آموزش زبان برنامه نویسی java

ایجاد متغیرهای کلاس

هر شیء به محض ایجاد ، شامل یک کپی مخصوص به خود ، از کلیه متغییرهایی است که بخشی از کلاس مربوط به آن شیء می باشند. به عنوان مثال ، هر شیء ایجاد شده از کلاس Virus شامل نسخهای مربوط به خود ، از متغییرهای newSeconds,maxFileSize,author می باشد . اگر یکی از این متغییرها را در یک شی ء تغییر دهید، تاثیری روی همان متغییر از شیء Virus دیگر نخواهد داشت .

گاهی یک خصوصیت بیش از آنکه مربوط به یک شیء خاص باشد، مربوط به خود کلاس است.

متغییرهایی که تاکنون برای اشیاء ایجاد می کردیم متغییرهای شیء (object variables) نامیده می شوند، چرا که وابسته به یک شیء خاص می باشند . متغییرهای کلاس(class variables) ، متغییرهایی هستند که وابسته اند به یک کلاس از اشیا و نه به یک شیء خاص .

هر دو نوع این متغیرها به یک شکل ایجاد می شوند و مورد استفاده قرار می گیرند ، با این تفاوت که متغییرهای کلاس با استفاده از جمله static ایجاد می شوند. مثال زیر ، یک متغیر کلاس را برای کلاس Virus ایجاد می کند :

```
Static int virusCount=۰;
```

نحوه تغییر مقدار یک متغییر کلاس ، تفاوتی با چگونگی تغییر یک متغیر شیء ندارد. اگر یک شیء Virus به نام tuberculosis داشته باشید ، می توانید کلاس virusCount را با استفاده از جمله زیر تغییر دهید:

```
Tuberculosis.virusCount++;
```

ولی از آنجایی که متغیرهای کلاس با کل کلاس ارتباط دارند و نه به یک شیء خاص، می توانید به جای نام شیء ، از نام کلاس استفاده کنید :

```
Virus.virusCount++;
```

هر دو جمله فوق عمل یکسانی انجام می دهند، لیکن استفاده از نام کلاس ، هنگام کار با متغیرهای کلاس بهتر است ، چرا که بلافاصله مشخص می کند که متغیر مورد نظر ، یک شیء نیست ، بلکه یک متغیر کلاس است . اگر هنگام کار با متغیرهای کلاس ، همواره از اسامی اشیا استفاده کنید، فهمیدن اینکه آیا یک متغیر کلاس است یا نه ، بدون بررسی دقیق کد مربوط به کلاس ، امکان پذیر نخواهد بود .

ایجاد رفتار با استفاده از متدها

خصوصیات هر کلاس، نگاهدارنده اطلاعات مربوط به آن کلاس می باشند، لیکن درعمل ، قادر به انجام هیچ کاری نیستند . برای اینکه یک کلاس بتواند کارهایی را که برای انجام آنها به وجود آمده ، انجام دهد ،باید رفتار لازم را برایش ایجاد کنید .

رفتار عبارت است از کلیه بخشهایی از یک کلاس که وظیفه معینی را انجام می دهند، هر یک از این بخشها را یک متد(method) می نامیم.

متدها نیز مانند متغییرها، در ارتباط با یک شیء یا یک کلاس به کار می روند. به نام متد، همراه با یک نقطه به دنبال نام شیء یا کلاس قرار می گیرد. مانند screen.drowString() یا Integer.parseInt() .

معرفی یک متد

جمله ای که متدها توسط آن ایجاد می شوند، شبیه جمله ای است که یک کلاس را آغاز می کند. هر دو بین پرانتزهایی که پس از نامشان قرار می گیرد، قادر به دریافت آرگومان بوده ، هر دو از علائم { و } برای مشخص کردن شروع و پایان خود استفاده می کنند .

تفاوت اصلی آن است که متدها قادر به بازگرداندن یک مقدار می باشند . این مقدار می تواند یکی از انواع ساده، مانند اعداد صحیح یا مقادیر Boolean و یا یک کلاس می باشد. برای تعریف متدی که مقداری را برنمی گرداند، از جمله void استفاده می کنیم .

نمونه زیر ، مثالی است از متدی که کلاس Virus می تواند برای آلوده کردن فایلها از آن استفاده کند:

```
Public Boolean intectFile(string filename) {
   Boolean success = false;
   // file-infecting statements would be here
 return success;
}
```

متد infectFile( ) برای افزودن ویروس به یک فایل به کار می رود . این متد فقط یک آرگومان می گیرد که متغیری است از نوع رشته به نام filename و مشخص کننده فایلی است که باید مورد حمله قرار گیرد .

متدهای مشابه با آرگومانهای مختلف

با ارسال آرگومان به یک متد ، می توان عملکرد آن را تحت قرار داد.متدهای مختلف در یک کلاس می توانند دارای اسامی مختلف باشند، لیکن امکان اینکه چند متد دارای نام یکسان باشند نیز وجود دارد ، به شرط اینکه آرگومانهای آنها متفاوت باشد .
دو متد می توانند دارای اسامی یکسان باشند، به این شرط که تعداد آرگومانهای آنها مختلف بوده و یا نوع برخی از آرگومانهای آنها فرق داشته باشد.
برای مثال، کلاس Virus می تواند دارای دو متد (tauntUser (به معنای طعنه زدن به کاربر) باشد که یکی از آنها آرگومانی نگرفته باشد و یک طعنه عمومی را نمایش دهد و دیگری متن طعنه را به صورت یک آرگومان از نوع رشته دریافت کند. مانند زیر:

```
Void tauntUser() {
   System.out.println("the problem is not with your set,but"+"with yourselves.");
}
void tauntUser(string taunt) {
    system.out.println(taunt);
}
```

متدهای سازنده

برای ایجاد یک شیء در برنامه، از جمله new استفاده می شود، مانند زیر:

```
Virus typhoid=new virus( );
```

این جمله، یک شیء virus جدید به نام typhoid ایجاد می کند. وقتی از جمله newاستفاده می کنید، متد خاصی از کلاس آن شیء فراخوانی می شود. این متد سازنده (constructor) نامیده می شود، چرا که عملیات مربوط به ایجاد شیء را انجام می دهد .هدف اصلی از وجود سازنده، مقداردهی کلیه متغیرهایی است که برای کارکرد صحیح شیء ، مورد نیاز می باشند .
متدهای سازنده ، مانند سایر متدها تعریف می شوند، با این تفاوت که قادر به بازگرداندن مقدار نمی باشند. در زیر دو متد سازنده برای کلاس virus را مشاهده می کنید:

```
Public virus() {
   author="Ignoto";
   maxFileSize=۳۰۰۰۰;
}
public virus(string name,int size) {
   author=name;
   maxFileSize=size;
}
```

با استفاده از آرگومانهای مختلف ، می توان برای یک کلاس، چندین سازنده تعریف کرد. در مثال بالا ، سازنده اول وقتی فراخوانی می شود که یک جمله new مانند زیر به کار رفته باشد:

```
Virus mumps=new virus();
```

سازنده دیگر در صورتی فراخوانده خواهد شد که در جمله new از یک رشته و یک عدد صحیح به عنوان آرگومان استفاده شده باشد، مانند :

```
Virus rubella=new virus("April Mayhem",٦۰۰۰۰);
```

اگر برای یک کلاس هیچ متد سازنده ای تعریف نکنید، این کلاس ، سازنده ای از کلاس ارشد خود که هیچ آرگومانی نمی گیرد، به ارث خواهد برد. در ضمن ، بسته به کلاس ارشد به کار رفته، کلاس فوق ممکن است متدهای سازنده دیگری را نیز به ارث ببرد.
هر کلاسی باید دارای یک متد سازنده باشد که تعداد و نوع آرگومانهای آن با آنچه در جمله new مربوط به ایجاد اشیای آن کلاس به کار رفته ، یکسان باشد.
مثلاً برای ایجاد اشیای از نوع virus که دارای دو سازنده به صورت virus() و virus(string name,int size) می باشد، تنها می توان از دو نوع جمله new استفاده کرد : یکی بدون آرگومان و دیگری دارای دو آرگومان که اولی از نوع رشته و دومی از نوع عدد صحیح می باشد .

متدهای کلاس

متدهای کلاس نیز مانند متغیرهای کلاس، روشی هستند برای ایجاد کارکردهایی که با کل یک کلاس در ارتباطند، نه با شیء خاص از آن کلاس.
اگر متدی یک شیء خاص از کلاس را تحت تاثیر قرار ندهد، باید به صورت متد کلاس تعریف شود .مانند متد parseInt() از کلاس Integer . این متد برای تبدیل یک رشته به متغییری از نوع int به کار می رود مانند:

```
Int time=integer.parsInt(timetext);
```

برای تبدیل یک متد به متد کلاس، قبل از نام آن از static استفاده کنید ، مانند زیر :

```
Static void showVirusCount() {
    System.out.println("there are"+virusCount+"viruses.");
}
```

متد ( )showVirusCount یک متد کلاس است که مقدار این متغییر را نمایش می دهد و باید آن را توسط جمله ای به صورت زیر فراخوانی کنید:

```
Virus. showVirusCount( );
```

حوزه دید متغیرها درون یک متد

وقتی یک متغیر یا شیء را درون یکی از متدهای یک کلاس ایجاد می کنید، فقط درون همان متد می توانید از آن استفاده کنید. علت این امر، مفهومی است به نام حوزه دید متغیر (variable scope) .

حوزه دید ، بخشی از برنامه است که یک متغیر در آن وجود دارد. با خروج از این بخش ، دیگر نمی توان از آن متغییر استفاده کرد. محدوده یک متغیر در برنامه با علائم { و } تعریف می شود. متغیری که بین این دو علامت ایجاد شده ، خارج از آنها نمی تواند مورد استفاده قرار گیرد .برای مثال جملات زیر را در نظر بگیرید:

```
If (numfile<۱) {
   String warning="no files remaining.";
}
system.out.println(warning);
```

این مثال عملکرد صحیحی نخواهد داشت، چرا که متغیر warning درون آکلادهای مربوط به جمله بلوک if تعریف شده است که حوزه دید آن را مشخص می سازند. این متغیر خارج از آکلادها وجود ندارد و لذا متد ( )system.out.println نمی تواند از آن به عنوان یک آرگومان استفاده کند.

قرار دادن یک کلاس درون کلاس دیگر

اگر چه هر برنامه جاوا معمولاً به عنوان یک کلاس در نظر گرفته می شود، در بسیاری از موارد ، یک برنامه ممکن است برای انجام کار خود به بیش از یک کلاس احتیاج داشته باشد. برنامه های جند کلاسه (multiclass) از یک کلاس اصلی و چندین کلاس کمکی (helper class) تشکیل می یابند.

به عنوان مثال می توان یک اپلت جاوا را در نظر گرفت که در بخشی از رابط کاربر گرافیکی خود، عنوان متحرکی را نمایش می دهد.

برای تعریف کلاسهای کمکی در برنامه ای که به چندین کلاس تقسیم شده است، دو روش وجود دارد ، روش اول آن است که هر کلاس به صورا جداگانه تعریف می شود، مانند:

```
Public class wreakHavoc {
   String author="Ignoto";

    Public void infectfile( ) {
          virusCode vic=new virusCode(۱۰۲٤);
    }
}
class virusCode {
    int vsize;

    virusCode(int size) {
         vsize=size;
     }
}
```

در این مثال virusCode به عنوان یک کلاس کمکی برایwreakHavoc به کار رفته است . کلاسهای کمکی اغلب در همان فایل مبدا java. که کلاس اصلی در آن واقع است ، تعریف می شوند. با کامپایل کردن این فایل مبدا، چندین فایل کلاس تولید خواهد شد .

روش دیگر برای تعریف کلاسهای کمکی آن است که آنها را درون کلاس اصلی قرار دهیم . در این حالت، کلاس کمکی یک کلاس درونی (inner class) نامیده می شود.

کلاس درونی ، بین آکلاد باز و بسته یک کلاس دیگر تعریف می شود، مانند:

```
Public class wreakMoreHavoc {
   String author="Ignoto";

   Public void infectFile( ) {
      virusCode vic=new virusCode(۱۰۲٤);
   }
  class virusCode {
       int vsize;
```

```
virusCode(int size) {
      vsize=size;
       }
   }
}
```

کلاسهای درونی را می توان مشابه سایر کلاسهای کمکی مورد استفاده قرار داد. تنها اختلاف آن است که کلاسهای درونی پس از کامپایل شدن، نام جدیدی دریافت می کنند که شامل نام کلاس اصلی آنها نیز می باشد

استفاده از کلمه کلیدی this

با توجه به اینکه درون یک کلاس می توان به متغییرها ومتدهای همان کلاس و کلاسهای دیگر رجوع کرد، در برخی شرایط متغیری که به آن رجوع می شود، ممکن است باعث ابهام وسردرگمی شود. یک روش برای کاستن از این گونه ابهامات ، استفاده از جمله this می باشد. با استفاده از این جمله، یک برنامه می تواند به شیء مربوط به خود رجوع کند.
هنگام استفاده از متدها یا متغیرهای یک شیء شما نام آن شیء را همراه بایک علامت نقطه، پیش از نام متد یا متغیر قرار می دهید.

مثال:

```
Virus chichenpox=new Virus( );
Chichenpox.name=”LoveHandles”;
Chichenpox.setSeconds(۷۰);
```

جملات فوق یک شیء جدید Virus به نام chickenpox ایجاد کرده ، متغیر name از chickenpox را مقداردهی می کنند و سپس متد setSeconds( ) از chickenpox را فرامی خوانند .
گاهی در یک برنامه لازم است که به شیء جاری یا به عبارت دیگر، به شیء مربوط به خود برنامه رجوع کنید.
برای مثال ، درون کلاس Virus ممکن است متدی داشته باشیم که دارای متغیری به نام author باشد :

```
Void public checkAuthor( ) {
    String athor=null;
}
```

متغیر author فقط در حوزه دید متد checkAuthor( ) وجود دارد و با متغیر شیء author متفاوت است . اگر بخواهید درون این متد به متغیرauthor از شیء جاری رجوع کنید، ناچار به استفاده از جمله this خواهید بود :

```
system.out.println(this.author);
```

با استفاده از this مشخص می کنید که به کدام متغیر یا متد رجوع کرده اید هر جایی از یک کلاس که امکان رجوع به یک شیء توسط نام آن وجود داشته باشد، می توانید از this استفاده کنید . می توانید از جمله به شکل زیر استفاده کنید :

```
verifyData(this);
```

به ارث بردن رفتارها و خصوصیات

رفتارها و خصوصیات هر کلاس ، ترکیبی هستند از دو چیز : رفتارها و خصوصیات خود آن کلاس و رفتارها و خصوصیاتی که از کلاسهای ارشد خود به ارث می برد.
برخی از رفتارها و خصوصیات Applet از این قرارند:

- متد equals( ) معین می کند که آیا شیء Applet دارای مقداری برابر با شیء دیگر هست یا نه .
- متد setBackground( ) رنگ زمینه پنجره اپلت را تنظیم می کند .
- متد add( ) برای افزودن عناصر رابط کاربر از قبیل دکمه ها و فیلدهای متنی به اپلت به کار می رود.
- متد showStatus( ) سطری از متن را درون نوار وضعیت مرورگر وب نمایش می دهد.

کلاس Applet می تواند از کلیه این متدها استفاده کند، گرچه تنها متدی که از کلاسهای دیگر به ارث نبرده showStatus( ) است . متد equals( ) در Object ، متد SetBackground( ) در Component( ) و متد add( ) در Container تعریف شده است .

دوباره نویسی متدها

برخی از متدهای تعریف شده در کلاس Applet، در یکی از کلاسهای ارشد آن نیز تعریف شده اند. مثلاً متد resize( ) هم در کلاس Applet وهم در کلاس Component تعریف شده است . این متد برای تغییر اندازه یک عنصر روی یک رابط کاربر گرافیکی به کار می رود. اگر متدی هم در زیرکلاس و هم در کلاس ارشد آن تعریف شده باشد، متد تعریف شده در زیرکلاس، مورد استفاده قرار می گیرد. این امر ، زیرکلاس را قادر می سازد که برخی از رفتارها و خصوصیات کلاسهای ارشد خود را تغییر دهد، جایگزین کند و یا به کلی از بین ببرد.

ایجاد یک متد جدید در یک زیر کلاس، به منظور تغییر رفتار به ارث برده شده از یک کلاس ارشد را دوباره نویسی (overriding) متد می نامند. در مواردی که رفتار به ارث برده شده، ممکن است منجر به نتایج نامطلوب شود، باید متد مربوطه را دوباره نویسی کنید.

برقراری ارث بری

برای قرار دادن یک کلاس به عنوان زیرکلاس کلاسی دیگر، از جمله extends استفاده می کنیم. مانند :

```
Class Animatedlogo extends java.applet.JApplet {
   // behavior and attributes go here
}
```

در اینجا جمله extends ،کلاس AnimatedLogo را به عنوان زیر کلاسJapplet قرار می دهد. کلیه اپلتهای جاوا ۲ باید زیرکلاسی از Japplet باشند، چرا که برای اجرا روی وب جهانی به کارکرد این کلاس نیازمندند.
برای دوباره نویسی یک متد، باید آن را به همان شکلی که در کلاس ارشد مربوطه آغاز می گردد، آغاز کنید . یک متد public بایدpublic باقی بماند، مقدار بازگردانده شده توسط متد باید یکسان باشد و تعداد و نوع آرگومانهای ارسالی به متد نیز نباید تغییر یابد.

استفاده از this و super در یک زیر کلاس

دو کلمه کلیدی بسیار مفید جهت استفاده در زیرکلاسها عبارتند از this و super . کلمه کلیدی this برای رجوع به شی ء جاری به کار می رود .هنگامی که کلاسی ایجاد می کنید و ناچار به رجوع به شی ء خاصی از این کلاس هستید می توانید از this استفاده کنید ، مانند جمله زیر :

```
This.title="Cagney";
```

این جمله متغییر title از این شی ء را برابر Cagney قرار می دهد.

کلمه کلیدی super نیز کاربرد مشابهی دارد و به کلاس ارشد مستقیم آن شی ء رجوع می نماید. Super را می توان به چندین روش مورد استفاده قرار داد :

- برای رجوع به یک متد سازنده از کلاس ارشد ، مانند super("Adam",۱۲) .
- برای رجوع به متغییری از کلاس ارشد، مانند super.Hawaii=۵۰ .
- برای رجوع به یک متد از کلاس ارشد ، مانند super.dragNet() .

یکی از کاربردهای کلمه کلیدی super در متد سازنده ، مربوط به یک زیر کلاس می باشد . از آنجایی که هر زیرکلاس کلیه رفتارها و خصوصیات کلاس ارشد خود را به ارث می برد ، شما باید هر متد سازنده از زیر کلاس را با یک متد سازنده از کلاس ارشد آن همراه کنید . در غیر اینصورت برخی از رفتارها و خصوصیات به درستی تنظیم نخواهد شد و زیر کلاس عملکرد صحیحی نخواهد داشت .
جهت حصول اطمینان از فراخوانی متد سازنده کلاس ارشد ، باید این متد را در اولین جمله از سازنه زیر کلاس فرا خوانی کنید و برای این منظور باید از کلمه کلیدی super استفاده کنید ، مانند جملات زیر :

```
Publiv ReadFile(String name,int length){
        Super(name,length);
}
```

مثال فوق ، متد سازنده ای است از یک زیر کلاس که با استفاده از super(name,lenth) متد سازنده متناظر از ارشد خود را فرا می خواند.

کار کردن با اشیای موجود

در آغاز معرفی جاوا ، به اشتراک گذاشتن اشیاء موضوعی غیر رسمی بود . برنامه نویسها اشیای خود را به گونه ای می نوشتند که حداکثر استقلال را داشته باشد و با استفاده از متغییرهای (private) و متدهای عمومی (public) جهت خواندن و نوشتن آن متغییرها ، اشیای خود را در برابر استفاده نادرست محافظت می کردند .
اگراستانداردی برای توسعه اشیای قابل کاربرد مجدد وجود داشته باشد به اشتراک گذاشتن اشیا قوت بیشتری خواهد گرفت .
مزایای وجود یک استاندارد عبارت است از :

- نیاز چندانی به مستند سازی چگونگی کارکرد اشیا وجود نخواهد داشت .
- با استفاده از استاندارد امکان طراحی ابزارهای توسعه (development tools) فراهم می شود که استفاده از این اشیا را آسانتر می سازد .
- اشیایی که از این استاندارد پیروی کرده اند ، بدون نیاز به برنامه نویسی خاص می توانند با یکدیگر تعامل و همکاری کنند

استاندارد مربوط به توسعه اشیای قابل کاربرد مجدد در جاوا ، JavaBeans نام دارد و هر شی ء منفرد از این مجموعه یک Bean نامیده می شود .

JavaBeans

JavaBeans کلاسهای جاوایی هستند که به منظور استفاده مجدد طراحی شده اند . این کلاسها که در بسیاری از زبانهای برنامه نویسی ، عناصر نرم افزاری (software components) نامیده می شوند ، تحت مجموعه ای از استاندارد از قواعد ، توسعه می یابند .
برای توسعه beans ، علاوه بر کیت توسعه جاوا (JDK) ، نیازمند ابزار برنامه نویسی دیگری نیز هستید . شرکت Sun Microsystems ابزار خود را تحت عنوان Beans Development Kit (کیت توسعه Beans) در سایت
http://java.sun.com/beans/software
ارائه می دهد .
کیت توسعه Beans که BDK نیز نامیده می شود ، شامل یک ابزار بصری (visual) به نام BeanBox است که برای افزودن Beans به برنامه های جاوا و فراهم کردن زمینه همکاری اشیای Bean با همدیگر به کار می رود .
برنامه نویسی Bean یکی از جنبه های تخصصی جاواست و یادگیری کامل آن تنها پس از تسلط بر اصول پایه ای زبان میسر است .
اما برنامه نویسان مبتدی با استفاده از Bean های موجود می توانند به نتایج قابل توجهی دست یابند پس خالی از فایده نخواهد بود اگر همزمان با فراگیری برنامه نویسی با JavaBean نیز آشنا شوید .

یادگیری چگونگی عملکرد اپلت ها

اپاتها برنامه هایی هستند که برای اجرا در صفحات وب جهانی ، طراحی شده اند. اگر یک صفحه وب حاوی یک اپلت جاوا باشد ، با مشاهده این صفحه ، اپلت فوق روی کامپیوتر کاربر download شده ، سپس اجرا می گردد.
برنامه نویسی اپلتها با جاوا و با ایجاد برنامه های کاربردی توسط آن ، بسیار متفاوت است .
از آنجا که اپلتها برای هر بار اجرا باید از یک صفحه وب download شوند ، لذا به سبب کاهش زمان download معمولا از برنامه های کاربردی کوچکترند .
همچنین ، اپلتها به علت اجرا شدن در کامپیوتر کاربر ، ناچارند محدودیتهای امنیتی متعددی را برای پیشگیری از اجرای برنامه های مزاحم یا مخرب رعایت کنند.

متدهای استاندارد اپلت

اولین کار در ایجاد اپلت آن است که آن را در یک زیر کلاس از jApplet در نظر بگیریم که بخشی از بسته com.sun.jsva.swing می باشد . این بسته که Swing نامیده می شود ، مجموعه از کلاسها را برای ایجاد یک رابط کاربر گرافیکی ، گرافیک کامپیوتری و دیگر عناصر بصری برای برنامه های کامپیوتری ، ارائه می هد.
یک اپلت به عنوان یک پنجره بصری ، داخل یک صفحه وب عمل می کند بنابراین JApplet بخشی است از Swing همراه با دکمه های قابل کلیک ، نوارهای مرورگر و دیگر اجزای یک رابط کاربر .
JApplet یک زیر کلاس از Applet می باشد که کلاسی است از بسته java.applet
قرار گرفتن در این سلسله مراتب اپلتهای نوشته شده را قادر می سازد که از تمامی رفتارها و خصوصیاتی که برای اجرا در صفحه جهانی وب مورد نیاز است ، استفاده کنند . اپلتهای شما ، بدون اینکه جمله ای به آنها بیفزایید . توانایی تعامل با مرورگرهای وب را خواهند داشت ، خود را load و unload خواهند کرد . در پاسخ به تغییرات در پنجره مرورگر ، پنجره های خود را دوباره ترسیم کرده ، وظایف ضروری دیگری را انجام خواهند داد.

ادامه متد های استاندارد اپلت

در برنامه های کاربردی، اجرای برنامه با اولین عبارت داخل بلوک ()main شروع می شود و با آخرین آکلاد بسته (}) که بلوک را می بندد ، پایان می پذیرد . در اپلت جاوا متدی تحت عنوان ()main موجود نیست و لذا مکان مشخصی برای شروع برنامه وجود ندارد .
در عوض ، اپلت دارای گروهی از متدهای استاندارد است که هنگام اجرای آن ، در پاسخ به رویدادهای مشخصی ، وارد عمل می شوند .
موارد زیر رویدادهایی هستند که می توانند یکی از متدهای اپلت را وارد عمل کنند :

- برنامه برای اولین بار Load می شود .
- رویدادهایی که مستلزم نمایش دوباره پنجره اپلت می باشند .
- برنامه توسط مرورگر متوقف می شود .
- برنامه پس از توقف ، دوباره کار خود را آغاز می کند .
- اجرای برنامه پایان یافته ، برنامه Unload می شود .

مثال زیر چارچوب یک اپلت را نشان می دهد :

```
Public class Skeleton extendes com.sun.java.swing.JApplet {
   //program will go here
}
```

برخلاف برنامه های کاربردی ، فایلهای کلاس اپلت برای اینکه بتوانند کار خود را انجام دهند .باید public باشند .
کلاس اپلت شما وارث تمام متدهایی است که هنگام نیاز به طور خودکار وارد عمل می شوند : ()init,()paint,()start,()stop,()destroy . لیکن هیچکدام از این متدها کاری انجام نمی دهند . اگر می خواهید که در اپلت شما اتفاق خاصی بیفتد ، باید این متدها را با نسخه های جدیدی در برنامه اپلت خود ، جایگزین کنید . متدهایی که اغلب جایگزین خواهید کرد ()pint و()init می باشند.

متد ( )paint

متد ()paint ، بخشی از تمامی اپلتهای شما خواهد بود ، چرا که بدون آن توانایی نمایش هیچ چیز را نخواهید داشت.
هر گاه چیزی در پنجره اپلت نیاز به نمایش و یا نمایش دوباره داشته باشد ، متد ( )paint انجام وظیفه را به عهده می گیرد.
متد ()paint یک آرگومان می گیرد .مورد زیر مثالی است از یک متد ()paint ساده :

```
public class paint(Graphics screen) {
  //display statements go here
}
```

در اپلتهای جاوا ۲ ، اولین جمله از متد ( )paint باید متد ( )paint را از کلاس ارشد خود صدا بزند . این کار تضمین می کند که پنجره اپلت به درستی ، به هنگام سازی شود . جمله زیر این وظیفه را انجام می دهد :

```
Super.paint(screen);
```

این مثال یک شی گرافیکی به نام screen را به متد ()paint از کلاس ارشد می فرستد . این شی باید همان شیئی باشد که به متد ()paintاپلت فرستاده شده است .

متد ( )init

متد()init تنها یکبار و در ابتدای اجرای اپلت ، وارد عمل می شود . در نتیجه ، مکانی مناسب برای تنظیم مقادیر اشیاء و متغییرهایی است که اپلت ، برای اجرای موفقیت آمیز به آنها نیاز دارد . این متد ، همچنین مکان مناسبی برای تنظیم فونتها ، رنگها و رنگ پس زمینه صفحه می باشد .
البته متغییرها و اشیاء نباید درون متد()init ایجاد شوند . چرا که در این صورت تنها در محدوده این متد وجود خواهند داشت .
برای مثال ، اگر یک متغییر از نوع عدد صحیح ، به نام display داخل متد ()init ایجاد کنید و بخواهید از آن در متد ()paint استفاده برید هنگام کامپایل کردن برنامه یک خطا دریافت خواهید کرد .
تمامی متغییرهایی را که می خواهید در یک کلاس ، به عنوان متغییرهای شی به کار برید ، درست بعد از عبارت class و قبل از هر متدی ، ایجاد کنید

متدهای ()start و ()stop

در هر نقطه ای که اجرای برنامه اپلت شروع شود ، متد ()start وارد عمل خواهد شد . هنگام شروع برنامه ، متد()init ، به دنبال متد () start فراخوانی می شود . پس از آن ، در بسیاری موارد هیچ عاملی موجب راه اندازی مجدد متد () start، باید اجرای اپلت در مقطع یا مقاطع خاصی متوقف گردد .
هنگامی که اجرای اپلت متوقف می شود ، متد () stop فراخوانی می گردد . ممکن است این اتفاق هنگامی رخ دهد که کاربر ، صفحهٔ وب شامل اپلت را ترک کرده ، وارد صفحهٔ دیگری بشود . همچنین فراخوانی مستقیم متد () stop نیز می تواند سبب وقوع این رویداد شود .
در برنامه هایی که ضمن آموختن زبان جاوا خواهید نوشت ، ()start و() stop بیشتر در متحرک سازی مورد استفاده قرار خواهند گرفت .

متد () destroy

متد () destroy نقطهٔ مقابل متد()init می باشد. این متد درست قبل از اتمام اجرا و بسته شدن اپلت به کار می افتد این متد در مواردی نادر که بعضی چیزها در طی یک برنامه تغییر می کنند و باید به حالت اولیه خود بازگردانده شوند ، به کار می رود . این متد نیز بیشتر در متحرک سازی مورد استفاده قرار خواهد گرفت ، تا دیگر انواع برنامه ها .

قرار دادن اپلت در صفحهٔ وب

اپلتها به همان روشی در صفحات وب قرار می گیرند که هر چیز غیر معمول دیگری در این صفحات قرار می گیرد . فرمانهای HTML برای توصیف اپلت به کار می روند و مرورگر وب آن را همراه سایر بخشهای صفحه Load می کند .

یکی از راههای قرار دادن اپلتها در صفحات وب ، استفاده از یک برچسب < APPLET> و چندین خصوصیت می باشد . مورد زیر مثالی است از HTML مورد نیاز برای قرار دادن یک اپلت را در یک صفحه وب :

```
<APPLET  CODE="Exampel.class" CODEBASE="javadir" HEIGHT=۳۰۰  WIDTH=٤۰۰>
Sory ...this requires a java-enabled browser.
</APPLET>
```

خصوصیت CODE نام فایل کلاس اپلت را تعیین می کند. اگر یک اپلت مشتمل بر چندین فایل کلاس باشد ، CODE باید به فایل کلاس اصلی ، که زیر کلاسی از کلاس JApplet می باشد ، اشاره کند.
اگر هیچ خصوصیتی ار نوعCODEBASE وجود نداشته باشد ، تمام فایلهای وابسته به اپلت ، باید در همان پوشه ای قرار گیرند که صفحه وب در آن واقع است . CODEBASE باید به پوشه یا زیر پوشه ای اشاره کند که بتوان اپلت و فایلهای وابسته را در آن یافت .

خصوصیات HEIGHT وWIDITH اندازه دقیق پنجره اپلت را در صفحه وب معین می کنند .
مابین برچسبهای <APPLET> و</APPLET> می توانید نوعی جایگزین ، برای کاربران وبی که نرم افزار مرورگرشان توانایی اجرای برنامه های جاوا را ندارد ، فراهم کنید.
خصوصیت دیگری که می توانید با اپلت به کار برید ، ALIGN می باشد . این خصوصیت معین می کند. که اپلت نسبت به آنچه در اطرافش قرار دارد ، از قبیل متن و گرافیک ، چگونه نمایش داده خواهد شد. مقادیر مربوطه شامل "ALIGN="left و"ALIGN="right و غیره می باشد .

استفاده از متد ()drawstring

برای نمایش یک متن در پنجره اپلت ، از متد ()drawstring از کلاس Graphics۲D و یا متد مشابه ، در کلاس Graphics استفاده می شود .
عملکرد ()drawstring مشابه متد ()System.out.println می باشد که اطلاعات را در خروجی استاندارد سیستم نمایش می دهد .
البته پیش از آنکه بتوانید متد ()drawstring از کلاس Graphics۲D را بکار ببرید ، باید یک شیء Graphics۲D داشته باشید که نمایشگر پنجره اپلت باشد .
متد ()paint در تمام اپلتها شامل یک شیء Graphics ، به عنوان تنها آرگومان خود ، می باشد . این شیءنمایانگر پنجره اپلت بوده و لذا می تواند برای ایجاد یک شیء Graphics۲D که آن نیز مشخص کننده این پنجره اپلت بوده و لذا می تواند برای ایجاد یک شیء Graphics۲D که آن نیز مشخص کننده این پنجره باشد ، به کار رود .
و حتما می دانید که از قالب ریزی (casting) برای تبدیل یک شیء Graphics به یک شیء Graphics۲D استفاده می کنیم .Graphics۲D یک زیر کلاس از Graphics می باشد و این امر استفاده از جمله زیر را ممکن می سازد :

```
Graphics۲D Screen۲D=(Graphics۲D)Screen;
```

این جمله یک شیء Graphics به نام Screen را به یک شیء Graphics۲D به نام Screen۲D تبدیل می کند .
پس از ایجاد یک شیء Graphics۲D ، می توانید متد drawstring را برای نمایش متن در ناحیه ای از صفحه که به این شیء مربوط می باشد ، فراخوانی کنید .
سه آرگومان زیر به متد ()drawstring فرستاده می شوند :

- متنی برای نمایش ، که می تواند از چندین رشته و متغییر که با عملگر + به هم پیوسته اند ، تشکیل شده باشد .
- موقعیت x (در سیستم مختصات دو بعدی(x,y)) محلی که رشته در آنجا نمایش داده خواهد شد .
- موقعیت y محلی که رشته در آنجا نمایش داده خواهد شد .

استفاده از برچسب جدید<OBJECT>

هنگامی که جاوا معرفی شد ، اپلتها تنها برنامه بودند که می توانستند در یک صفحه وب اجرا شوند ، به همین دلیل، بر چسب <APPLET> به طور خاص برای کار با این برنامه ها به HTML افزوده شد .
امروزه ، انواع مختلفی از برنامه های تعاملی روی وب به وجود آمده اند ، مانند پیتون(Ptyhon) ، کنترلهای ActiveX و اپلتهایNetRexx (bytecode جاوای قابل اجرا که به جای جاوا با زبان NetRexx ایجاد شده است).
برای رسیدگی به این قبیل برنامه ها و برنامه هایی که در آینده معرفی خواهند شد ، برچسب <object> برای کار با کلیه برنامه های تعاملی و سایر عناصر خارجی که از یک فایل مجزا می توانند در صفحه وب قرار گیرند، بکار می رود. این عناصر نیز مانند اشیاء نرم افزاری ، شیء نامیده می شوند و در واقع دارای معنای مشابهی نیز هستند . برچسب <object> توسط نسخه های ٤ و بالاتر NetScapeNavigator و Microsoft Internet Explorer پشتیبانی می شوند .
برچسب <object> دارای شکل کلی زیر است :

```
<object classid="java:face.class" codebase="jdir" height=٥٠ width=٤٥٠">
</object>
```

انتقال از برچسب <APPLET> به برچسب <OBJECT> نیازمند تغییرات زیر است:

- به جای <APPLET> باید از برچسب <OBJECT> استفاده شود .
- خصوصیت CLASSID باید جانشین خصوصیت CODE شود و متن "java:" باید قبل از نام فایل کلاس اپلت قرار بگیرد .

سایر خصوصیات این برچسبها مانند HEIGHT, CODEBASE, ALIGN و WIDTH یکسان می باشد.

ایجاد اپلتهای ریسمانی

یکی از جنبه های بسیار پیچیده زبان جاوا، توانایی آن در نوشتن برنامه هایی است که می توانند چند وظیفه ای (multitasking) باشند . در جاوا وظایف همزمانی که کامپیوتر به دست می گیرد ، ریسمان (thread) نامیده می شود و کل این فرایند، چند ریسمانی شدن(multithreading) نامیده می شود. استفاده از ریسمانها در متحرک سازی و بسیاری برنامه های دیگر مفید می باشد .
خوب است بدانیم که پارامترها پاسخ اپلت به آرگومانها می باشند ، اطلاعاتی که می توانند هنگام اجرای برنامه ، از یک خط فرمان به آن فرستاده شوند . پارامترها در صفحه وبی که اپلت اجرا می شود ، قرار می گیرند و هنگام شروع اجرای اپلت ، خوانده می شوند .

فرستادن پارامترها از صفحه وب

یکی از دلایل استفاده از پارامتر در اپلتهای جاوا ، بیزاری و هراس از کامپایل کردن می باشد . پارامترها شما را قادر می سازند که عناصر اپلت را بدون هیچ گونه ویرایش و یا کامپایل کردن مجدد تغییر دهید . آنها همچنین برنامه را بسیار مفیدتر می سازند .
پارامترها در صفحه وبی که شامل اپلت می باشد، ذخیره می شوند. آنها با استفاده از برچسب <PARAM> در HTML و دو خصوصیت آن : NAME و VALUE ، ایجاد می شوند. یک اپلت می تواند می تواند بیش از یک برچسب <PARAM> باشد ، اما همه آنها باید بین دو برچسب <APPLET> و </APPLET> (یا برچسبهای <OBJECT> و </OBJECT> که آنها نیز از پارامترها پشتیبانی می کنند) قرار داشته باشند. مانند مثال زیر:

```
<APPLET  CODE="ScrollingHeadline.class" HEIGHT=٥٠ WIDTH=٤٠٠>
<PARAM  NAME="Headline١"   VALUE="Dewey defeats Truman">
<PARAM  NAME=" Headline٢"   VALUE="Stix nix hix pix">
<PARAM  NAME=" Headline٣"   VALUE="Man bites dog">
</APPLET>
```

این مثال می تواند برای فرستادن عناوین اخبار به اپلتی که آنها را روی صفحه به حرکت در آورد، به کار رود. تنها روش ایجاد برنامه از این نوع ، استفاده از پارامترها می باشد.
برای نامیدن یک پارامتر ، از خصوصیت NAME استفاده می شود . این خصوصیت را می توان با دادن نام به یک متغیر مقایسه کرد. خصوصیت VALUE مقداری را به پارامتر نامدار تخصیص می دهد .

دریافت پارامترها در اپلت

برای بازیابی پارامترها از صفحه وب ، ناچارید تغییراتی در برنامه خود اعمال کنید ، در غیر این صورت این پارامترها نادیده گرفته خواهد شد.
متد ()getParameter از کلاسJApplet ، یک پارامتر را از برچسب <PARAM> موجود روی یک صفحه وب بازیابی می کند . نام پارامتر که توسط خصوصیت NAME در صفحه وب مشخص می شود ، به عنوان آرگومان ()getParameter به کار می رود . آنچه در زیر آمده مثالی است از ()getParameter در عمل :

```
String display١= getParameter("Headinel");
```

متد ()getParameter تمامی پارامترها را به شکل رشته باز می گرداند ، لذا در صورت نیاز ناچارید آنها را به انواع دیگر تبدیل کنید .

متوقف ساختن ریسمان

متد ()stop ، هرگاه که خروج از صفحه مربوط به اپلت موجب توقف آن گردد ، اجرا می شود و بهترین جا برای متوقف ساختن ریسمان در حال اجرا می باشد .
متد ()stop مربوط به اپلت Revolve شامل جملات زیر است :

```
Public void stop() {
      If (runner != null) {
        Runner = null;
      }
}
```

بازیابی و پخش صدا در اپلت

تمام تواناییها صوتی زبان جاوا در کلاسJApplet (کلاس ارشد کلیه اپلتهای جاوا ۲) قرار دارد . این امر ممکن است باعث ایجاد این تصور شود که تنها در اپلتها می توان از صدا استفاده کرد ، حال آنکه فایلهای صوتی را می توان در هر برنامه جاوایی باز و پخش کرد .
دو راه برای پخش صدا در یک برنامه وجود دارد : به صورت دفعی و یا درون حلقه تکرار شونده .
صداها باید از فایلهایی خوانده شوند که قالب آنها توسط کلاس JApplet پشتیبانی می شود . جاوا ۲ به گونه ای بسط یافته که بتواند با کلیه قالبهای زیر کار کند :

- فایلهای AU
- فایلهای AIFF
- فایلهای WAV

برای ایجاد یک فایل صوتی با یکی از این قالبها ، صدای واقعی توسط کامپیوتر ضبط شده ، سپس به شکلی در می آید که بتواند درون یک فایل ذخیره شود . نام این قالبهای صوتی به عنوان پسوند نام فایل آنها به کار می رود .
جاوا همچنین می تواند با سه قالب فایل صوتی مبتنی بر MIDI ، کار کند : MIDI نوع ۰ ، MIDI نوع ۱ و RMF . این قالبها صدا را به نتهای موسیقی ، آلات موسیقی و ارتفاع به کار رفته برای ایجاد آنها ، تجزیه می کنند . کامپیوتری که توانایی پخش فایلهای MIDI را داشته باشد ، هنگام مواجهه با این اطلاعات ، قادر به ارائه آنها خواهد بود .