طراحی سیستمهای شئ گرا

مدرس استاد مهدیخانی

دانشگاه ادیبان گرمسار

www.irstu.com

به نام خداوند بخشنده ومهربان

مقدمه

شرکت Sun Microsystems زبان برنامه نویسی جاوا را در سال 1995 ارائه نمود, قواعد و دستور زبان جاوا بسیار شبیه به زبان Cو++Cمیباشد این زبان ابتدا برای کنترل و برنامه ریزی لوازمات الکترونیکی لوازم خانگی و سایر دستگاههای الکترونیکی تعبیه گردید تا قبل ازپیدایش زبان جاوا برای برنامه ریزی این تجهیزات از زبان C استفاده میگردید.

به علت متفاوت بودن نوع سخت افزار برای انجام یک کار واحد میبایستی کدهای متفاوتی تولید کرد برای مثال فرض کنید میخواهیم برای یک دستگاه تراشکاری و یک ماشین لباسشوئی یک تایمری به زبان سی بنویسیم به علت متفاوت بودن این دو دستگاه میبایستی دو برنامه مجزا بنا به ساختار سخت افزاری آن دو دستگاه نوشته شود و این کار هزینه تولید شده نرم افزار را بالا میبرد با این رویکرد در سال 1991 مهندسین شرکت سان تصمیم گرفتند زبانی را طراحی نمایند که مستقل از سخت افزار باشد این زبان ابتدا برای برنامه ریزی لوازم خانگی و اسباب بازیها طراحی گردید نام اولیه آن Oakبود ولی در سال 1995با نام جاوامعرفی گردید.

همزمان با پیدایش جاوا , اینترنت نیز با پیدایش ابزارهای جستجو مثل گوفِر در سال 1991 و تولید اولین مرورگر گرافیکی در سال1993با نام موزائیک متحول گردید نکته جالب این بود که کلاینتهای اینترنت هم بسیار متفاوت بودند کاربران اینترنت دارای سخت افزار و سیستم عاملهای مختلفی بودند بنابراین نیاز به زبانی بود که با استفاده از آن بتوان برنامه ای نوشت که روی کلیه سخت افزارهای مختلف قابل اجرا باشد. انتخابی بهتر از جاوا نبود به همین علت جاوا مناسبترین و محبوبترین زبان برنامه نویسی بروی اینترنت قرارگرفت و همین مسئله باعث فراگیر شدن این زبان در سطح دنیا گردید.

فصل اول : معرفی زبان جاوا

ویرایشهای زبان جاوا:

جاوا در سه ویرایش کلی عرضه میگردد.

1. نسخه استاندارد جاوا Java Standard Edition = JSE or JS2E

از این ویرایش جهت نوشتن برنامه های کامپیوترهای رومیزی استفاده میشود و شامل امکانات اصلی زبان جاوا میباشد.

2. نسخه سازمانی یا پیشرفته تر Java Enterprise Edition = JEE

از این ویرایش جهت نوشتن برنامه های توزیع شده تحت شبکه و تحت وب استفاده میشود مثل برنامه های مربوط به بانکها وبرنامه های مربوط به اداره راهنمائی و رانندگی و ..... .

3. نسخه میکرو (موبایل) Java Micro Edition = Java ME

از این ویرایش جهت نوشتن دستگاههای کوچک با منابع محدود استفاده میشود مثل نوشتن برنامه های مربوط به موبایل پیجر یا PDA
(PDA = Personal Digital Assistant)

توابع کتابخانه ای جاوا: API Application Programming Interface

جاوا نیز مانند سایر زبانهای برنامه نویسی با مجموعه ای غنی از توابع کتابخانه ای عرضه میگردد معمولا برای نوشتن برنامه های جاوا ازبلوکهای سازنده زیر استفاده میگردد.

1-کلاسها ومتدهای موجود در API جاوا

2- کلاسها و متدهای که خود برنامه نویس ایجاد میکند.

3- کلاسها ومتدهای که سایر برنامه نویسان نوشته اند و معمولا از طریق اینترنت قابل دسترسی میباشند.

مکانیسم نوشتن واجرای یک برنامه جاوا:

قبل از ورود به این بخش لازم است اندکی بیشتر با زبان جاوا آشنا بشویم همانطور که میدانیم در ابتدا برنامه نویسان برای نوشتن برنامه از زبان ماشین استفاده مینمودند بعد از آن زبان اسمبلی که بسیار وابسته به سخت افزار بود تولید گردید که دستورات توسط اسمبلر به زبان ماشین ترجمه میشدند برای ساده شدن کار, در سال 1957 اولین زبان برناامه نویسی درارای کامپایلر با نام فرترن نوشته شد و زبانهایی مثل پاسکال و کوبل هم با کامپایلرهای مخصوص به خود به میدان آمدند و در آخر در سال 1972 دنیس ریچی ربان کامل سی را با کامپایلر مخصوص به خود ارائه نمود همگی این زبانها با استفاده از کامپایلر دستورات را به زبان ماشین ترجمه میکردند تا برنامه قابل اجرا باشد.

نوع دوم هم زبانهای تفسیری میباشند که مفسر (Interpreter) خط به خط دستورات برنامه را ترجمه و اجرا میکند مثل زبان بیسیک که توسط بیل گیتس تولید گردید.

همه زبانهای اشاره شده شدیدا به سخت افزار وابسطه بوده و در موقع اجرا به روی سخت افزارهای متفاوت کامپایلر یا مفسر مربوط به خود را نیاز دارا هستند.

معرفی زبان کامپایلی - تفسیری جاوا:

زبان جاوا در دو مرحله کامپایل و اجرا میگردد و همین طراحی است که اجرای این زبان را از نوع سخت افزار مستقل نموده است.

بعد از نوشتن یک برنامه جاوا و کامپایل آن دستورات نوشته شده توسط کامپایلر جاوا به بایت کد تبدیل میگردند این بایت کدها در فایلی با پسوند class. ذخیره میگردند اگر برنامه نوشته شده جاوا دارای چندین فایل کلاس باشد (برنامه دارای کلاسهای بیشتری باشد) برای سهولت کار همه این فایلها در یک

پک بسته بندی یا آرشیو میشوند در این حالت بسته حاوی فایلها مانند فایلهای فشرده شده زیپ که با پسوند ZIP بسته بندی میگردند فایلهای جاوا با پسوند JAR بسته بندی میگردند JAR = Java Archive

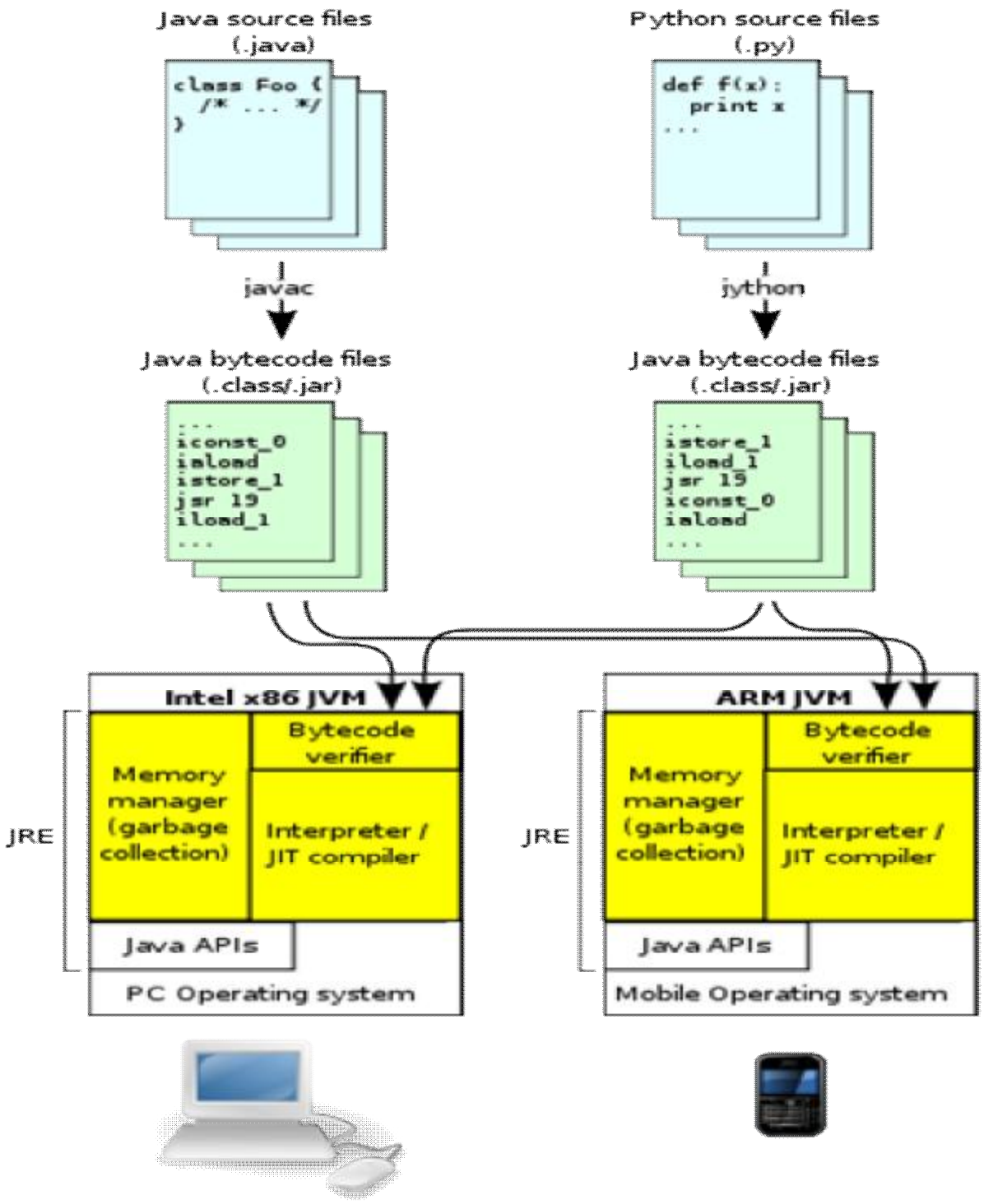
با کامپایل برنامه کدهای نوشته شده توسط برنامه نویس (سورس کد) به کدهای واسط (بایت کد) تبدیل میگردند .

در مرحله بعد این بایت کدها توسط ماشین مجازی جاوا یا همان JVM =Java Virtual Machine تفسیر و اجرا میگردند این دو مرحله ای بودن زبان جاوا قابلیت Portable این زبان را فراهم میسازد به طوری که با داشتن بایت کدها برنامه بروی کلیه سخت افزارها قابل اجرا میباشد به زبان ساده تر اجرای برنامه جاوا اینگونه میباشد.

1- نوشتن برنامه به زبان جاوا توسط برنامه نویس
2- کامپایل برنامه نوشته شده توسط برنامه نویس بروی کامپیوتر خود و تولید شدن بایت کد
3- تفسیر و اجرای بایت کد تولید شده توسط ماشین مجازی مختلف روی سخت افزارهای مختلف

شعار جاوا :"یک با بنویس همه جا اجرا کن"

نکته مهم اینجاست که بایت کدها توسط برنامه نویس تولید شده و در همه سخت افزارها قابل اجرا هستند و نکته جالبتر اینکه این قابلیت در لایه برنامه نویس هم امتیازات ارزنده ای به همراه دارد مثلا زبان برنامه نویسی پایتون هم بایت کد تولید میکند که توسط ماشین مجازی جاوا قابل اجرا میباشد.



نحوه اجرای یک برنامه جاوا :

ابتدا در یک ویرایشگر متن مثل Note Pad برنامه مورد نظر را نوشته و با پسوند JAVA. فایل مورد نظر را ذخیره میکنیم .

[redacted]

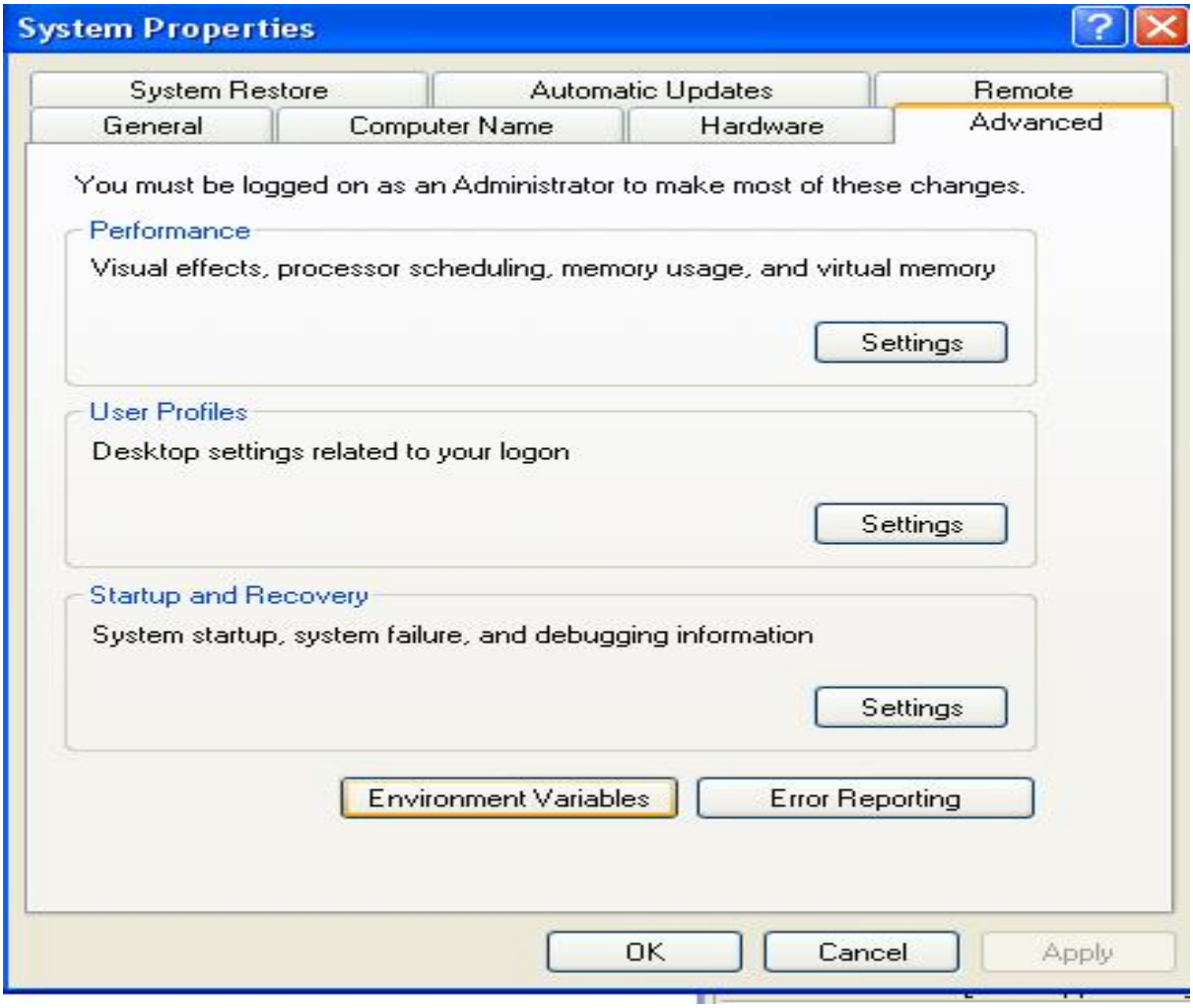
```
public class First {
        public static void main(String[] args) {
        System.out.println("In the name of God");
        }
}
```

برای اجرای برنامه های جاوا همانطور که قبلا هم اشاره شد میبایستی کامپایلر جاوا – ماشین مجازی جاوا و توابع کتابخانه ای جاوا بروی سیستم عامل نصب باشند برای سهولت کار شرکت سان یک کیت کاملی از موارد فوق تعبیه نموده و با نام JDK= Java Development Kit ارائه نموده است با نصب این کیت ملزومات مورد نیاز آماده به کار میباشند. .....

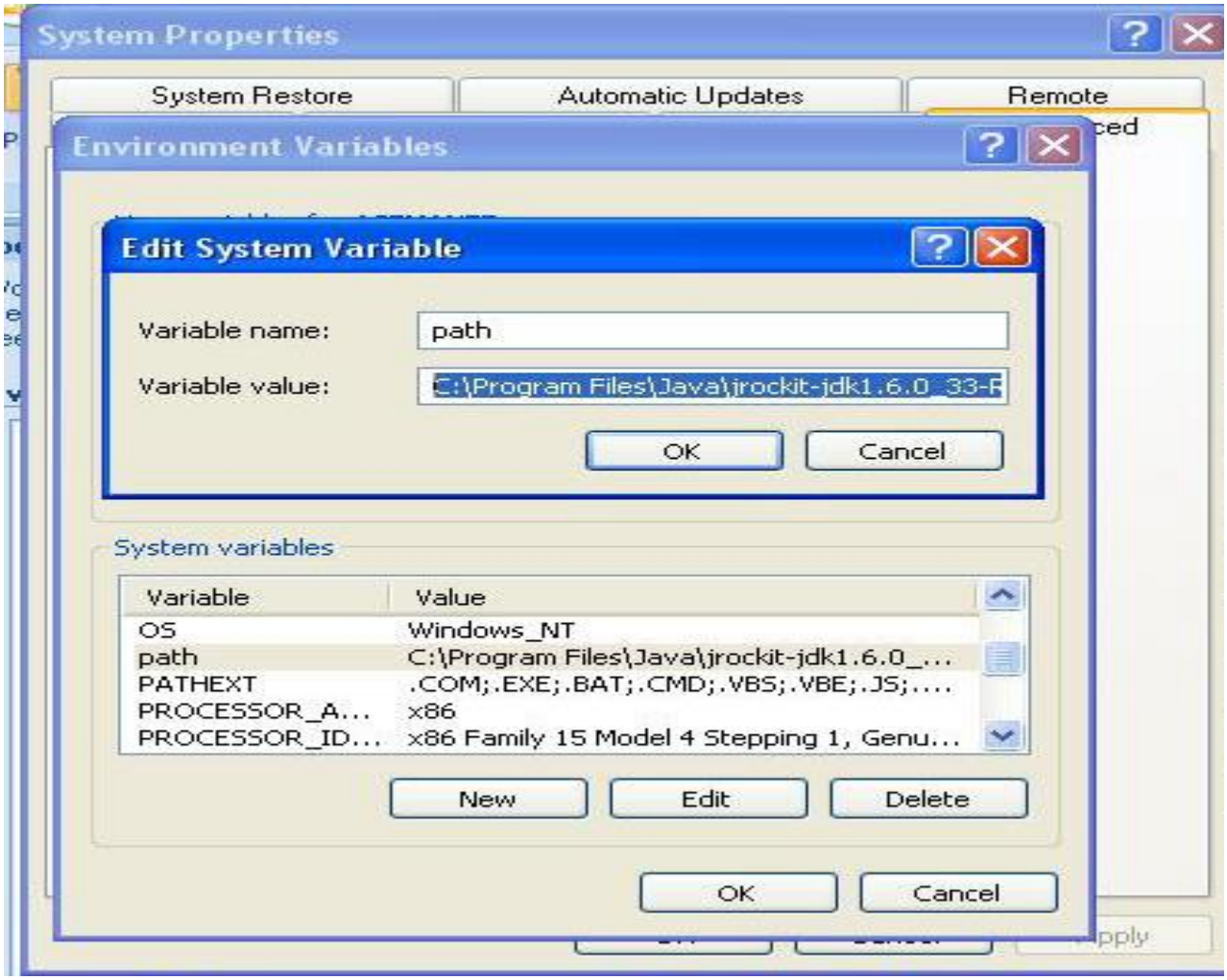
در پنجره Commandبرنامه نوشته شده را با کامپایلر جاوا فراخوانی میکنیم .

```
Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\ASEMAN77>javac
'javac' is not recognized as an internal or external command,
operable program or batch file.

C:\Documents and Settings\ASEMAN77>javac  d:\First.java
'javac' is not recognized as an internal or external command,
operable program or batch file.

C:\Documents and Settings\ASEMAN77>^P^P_
```

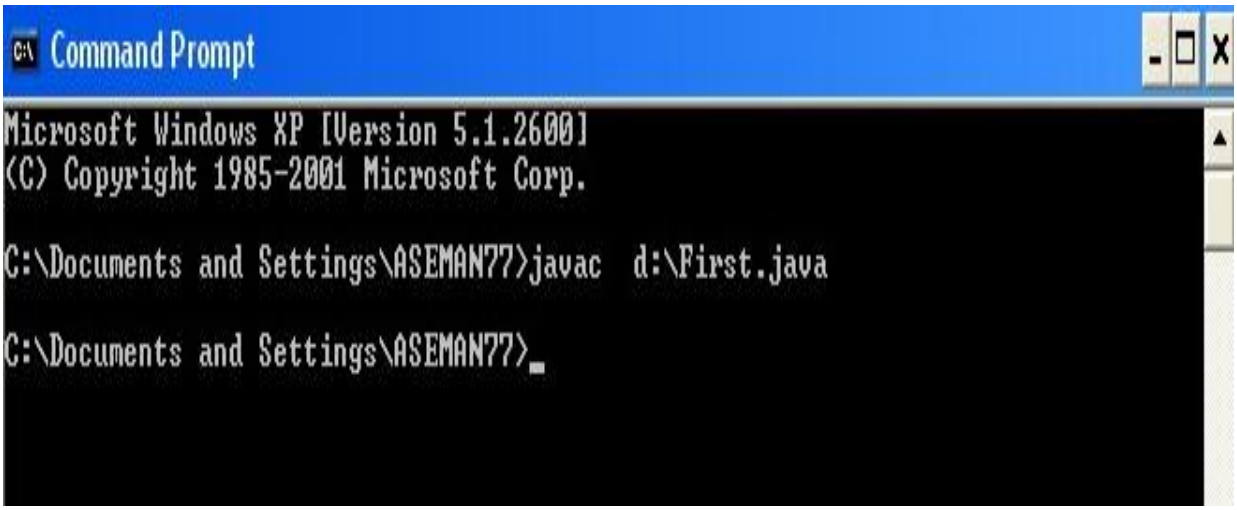
Javac نام کامپایلر جاوا است پیام خطای بالا مربوط به جسجوی سیستم عامل است که قادر به پیدا کردن مسیر نبوده است برای رفع این مشکل دامنه جستجو در سیستم عامل را افزایش داده و مسیر نصب JDK را معرفی میکنیم برای انجام این کار روی My Computer راست کلیک و سپس Propertiesو Advance



کلیک نموده و از پنجره باز شده تب Environment Variables را انتخاب و آدرس مربوطه را اضافه میکنیم در این حالت سیستم عامل در هنگام جستجو به این مسیر هم سرکشی مینماید.....



حالا مجدداً برنامه First.java را با کاماپایلر جاوا فراخوانی میکنیم :



در اینجا برنامه بدون پیام خطا کامپایل شده و فایلی با نام برنامه و پسوند کلاس ساخته میشود First.class برای اجرای برنامه, بایت کد یا همان First.class توسط JVM یا همان ماشین مجازی فراخوانی میگردد.

```
Command Prompt
C:\Program Files\Java\jrockit-jdk1.6.0_33-R28.2.4-4.1.0\jre\bin>java First
In the name of God

C:\Program Files\Java\jrockit-jdk1.6.0_33-R28.2.4-4.1.0\jre\bin>
```

در این مرحله بایت کدها به زبان ماشین ترجمه و اجرا شدند و خروجی برنامه در خط فرمان قابل مشاهده میباشد.

آنالیز مراحل اجرای برنامه فوق :

1-ویرایش برنامه Edit

2-کامپایل Compile

3-بارگذاری Loading

4- بررسی صحت Verify

5- اجرا Execution

با نوشتن برنامه در ویرایشگر متن و کامپایل برنامه توسط کامپایلر javac مراحل اول و دوم انجام گرفته است و سه مرحله آخر توسط ماشین مجازی انجام میپذیرد.

برای سهولت ویرایش- اجرای برنامه و استفاده بهینه از امکانات توسعه یافته میتوان از IDE استفاده نمود.

IDE= Integrated Development Environment

از IDE های معروف و محبوب میتوان به eclipse اشاره نمود .

فصل دوم: مقدمه‌ای بر برنامه‌های جاوا

یک برنامه جاوا معمولاً از تعدادی فایل با پسوند java. تشکیل شده است. در داخل یک فایل جاوا معمولاً تعدادی دستور import در ابتدای فایل وجود دارد. هر فایل جاوا می‌تواند شامل یک یا چند کلاس باشد که فقط یکی از آنها عمومی public می‌باشد. داخل یک کلاس می‌توانیم چندین attribute (صفت) و چندین متد داشته باشیم هر کلاس جاوا الزاماً یک متد اصلی به نام main دارد که نقطه شروع برنامه میباشد.

توجه به نکات زیر ضروری است:

1- نام کلاسی که به صورت public تعریف می‌شود دقیقاً باید همنام فایل جاوایی باشد که بر روی دیسک ذخیره شده است.

2- جاوا یک زبان Case Sensitive است و تمامی کلمات کلیدی در این زبان با حروف کوچک نوشته می‌شوند. مانند:

void, public, …

برای روشن شدن مطلب برنامه زیر را مورد بررسی قرار میدهیم.

```
1  // Fig. 2.1: Welcome1.java
2  // Text-printing program.
3
4  public class Welcome1
5  {
6     // main method begins execution of Java application
7     public static void main( String args[] )
8     {
9        System.out.println( "Welcome to Java Programming!" );
10
11    } // end method main
12
13 } // end class Welcome1
```

```
Welcome to Java Programming!
```

Fig. 2.1 | Text-printing program.

سطر 4) تعریف کلاس در این سطر شروع می‌شود public بیانگر این است که این کلاس عمومی می‌باشد و می‌توان در سایر قسمت‌های برنامه از آن استفاده کرد و به آن access modifier گفته می‌شود، در فصل‌های بعدی با سایر access modifier آشنا می‌شویم.

class کلمه کلیدی است که بعد از آن، نام کلاس می‌آید.

سطر 7) کلاس Welcom1 تنها یک متد دارد که نام آن متد main می‌باشد در هر برنامه جاوا برای اجرای برنامه حداقل یک متد main میبایستی وجود داشته باشد. این متد نقطهٔ شروع برنامه بوده و در واقع JVM برنامه را با فراخوانی این متد اجرا می‌کند. این متد مشابه متد main در برنامه C++ می‌باشد.

این متد باید دقیقاً به همین شکل نوشته شود در غیر این صورت JVM قادر به اجرای برنامه نخواهد بود.

Access modifier-public: معرف دسترسی متد می‌باشد و چنانچه یک متد public باشد می‌توان آن را از داخل سایر کلاس‌ها در برنامه صدا زد.

static: بیانگر این است که این متد یک متد static می‌باشد.

(در مورد متدهای static در فصل‌های آینده صحبت خواهد شد.)

void قبل از نام متد نوع داده برگشتی از متد مشخص می‌گردد، void یعنی این متد هیچ مقداری برنمی‌گرداند.

main : نام متد اصلی میباشد.

String args[] در صورتی که متد پارامتر بپذیرد این پارامترها در داخل پرانتز جلوی نام متد نوشته می‌شود. در غیر این صورت داخل پرانتز خالی خواهد بود. در متد main، پارامتری به نام args از نوع آرایه رشته‌ای تعریف شده است.

سطر 9) System.out شی است که به خروجی استاندارد معروف است و دارای متدهایی است که برای چاپ مقدار (عبارات) در خروجی استاندارد (کنسول یا خط فرمان) مورد استفاده قرار می‌گیرد.

متد ()println یکی از متدهای این شی می‌باشد که یک آرگومان را دریافت کرده و آن را چاپ می‌کند.

()println : بعد از چاپ، مقدار اشاره‌گر را به خط بعد می‌برد.

```
1  // Fig. 2.4: Welcome3.java
2  // Printing multiple lines of text with a single statement.
3
4  public class Welcome3
5  {
6     // main method begins execution of Java application
7     public static void main( String args[] )
8     {
9        System.out.println( "Welcome\nto\nJava\nProgramming!" );
10
11    } // end method main
12
13 } // end class Welcome3
```

Fig. 2.4 | Printing multiple lines of text with a single statement. (Part 1 of 2.)

```
Welcome
to
Java
Programming!
```

در شکل2,4 در سطر 9

درداخل آرگومان دستور println کاراکتر \n خط را می‌شکند و اشاره‌گر را به خط بعدی می‌برد.

```
1  // Fig. 2.6: Welcome4.java
2  // Printing multiple lines in a dialog box.
3
4  public class Welcome4
5  {
6     // main method begins execution of Java application
7     public static void main( String args[] )
8     {
9        System.out.printf( "%s\n%s\n",
10          "Welcome to", "Java Programming!" );
11
12    } // end method main
13
14 } // end class Welcome4
```

```
Welcome to
Java Programming!
```

Fig. 2.6 | Displaying multiple lines with method `System.out.printf`.

در سطر 9 در شکل 6-2 در دستور prinf دو آرگومان رشته ای فرمت بندی میشوند

printf برای چاپ خروجی فرمت شده استفاده می‌شود و شبیه دستور printf در زبان C می‌باشد و مـی‌تـوان

در عبارت مورد نظر برای چاپ از کاراکترهای فرمت‌دهی مانند %s و %f و %d و ... استفاده کرد.

```java
// Fig. 2.7: Addition.java
// Addition program that displays the sum of two numbers.
import java.util.Scanner; // program uses class Scanner

public class Addition
{
   // main method begins execution of Java application
   public static void main( String args[] )
   {
      // create Scanner to obtain input from command window
      Scanner input = new Scanner( System.in );

      int number1; // first number to add
      int number2; // second number to add
      int sum; // sum of number1 and number2

      System.out.print( "Enter first integer: " ); // prompt
      number1 = input.nextInt(); // read first number from user

      System.out.print( "Enter second integer: " ); // prompt
      number2 = input.nextInt(); // read second number from user

      sum = number1 + number2; // add numbers

      System.out.printf( "Sum is %d\n", sum ); // display sum

   } // end method main

} // end class Addition
```

Fig. 2.7 | Addition program that displays the sum of two numbers. (Part 1 of 2.)

```
Enter first integer: 45
Enter second integer: 72
Sum is 117
```

شکل 7-2 سطر 3 دستور nimport Java.util.scanner

باید در ابتدای فایل جاوا با استفاده از import نام و مسیر این کلاس را به کامپایلر معرفی می‌کنیم. این خط (3) نشان می‌دهد که در این برنامه از کلاس scanner که در پکیج (package) Java.util قرار دارد استفاده کرده‌ایم. کلاس‌های API جاوا برای سهولت کار کردن با آنها در داخل پوشه‌هایی سازماندهی شده‌اند، که به این پوشه‌ها در ادبیات Java پکیج گفته می‌شود.

این دستور مشابه عملی را انجام می‌دهد که include در C و C++ و using در دلفی انجام می‌دادند.

سطر 11) Scanner input = new Scanner (System.in);

در این سطر متغیری به نام input از نوع Scanner تعریف شده است و در همین سطر نیز مقداردهی شده است. برای استفاده از یک کلاس باید از روی آن یک شی ساخته شود. دستور New، شیی از کلاسی که نام آن بعد از New آمده ایجاد می‌کند. در این دستور آدرس شی ایجاد شده در حافظ در متغیر input قرار می‌گیرد و در ادامهٔ برنامه می‌توان با استفاده از این متغیر به شی ایجاد شده دسترسی داشت و متدهای آن (در واقع متدهای کلاسی که شی از روی آن ساخته شده) را فراخوانی کرد.

کلاس scanner کلاسی است که برای خواندن مقادیر از ورودی استاندارد فایل و ... استفاده می‌شود. در زمان ایجاد شی از روی این کلاس مشخص کرد که مقادیر باید از کجا خوانده شوند این کار با استفاده از عبارت system.in که شی ورودی استاندارد در زبان جاوا می‌باشد مشخص می‌گردد.

سطر 18) number 1 = input.nextInt ( );

در دستور فوق یک مقدار اینتجر از ورودی دریافت و در متغیر number1 قرار میگیرد.

nextInt() یکی از متدهایی است که در داخل کلاس Scanner قرار دارد و ما می‌توانیم با استفاده از متغیر (شئ) input آن را صدا بزنیم. این متد وقتی صدا زده شود، در خط فرمان منتظر می‌ماند تا کاربر یک عدد صحیح را وارد کند و Enter را فشار دهد. سپس مقدار وارد شده را برمی‌گرداند. که این مقدار برگشتی در این دستور در متغیر namber1 ریخته می‌شود.

در دستور پرینت (printf)داریم:

%s چاپ رشته

%d چاپ عدد صحیح

%f چاپ عدد اعشاری

فصل سوم: برنامه‌نویس شی‌گرا

در برنامه‌نویس شی‌گرا مدل برنامه‌نویسی تا حدودی با روش رویه‌ای Procedural مانند زبان‌های C و پاسکال تفاوت دارد. در این روش نوع برنامه‌نویسی مشابه چیزی است که در دنیای واقعی با آن سروکار داریم، در دنیای واقعی ما با مجموعه‌ای از اشیاء سروکار داریم که هر شی برای خود خصوصیاتی دارد و نیز می‌تواند اعمالی را انجام دهد؛ مانند خودکار که دارای خصوصیتی مانند رنگ است و عملی مانند نوشتن را انجام می‌دهد، و یا حساب بانکی که دارای خصوصیتی به نام میزان موجودی (balance) بوده و می‌توان اعمالی مانند واریز وجه به حساب را انجام داد.

در برنامه‌نویسی شی‌گرا به طور مشابه عمل می‌شود و یعنی یک برنامهٔ بزرگ به تعدادی کلاس شکسته می‌شود که هر کلاس دارای خصوصیاتی است attribute و نیز کارهایی را انجام می‌دهد (Method) کلاس‌ها را می‌توان با نقشه‌های طراحی شده یک ماشین مقایسه کرد. همانگونه که از روی نقشهٔ ماشین نمونه‌های واقعی از

ماشین ساخته می‌شوند، از روی کلاس‌ها نیز در برنامه می‌توان نمونه‌هایی ایجاد کرد که به این نوع نمونه‌ها object گفته می‌شود.

```
1  // Fig. 3.1: GradeBook.java
2  // Class declaration with one method.
3
4  public class GradeBook
5  {
6     // display a welcome message to the GradeBook user
7     public void displayMessage()
8     {
9        System.out.println( "Welcome to the Grade Book!" );
10    } // end method displayMessage
11
12 } // end class GradeBook
```

g. 3.1 | Class declaration with one method.

Chapter 3 fig 3.1

سطر 7): متدی به نام ()displaymessage تعریف شده است، که دسترسی به آن به صورت public است.

(access Modifire)

این بدین معنی است که این متد عمومی بوده و از داخل سایر کلاس‌ها در برنامه می‌توان آن را صدا زد.

```
1  // Fig. 3.2: GradeBookTest.java
2  // Create a GradeBook object and call its displayMessage method.
3
4  public class GradeBookTest
5  {
6     // main method begins program execution
7     public static void main( String args[] )
8     {
9        // create a GradeBook object and assign it to myGradeBook
10       GradeBook myGradeBook = new GradeBook();
11
12       // call myGradeBook's displayMessage method
13       myGradeBook.displayMessage();
14    } // end main
15
16 } // end class GradeBookTest
```

```
Welcome to the Grade Book!
```

Fig. 3.2 | Creating an object of class `GradeBook` and calling its `displayMessage` method.

سطر 7 (fig: 3.2): کلمهٔ کلیدی static مشخص می‌کند که این متد متد استاتیک است. متدهای استاتیک متدهای خاصی هستند که برای صدا زدن آنها نیاز به ساخت شی از روی کلاس وجود نـدارد و مـی‌تـوان آنهـا را مستقیماً با اسم کلاس صدا زد. در این مورد در فصل‌های بعدی بیشتر بحث خواهد شد. این مسئله بـه Jvm ایـن امکان را می‌دهد که بتواند بدون نیاز به ساختن شـی از روی کـلاس GradeBookTtest متـد Main آن را صـدا بزند.

برای تعریف یک کلاس به صورت استاتیک کافیست قبل از نوع برگشتی متد از کلمهٔ کلیدی static استفاده کرد.

سطر 10: **Gread Book my Gread Book = new Gread Book ( )**

متغیری به نام My Gread Book از نوع کلاس Gread Book معرفی شده است در واقع این متغیر می‌تواند اشاره‌گری به یک شی از نوع Gread Book را در خود ذخیره کند. (آدرس یک شی از نـوع Gread Book) البتـه برای سهولت کار My Gread Book را یک شی از نوع Gread Book به حساب می‌آوریم.

دستور New همان طور که قبلاً اشاره شد برای ایجاد یک شی از روی کلاس استفاده می‌شود. و شی از نـوع کلاس که اسم آن بعد از این دستور آمده در حافظه ایجاد می‌کند و آدرس آن را برمی‌گرداند. که در ایـن دسـتور این آدرس در داخل متغیر My Gread Book ریخته می‌شود.

وجود پرانتزها بعد از نام کلاس در دستور New اجباری است و برای صدا زدن متد سـازنده کـلاس اسـتفاده می‌شود. در این مورد در قسمت‌های بعدی بحث خواهد شد.

13) My Gread Book. display Message ( );

خط 13: از شی ساخته شده برای فراخوانی متدهای موجود در کلاس استفاده می‌گردد. به ایـن ترتیـب کـه نام شی را ذکر کرده و نام متد را بعد از دات. می‌آوریم. وجود پرانتز در جلوی نام متد در هنگام فراخوانی اجبـاری است. پرانتز خالی نشان می‌دهد که در زمان فراخوانی متد هیچ آرگومانی به آن پاس داده نمی‌شود. و در واقـع در تعریف متد پارامتری مشخص نشده است.

```java
// Fig. 3.5: GradeBookTest.java
// Create GradeBook object and pass a String to
// its displayMessage method.
import java.util.Scanner; // program uses Scanner

public class GradeBookTest
{
   // main method begins program execution
   public static void main( String args[] )
   {
      // create Scanner to obtain input from command window
      Scanner input = new Scanner( System.in );

      // create a GradeBook object and assign it to myGradeBook
      GradeBook myGradeBook = new GradeBook();

      // prompt for and input course name
      System.out.println( "Please enter the course name:" );
      String nameOfCourse = input.nextLine(); // read a line of text
      System.out.println(); // outputs a blank line

      // call myGradeBook's displayMessage method
      // and pass nameOfCourse as an argument
      myGradeBook.displayMessage( nameOfCourse );
   } // end main

} // end class GradeBookTest
```

```
Please enter the course name:
CS101 Introduction to Java Programming

Welcome to the grade book for
CS101 Introduction to Java Programming!
```

Fig. 3.5 | Creating a GradeBook object and passing a String to its displayMessage method.

در شکل 5-3 خط 19: متد ( ) next line یکی دیگر از متدهای کلاس Scanner می‌باشد که با استفاده از شی ساخته شده از روی آن، آن را فراخوانی می‌کنیم. این متد در خط فرمان منتظر می‌ماند و رشته وارد شده توسط کاربر را برمی‌گرداند.

سطر 20: سطر خالی چاپ می‌کند.

سطر 24: متد display message() را فراخوانی می‌کند و متغیر nameOfcource را به عنوان آرگومان به آن ارسال می‌کند در این مرحله کنترل اجرا به ابتدای متد display message () در کلاس GreadeBook منتقل می‌شود (سطر 7 از شکل 4-3) و مقدار متغیر ارسال شده در داخل پارامتر course name کپی می‌شود. و بدنه متد اجرا می‌شود و در نهایت بعد از اتمام متد در سطر 11 به جای قبلی (سطری که متد را صدا زده بود یعنی سطر 24 از شکل 5-3) بازمی‌گردیم و برنامه ادامه پیدا می‌کند

نکته: کلاس‌های داخل یک پکیج نیازی به import کردن ندارند و به صورت ضمنی import می‌شوند.

نکته: هدف از دستور import مشخص کردن مسیر کامل کلاس است تا زمانی که چندین کلاس همنام در برنامه وجود دارد و تداخلی نداشته باشیم به جای دستور import می‌توان در زمان استفاده از یک کلاس مسیر کامل آن را مشخص کرد.

مثال Java.util.scanner in = new Java.util.scanner (system.in);

نکته: **سطر** 19 *: برخلاف سایر objectها objectهای از نوع string نیازی به ایجاد شدن توسط دستور New ندارند و Java به صورت ضمنی آنها را ایجاد می‌کند.

نکته: همان گونه که ملاحظه می‌شود برای استفاده از کلاس‌های System و String آنها را import نکردیم، این کلاس‌ها به همراه تعدادی از سایر کلاس‌های پر استفاده جاوا در داخل پکیجی به نام Java.lang قرار دارند و این پکیج به طور ضمنی در تمام برنامه‌های جاوا import می‌شود.

متغیرهای نمونه Instance Variables

و متدهای Setter و Getter

متغیرهایی که در داخل بدنهٔ یک متد تعریف می‌شوند متغیرهای محلی local variables نامیده می‌شوند. این متغیرها قبل از استفاده حتماً باید توسط برنامه‌نویس مقداردهی اولیه شوند و فقط در همان متد که تعریف شده‌اند قابل دسترسی می‌باشند. در مقابل متغیرهایی که در داخل کلاس و خارج از بدنهٔ متدها تعریف می‌شوند field نامیده می‌شوند.

زمانی که از روی یک کلاس شی یا نمونه (Instance) ساخته می‌شود هر شی کپی مختص به خود را از این فیلدها خواهد داشت، به همین خاطر به این فیلدها متغیرهای نمونه یا Instance Variable گفته می‌شود.

Instance Variableها در زمان ایجاد شی به طور پیش فرض مقداردهی می‌گردند.

```
1   // Fig. 3.7: GradeBook.java
2   // GradeBook class that contains a courseName instance variable
3   // and methods to set and get its value.
4
5   public class GradeBook
6   {
7      private String courseName; // course name for this GradeBook
8
9      // method to set the course name
10     public void setCourseName( String name )
11     {
12        courseName = name; // store the course name
13     } // end method setCourseName
14
15     // method to retrieve the course name
16     public String getCourseName()
17     {
18        return courseName;
19     } // end method getCourseName
20
21     // display a welcome message to the GradeBook user
22     public void displayMessage()
23     {
24        // this statement calls getCourseName to get the
25        // name of the course this GradeBook represents
26        System.out.printf( "Welcome to the grade book for\n%s!\n",
27           getCourseName() );
28     } // end method displayMessage
29
30  } // end class GradeBook
```

Fig. 3.7 | GradeBook class that contains a courseName instance variable and methods to set and get its value.

در شکل 7-3سطر 7: فیلد course name از نوع رشته و به صورت private (خصوصی) تعریف شده است. فیلدهای private توسط متدهای همان کلاس قابل استفاده می‌شود، ولی از سایر کلاس‌ها قابل استفاده به صورت مستقیم نخواهد بود.

سطر 10: برای set کردن فیلد CourseName از یک متد setter استفاده کرده‌ایم که یک رشته را گرفته داخل CourseName ذخیره می‌کند. یکی از مزایای استفاده از متدهای setter این است که می‌توان قبل از ذخیره مقدار آن را اعتبارسنجی کرد و سپس در فیلد ذخیره کرد تا از ورود داده‌های اشتباه به فیلدها جلوگیری شود.

سطر 16:

برای خواندن مقدار فیلد course name از یک متد getter استفاده شده است که یک مقدار از نـوع رشـته را برمی‌گرداند. مزیت استفاده از متدهای getter این است که چنانچه بخواهیم تغییری در یک مقدار یک فیلد قبل از استفاده از آن بدهیم می‌توان در این متد این کار را انجام داد. به جای اینکه مجبور باشیم در سایر قسمت‌هـای برنامه تغییرات اعمال کنیم به عنوان مثال فیلـد حقـوق را در نظـر بگیریـد کـه ممکـن اسـت بخـواهیم قبـل از برگرداندن آن درصدی از پاداش را به آن اضافه کنیم.

سطر 18:

برای برگرداندن مقدار، از یک متد از دستور return استفاده می‌شود.

سطر 27: **get**CourseName **( )**

برای فراخوانی یک متد از داخل متد دیگر در همان کلاس نیازی به ساختن شی نیست و می‌توان این کار را مستقیماً انجام داد.

```java
1  // Fig. 3.8: GradeBookTest.java
2  // Create and manipulate a GradeBook object.
3  import java.util.Scanner; // program uses Scanner
4
5  public class GradeBookTest
6  {
7     // main method begins program execution
8     public static void main( String args[] )
9     {
10       // create Scanner to obtain input from command window
11       Scanner input = new Scanner( System.in );
12
13       // create a GradeBook object and assign it to myGradeBook
14       GradeBook myGradeBook = new GradeBook();
15
16       // display initial value of courseName
17       System.out.printf( "Initial course name is: %s\n\n",
18          myGradeBook.getCourseName() );
19
20      // prompt for and read course name
21      System.out.println( "Please enter the course name:" );
22      String theName = input.nextLine(); // read a line of text
23      myGradeBook.setCourseName( theName ); // set the course name
24      System.out.println(); // outputs a blank line
25
26      // display welcome message after specifying course name
27      myGradeBook.displayMessage();
28    } // end main
29
30 } // end class GradeBookTest
```

```
Initial course name is: null

Please enter the course name:
CS101 Introduction to Java Programming

Welcome to the grade book for
CS101 Introduction to Java Programming!
```

Fig. 3.8 | Creating and manipulating a GradeBook object.

در شکل 8-3 سطر 14:

دستور New یک شی از کلاس GradeBook در حافظه ایجاد می‌کند نام این شی myGradeBook است بنابراین شی ایجاد شده یک کپی مخصوص به خود از فیلد CourseName را خواهد داشت.

سطر 18 myGreadBook. get CourseName ()

متد getCourseName () از شی myGradeBook فراخوانی یا احضار (invoke) می‌شود. در واقع با این فراخوانی به سطر 16 (public ...) از شکل 3.7 منتقل می‌شویم. مقدار برگشتی از دستور return به جای عبارت فراخوانی قرار می‌گیرد و در نهایت توسط دستور printf چاپ می‌شود.

شکل 3.7 سطر 7 به این قضیه پنهان‌سازی داده یا data hiding گفته می‌شود و باعث افزایش امنیت دسترسی به داده‌ها می‌شود.

انواع داده‌ای در Java:

در زبان جاوا دو نوع داده‌ای وجود دارد، نوع داده اصلی (primitive) و نوع دادهٔ ارجاعی (Reference)

انواع داده‌های اصلی عبارتند از:

char, byte, boolean, short, int, long, float, double.

سایر انواع داده‌ای، انواع داده‌ای ارجاعی می‌باشند. بنابراین کلاس‌هایی که از روی آن‌ها شی ایجاد کردیم جزء انواع ارجاعی می‌باشند مانند کلاس Scanner، کلاس GradeBook، String با توجه به آنچه در خصوص مقداردهی اولیه instance variableها گفته شد.

انواع داده‌ای اصلی عددی یعنی:

Byte, char, short, int, long, float, double

مقدار پیش فرض صفر نوع boolean نوع پیش فرض false به خود می‌گیرد.

انواع داده‌ای ارجاعی به مقدار null مقداردهی می‌شود.

* (متغیرهای local باید توسط برنامه‌نویس مقداردهی شوند.)*

متدهای سازنده یا Constructors:

برای مقداردهی اولیه به فیلدهای یک شی می‌توان از متد سازنده استفاده کرد. هر کلاس باید حداقل یک متد سازنده داشته باشد.

چنانچه متد سازنده‌ای برای یک کلاس ایجاد نشود کامپایلر به طور پیش فرض یک متد سازندهٔ بدون پارامتر برای آن اضافه می‌کند.

متدهای سازنده مانند متدهای معمولی می‌توانند دارای پارامتر یا بدون پارامتر باشند. متد سازنده توسط دستور New و بعد از ایجاد شی به طور اتوماتیک صدا زده می‌شود.

سطر 9 *:

این متد، متد سازنده می‌باشد و چنانچه مشخص است نوع برگشتی ندارد و دقیقاً همنام کلاس است. دسترسی به این متد غالباً public است اما برای استفاده‌های خاص (مثلاً الگوی طراحی singleton) می‌توان آن را private نیز تعریف کرد که در این صورت نمی‌توان از سایر کلاس‌ها از روی این کلاس شئ ساخت.

```java
// Fig. 3.10: GradeBook.java
// GradeBook class with a constructor to initialize the course name.

public class GradeBook
{
   private String courseName; // course name for this GradeBook

   // constructor initializes courseName with String supplied as argument
   public GradeBook( String name )
   {
      courseName = name; // initializes courseName
   } // end constructor

   // method to set the course name
   public void setCourseName( String name )
   {
      courseName = name; // store the course name
   } // end method setCourseName

   // method to retrieve the course name
   public String getCourseName()
   {
      return courseName;
   } // end method getCourseName

   // display a welcome message to the GradeBook user
   public void displayMessage()
   {
      // this statement calls getCourseName to get the
      // name of the course this GradeBook represents
      System.out.printf( "Welcome to the grade book for\n%s!\n",
         getCourseName() );
   } // end method displayMessage

} // end class GradeBook
```

Fig. 3.10 | GradeBook class with a constructor to initialize the course name.

اعداد اعشاری با ممیز شناور در Java:

برای کار با اعداد اعشاری در زبان java مـی‌تـوان از دو نـوع float و double اسـتفاده کـرد. نـوع float، 32 بایتی و نوع double، 64 بایتی می‌باشد و دقت آن از نوع float بیشتر است.

```java
// Fig. 3.14: AccountTest.java
// Inputting and outputting floating-point numbers with Account objects.
import java.util.Scanner;

public class AccountTest
{
   // main method begins execution of Java application
   public static void main( String args[] )
   {
      Account account1 = new Account( 50.00 ); // create Account object
      Account account2 = new Account( -7.53 ); // create Account object

      // display initial balance of each object
      System.out.printf( "account1 balance: $%.2f\n",
         account1.getBalance() );
      System.out.printf( "account2 balance: $%.2f\n\n",
         account2.getBalance() );

      // create Scanner to obtain input from command window
      Scanner input = new Scanner( System.in );
      double depositAmount; // deposit amount read from user

      System.out.print( "Enter deposit amount for account1: " ); // prompt
      depositAmount = input.nextDouble(); // obtain user input
      System.out.printf( "\nadding %.2f to account1 balance\n\n",
         depositAmount );
      account1.credit( depositAmount ); // add to account1 balance

      // display balances
      System.out.printf( "account1 balance: $%.2f\n",
         account1.getBalance() );
      System.out.printf( "account2 balance: $%.2f\n\n",
         account2.getBalance() );

      System.out.print( "Enter deposit amount for account2: " ); // prompt
      depositAmount = input.nextDouble(); // obtain user input
      System.out.printf( "\nadding %.2f to account2 balance\n\n",
         depositAmount );
      account2.credit( depositAmount ); // add to account2 balance

      // display balances
      System.out.printf( "account1 balance: $%.2f\n",
         account1.getBalance() );
      System.out.printf( "account2 balance: $%.2f\n",
         account2.getBalance() );
   } // end main

} // end class AccountTest
```

```
account1 balance: $50.00
account2 balance: $0.00

Enter deposit amount for account1: 25.53

adding 25.53 to account1 balance

account1 balance: $75.53
account2 balance: $0.00

Enter deposit amount for account2: 123.45

adding 123.45 to account2 balance

account1 balance: $75.53
account2 balance: $123.45
```

Fig. 3.14 | Inputting and outputting floating-point numbers with Account objects. (Part 2 of 2.)

در شکل 14-3 خط 10:

ثابت‌های اعشاری در زبان java به طور پیش فرض double در نظر گرفته می‌شوند.

سطر 36:

یکی دیگر از متدهای کلاس Scanner می‌باشد و برای خواندن اعداد اعشاری از ورودی استفاده می‌گردد.

سطر 30:

بعد از % می‌توان اطلاعات فرمت کردن را مشخص کرد تا عدد اعشاری با فرمت مشخص شده چاپ شود.

.2f یعنی عدد اعشاری از نوع float با دو رقم اعشار چاپ شود.

```
1  // Fig. 3.17: Dialog1.java
2  // Printing multiple lines in dialog box.
3  import javax.swing.JOptionPane; // import class JOptionPane
4
5  public class Dialog1
6  {
7     public static void main( String args[] )
8     {
9        // display a dialog with a message
10       JOptionPane.showMessageDialog( null, "Welcome\nto\nJava" );
11    } // end main
12 } // end class Dialog1
```

Fig. 3.17 | Using JOptionPane to display multiple lines in a dialog box.

در شکل 17-3

کلاس JOptionPane یکی از کلاس‌های API جاوا است که شامل متدهایی است که برای نمـایش پیغـام از خروجی یا خواندن مقدار از خروجی در قالب یک پنجره استفاده می‌شود.

سطر 10:

()showMassegDialog یکی از متدهای کلاس :JOptionPane می‌باشـد کـه بـرای نمـایش پیـام در یـک پنجره dialog استفاده می‌شود.

... Welcome و null موقعیت قرار گرفتن پنجره روی صفحه را مشخص می‌کند

Null به معنی این است که این پنجـره هـیچ parent (والـدی) نـدارد. بنـابراین وسـط صـفحه نمـایش داده می‌شود.

2) آرگومان دوم پیامی است که باید نمایش داده شود.

```
1  // Fig. 3.18: NameDialog.java
2  // Basic input with a dialog box.
3  import javax.swing.JOptionPane;
4
5  public class NameDialog
6  {
7     public static void main( String args[] )
8     {
9        // prompt user to enter name
10       String name =
11          JOptionPane.showInputDialog( "What is your name?" );
12
13       // create the message
14       String message =
15          String.format( "Welcome, %s, to Java Programming!", name );
16
17       // display the message to welcome the user by name
18       JOptionPane.showMessageDialog( null, message );
19    } // end main
20 } // end class NameDialog
```

در شکل 3- 18 سطر 11:

این متد یکی دیگر از متدهای JOptionPane می‌باشد و برای گرفتن مقدار از ورودی در قالب یک پنجره یا دیالوگ استفاده می‌شود و آرگومان مشخص شده به عنوان یک پیام نمایش می‌دهد.

در سطر15 : متد format یکی از متدهای کلاس String می‌باشد و عملکرد آن مشابه عملکرد متد printf می‌باشد، با این تفاوت که رشته حاصل را به جای چاپ کردن برمی‌گرداند. و می‌توان آن را ذخیره و در جاهای دیگر برنامه از آن استفاده کرد.

فصل چهارم: ساختارهای کنترلی

```
1   // Fig. 4.6: GradeBook.java
2   // GradeBook class that solves class-average problem using
3   // counter-controlled repetition.
4   import java.util.Scanner; // program uses class Scanner
5
6   public class GradeBook
7   {
8      private String courseName; // name of course this GradeBook represents
9
10     // constructor initializes courseName
11     public GradeBook( String name )
12     {
13        courseName = name; // initializes courseName
14     } // end constructor
15
16     // method to set the course name
17     public void setCourseName( String name )
18     {
19        courseName = name; // store the course name
20     } // end method setCourseName
21
22     // method to retrieve the course name
23     public String getCourseName()
24     {
25        return courseName;
26     } // end method getCourseName
27
28     // display a welcome message to the GradeBook user
29     public void displayMessage()
30     {
31        // getCourseName gets the name of the course
```

```
32         System.out.printf( "Welcome to the grade book for\n%s!\n\n",
33            getCourseName() );
34      } // end method displayMessage
35
36      // determine class average based on 10 grades entered by user
37      public void determineClassAverage()
38      {
39         // create Scanner to obtain input from command window
40         Scanner input = new Scanner( System.in );
41
42         int total; // sum of grades entered by user
43         int gradeCounter; // number of the grade to be entered next
44         int grade; // grade value entered by user
45         int average; // average of grades
46
47         // initialization phase
48         total = 0; // initialize total
49         gradeCounter = 1; // initialize loop counter
50
51         // processing phase
52         while ( gradeCounter <= 10 ) // loop 10 times
53         {
54            System.out.print( "Enter grade: " ); // prompt
55            grade = input.nextInt(); // input next grade
56            total = total + grade; // add grade to total
57            gradeCounter = gradeCounter + 1; // increment counter by 1
58         } // end while
59
60         // termination phase
61         average = total / 10; // integer division yields integer result
62
63         // display total and average of grades
64         System.out.printf( "\nTotal of all 10 grades is %d\n", total );
65         System.out.printf( "Class average is %d\n", average );
66      } // end method determineClassAverage
67
68   } // end class GradeBook
```

Fig. 4.6 | Counter-controlled repetition: Class-average problem. (Part 2 of 2.)

در شکل 6-4 سطر 61:

با توجه به اینکه هر دو عملوند از نوع صحیح می‌باشند تقسیم به صورت صحیح انجام مـی‌گیـرد و از قسـمت اعشاری صرفنظر می‌گردد.

java قادر است عملیاتی را انجام دهـد کـه در آن عملونـدها از یـک نـوع باشـند در غیـر ایـن صـورت بایـد عملوندها به صورت ضمنی توسط جاوا و یا به صورت صریح توسط برنامه‌نویس به نوعی دیگر تبدیل گردند.

```
1  // Fig. 4.9: GradeBook.java
2  // GradeBook class that solves class-average program using
3  // sentinel-controlled repetition.
4  import java.util.Scanner; // program uses class Scanner
5
6  public class GradeBook
7  {
8     private String courseName; // name of course this GradeBook represents
9
10    // constructor initializes courseName
11    public GradeBook( String name )
12    {
13       courseName = name; // initializes courseName
14    } // end constructor
15
16    // method to set the course name
17    public void setCourseName( String name )
18    {
19       courseName = name; // store the course name
20    } // end method setCourseName
21
22    // method to retrieve the course name
23    public String getCourseName()
```

```
24      {
25         return courseName;
26      } // end method getCourseName
27
28      // display a welcome message to the GradeBook user
29      public void displayMessage()
30      {
31         // getCourseName gets the name of the course
32         System.out.printf( "Welcome to the grade book for\n%s!\n\n",
33            getCourseName() );
34      } // end method displayMessage
35
36      // determine the average of an arbitrary number of grades
37      public void determineClassAverage()
38      {
39         // create Scanner to obtain input from command window
40         Scanner input = new Scanner( System.in );
41
42         int total; // sum of grades
43         int gradeCounter; // number of grades entered
44         int grade; // grade value
45         double average; // number with decimal point for average
46
47         // initialization phase
48         total = 0; // initialize total
49         gradeCounter = 0; // initialize loop counter
50
51         // processing phase
52         // prompt for input and read grade from user
53         System.out.print( "Enter grade or -1 to quit: " );
54         grade = input.nextInt();
55
56         // loop until sentinel value read from user
57         while ( grade != -1 )
58         {
59            total = total + grade; // add grade to total
60            gradeCounter = gradeCounter + 1; // increment counter
61
62            // prompt for input and read next grade from user
63            System.out.print( "Enter grade or -1 to quit: " );
64            grade = input.nextInt();
65         } // end while
66
67         // termination phase
68         // if user entered at least one grade...
69         if ( gradeCounter != 0 )
70         {
71            // calculate average of all grades entered
72            average = (double) total / gradeCounter;
```

```
72          average = (double) total / gradeCounter;
73
74          // display total and average (with two digits of precision)
75          System.out.printf( "\nTotal of the %d grades entered is %d\n",
76             gradeCounter, total );
77          System.out.printf( "Class average is %.2f\n", average );
78       } // end if
79       else // no grades were entered, so output appropriate message
80          System.out.println( "No grades were entered" );
81    } // end method determineClassAverage
82
83 } // end class GradeBook
```

Fig. 4.9 | Sentinel-controlled repetition: Class-average problem. (Part 2 of 2.)

در شکل 9-4 سطر 72:

(double) این عملگر، عملگر تبدیل نوع یا type casting می‌باشد و نوع عملوند سمت راست خود را تغییر می‌دهد از این عملگر برای تبدیل نوع، به صورت صریح explicit استفاده می‌شود. با توجه به آنچه گفته شد جاوا عملیاتی را می‌تواند انجام دهد که نوع عملوندها یکسان باشد بنابراین در این دستور gradeCounter نیز باید به نوع اعشاری تبدیل گردد، این کار توسط Java و به صورت ضمنی implicit انجام می‌گیرد چون با از دست رفتن داده همراه نیست.

در شکل 15-4

++a ابتدا مقدار عملوند را یک واحد اضافه می‌کند سپس مقدار جدید عملوند در عبارت استفاده می‌شود.

a++ ابتدا مقدار فعلی عملوند در عبارت استفاده می‌شود سپس مقدار عملوند افزوده می‌شود.

Operator	Operator name	Sample expression	Explanation
++	prefix increment	++a	Increment a by 1, then use the new value of a in the expression in which a resides.
++	postfix increment	a++	Use the current value of a in the expression in which a resides, then increment a by 1.
--	prefix decrement	--b	Decrement b by 1, then use the new value of b in the expression in which b resides.
--	postfix decrement	b--	Use the current value of b in the expression in which b resides, then decrement b by 1.

Fig. 4.15 | Increment and decrement operators.

```
1  // Fig. 5.5: Sum.java
2  // Summing integers with the for statement.
3
4  public class Sum
5  {
6     public static void main( String args[] )
7     {
8        int total = 0; // initialize total
9
10       // total even integers from 2 through 20
11       for ( int number = 2; number <= 20; number += 2 )
12          total += number;
13
14       System.out.printf( "Sum is %d\n", total ); // display results
15    } // end main
16 } // end class Sum
```

```
Sum is 110
```

در شکل 5-5 سطر 11

دستور حلقهٔ for دارای 3 قسمت می‌باشد:

- **قسمت اول**: مقداردهی متغیر را انجام می‌دهد و فقط یک بار در ابتدای اجرای حلقهٔ for اجرا می‌شود.
- **قسمت دوم**: شرط کنترلی حلقه می‌باشد هر بار قبل از اجرای بدنهٔ حلقه شرط کنترلی بررسی شده و در صورت true بودن بدنه اجرا می‌شود.
- **قسمت سوم**: گام حرکتی می‌باشد و هر بار که اجرای حلقه تمام شد و به انتهای حلقهٔ for رسیدیم، به ابتدای حلقه for وارد می‌شویم ابتدا گام حرکتی اجرا شده سپس شرط حلقه بررسی، و در صورت true بودن همین روند تکرار می‌شود.

```java
// Fig. 5.6: Interest.java
// Compound-interest calculations with for.

public class Interest
{
   public static void main( String args[] )
   {
      double amount; // amount on deposit at end of each year
      double principal = 1000.0; // initial amount before interest
      double rate = 0.05; // interest rate

      // display headers
      System.out.printf( "%s%20s\n", "Year", "Amount on deposit" );

      // calculate amount on deposit for each of ten years
      for ( int year = 1; year <= 10; year++ )
      {
         // calculate new amount for specified year
         amount = principal * Math.pow( 1.0 + rate, year );

         // display the year and the amount
         System.out.printf( "%4d%,20.2f\n", year, amount );
      } // end for
   } // end main
} // end class Interest
```

```
Year   Amount on deposit
   1            1,050.00
   2            1,102.50
   3            1,157.63
   4            1,215.51
   5            1,276.28
   6            1,340.10
   7            1,407.10
   8            1,477.46
   9            1,551.33
  10            1,628.89
```

Fig. 5.6 | Compound-interest calculations with **for**.

توضیح شکل 5-6 برناهمه ای که موجودی حساب بانکی برای یک حساب را در پایـان هـر سـال و بـرای 10 سال چاپ می‌کند.

فرض شده است موجودی اولیه $100 می‌باشـد و سـود حاصـل دسـت نخـورده در حسـاب بـاقی مـی‌مانـد همچنین سود سالیانه 5% می‌باشد.

میزان موجودی a = p 1000$ اصل پول (1 + r نرخ سود)^n

p: موجودی اولیه حساب که 1000 دلار است.

r: نرخ سود سالیانه که 5% یا 0.05 می‌باشد.

n: شمارنده سال می‌باشد.

a: میزان موجودی در پایان سال nام می‌باشد.

سطر 13:

فیلدی (فضایی) به طول 20 کاراکتر در نظر گرفته می‌شود و رشته مورد نظر به صورت تراز شده، از راست چاپ می‌شود.

سطر 19:

کلاس Math یکی از کلاس‌های API جاوا است که در پکیج java.lang قرار دارد و شامل متدهایی است مانند pow (توان) و سایر متدها برای قدر مطلق، جذر، Sin، Cos و ... و به شکل زیر عمل می‌کند:

$$\text{Math.pow (a, b)} \xrightarrow{\text{یعنی}} a^b$$

سطر 22:

مشخص می‌کند که عدد باید به صورت پولی چاپ شود یعنی؛ سه رقم سه رقم جدا شود کاراکتر مورد استفاده برای جدا کردن از local سیستم مشخص می‌گردد. فضایی به اندازهٔ 20 کاراکتر در نظر گرفته می‌شود و عدد مورد نظر در سمت راست قرار می‌گیرد و 52 عدد را تا 2 رقم اعشار گرد می‌کند.

سطر 16 **شکل** (6-5)

متغیری که در دستور for تعریف می‌گردد، فقط در بدنهٔ همان دستور معتبر بوده و در قسمت‌های دیگر برنامه شناخته شده نیست.

```java
1  // Fig. 5.7: DoWhileTest.java
2  // do...while repetition statement.
3
4  public class DoWhileTest
5  {
6     public static void main( String args[] )
7     {
8        int counter = 1; // initialize counter
9
10       do
11       {
12          System.out.printf( "%d  ", counter );
13          ++counter;
14       } while ( counter <= 10 ); // end do...while
15
16       System.out.println(); // outputs a newline
17    } // end main
18 } // end class DoWhileTest
```

```
1  2  3  4  5  6  7  8  9  10
```

سطر 10:

مشابه حلقه while می‌باشد با این تفاوت که شرط در انتهای حلقه بررسی می‌شود، بنابراین بدنه حلقه حداقل یک بار اجرا می‌شود.

دستور switch: در واقع می‌تواند توسط if else if نیز پیاده‌سازی شود اما گاهی کوتاه‌تر و گویاتر بوه و می‌توان از آن استفاده کرد.

switch statement

```
import java.util.Scanner;

public class SwitchTest {

        public static void main(String args[]) {

                int grade;
                Scanner i = new Scanner(System.in);

                System.out.println("Enter a number between 0 and 10: ");
                grade = i.nextInt();

// controlling expression of the switch could be of type char, sort, byte, and int
                switch (grade) {
                case 10:
                case 9:
                case 8:
                        System.out.printf("%s %s \n", grade, "is a very good grade!");
                        break;

                case 7:
                case 6:
                        System.out.printf("%s %s \n", grade, "is a good grade!");
```



```
                        break;

                case 5:
                        System.out.printf("%s %s \n", grade, "is not bad!");
                        break;

                default:
                        System.out.printf("%s %s \n", grade, "is a bad grade!");
                        System.out.printf("%s\n", "try to do better next time!");
                        break;
                } // end of switch
        } // end of main method
} // end of class
```

برای استفاده از دستور سوئیچ عبارت داخل پرانتز بعد از سوئیچ یعنی عبارت مورد سنجش دستور سوئیچ فقط میتواند از جنس char، short، byte و int باشد البته در جاوای نسخه 1,7 میتوان از رشته هم استفاده نمود.

default: مشابه دستور else، در دستور if می‌باشد. و در صورتی که هیچ کدام از شرط‌ها درست نباشد این قسمت اجرا می‌شود.

دستور break: چنانچه این دستور در حلقهٔ while، for، do while یا switch اجرا شود، باعث خروج آنی از حلقه می‌شود.

دستور Continue:

این دستور زمانی که در حلقه‌های while، for و do while استفاده شود باعث می‌شود تا از اجرای ادامهٔ دستورات تا انتهای حلقه صرف نظر شود و دور جدیدی از حلقه تکرار شود. در حلقه‌های while و do while ابتدا شرط بررسی شده و سپس وارد حلقه می‌شویم، اما در حلقهٔ for ابتدا گام حرکت اجرا شده و سپس شرط بررسی می‌شود.

عملگرها در زبان جاوا :

معنی	عملگر
AND منطقی	**&**
OR منطقی	**|**
XOR منطقی	**^**
NOT منطقی	**!**
OR شرطی	**||**
AND شرطی	**& &**

هنگامی که از AND منطقی (&) یا OR منطقی (|) استفاده میشود ابتدا دو طرف این عملگرها ارزیابی شده و سپس از روی نتیجه ارزیابی کل عملگر مقدار میگیرد برای وضوح بیشتر مثال زیـر را مـورد بررسـی قـرار میدهیم.

```
if  (( a==20 ) & (b!= 10)) {
.........................
}
```

در این مثال ابتدا a==20 و b!= 10 هردو مورد بررسی قرار گرفته و سپس نتیجه کل در دستور if قرار میگیرد.

ولی در AND و OR شرطی ابتدا عبارت سمت چپی عملوند بررسی شده و در صورت لزوم؛ عبارت سمت راست بررسی میشود. در مثال زیر

```
if  (( a==20 ) && (b!= 10))
```

ابتدا a==20 مورد بررسی قرار گرفته اگر مقدار a برابر عدد 20 نباشد عبارت سمت راستی مورد بررسی قرار نمیگیرد چون در هر صورت عبارت اول نتجه AND را مشخص کرده است .

در هنگام استفاده از عملگر ^ " یا انحصاری" Exclusive OR تعداد عبارتهای True مورد بررسی نتیجه را مشخص میکنند اگر تعداد عبارتهای True فرد باشد نتیجه ارزیابی True میشود.

تعدادی از کلیدهای ترکیبی مورد استفاده در برنامه eclipse

Contl +space	تکمیل اتومات عبارتها
Contl + Shift + f	تنظیم فرورفتگی
Contl + /	کامنت کردن
Contl + Shift + /	کامنت نوع دوم
Contl + d	سطر را پاک می‌کند
Contl + Shift + p	آخر بلوک را مشخص می‌کند
Contl + Alt	کپی
Contl + L	پرش به سطر مورد نظر
Contl + J	جستجو

فصل ششم: نگاهی عمیق‌تر به متدها

تجربه نشان داده است که تنها راه نوشتن و نگهداری برنامه تقسیم کردن آن به بخش‌های کوچک‌تر بـه نـام ماجول است. این تکنیک تقسیم و غلبه divide and conquer نامیده می‌شود. سه نوع از ماجول‌ها در زبان جاوا وجود دارد: متدها، کلاس‌ها، پکیج‌ها. برنامه‌های جاوا با ترکیب کردن کلاس‌هـا و متـدهای نوشـته شـده توسـط کاربر کلاس و متدهای موجود در API جاوا و کلاس‌ها و متدهای نوشته شده توسط سایرین ایجاد می‌گردد.

دلایل استفاده از متدها برای ماجول کردن برنامه:

1) استفاده از تکنیک تقسیم و غلبه
2) قابلیت استفاده مجدد از نرم‌افزار: به این صورت که از متدهای قبلاً نوشته شده برای ایجـاد برنامـه‌هـای جدید استفاده می‌کنیم. به عنوان مثال برای خواندن از ورودی از کلاس scanner و متدهای موجـود در آن استفاده می‌کنیم.
3) جلوگیری از تکرار کد.

متدهای استاتیک

گاهی بهتر است تا متدهای پر استفاده را در صورت امکان به صورت استاتیک تعریف کـرد بـه عنـوان مثـال قبلاً با متد pow (توان) که در کلاس Math قرار دارد آشنا شدید.

برای تعریف متد به صورت استاتیک باید کلمه کلیدی static را قبل از نوع برگشتی متد قرار داد.

برای فراخوانی متدهای استاتیک نیازی به ساختن شـی از روی کـلاس وجـود نـدارد و مـی‌تـوان آنهـا را بـا استفاده از نام کلاس و به شکل زیر فراخوانی کرد:

Class Name.method Name (arguments)

* (بعضی از متدهای کلاس Math در جدول 6 chapter و 6.2 fig آمده است) *

(اگر متد باشد باید پرانتز جلوش باشد اما فیلد نه)

کلاس Math دارای دو فیلد به نام‌های Math.PI و Math.E می‌باشد که مقادیر عدد π و عدد نپر را در خود دارد. این فیلدها به صورت public، Final و static تعریف شده‌اند.

Public بودن باعث می‌شود که از سایر کلاس‌ها به آنها دسترسی داشته باشیم.

کلمهٔ کلیدی Final برای تعریف ثابت‌ها در زبان جاوا استفاده می‌شود.

مثال Final int num

و static باعث می‌شود که بتوان مستقیماً با نام کلاس آنها را صدا زد.