دانشکده مهندسی و علوم کامپیوتر

برنامه نویسی پیشرفته

وحیدی اصل

جلسه دوم- آشنایی با جاوا

مروری بر جاوا

- اولین بار توسط شرکت مایکروسیستم Sun (James Gosling) ارایه شده است.
- یک زبان شیئ گرای همه منظوره می‌باشد.
- مبتنی بر زبانهای C/C++ می‌باشد.
- در سال ۲۰۰۹ شرکت مایکروسیستم Sun توسط شرکت Oracle (اوراکل) خریداری شده و از آن پس، اوراکل متولی جاوا به حساب می‌آید.

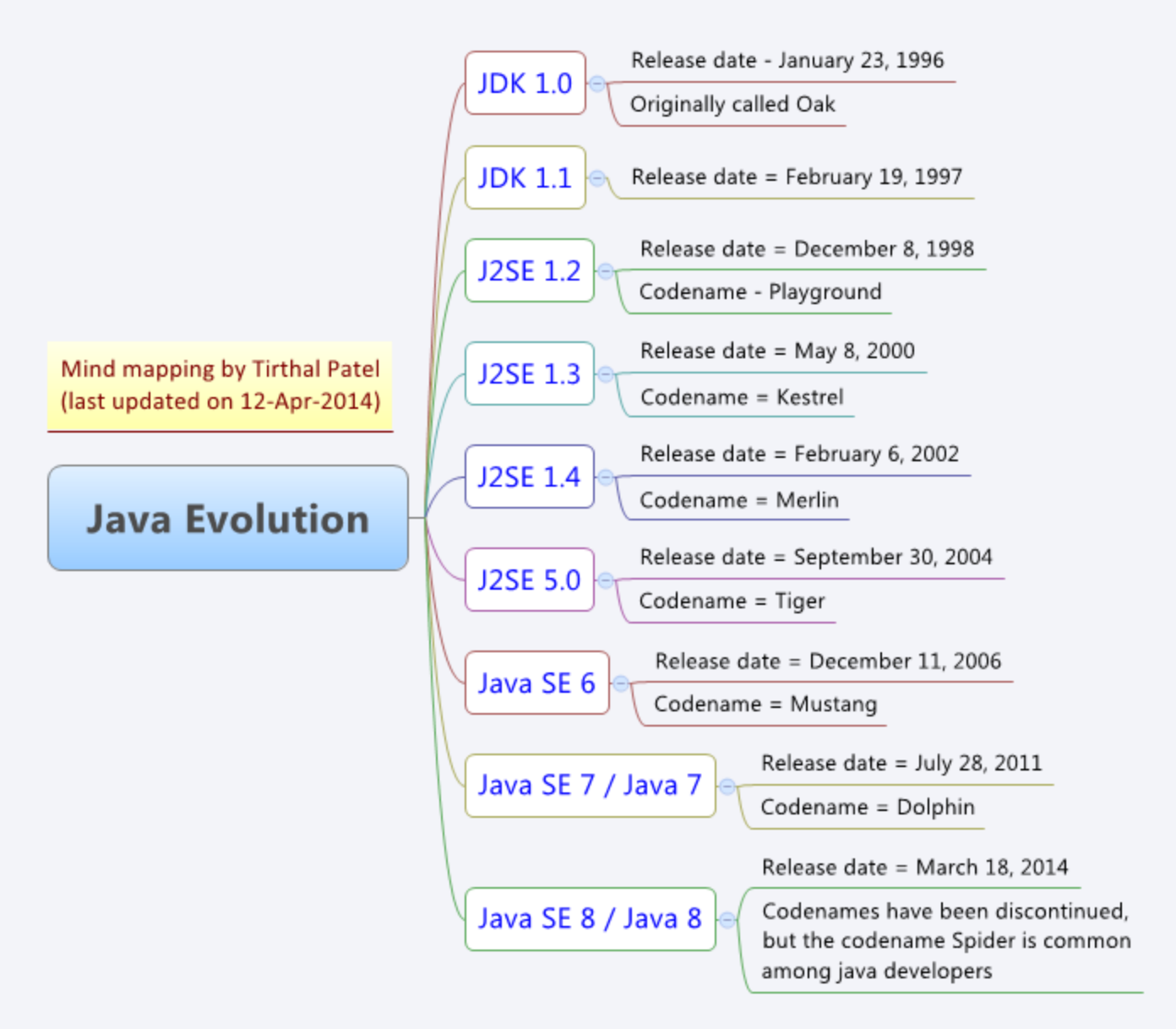
مروری برجاوا– جاوا چه زمانی متولد شد؟

1980 - C++ (as C with
1983 - Ada
1984 - Common Lisp
1984 - MATLAB
1985 - Eiffel
1986 - Objective-C
1986 - Erlang
1987 - Perl
1988 - Tcl
1988 - Mathematica
1989 - FL (Backus);

1990 - Haskell
1991 - Python
1991 - Visual Basic
1991 - HTML (Mark-up Language)
1993 - Ruby
1993 - Lua
1994 - CLOS (part of ANSI Common Lisp)
1995 - Java
1995 - Delphi (Object Pascal)
1995 - JavaScript
1995 - PHP
1996 - WebDNA
1997 - Rebol
1999 - D

2000 - ActionScript
2001 - C#
2001 - Visual Basic .NET
2002 - F#
2003 - Groovy
2003 - Scala
2003 - Factor
2007 - Clojure
2009 - Go
2011 - Dart

http://en.wikipedia.org/wiki/History_of_programming_languages


Mind mapping by Tirthal Patel
(last updated on 12-Apr-2014)
Java Evolution
JDK 1.0
Release date - January 23, 1996
Originally called Oak
JDK 1.1
Release date = February 19, 1997
J2SE 1.2
Release date = December 8, 1998
Codename - Playground
J2SE 1.3
Release date = May 8, 2000
Codename = Kestrel
J2SE 1.4
Release date = February 6, 2002
Codename = Merlin
J2SE 5.0
Release date = September 30, 2004
Codename = Tiger
Java SE 6
Release date = December 11, 2006
Codename = Mustang
Java SE 7 / Java 7
Release date = July 28, 2011
Codename = Dolphin
Java SE 8 / Java 8
Release date = March 18, 2014
Codenames have been discontinued, but the codename Spider is common among java developers

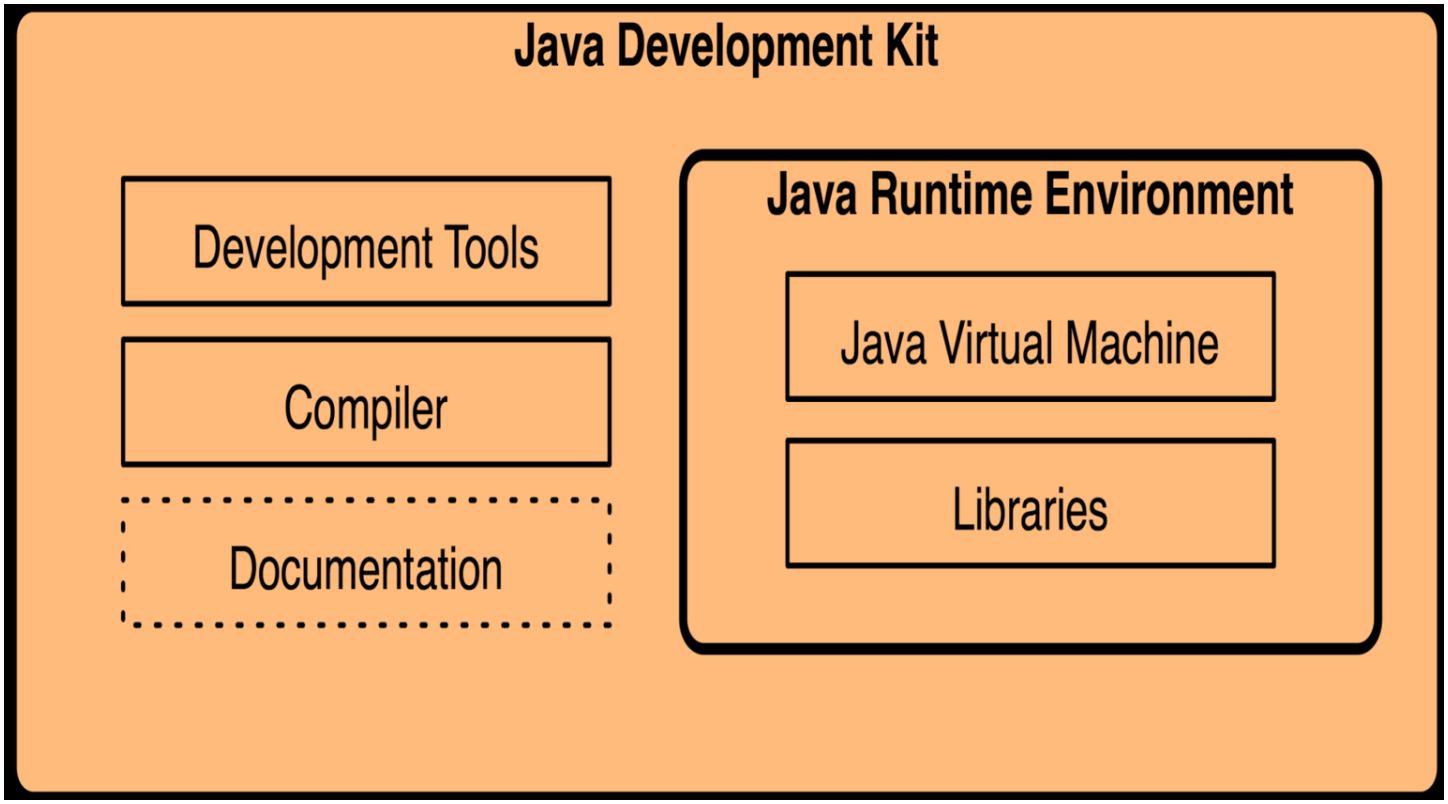
امروز چه می آموزیم؟

- با نحوه کامپایل و اجرای جاوا آشنا می‌شویم.
- یک برنامه ساده می‌نویسیم.
- با متغیرها و انواع ساده آشنا می‌شویم.
- با عملگرهای حسابی، رابطه ای و تقدم آنها آشنا می‌شویم.
- با ساختارهای اولیه این زبان آشنا می‌شویم.

مراحل تولید و اجرای یک برنامه جاوا

- در این بخش می خواهیم مراحلی را که برای ایجاد و اجرای یک برنامه جاوا در یک محیط توسعه جاوا موردنیاز است، بیان کنیم.
- برنامه های جاوا عموماً در طی ۵ فاز اجرا می‌شوند.
- این ۵ فاز عبارتند از نوشتن/ویرایش، کامپایل، بارکردن، درستی سنجی (verify) و اجرا
- ما این مراحل را در قالب Java SE Development Kit (JDK) تشریح می‌کنیم.

برای کامپایل و اجرای یک برنامه جاوا به jdk نیاز داریم



مراحل تولید و اجرای یک برنامه جاوا–نوشتن

- فاز اول: نوشتن یک برنامه

– در این فاز در یک نرم افزار ویرایشگر، فایل برنامه را ایجاد می کنیم. در ابتدا با استفاده از ویرایشگر (مثل wordpad) یک برنامه جاوا را درون فایل تایپ می کنیم (به این برنامه، کد منبع هم گفته می شود).

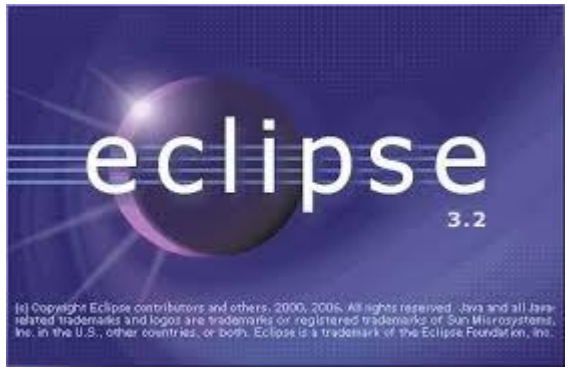
– سپس تصحیحات لازم را انجام داده و برنامه را برروی یک حافظه جانبی (مثلا در درایو D) ذخیره می کنیم.

– توجه داریم که نام فایل با پسوند .java ذخیره شده باشد.

```
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    } // end method main
11 } // end class Welcome3
```

مراحل تولید و اجرای یک برنامه جاوا- IDEها

- برای نوشتن، توسعه و اشکالزدایی برنامه ها اغلب از محیطهای توسعه یکپارچه (IDE) استفاده می شود.
- IDE ها ابزارهایی را به هدف پشتیبانی فرآیند توسعه نرم افزار فراهم می سازند.
- این ابزارها شامل ویرایشگرهایی جهت نوشتن و ویرایش برنامه ها، اشکالزداهایی (debuggers) برای یافتن خطاهای منطقی (خطاهایی که سبب می شوند برنامه به درستی کار نکند) هستند.

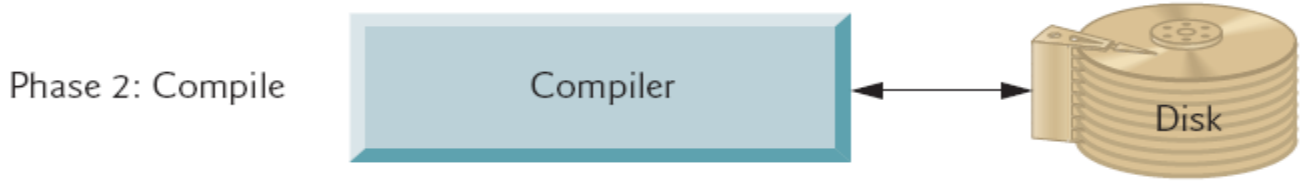


مراحل تولید و اجرای یک برنامه جاوا-کامپایل یک برنامه جاوا به بایت کد

- در فاز دوم با استفاده از دستور javac (کامپایلر جاوا) برنامه نوشته شده کامپایل می‌شود.
- برای مثال، برای کامپایل یک برنامه به نام Welcome.java به شکل زیر عمل می‌کنیم:

```
javac Welcome.java
```

- این دستور در پنجره command سیستم شما (command prompt) تایپ می‌شود.
- کامپایلر فایل برنامه شما را دریافت کرده و پس از کامپایل آن، فایلی با پسوند Welcome.class ایجاد می‌کند. این فایل حاوی نسخه کامپایل شده از برنامه خواهد بود.



مراحل تولید و اجرای یک برنامه جاوا–شروع کار JVM

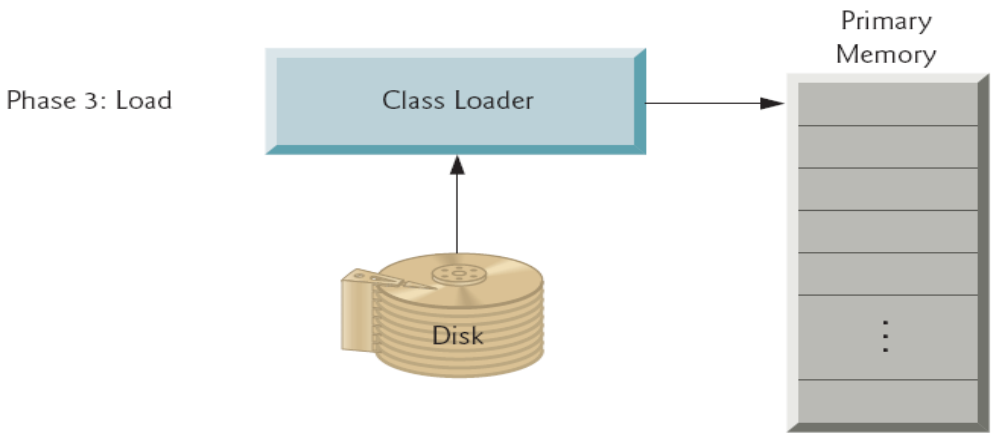
- بایت کدها به وسیله JVM اجرا می‌شوند.
- ماشین مجازی جاوا بخشی از JDK و فنداسیون سکوی جاوا (Java Platform) به حساب می‌آید.
- ماشین مجازی یک برنامه نرم افزاری است که یک کامپیوتر را شبیه سازی می کند، اما سیستم عامل و سخت افزار را که با او در تعامل است از دید کاربر پنهان می‌سازد.
- ماشین مجازی جاوا لایه ای است برروی لایه سیستم عامل که سبب می شود برنامه های نوشته شده به جاوا، دغدغه جزئیات لایه های پایینتر (سیستم عامل و سخت افزار) را نداشته باشند.
- JVM یکی از پرطرفدارترین ماشینهای مجازی است. مایکروسافت .NET نیز دارای ساختاری مشابه است.
- برخلاف ماشین مجازی که به سیستم عامل و سخت افزار وابسته هستند، بایت کدها مستقل از سکو به حساب می‌آیند.
- بنابراین یک بایت کد در سکوهای مختلف بدون تغییر قابل اجرا می‌باشد.
- آنها به سخت افزار خاصی وابستگی ندارند و لذا قابل حمل می‌باشند.

- برای فراخوانی ماشین مجازی به صورت زیر عمل می‌کنیم:

```
java Welcome
```

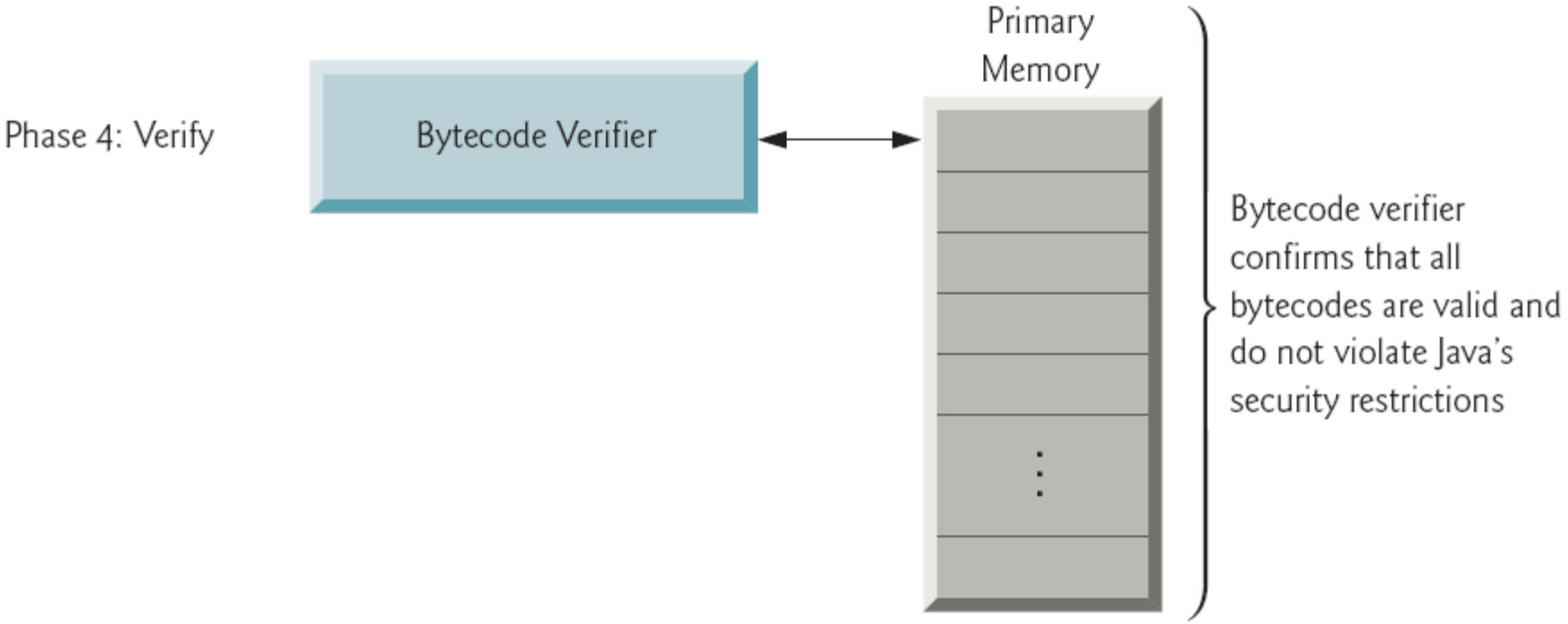
مراحل تولید و اجرای یک برنامه جاوا–بارکردن برنامه به حافظه

- در فاز سوم، JVM برنامه (بایت کدها) را در حافظه قرار می دهد تا آن را اجرا نماید.
- به این کار بارکردن loading می‌گویند.
- لودر کلاس JVM فایلهای .class ای را که حاوی بایت کدهای برنامه هستند گرفته و آنها را به حافظه اصلی (RAM) منتقل می‌کند.
- لودر کلاس همچنین همه فایلهای .class ای را که برنامه شما به آنها احتیاج دارد و در کتابخانه جاوا موجود است را به حافظه می‌برد.



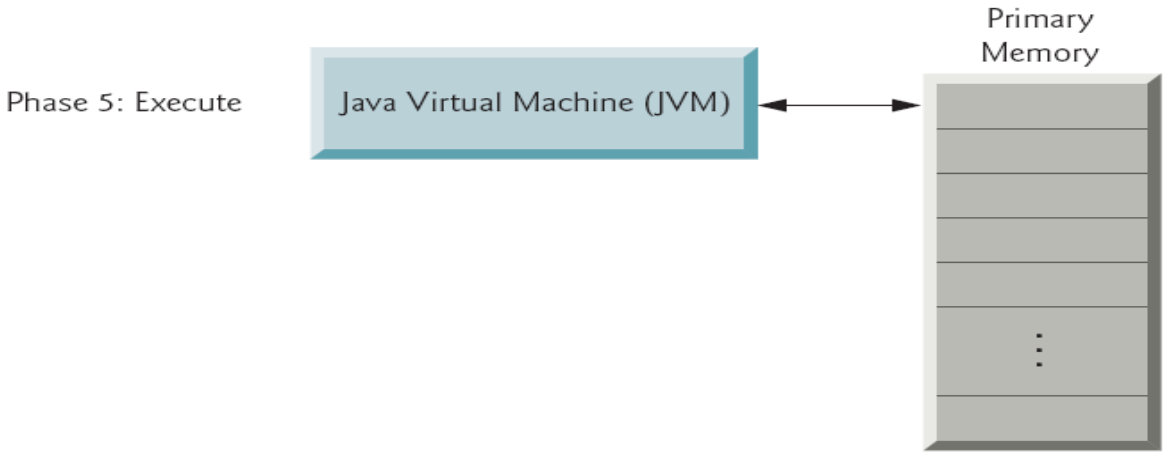
مراحل تولید و اجرای یک برنامه جاوا–درستی سنجی

- در مرحله ۴ بررسی می شود که بایت کدها قواعد امنیتی جاوا را خدشه دار نمی‌کنند.
- در صورت تخطی از اصول، خطاهای زمان اجرا (runtime errors) صادر خواهد شد.



مراحل تولید و اجرای یک برنامه جاوا-اجرا

- در فاز پنجم، JVM بایت کدهای برنامه را اجرا می‌کند.
- در نسخه های اولیه جاوا، JVM تنها یک مفسر ساده برای اجرای تک به تک بایت کدهای برنامه بود.
- این امر سبب کاهش سرعت زمان اجرا می شد، چون در هر لحظه فقط یک بایت کد تفسیر و اجرا می‌شد.
- معماری های جدیدتر کامپیوتر امکان اجرای همزمان چندین دستور را در هر لحظه فراهم می‌کنند.
- JVM های امروزی معمولا بایت کدها را به صورت ترکیبی از مفسر و کامپایل لحظه ای (just in-time-JIT) اجرا می‌کنند.
- در این فرآیند، JVM بایت کدها را در قالب تفسیری تحلیل می کند تا نقاط داغ را (قسمتهایی را که به دفعات اجرا می شوند) شناسایی کند.
- در این لحظه کامپایلر لحظه ای (JIT) فعال می شود و بایت کدها را به زبان ماشین کامپیوتر تبدیل می کند. قسمتهای داغ یکبار به زبان ماشین تبدیل می شوند و بارها اجرا می‌شوند.



```
C:\sample1>dir
 Volume in drive C is C
 Volume Serial Number is 04B8-352A

 Directory of C:\sample1

10/08/2010  04:53 PM    <DIR>          .
10/08/2010  04:53 PM    <DIR>          ..
10/08/2010  04:47 PM               113 First.java
               1 File(s)            113 bytes
               2 Dir(s)  22,242,201,600 bytes free

C:\sample1>javac First.java

C:\sample1>dir
 Volume in drive C is C
 Volume Serial Number is 04B8-352A

 Directory of C:\sample1

10/08/2010  04:54 PM    <DIR>          .
10/08/2010  04:54 PM    <DIR>          ..
10/08/2010  04:54 PM               412 First.class
10/08/2010  04:47 PM               113 First.java
               2 File(s)            525 bytes
               2 Dir(s)  22,242,201,600 bytes free

C:\sample1>java First
Salam!!!

C:\sample1>_
```

- یک برنامه ساده جاوا تنها از یک کلاس تشکیل شده است.
- نام کلاس با نام فایل برنامه برابر است.
- اسامی case sensitive هستند.
  - حرف اول اسامی کلاسها بهتر است حرف بزرگ (capital) باشد.
- کلاس حاوی یک متد main است.
- هرگاه برنامه اجرا شود، شروع اجرا از متد main خواهد بود.

نوشتن یک برنامه ساده به زبان جاوا–طرز تهیه!

- می خواهیم برنامه ای بنویسیم که یک رشته را در خروجی چاپ کند.
- برای اعلان تنها کلاس برنامه، پیش از نام کلاس از دو لغت public class استفاده کنید و در ادامه نام کلاس را بنویسید.
- آنچه در داخل کلاس می نویسید باید میان {و} قرار گیرد.
- کلاس حاوی یک متد main است.
  - کلمه main بعد از سه لغت public static void می آید.
  - آنچه در داخل هر متد مانند main می نویسید باید میان {و} قرار گیرد.
- در داخل پرانترهای main عبارت String[] args را بنویسید.
- در داخل متد main یک دستور بنویسید که نام شما را چاپ کند.
- نام شما در بین " و " قرار بگیرد و رشته حاصل در داخل پرانتز دستور زیر قرار می‌گیرد:

System.out.println() ;

- بعد از هر دستور غیر شرطی و غیرحلقه یک ; قرار می‌دهیم.

نوشتن یک برنامه ساده به زبان جاوا–برنامه فردی به نام Salam:

```
public class First {
    public static void main(String[] args) {
        System.out.println("Salam!!!");
    }

}
```

- یک شناسه دنباله ای از کاراکترها است که می تواند حاوی حروف، ارقام، خط زیرین (_) و نماد $ باشد.
- یک شناسه باید با یک حرف یا یک خط زیرین یا یک $ آغاز شود. نمی تواند با یک رقم شروع شود.
  - یک شناسه نمی تواند یک کلمه رزرو شده باشد.
- یک شناسه نمی تواند یکی از کلمات true، false یا null باشد.
- محدودیتی روی طول شناسه وجود ندارد.

لیست کلمات رزرو شده در جاوا

boolean	byte	char	double	float	int	long	short	public	private
protected	abstract	final	native	static	strictfp	synchronized	transient	volatile	if
else	do	while	switch	case	default	for	break	continue	assert
class	extends	implements	import	instanceof	interface	new	package	super	this
catch	finally	try	throw	throws	return	void	const	goto	enum

متغیر ها

- متغیر چیست؟
- متغیر مانند یک لیوان است (یک محفظه) که می تواند چیزی را در خود نگهداری کند.
- متغیر، قطعه ای از حافظه می باشد که داده ای را نگهداری می‌کند.
- مثلاً یک عدد، یک رشته یا یک مقدار بولین
- متغیرهای جاوا دارای اندازه مشخص هستند(مستقل از سخت افزار و سیستم عامل –سکو)
- در تعریف متغیر باید حتما نوع آن مشخص شده باشد، مانند

```
int x;
```

- یک متغیر باید حتماً پیش از استفاده مقداردهی شود.
  - برای مقدار دهی از انتساب (عملگر =) استفاده می‌شود. مثال:

```
x=2;
```

انواع در

- انواع اولیه (primitives)
- انواع ارجاعی (reference)

انواع اصلی (primitive) در جاوا

Type	Contains	Default	Size	Range
`boolean`	`true` or `false`	`false`	1 bit	NA
`char`	Unicode character	`\u0000`	16 bits	`\u0000` to `\uFFFF`
`byte`	Signed integer	0	8 bits	-128 to 127
`short`	Signed integer	0	16 bits	-32768 to 32767
`int`	Signed integer	0	32 bits	-2147483648 to 2147483647
`long`	Signed integer	0	64 bits	-9223372036854775808 to 9223372036854775807
`float`	IEEE 754 floating point	0.0	32 bits	±1.4E-45 to ±3.4028235E+38
`double`	IEEE 754 floating point	0.0	64 bits	±4.9E-324 to ±1.7976931348623157E+308

عملگرهای حسابی در جاوا

نماد	مفهوم	مثال	حاصل
+	Addition	34 + 1	35
-	Subtraction	34.0 - 0.1	33.9
*	Multiplication	300 * 30	9000
/	Division	1.0 / 2.0	0.5
%	Remainder	20 % 3	2

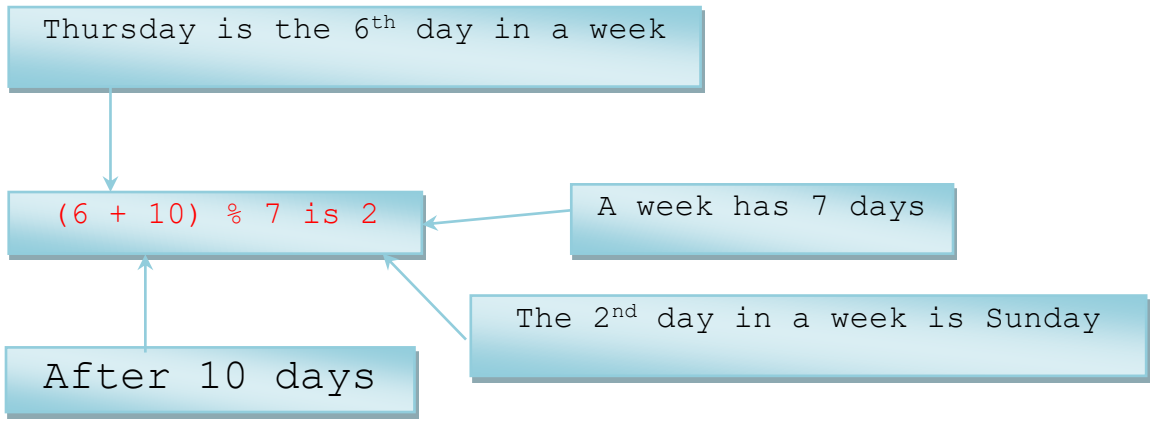
+, -, *, /, and %

5 / 2 → integer 2.

5.0 / 2 → double 2.5

5 % 2 → 1

- کاربردهای زیادی داره!
- مثلا فهمیدن اینکه عددی زوجه یا فرد
- حالا فرض کنید امروز پنجشنبه هست و می خواهید ۱۰ روز دیگه یه قرار با دوستانتون بگذارید.
- فرمولی ارایه کنید که روز هفته رو مشخص کنه:



Operator(s)	Operation(s)	Order of evaluation (precedence)
* / %	Multiplication Division Remainder	Evaluated first. If there are several operators of this type, they are evaluated from left to right.
+ -	Addition Subtraction	Evaluated next. If there are several operators of this type, they are evaluated from left to right.

- 1 + 2 * 3 = ?
  - is treated as 1 + (2 * 3)

```
3 + 4 * 4 + 5 * (4 + 3) - 1
```

$$\frac{3+4x}{5} - \frac{10(y-5)(a+b+c)}{x} + 9(\frac{4}{x} + \frac{9+x}{y})$$

تبدیل می‌شود به:

```
(3+4*x)/5 – 10*(y-5)*(a+b+c)/x + 9*(4/x + (9+x)/y)
```

عملگرهای تساوی و رابطه ای

Standard algebraic equality or relational operator	Java equality or relational operator	Sample Java condition	Meaning of Java condition
Equality operators			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y

- هرگاه دو عملگر دارای اولویت یکسان باشند، عبارت براساس شرکت پذیری ارزیابی می‌شود.

- x = y = z = 17
  - is treated as x = (y = (z = 17))
    - چون عملگر انتساب دارای شرکت پذیری راست به چپ است.
- 72 / 2 / 3
  - is treated as (72 / 2) / 3
    - چون عملگر تقسیم دارای شرکت پذیری چپ راست به است.

عملگر	مثال	معادل است با
+=	i += 8	i = i + 8
-=	f -= 8.0	f = f - 8.0
*=	i *= 8	i = i * 8
/=	i /= 8	i = i / 8
%=	i %= 8	i = i % 8

Operator	Name	Description
++var	preincrement	The expression (++var) increments var by 1 and evaluates to the *new* value in var *after* the increment.
var++	postincrement	The expression (var++) evaluates to the *original* value in var and increments var by 1.
--var	predecrement	The expression (--var) decrements var by 1 and evaluates to the *new* value in var *after* the decrement.
var--	postdecrement	The expression (var--) evaluates to the *original* value in var and decrements var by 1.

تاثیر استفاده از عملگرهای پیشوندی و پسوندی

```
int i = 10;
int newNum = 10 * i++;
```

Same effect as

```
int i = 10;
int newNum = 10 * (++i);
```

Same effect as

عملگر + برروی رشته (String)

- عملگر + برروی اشیایی از نوع String سربارگذاری شده است:
  - مشابه C++
  - این عملگر به ما امکان می دهد چند رشته را بهم هم بچسبانیم.
- اگر یک عبارت با یک String شروع شده باشد و بعد از آن از + استفاده کنیم. تمامی عملوندهای بعدی هم به رشته تبدیل خواهند شد. یک مثال:

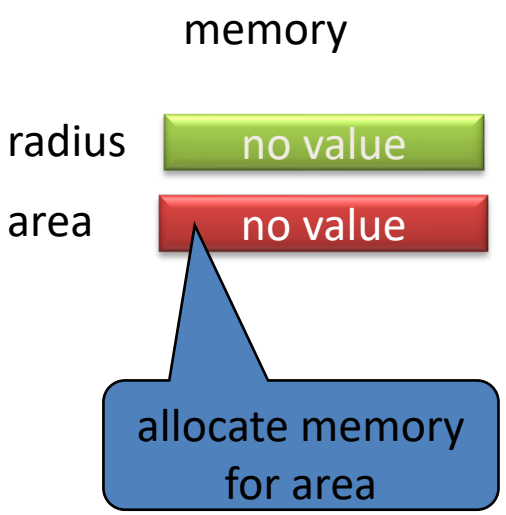
```
int x = 0, y = 1, z = 2;
String myString = “x, y, z ”;
System.out.println(myString + x + y + z);
```

برنامه ای بنویسید که براساس مقدار شعاع مساحت دایره را حساب کند:

```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double radius;
    double area;

    // Assign a radius
    radius = 20;

    // Compute area
    area = radius * radius * 3.14159;

    // Display results
    System.out.println("The area for the circle of radius " +
      radius + " is " + area);
  }
}
```

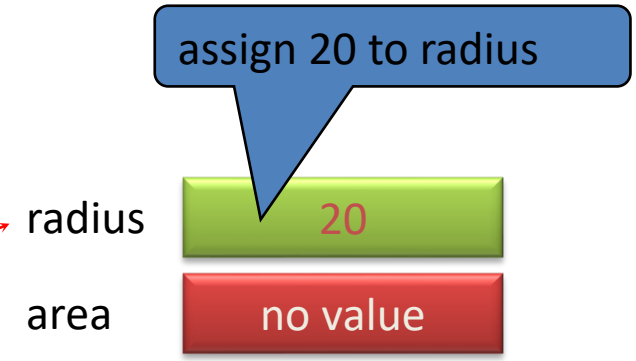
برنامه ای بنویسید که براساس مقدار شعاع مساحت دایره را حساب کند:

```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double radius;
    double area;

    // Assign a radius
    radius = 20;

    // Compute area
    area = radius * radius * 3.14159;

    // Display results
    System.out.println("The area for the circle of radius " +
      radius + " is " + area);
  }
}
```



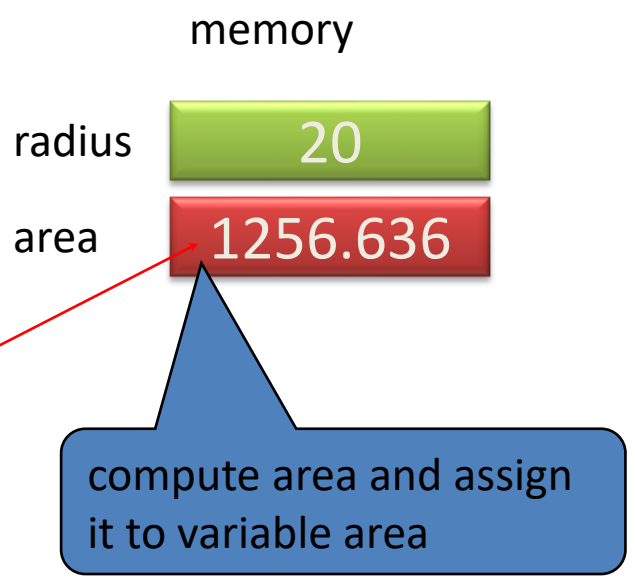
برنامه ای بنویسید که براساس مقدار شعاع مساحت دایره را حساب کند:

```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double radius;
    double area;

    // Assign a radius
    radius = 20;

    // Compute area
    area = radius * radius * 3.14159;

    // Display results
    System.out.println("The area for the circle of radius " +
      radius + " is " + area);
  }
}
```



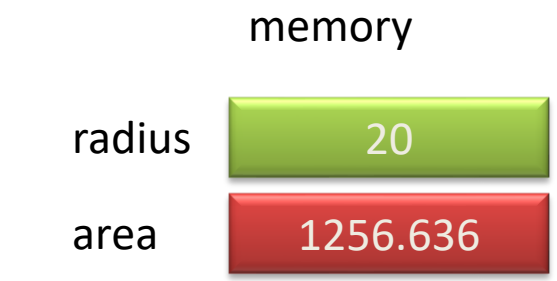
برنامه ای بنویسید که براساس مقدار شعاع مساحت دایره را حساب کند:

```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double radius;
    double area;

    // Assign a radius
    radius = 20;

    // Compute area
    area = radius * radius * 3.14159;

    // Display results
    System.out.println("The area for the circle of radius " +
      radius + " is " + area);
  }
}
```

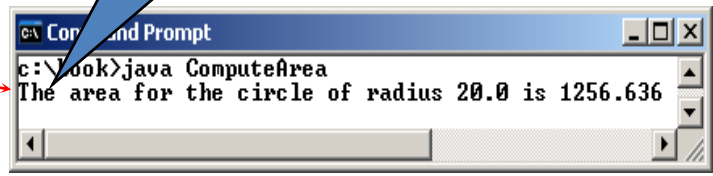


برنامه ای بنویسید که براساس مقدار شعاع مساحت دایره را حساب کند:

```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double radius;
    double area;

    // Assign a radius
    radius = 20;

    // Compute area
    area = radius * radius * 3.14159;

    // Display results
    System.out.println("The area for the circle of radius " +
      radius + " is " + area);
  }
}
```



print a message to the console



خواندن ورودی از کنسول (از کاربر)

- ابتدا باید کتابخانه java.util.Scanner; را در ابتدای برنامه import کنیم:
- سپس یک شیئ از نوع Scanner ایجاد می کنیم

```
Scanner input = new Scanner(System.in);
```

- از متدهای زیر برای خواندن مقادیری از انواع مختلف استفاده می کنیم:

next() برای خواندن یک رشته

nextByte() برای خواندن یک مقدار از نوع بایت

nextInt() برای خواندن یک مقدار از نوع int

nextShort() برای خواندن یک مقدار از نوع short

nextLong() برای خواندن یک مقدار از نوع long

nextFloat() برای خواندن یک مقدار از نوع float

nextDouble() برای خواندن یک مقدار از نوع double

nextBoolean() برای خواندن یک مقدار از نوع boolean

برای مثال

```
System.out.print("Enter a double value: ");
Scanner input = new Scanner(System.in);
double d = input.nextDouble();
```

برنامه ای بنویسید که ثانیه را از کاربر بگیرد و به دقیقه و ثانیه تبدیل کند.

```
import java.util.Scanner;

public class DisplayTime {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    // Prompt the user for input
    System.out.print("Enter an integer for seconds: ");
    int seconds = input.nextInt();

    int minutes = seconds / 60; // Find minutes in seconds
    int remainingSeconds = seconds % 60; // Seconds remaining
    System.out.println(seconds + " seconds is " + minutes +
      " minutes and " + remainingSeconds + " seconds");
  }
}
```

محاسبه مساحت با گرفتن اندازه شعاع از کاربر

```
import java.util.Scanner; // Scanner is in the java.util package

public class ComputeAreaWithConsoleInput {
  public static void main(String[] args) {
    // Create a Scanner object
    Scanner input = new Scanner(System.in);

    // Prompt the user to enter a radius
    System.out.print("Enter a number for radius: ");
    double radius = input.nextDouble();

    // Compute area
    double area = radius * radius * 3.14159;

    // Display result
    System.out.println("The area for the circle of radius " +
      radius + " is " + area);
  }
}
```

```
final datatype CONSTANTNAME = VALUE;

final double PI = 3.14159;
final int SIZE = 3;
```

دقت ممیز شناور- چند نکته

- محاسبات برروی انواع اعشاری ممیز شناور تقریبی و نه قطعی هستند.
  - چون این اعداد با دقت کامل در حافظه ذخیره نمی شوند!
  - برای مثال:

```
System.out.println(1.0 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1);
```

- مقدار 0.5000000000000001 را چاپ می کند نه 0.5 را
- یا

```
System.out.println(1.0 - 0.9);
```

- مقدار 0.09999999999999998را چاپ می کند نه 0.1 را
- مقادیر صحیح (integer) دقیق ذخیره میشوند.
- در نتیجه عملیات روی آنها به نتایج دقیق ختم می شود.

یک لیترال مقداری ثابت است که مستقیماً در برنامه ظاهر می شوند.

برای مثال ۴۱، ۲۳۰۹۹۹و ۵.۰ در دستورات زیر لیترال هستند.

```
int i = 41;

long x = 230990;

double d = 5.0;
```

- لیترالهای ممیز شناور با نقطه ممیز نوشته می شوند.
- به طور پیش فرض از نوع double درنظر گرفته می شود.
- برای مثال، 5.0 یک double فرض می شود نه یک float
- اگر می خواهیم مقدار float در نظر گرفته شود باید حرف f یا F را در انتهای آن تایپ کنیم. مثال:
  - 100.2f یا 100.2F
- اگر بخواهیم مقداری double در نظر گرفته شود باید D یا d را به آن اضافه کنیم. مثال:
  - 100.2d یا 100.2D

- لیترالهای ممیز شناور می توانند به صورت نماد علمی هم بیان شوند.
- برای مثال 1.23456e+2 معادل است با 123.456
- یا 1.23456e-2 معادل است با 0.0123456
- نماد E یا e توان را مشخص می کند و می تواند به صورت کوچک یا بزرگ نوشته شود.

تبدیل نوع عددی (numeric type conversion)

- دستورات زیر را درنظر بگیرید:

```
byte i = 100;
long k = i * 3 + 4;
double d = i * 3.1 + k / 2;
```

تبدیل نوع عددی (numeric type conversion)

- هرگاه بخواهیم عملیاتی برروی دو عملوند از دو نوع مختلف انجام دهیم، جاوا به صورت خودکار عملوند را طبق قوانین زیر به نوع مناسب تبدیل می کند:

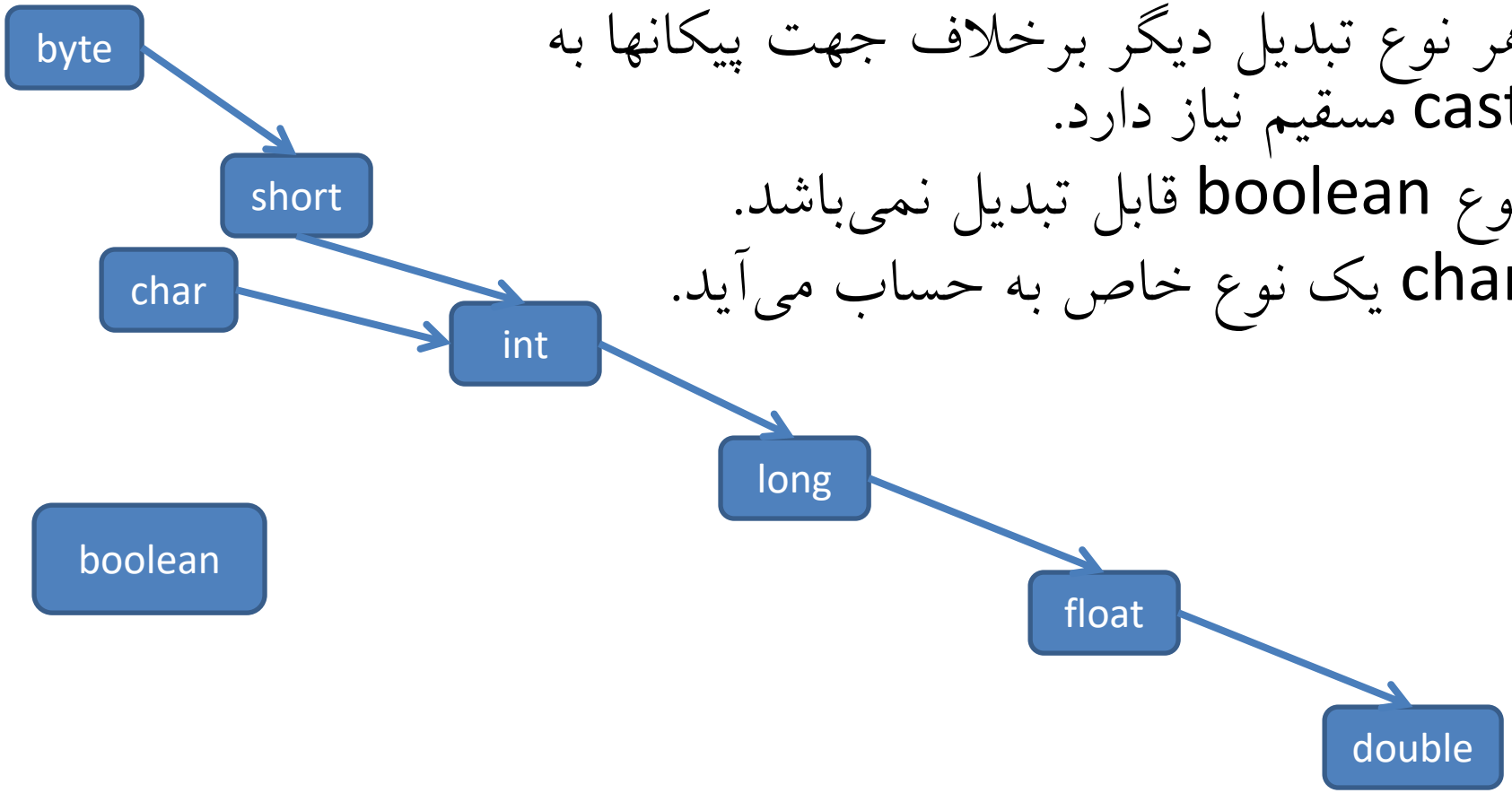
1. اگر یکی از عملوندها از نوع double باشد، دیگری نیز به double تبدیل می‌شود.
2. درغیراینصورت، اگر یکی float باشد، دیگری هم به float تبدیل می‌شود.
3. درغیراینصورت، اگر یکی long باشد، دیگری به long تبدیل می‌شود.
4. درغیراینصورت، هردو به int تبدیل می‌شوند.

توجه کنید که سرریزی رخ ندهد!

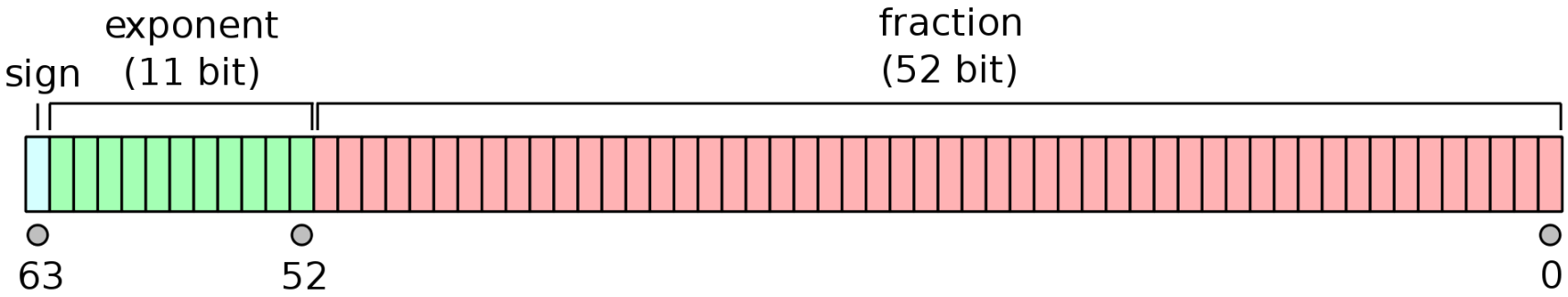
- شما نمی توانید مقداری بزرگتر از اندازه یک لیوان در آن قرار دهید.
- اگر این کار را انجام دهید، ممکن است بخشی از داده را از دست بدهید.
- مثال

```
int x=1000;
byte y=x;
```

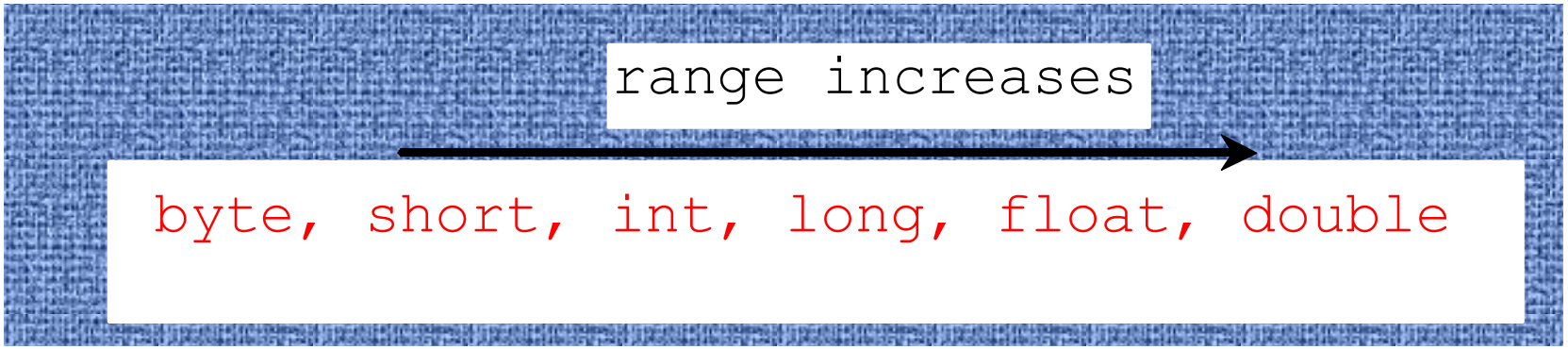
- تبدیلهایی که در جهت پیکانها انجام شوند، توسط جاوا وخودکار انجام می‌شوند.
- پیکانها خاصیت تعدی دارند.
- هر نوع تبدیل دیگر برخلاف جهت پیکانها به cast مسقیم نیاز دارد.
- نوع boolean قابل تبدیل نمی‌باشد.
- char یک نوع خاص به حساب می‌آید.



(numeric type conversion) تبدیل نوع عددی




```
Implicit casting
  double d = 3; (type widening)

Explicit casting
  int i = (int)3.0; (type narrowing)
  int i = (int)3.9; (Fraction part is
  truncated)
```

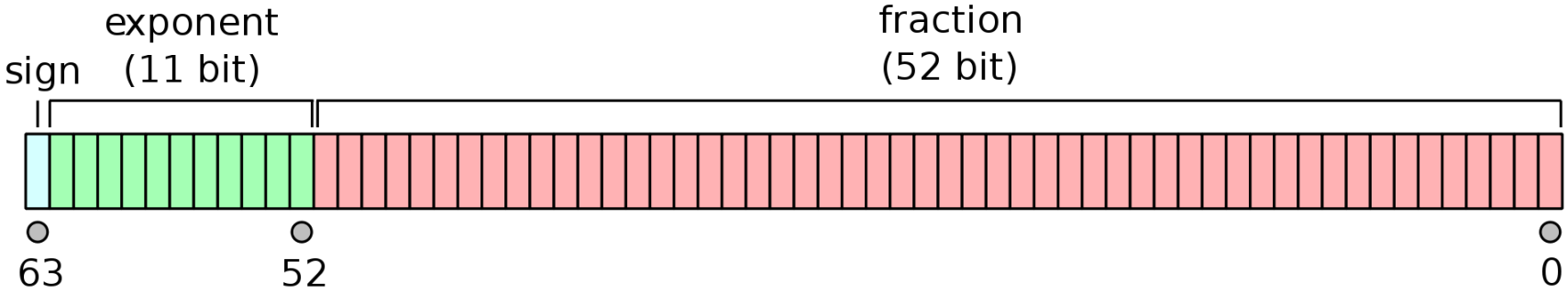


جدول تبدیل نوع

Convert From:	Convert To: boolean	byte	short	char	int	long	float	double
boolean	-	N	N	N	N	N	N	N
byte	N	-	Y	C	Y	Y	Y	Y
short	N	C	-	C	Y	Y	Y	Y
char	N	C	C	-	Y	Y	Y	Y
int	N	C	C	C	-	Y	Y*	Y
long	N	C	C	C	C	-	Y*	Y*
float	N	C	C	C	C	C	-	Y
double	N	C	C	C	C	C	C	-

- N: تبدیل نمی‌تواند انجام شود.
- Y: تبدیل به طور خودکار و توسط جاوا انجام می‌شود.
- C: تبدیل کوتاه‌کننده (درخلاف جهت پیکانها) است و به cast مستقیم نیاز است.
- Y*: تبدیل نوع تعریض‌کننده است، اما ارقام کم‌ارزش‌تر ممکن است براثر تبدیل، از بین بروند.

مثالهایی از تبدیل




```
i = 123456789; //a big integer
f = i; //f stores and approximation of i
System.out.println(f);//output :
  1.23456792E8
i = (int) f;
System.out.println(i); //output :
  123456792
```

- floating-point types are **approximations of numbers**
- They cannot always hold as many significant digits as the integer types

ارقام در مبنای ۱۶ هستند

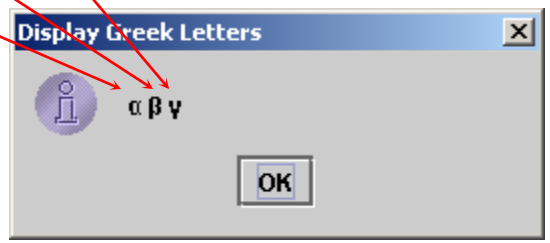
```
char letter = 'A'; (ASCII)
char numChar = '4'; (ASCII)
char letter = '\u0041'; (Unicode)
char numChar = '\u0034'; (Unicode)
```

نکته: توجه داشته باشید که عملگرهای افزایشی و کاهشی می توانند برروی متغیرهای نوع char نیز اعمال شوند تا کارکتر بعدی یا قبلی Unicode را تولید کنند. برای مثال دستور زیر کاراکتر b را نمایش می دهد.

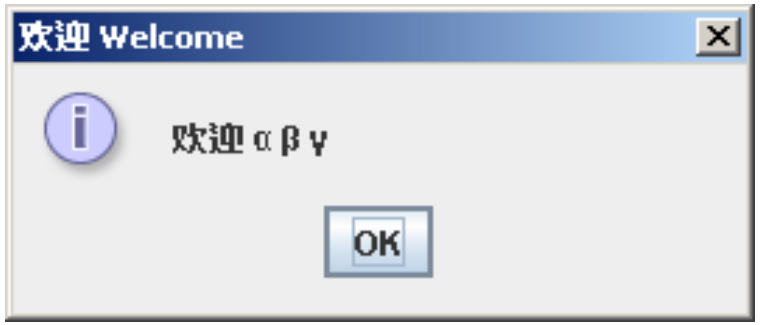
```
char ch = 'a';
        System.out.println(++ch);
```

- کاراکترهای جاوا از *Unicode* استفاده می‌کنند.
- *Unicode* یک فرمت ۱۶بیتی است که از سوی کنسرسیوم یونیکد ارایه شده است تا تبادل، پردازش و نمایش متون به زبانهای مختلف جهان را پشتیبانی کند.
- کدهای یونیکد از دوبایت (۱۶بیت) تشکیل شده‌اند که پیش از کد، نماد \u قرار می‌گیرد.
- این ۱۶ بیت در مبنای ۱۶ با اعدادی چهار رقمی مشخص می‌شوند که محدوده ای بین '\u0000' تا '\uFFFF' را در برمی‌گیرد. در نتیجه قادر به نمایش حدود 65535 کاراکتر می‌باشد.
- مجموعه کاراکترهای ASCII زیرمجموعه‌ای از Unicode است که کاراکترهای \u0000 تا \u007f را در بر می‌گیرد.

Unicode \u03b1 \u03b2 \u03b3 for three Greek letters



برنامه‌ای بنویسید که دو کاراکتر چینی و سه کاراکتر یونانی نمایش دهد!



تبدیل (cast) بین نوع کاراکتر و انواع عددی

```
int i = 'a'; // Same as int i = (int)'a';

char c = 97; // Same as char c = (char)97;
```

مجموعه کاراکترهای ASCII زیرمجموعه‌ای از UNICODE بین \u0000 تا \u007f هستند

TABLE B.1 ASCII Character Set in the Decimal Index

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
1	nl	vt	ff	cr	so	si	dle	dcl	dc2	dc3
2	dc4	nak	syn	etb	can	em	sub	esc	fs	gs
3	rs	us	sp	!	″	#	$	%	&	’
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	‘	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~	del		


The End