متغیرها که در هیچ گونه قالب ریزی نمی تواند شرکت کند ، نوع Boolean است .
برای قالب ریزی اطلاعات ، کافی است نام قالب جدید را بین دو پرانتز و پیش از نام متغیر مورد نظر قرار دهیم . برای مثال ، جهت تبدیل یک متغیر به متغیری از نوع long ، باید عبارت (long) را قبل از نام آن قرار دهیم .

مثال:

```
Float source =٢,٠٦ ;
Int destination = ( int ) source ;
```

جملات بالا، یک مقدار float را به int تبدیل می کنند
گاهی می توان بدون استفاده از قالب ریزی، متغییری را در قالبی متفاوت به کار برید
مثلاً می توانید متغییرهای char را به جای متغییرهای int به کار برد. همچنین متغییرهای int به جای متغییرهای long و کلیه متغییرهای عددی به جای متغییرهای dubele قابل استفاده می باشند.
در این حالت ، چون مقصد در مقایسه با مبدا دارای ظرفیت بیشتری است ، معمولاً تبدیل اطلاعات بدون تغییر مقدار اولیه متغییر صورت می پذیرد .
مهمترین موارد استثنا، مربوط به هنگامی می شود که یک متغییر int با long ، به float و یا یک long به bubel قالب ریزی می گردد .
برای تبدیل اطلاعات از یک نوع متغییر به نوع کوچکتر ، باید به طور صریح از قالب ریزی استفاده کرد .

مثال :

```
Int  num=١٤;
Byte  val=(byte)num;
```

قالب ریزی اشیا

اشیا را می توان به اشیای دیگر قالب ریزی کرد ، به شرطی که شی مبدا و مقصد از طریق ارث بری با هم مرتبط باشند . به عبارت دیگر ، یکی از کلاسها باید زیر کلاس دیگری باشد.
برخی اشیا هیچ نیازی به قالب ریزی ندارند و می توان از آنها به جای هر یک از کلاسهای ارشدشان استفاده کرد .
برای مثال ، از آنجایی که کلیه اشیا در جاوا زیر کلاسهای کلاس Object هستند ، هر جا که آرگومانی از نوع Object مورد نیاز باشد ، می توان از هر شیی استفاده کرد .
در ضمن ، هر جا یکی از کلاسها یک شی مورد احتیاج باشد، می توان از خود آن شی استفاده کرد.
جهت استفاده از یک شی به جای یکی از زیرکلاسهای آن ، باید از طریق جملاتی مانند زیر ، آن را صریحاً قالب ریزی کنید:

```
Window  win = new Window();
Frame top = new (Frame)win;
```

این جملات ، سبب قالب ریزی یک شی Window به نام win ، درون یک شی Frame می شوند.

تبدیل متغیرهای ساده به اشیا و بالعکس

تبدیل یک شی به یک متغیر ساده و یا عکس آن ، از طریق قالب ریزی امکان پذیر نمی باشد ، چرا که انواع اطلاعاتی در جاوا بسیار متفاوت می باشند .
در عوض ، بسته java.long به ازای هر نوع متغیر ساده ، دارای یک کلاس متناظر می باشد.

این کلاسها عبارتند از : Short , Long ,Integer , Float , Double , Character , Byte , Boolean
همانطور که می بینید نام کلیه این کلاسها با حروف بزرگ شروع می شود و این امر یادآور این نکته است که اینها همگی شیء می باشند ، نه متغییر ساده .
با استفاده از متدهای تعریف شده در هر یک از کلاسها و ارسال مقدار یک متغییر به عنوان آرگومان ، می توانید شیء متناظر با آن متغییر ایجاد کنید .

مثال :

```
Integer suffix = new Integer (۹۰۷);
```

مثال بالا ، برای مقدار ۹۰۷ یک شیء Integer ایجاد می کند .
پس از ایجاد این شیء می توانید از آن مانند اشیای دیگر استفاده کنید.
به عنوان مثال ، برای به دست آوردن یک مقدار int از شیء suffix می توانید از جملات زیر استفاده کرد :

```
Int newSuff=suffix.netValue();
```

یکی از قالب ریزهای معمول از بک شیء به یک متغییر ، مربوط به هنگامی است که شیی از نوع رشته باید به صورت یک عدد مورد استفاده قرار گیرد . این امر ا استفاده از متد( )parselnt از کلاس Integer انجام می شود.

مثال :

```
String count="٤٦";
Int myCunt=Integer.parselnt(count);
```

ایجاد متغییرها

خصوصیات یک شیء مشخص کننده کایه متغییرهایی هستند که این شیء برای عملکرد خود بدانها نیازمند است . این متغییرها می توانند انواع داده ای ساده ای از قبیل String یا Graphics باشند.متغییرهای یک شیء را در تمامی برنامه مربوط به آن ، یعنی در تمامی متدهای آن شیء، می توان مورد استفاده قرار داد. این متغییرها بلافاصله پس از جمله class مربوط به شیء و قبل از کلیه متدهای آن تعریف می شوند.
یکی از چیزهای مورد نیاز برای یک شیء Virus ، روشی است برای دانستن اینکه آیا یک فایل قبلاً آلوده شده است یا نه . برخی از ویروسهای کامپیوتری برای این منظور، محتویات فیلدی از فایل را که زمان آخرین تغییر آن فایل را نگهداری می کند، تغییر می دهند.
برای مثال یک ویروس ممکن است زمان فوق را از ۱۳:٤۱:۲۰ به ۱۳:٤۱:٦۱ تغییر می دهد . از آنجایی که هیچ فایلی در شرایط عادی در ثانیه ٦۱ تغییر نمی یابد ، این زمان می تواند تعیین کننده آلودگی فایل باشد .
شیء Virus فیلد ثانیه از زمان تغییر فایل را به عدد ۸٦ تغییر می دهد(علت انتخاب این عدد ، اصطلاح عامیانه ای است در زبان انگلیسی که در آن ، این عدد به معنای دور انداختن و طرد کردن به کار می رود). این مقدار در یک متغییر صحیح به نام newSeconds ذخیره خواهد شد.
جملات زیر ، تعریف کلاس به نام Virus را با یک خصوصیت به نام newSeconds و دو خصوصیت دیگر آغاز می کنند :

```
Public class Virus {
        Public int newSeconds = ۸٦;
        Public String author = "Sam Snett";
        Int maxFileSize = ۳۰۰۰۰;
```

هر سه متغییر newSeconds و maxFileSize و author جزو خصوصیات این کلاس می باشند . قرار دادن جملاتی مانند public را در معرفی یک متغییر ، کنترل دسترسی (access control) می نامیم . چرا که این جملات چگونگی دسترسی سایر کلاسها به این متغییر را مشخص می سازند .
متغییر newSeconds دارای مقدار اولیه ۸٦ است و جمله ای که آن را ایجاد می کند ، با public شروع می شود . public کردن به این معنا است که هر برنامه ای که از اشیای Virus استفاده می کند، می تواند این متغییر را تغییر دهد . مثلاً اگر عدد ۹۲ برای این برنامه حائز اهمیت باشد ، می تواند مقدار newSeconds را به این عدد تغییر دهد. فرض کنید که برنامه ای یک شی Virus به نام influenza ایجاد کند. این برنامه می تواند مقدار متغییر newSeconds این شیء را توسط جمله زیر تنظیم کند :

```
influenza.newSeconds=۹۲;
```

در کلاس Virus، متغییر author نیز public است و لذا برنامه های دیگر می توانند آزادانه مقدار آن را تغییر دهند. لیکن متغییر maxFileSize تنها در درون خود این کلاس قابل استفاده است .
با public کردن یک متغییر از یک کلاس ، این کلاس کنترل خود را روی چگونگی استفاده از آن متغییر، توسط برنامه های دیگر از دست می دهد. در بسیاری از موارد ، این امر مساله خاصی ایجاد نخواهد کرد. برای مثال ، متغییر author می تواند مساوی هر نام یا لقبی که مو لف ویروس را مشخص کند ، قرار گیرد.
محدود کردن دسترسی به یک متغییر باعث جلوگیری از خطاهایی می شود که ممکن است در اثر تخصیص مقادیر نادرست به آن متغییر رخ دهد. در مورد کلاس Virus، اگر متغییر newSeconds برابر عدد ٦۰ یا عددی کوچکتر از آن قرار داده شود ،

دیگر این کلاس به طور صحیح قادر به تعیین آلودگی فایلها نخواهد بود ، چرا که برخی از فایلها که بدون دخالت ویروس ، در همان ثانیه تغییر یافته اند ، از نظر این کلاس به عنوان فایلهای آلوده شناسایی خواهند شد . اگر کلاس Virus بخواهد از این مساله اجتناب کند، باید دو عمل زیر انجام پذیرد :

- متغییر newSeconds را از حالت public به حالت protected یا private تبدیل کنید. این جملات دسترسی به متغییر را محدودتر می سازند.
- رفتارهای لازم را برای تغییر مقدار این متغییر و گزارش کردن مقدار آن به دیگر برنامه ها به ابن کلاس بیفزایید.

یک متغییر protected فقط توسط خود کلاس ، کلیه زیر کلاسهای آن و کلیه کلاسهای موجود در همان بسته (package) قابل استفاده می باشد، بسته گروهی از کلاسهای مرتبط است که هدف مشترک را تعقیب می کنند.
یک متغییر private از نظر دسترسی، حتی از متغییر protected نیز محدودتر است و فقط توسط خود کلاس قابل استفاده می باشد . بهتر است که متغییر ها را عموماً به صورت protected یا private تعریف کنید . مگر در مواردی که بدانید تغییر یک متغییر توسط برنامه های دیگر، خللی به عملکرد کلاس مربوطه وارد نخواهد کرد .
جمله زیر newSecond را به یک متغییر private تبدیل می کند:

```
private int newSeconds=٨٦;
```

نوع دیگری از کنترل دسترسی نیز وجود دارد که عبارت است از عدم استفاده از هیچ یک از جملات public,private,protected هنگام ایجاد متغییر .
وقتی یک متغییر بدون استفاده از جملات کنترل دسترسی را اغلب دسترسی پیش فرض (default access) یا دسترسی بسته (package access) می نامند ، هر چند جمله خاصی برای آن در نظر گرفته نشده است .

مرکز فن آوری اطلاعات