

**Less is more ...**

- **Effective**
- **Attractive**
- **Impactive**

# Introduction to Java Programming and Data Structures

Presented to:

Amol University of Special Modern Technologies

**Vahid.khazaeinezhad@gmail.co**

**m**

# Introduction to Java Programming and Data Structures

## فصل دوم

### Elementary Programming

# اهداف فصل

- نوشتن برنامه های جاوا که محاسبات ساده ای انجام بدهند
- دریافت ورودی از کاربر با استفاده از کلاس `Scanner`
- استفاده از `Identifiers` برای نام گذاری متغیرها، ثابت ها، متدها و کلاس ها
- استفاده از متغیرها برای ذخیره سازی مقادیر
- معرفی و کار با انواع پایه (`Primitive Data Types`)
- نوشتن برنامه ای برای ارزیابی عبارات (`Expression`)
- کار با ساعت سیستم
- شرح فرایند توسعه نرم افزار با برنامه پول خرد کن

# 1-2) مقدمه

- تمرکز این فصل روی یادگیری تکنیک های پایه برنامه نویسی برای حل مساله است
- در طول این فصل مقدمات کار با داده های اولیه، متغیرها، ثابت ها، اپراتورها، عبارات و ورودی و خروجی شرح داده می شود

## 2-2) نوشتن یک برنامه ساده

- برنامه نویسی شامل طراحی یک استراتژی برای حل مساله و سپس پیاده سازی آن با استفاده از یک زبان برنامه نویسی است.
- برنامه ساده محاسبه مساحت یک دایره را در نظر می گیریم.
- برای این کار ابتدا یک الگوریتم طراحی می کنیم و سپس الگوریتم را به دستورات / کدهای برنامه تبدیل (ترجمه) می کنیم
- الگوریتم گام هایی که برای حل مساله باید دنبال بشود را لیست می کند.
- الگوریتم امکان برنامه ریزی برای برنامه نویس پیش از شروع را می دهد
- الگوریتم می تواند به زبان طبیعی، شبه کد (Pseudo Code) (به معنی ترکیب زبان طبیعی با کدهای برنامه نویسی)
- الگوریتم محاسبه مساحت دایره:

1. شعاع دایره را بخوان

2. با استفاده از فرمول رو برو مساحت را محاسبه کن

3. نتیجه را نمایش بده

مساحت = شعاع \* شعاع \*

π

## 2-2) نوشتن یک برنامه ساده

```
1  /*
2   * This program is wrote for educational purpose
3   * Advance Programmin Course, AUSMT
4   */
5  package computecirclearea;
6
7  /**
8   * This program calculates the area of an
9   * arbitrary circle, at first read its radius
10  * and then presents the result *
11  *
12  * @author Khazaeinezhad_Vahid
13  */
14  public class ComputeCircleArea {
15
16     /**
17      * @param args the command line arguments
18      */
19     public static void main(String[] args) {
20         // 1) Read in radius
21         // 2) Compute area
22         // 3) Dispaly the result
23
24     } //end of main() method
25
26 } //end of class
27
```

- در گام اول نیاز است که مقدار شعاع را از کنسول ورودی (کیبورد) بخوانیم

- برای این کار نیاز به یک **متغیر** داریم

- متغیر مقداری را که در حافظه ذخیره شده نمایه می کند.

- برای نام گذاری متغیر از یک عبارت **توصیفی** استفاده کنید (از نام گذاری  $x,y$  ... اجتناب کنید)

- برای شعاع از **radius** و برای مساحت از **area** استفاده می کنیم

- برای این که کامپایلر متغیرها را بشناسد باید آن ها را **اعلان کنیم (Declaration)**

- باید **نوع و نام متغیر** مشخص بشود

- به همین خاطر جاوا یک زبان **static Typed** است.

- هر متغیر **نوع (Type)**، نام و **مقدار (Value)** دارد.

## 2-2) نوشتن یک برنامه ساده

```
14 public class ComputeCircleArea {
15
16     /**
17     * @param args the command line arguments
18     */
19     public static void main(String[] args) {
20         // 1) Read in radius
21         //1-1)
22         double radius; ◀ Declaration
23         double area;
24         //1-2)
25         radius = 30; ◀ Assignment
26         // 2) Compute area
27         area = radius * radius * 3.141596; ◀ Concatenation
28         // 3) Display the result
29         System.out.println("The area of circle with radius " + radius + " are " + area);
30     }
31 }
```

Console

run:

The area of circle with radius 30.0 are 2827.4364  
BUILD SUCCESSFUL (total time: 0 seconds)



## 2-3) دریافت ورودی از کنسول

```
16 public class ComputeCircleArea {
17
18     /**
19     * @param args the command line arguments
20     */
21     public static void main(String[] args) {
22         // 1) run:
23         // 1) Enter radius:
24         // 1) 30.0
25         double area = 2827.4364;
26         Scanner input = new Scanner(System.in);
27         // 1) Enter radius:
28         System.out.println("Enter radius:");
29         radius = input.nextDouble();
30         // 2) Compute area
31         area = radius * radius * 3.141596;
32         // 3) Display the result
33         System.out.println("The area of circle with radius " + radius + " are " + area);
34
35     } //end of main() method
36
37 } //end of class
```

# 3-2) نکته های برنامه

- کنسول استاندارد خروجی **مونیتر** است که استفاده از System.out می توانیم با آن کار کنیم.
- کنسول استاندارد ورودی **کیبورد** است که می توانیم با استفاده از System.in با آن کار کنیم.
- برای استفاده از کنسول استاندارد ورودی می توانیم از کلاس Scanner بهره بگیریم.
- دستور new Scanner (System.in) یک شی ایجاد می نماید.

- دستور Scanner input یک متغیر با نام input از نوع کلاس Scanner می سازد.
- یک رفرنس از شی ساخته شده به متغیر ایجاد شده انتساب داده می شود.
- متد nextDouble در کلاس Scanner تعریف شده است.
- کلاس Scanner در بسته java.util (Package) قرار دارد.
- برای استفاده از بسته های از پیش تعریف شده ابتدا آن ها را import کنیم.

• import java.util.Scanner;

- دستور import داخل بسته جاری و خارج از همه ی بلوک ها نوشته می شود.

```
1  /*
2   * This program is wrote for educational purpose
3   * Advance Programmin Course, AUSMT
4   */
5  package computecirclearea;
6
7  import java.util.Scanner;
8
9  /**
10 * This program calculates the area of an
11 * arbitrary circle, at first read its radius
12 * and then presents the result *
13 *
14 * @author Khazaeinezhad_Vahid
15 */
16 public class ComputeCircleArea {
17
```

# مثال 1) برنامه بنویسید که سه عدد از ورودی گرفته و میانگین آن ها نمایش دهد.

```
19 public static void main(String[] args) {
20     //1) Declaring all variables
21     Scanner input = new Scanner(System.in);
22     double number1;
23     double number2;
24     double number3;
25     double average;
26
27     System.out.print("Please enter first number:");
28     number1 = input.nextDouble();
29     System.out.print("Please enter second number:");
30     number2 = input.nextDouble();
31     System.out.print("Please enter third number:");
32     number3 = input.nextDouble();
33
34     average = (number1 + number2 + number3) / 3;
35
36     System.out.println("For these 3 [" + number1 + ", " + number2 +
37         ", " + number3 + "] numbers the average are " + average);
38
39 } //end of main() method
40
41
42
43 } //end of class
44
```

run:  
Please enter first number:10  
Please enter second number:11  
Please enter third number:12  
For these 3 [10.0, 11.0, 12.0] numbers the average are 11.0  
BUILD SUCCESSFUL (total time: 10 seconds)

# 2-4) شناسه ها (IDENTIFIERS)

- شناسه ها نام هایی هستند که برای شناسایی عناصری مانند کلاس ها و متدها و متغیرها در برنامه مورد استفاده قرار می گیرند.
- Number1, input, main, computeAverage, ComputeAverage
- قوانین نام گذاری شناسه ها:
- شناسه دنباله ای از کاراکترها شامل حروف، عدد و `under_score` و `dollar sign` (\$) می تواند باشد.
- یک شناسه می تواند با `underscore`, \$ و حرف شروع بشود.
- نمی تواند با عدد شروع بشود
- یک شناسه نمی تواند یک کلمه رزرو شده باشد.
- شناسه نمی تواند `true`, `false`, `null` باشد.
- یک شناسه هر طولی (تعداد کاراکتری) می تواند داشته باشد.
- `ComputeArea`, `print`, `_all`, `$2` نام های مجاز هستند.
- `2a`, `c+d` نام های غیر مجاز هستند
- کامپایلر وظیفه تطبیق نام ها با قوانین نام گذاری را بر عهده دارد و در صورت تخطی از قوانین تولید خطای نحوی (Syntax Error) می کند.

# (2-4) شناسه ها (IDENTIFIERS)

- توصیه ها:
- به منظور افزایش خوانایی برنامه از نام های توصیف کننده استفاده کنید
- نام ها در بردارنده کلمات معنی دار و کامل و توصیفی باشند. (Self\_Documented)
- از به کار بردن نام های اختصاری بپرهیزید.
- از کاراکتر \$ برای نام گذاری استفاده نکنید.
- به عنوان یک قرارداد این کاراکتر برای نام گذاری در سورس کد های تولید شده توسط ماشین استفاده می شود.

# متغیرها (2-5) (VARIABLES)

- متغیرها برای نمایه کردن مقادیری که ممکن است در طول برنامه تغییر کنند استفاده می شوند.
- در جاوا متغیرها می توانند یک نوع از قبل اعلان شده را نمایه کنند (Static Typed)
- پیش از استفاده از یک متغیر نام آن و نوعی که می تواند در خود ذخیره کند باید به کامپایلر اطلاع داده شود (Declaration)
- اعلان متغیر به کامپایلر می گوید که فضای مناسب را در حافظه برای ذخیره سازی متغیر تخصیص بدهد.
- انواع اولیه (Primitive) در جاوا:

```
int count;  
double radius;  
double interestRate;
```

```
int i, j, k;
```

```
int i = 0;
```

```
int i = 0, j = 0; //shorthand
```

• byte, char, int, double, float, short, long, Boolean

• مقدار دهی اولیه متغیر (Initialization):

• در زمان اعلان متغیر مقدار دهی هم انجام بشود.

• محدوده متغیر (Scope):

• محدوده متغیر قسمتی از برنامه است که در آن می توان به متغیر ارجاع کرد.

• هر متغیر یک محدوده دارد.

• نکته

• سعی کنید همواره زمان اعلان متغیر آن را مقدار دهی هم بکنید.

• باعث افزایش خوانایی و جلوگیری از تولید خطا می شود.

# 2-6) دستور مقدار دهی و عبارت مقدار دهی

## ASSIGNMENT STATEMENT

## ASSIGNMENT EXPRESSION

- پس از اعلان متغیر می توان یک مقدار (Value) به آن نسبت داد.
- در جاوا از اپراتور (=) برای نسبت دهی مقدار به متغیر استفاده می شود.
- اگر طرف دوم اپراتور = یک مقدار محاسباتی (شامل مقدار، اپراتور و متغیر) باشد به آن عبارت نسبت دهی گفته می شود.
- به طور کلی در جاوا (++C) هر نوع دستوری (Statement) یک عبارت (Expression) است.
- در نسبت دهی **چندگانه** باید دقت نمود که نوع متغیرهای سمت راست اپراتور با نوع متغیرها سمت چپ اپراتور سازگار باشد

```
int i = 0;  
int j = 10;  
i = (j*j) / 2;
```



```
int x,y, z;//Declaration  
  
x = y = z = 2; //Assignment  
  
System.out.println(x); // x value is 2  
  
z = (x + y ) / (2 * z);//Assignment Expression  
  
System.out.println(z); // z value is 1
```

# 7-2) ثابت‌ها (CONSTANTS)

- ثابت‌های نام‌دار شناسه‌هایی هستند که مقادیر دائمی / ثابت را نمایه می‌کنند.
- مقدار متغیرها در زمان اجرای برنامه ممکن است تغییر بکند.
- ثابت‌های نام‌دار و یا ساده هرگز تغییر نمی‌کنند.
- یک ثابت به عنوان Final Variable نیز در جاوا شناخته می‌شود.
- ثابت‌ها باید هم‌زمان با اعلان مقدار دهی اولیه (initialize) بشوند.
- کلیدواژه تعریف یک ثابت final است.
- به عنوان یک قرارداد ثابت‌ها با حروف بزرگ (تمامی حروف) نام‌گذاری می‌شوند.

```
final double PI = 3.141596;
```





# 8-2) قرارداد نام گذاری (NAMING CONVENTION)

- رعایت قرارداد نام گذاری (به نوعی استاندارد) **خوانایی** برنامه را افزایش می دهد و احتمال بروز خطاها کاهش می دهد.
- اطمینان حاصل کنید که نام توصیفی ساده و **قابل فهم** برای متغیرها، ثابت ها، متدها و کلاس های خود انتخاب کرده اید.
- جاوا یک زبان Case Sensitive است.

## • قواعد نام گذاری:

- برای نام گذاری متغیرها و متدها از **حروف کوچک** استفاده کنید. `radius, area`
- اگر نام شما از **چند کلمه** تشکیل شده است **اولین** کلمه با **حروف کوچک** و **اولین** حرف **باقی** کلمات با **حروف بزرگ** نوشته بشوند. (camelCase)
- **اولین** حرف هر کلمه بکار رفته در نام کلاس را با **حروف بزرگ** بنویسید `Circle`
- برای نام گذاری **ثابت** های به طور **کامل** از **حروف بزرگ** استفاده کنید `SIZE, RATE`
- در صورتی که نام ثابت **متشکل** از **چند کلمه** است بین آن ها آن ها `underscore ( _ )` قرار دهید. `SIZE_OF_STUDENTS`
- از به کار بردن نام های استفاده شده در کتابخانه جاوا **پرهیز** کنید. `print`
- در نام گذاری **متدها** اولین کلمه **فعل** باشد. `computeArea, printMyValue`

## 9-2) انواع عددی و اپراتورها

```
nextByte()  
nextShort()  
nextInt()  
nextLong()  
nextFloat()  
nextDouble()
```

```
Scanner input = new Scanner(System.in);  
System.out.print("Enter a byte value: ");  
byte byteValue = input.nextByte();
```



- جاوا شش نوعی داده ای عددی دارد.
- هر نوع توانایی نگهداری یک رنج عددی را دارد.
- چهار نوع `byte`, `short`, `int`, `long` برای کار با اعداد صحیح است.
- دو نوع `float`, `double` برای کار با اعداد اعشاری است

<code>byte</code>	$-2^7$ to $2^7 - 1$ (-128 to 127)	8-bit signed
<code>short</code>	$-2^{15}$ to $2^{15} - 1$ (-32768 to 32767)	16-bit signed
<code>int</code>	$-2^{31}$ to $2^{31} - 1$ (-2147483648 to 2147483647)	32-bit signed
<code>long</code>	$-2^{63}$ to $2^{63} - 1$ (i.e., -9223372036854775808 to 9223372036854775807)	64-bit signed
<code>float</code>	Negative range: $-3.4028235E + 38$ to $-1.4E - 45$ Positive range: $1.4E - 45$ to $3.4028235E + 38$	32-bit IEEE 754
<code>double</code>	Negative range: $-1.7976931348623157E + 308$ to $-4.9E - 324$ Positive range: $4.9E - 324$ to $1.7976931348623157E + 308$	64-bit IEEE 754

# 1-9-2) اپراتورهای عددی

Name	Meaning	Example	Result
+	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

اپراتورهای عددی تقویت شده

(Augmented)

<code>++var</code>	preincrement	Increment <code>var</code> by 1, and use the new <code>var</code> value in the statement	<code>int j = ++i;</code> // j is 2, i is 2
<code>var++</code>	postincrement	Increment <code>var</code> by 1, but use the original <code>var</code> value in the statement	<code>int j = i++;</code> // j is 1, i is 2
<code>--var</code>	predecrement	Decrement <code>var</code> by 1, and use the new <code>var</code> value in the statement	<code>int j = --i;</code> // j is 0, i is 0
<code>var--</code>	postdecrement	Decrement <code>var</code> by 1, and use the original <code>var</code> value in the statement	<code>int j = i--;</code> // j is 1, i is 0

Operator	Name	Example	Equivalent
<code>+=</code>	Addition assignment	<code>i += 8</code>	<code>i = i + 8</code>
<code>-=</code>	Subtraction assignment	<code>i -= 8</code>	<code>i = i - 8</code>
<code>*=</code>	Multiplication assignment	<code>i *= 8</code>	<code>i = i * 8</code>
<code>/=</code>	Division assignment	<code>i /= 8</code>	<code>i = i / 8</code>
<code>%=</code>	Remainder assignment	<code>i %= 8</code>	<code>i = i % 8</code>

مثال 2) برنامه ای بنویسید که بر اساس عدد دریافتی از ساعت هوشمند محاسبه نماید کاربر چه مدت زمانی مشغول تمرین بوده است. (معمولاً اعداد بر حسب میلی ثانیه داده می شوند)

```
public static void main(String[] args) {
    //Create an object as Scanner Class by System.in
    Scanner input = new Scanner(System.in);
    //Variables declaration
    int totalMiliseconds;
    int hours, minutes, seconds, miliseconds;
    //Getting input from user
    System.out.print("Please enter total msec's: ");
    totalMiliseconds = input.nextInt(); //read an integer from console

    miliseconds = totalMiliseconds % 1000; //Computing remained ms
    sec run:
    Please enter total msec's: 1257130
    Total 1257130ms means it's been working for 0:20:57 130ms (h:m:s ms)
    sec
    BUILD SUCCESSFUL (total time: 3 seconds)

    hours = minutes / 60;
    minutes = minutes % 60;

    System.out.print("Total " + totalMiliseconds +
        "ms means it's been working for ");

    System.out.print(hours + ":");
    System.out.print(minutes + ":");
    System.out.print(seconds + " ");
    System.out.println(miliseconds + "ms (h:m:s ms)\n");
} //end of main() method
```

تمرین 3) امروز (روز  $m$  ام هفته) با دوست خود قرار می  
گذارید که  $n$  روز بعد او را ملاقات کنید. برنامه ای بنویسید که  
مشخص کند  $n$  روز بعد چه روزی از هفته خواهد بود. ورودی  
از کاربر در زمان اجرای برنامه گرفته شود و روز شنبه روز  
اول هفته در نظر گرفته شود.

به عنوان مثال امروز دو شنبه است و با دوست خود قرار گذاشته اید که 11 روز بعد او را  
ملاقات کنید. 11 روز بعد چه روزی از هفته خواهد بود!؟

# 10-2) ثابت های عددی (NUMERIC LITERALS)

- لیترال ها ثابت های عددی (اعدادی) هستند که در برنامه به صورت **مستقیم** ظاهر می شوند.
- اگر لیترال نسبت داده شده به یک متغیر از محدوده نگهداری نوع متغیر بیشتر باشد کامپایلر تولید خطای کامپایل (Compiler Error) می کند.
- برای استفاده یا نمایش اعداد در مبناهای مختلف از روش زیر استفاده می شود:

```
int numberOfYears = 34;  
double weight = 0.305;
```

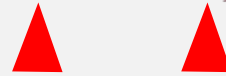
```
System.out.println(0B1111); // Displays 15  
System.out.println(07777); // Displays 4095  
System.out.println(0xFFFF); // Displays 65535
```

- افزودن 0b به ابتدای اعداد مبنای دو (Binary)
- افزودن 0 در ابتدای اعداد مبنای هشت (Octal)
- افزودن 0x ابتدای اعداد مبنای 16 (Hex)

- می توانیم برای خوانایی بیشتر اعداد بین آن ها از `underscore` استفاده کنیم. مانند `25_1455_123`
- اعداد اعشاری به صورت **پیش فرض** از نوع `double` در نظر گرفته می شوند برای جلوگیری از این کار می توانیم انتهای اعداد اعشاری از `f` یا `F` استفاده کنیم.

# 10-2) نماد علمی

- اعداد اعشاری می توانند به شکل  $a * 10^b$  نوشته شوند.
- نمادهای علمی را می توان به صورت قسمت `integer.fractionEpower` نوشت. مانند 1.23456E5



5.2534e+1, 0.52534e+2, 525.34e-1, 5.2534e+0

# ارزیابی عبارات و حق تقدم اپراتورها (11-2)

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

• عبارات در جاوا مشابه عبارات محاسباتی ارزیابی می شوند.

• در عبارات محاسباتی ابتدا عبارت داخل پرانتز ارزیابی می شود.

• اگر در یک عبارت بیش از یک اپراتور وجود داشته باشد:

• ابتدا اپراتور های `*`، `/`، `%` مورد ارزیابی قرار می گیرند.

• اگر در یک عبارت چند اپراتور ضرب و تقسیم و باقیمانده وجود داشته باشد به ترتیب از چپ به راست پردازش می شوند.

• اپراتور های جمع و تفریق در انتها ارزیابی می شوند.

• اگر در یک عبارت چند اپراتور جمع و تفریق وجود داشته باشد

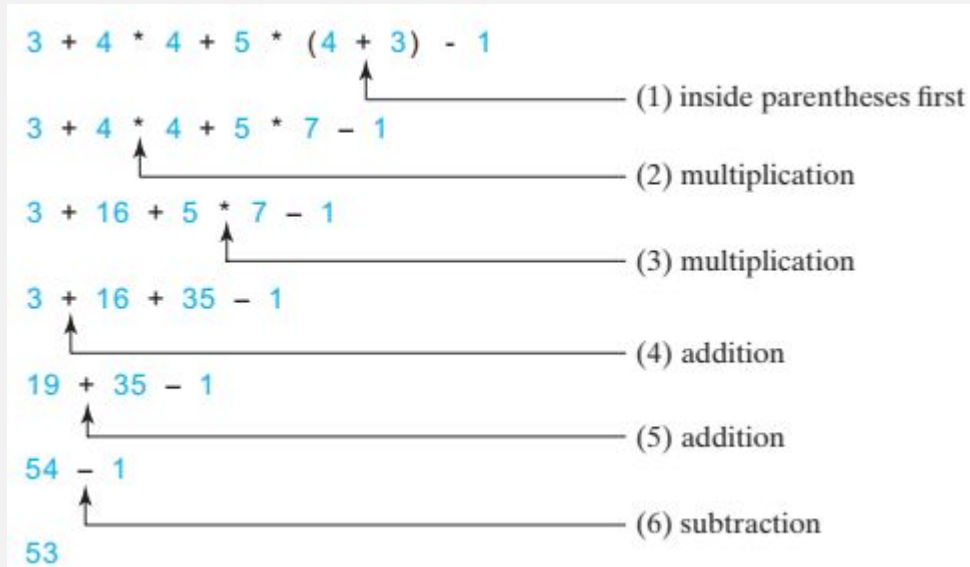
به ترتیب از چپ به راست پردازش می شوند.

• پس از ارزیابی عبارت محاسباتی اپراتور های تقویت شده ارزیابی می شوند.

• اپراتور های تقویت شده با اپراتور انتساب ترکیب شده اند. (در مرحله انتساب عمل می کنند)

• همه عملوندها از چپ به راست ارزیابی می شوند. در عملوندهای باینری اول عملوند (عبارت)

سمت چپ ارزیابی می شود.



```
int i = 1;
int k = ++i + i * 3; // k = ?
```



مثال 3) برنامه ای بنویسید که از کاربر درجه دما به فارانهیت را از کاربر دریافت نموده و به سیلیسیوس تبدیل کند.

```
public static void main(String[] args) {  
  
    double fahrenheit;  
    double celsius;  
    Scanner input = new Scanner(System.in);  
    System.out.print("Please enter a degree in fahrenheit: ");  
    fahrenheit = input.nextDouble();  
  
    celsius = (5.0 / 9) * (fahrenheit - 32);  
  
    System.out.println("Degree in " + fahrenheit + " fahrenheit is equal to "  
        + celsius + " celsius");  
}  
//end of main() method
```

run:  
Please enter a degree in fahrenheit: 100  
Degree in 100.0 fahrenheit is equal to 37.77777777777778 celsius  
BUILD SUCCESSFUL (total time: 22 seconds)

h:m: نمایش

مثال (4) بر دهد.

```
public static void main(String[] args) {
    //Create an object as Scanner Class by System.in
    Scanner input = new Scanner(System.in);
    //Variables declaration
    long totalMilliseconds;
    long hours, minutes, seconds, milliseconds;
    long currentHour, currentMinute, currentSecond;

    totalMilliseconds = System.currentTimeMillis();

    milliseconds = totalMilliseconds % 1000; //Computing remained ms
    seconds = totalMilliseconds / 1000;      //Computing seconds

    run:
    minutes : 441251:18:0 600ms (h:m:s ms) past from UNIX epoch!
    seconds  →                                     d sec
    GMT Now: 11:18:0

    hours = 1
    minutes : BUILD SUCCESSFUL (total time: 0 seconds)

    System.out.print(hours + ":" + minutes + ":" + seconds + " ");
    System.out.println(milliseconds +
        "ms (h:m:s ms) past from UNIX epoch!\n");

    currentSecond = seconds % 60;
    currentMinute = minutes % 60;
    currentHour = hours % 24;

    System.out.println("GMT Now: " + currentHour + ":" + currentMinute +
        ":" + currentSecond + "\n");
} //end of main() method
```

--

ime

## 2-15) تبدیل انواع عددی

- تبدیل نوع (Type Casting):

- به عملیاتی گفته می شود که در آن مقدار یک متغیر از یک نوع به نوع دیگر تبدیل می شود. `int -> double`

- به طور کلی دو نوع تبدیل داریم:

- گسترش یک نوع (Widening a Type): تبدیل (Cast) یک نوع با محدوده کوچک تر به نوع دارای محدوده بزرگتر

- کاهش یک نوع (Narrowing a Type): تبدیل (Cast) یک نوع دارای محدوده بزرگتر به نوع دارای محدوده کوچکتر

- جاوا به صورت اتوماتیک تبدیل از نوع گسترش - نوع را انجام می دهد.  $\frac{3}{int} * \frac{4.5}{float} = \frac{12.5}{float}$

- عملیات تبدیل از نوع کاهش - نوع لازم است صراحتاً توسط برنامه نویس انجام بشود.

```
double d = 4.5;  
int i = (int)d; // i becomes 4, but d is still 4.5
```

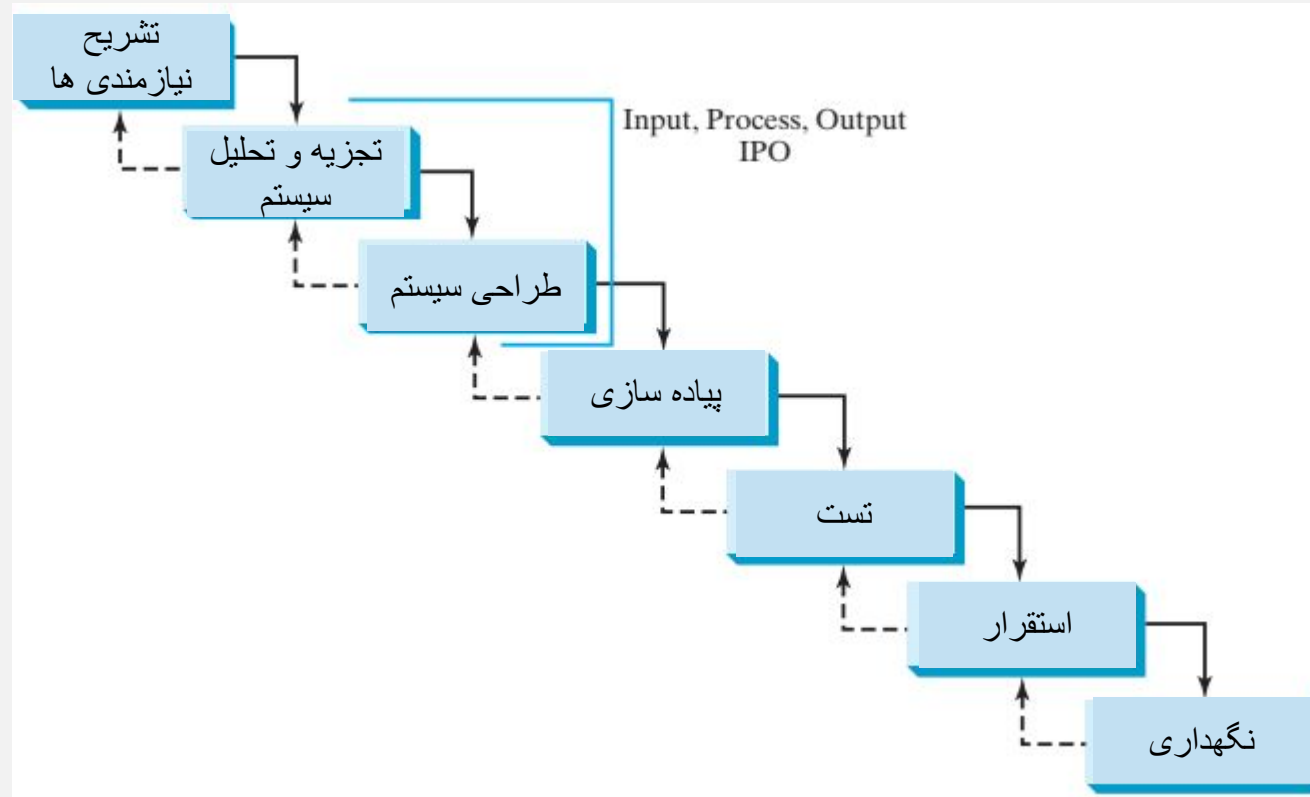
- در صورت عدم استفاده صریح از فرمت casting کامپایلر تولید خطای نحوی (Syntax) می کند.

مثال 5) برنامه ای بنویسید که مقدار مالیات بر مصرف کننده (ارزش افزوده ایرانی) را به برای کالای خریداری شده محاسبه نماید. (9%)

```
public static void main(String[] args) {  
  
    double price, tax, total;  
  
    Scanner input = new Scanner(System.in);  
  
    System.out.print("Please enter a purchase amount: ");  
    run:  
    Please enter a purchase amount: 87000  
    Your total payment: 94830.0 Rials. 87000.0 your sale and 7830.0 for taxes  
    BUILD SUCCESSFUL (total time: 4 seconds)  
    total = price + tax;  
    total = (int)(total * 100) / 100.0;  
  
    System.out.print("Your total payment: " + total + " Rials. ");  
    System.out.println(price + " your sale and " +  
        (int)(tax * 100) / 100.0 + " for taxes");  
}  
//end of main() method
```

# 2-16) فرایند توسعه نرم افزار

- چرخه حیات توسعه نرم افزار یک فرایند چند مرحله ای شامل تشریح نیازمندی ها، تجزیه و تحلیل، پیاده سازی، تست، استقرار و نگهداری است.
- توسعه نرم افزار یک فرایند مهندسی است.



# 16-2) فرایند توسعه نرم افزار

- Requirement Specification: بخشی از روند اصلی توسعه نرم افزار است و طی آن مساله ای که قرار است توسط نرم افزار حل شود به خوبی و با جزئیات شناسایی و توسط مستندات تشریح می شود. در این مرحله مشخص می شود که سیستم نرم افزاری چه نیازهایی دارد.
- نیازمند ارتباط نزدیک تیم توسعه و کار بر است
- System Analysis: در این مرحله جریان داده در سیستم نرم افزاری به خوبی شناسایی می شود و ورودی / خروجی های سیستم مشخص می شود
- System Design: شامل طراحی فرایندی می شود که بتوان طی آن از ورودی ها به خروجی های تعیین شده دست یافت.
- این مرحله شامل انتزاع (Abstraction)، خرد کردن مساله به اجزا (Components) قابل مدیریت و طراحی استراتژی برای پیاده سازی هر یک از اجزا می شود.
- هر جز می تواند به عنوان یک زیر سیستم یا یک کارکرد (Function) در نظر رگفته شود.
- Implementation: شامل ترجمه / تبدیل طراحی سیستم انجام شده به برنامه است.
- تجزیه سیستم به اجزا پیاده سازی شده (نوشته شده) و سپس مجتمع سازی اجزا برای کار کردن با یک دیگر
- این مرحله شامل کدنویسی، تست و خطایابی (Debug) می شود.
- Testing: این مرحله شامل اطمینان پیدا کردن از در نظر گرفتن تمام نیازمندی های تشریح شده در سند نیازمندی ها و بدون عیب بودن کدها است

# 16-2) فرایند توسعه نرم افزار

- Deployment: شامل در دسترس قرار دادن نرم افزار برای کاربر است.
- بسته به نوع نرم افزار می تواند روی کامپیوتر کاربر و یا روی سرور استقرار صورت گیرد
- Maintenance: ناظر بر بهبود و بروز رسانی محصول است.
- فرایند بهبود و بروز رسانی به صورت دائمی باید انجام بشود.
- به دلیل تغییر نیازمندی های کاربر و کشف خطاهای جدید در نرم افزار

## مثال 6) به منظور درک بهتر چرخه توسعه نرم افزار نرم افزار پرداخت وام را گام به گام پیاده سازی می کنیم. وام خودرو، وام دانشجویی و وام رهن منزل می تواند نمونه ای از انواع وام باشد.

- مرحله ۱) Requirement specification: نرم افزار باید نیازمندی های زیر را ارضا نماید.
  - باید به کاربر اجازه دهد که نرخ، مقدار و تعداد سال های وام مورد نظر خود را وارد نماید.
  - نرم افزار باید اقساط ماهانه و کل مبلغ بازپرداخت را نمایش دهد.
- مرحله ۲) System analysis: خروجی باید شامل اقساط پرداختی ماهانه و کل مبلغ وام باشد که از فرمول زیر محاسبه می شود.

$$monthlyPayment = \frac{loanAmount * monthlyInterestRate}{1 - \frac{1}{(1 + monthlyInterestRate)^{numberOfYears * 12}}}$$

$$totalPayment = monthlyPayment * numberOfYears * 12$$

❖ ممکن است در طی انجام این مرحله نیازمندی و یا اطلاعات جدیدی به دست بیآورد و نیاز باشد که به مرحله قبل بازگردید و نیازمندی ها را اصلاح کنید.

❖ به عنوان یک برنامه نویس ممکن است با افراد و تخصص های مختلف برای توسعه نرم افزار همکاری کنید. نیازی نیست متخصص هر کدام از آن رشته ها باشید



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

```
    int numberOfYears;
```

```
    double annualInterestRate;
```

```
    double monthlyInterestRate;
```

```
    double monthlyPayment;
```

```
    double loanAmount;
```

```
    double totalPayment;
```

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

```
    //Stage 1: Obtaining required info
```

```
    System.out.print("Please enter your desire annual rate (eg. 4.5): ");
```

```
    annualInterestRate = input.nextDouble();
```

```
System run:
```

```
number: Please enter your desire annual rate (eg. 4.5): 4
```

```
        Please enter number of years as an integer: 7
```

```
System: Please enter your desire loan amount: 50000000
```

```
loanAm: The monthly payment is 683440.31
```

```
//Stage: The total payment is 2.147483647E7
```

```
monthl: BUILD SUCCESSFUL (total time: 16 seconds)
```

```
//Stage 2: Calculating monthly payment according to the given formula
```

```
monthlyPayment = loanAmount * monthlyInterestRate /
```

```
    (1 - 1 / Math.pow(1 + monthlyInterestRate, numberOfYears * 12));
```

```
//Stage 4: Calculating total payment after given years
```

```
totalPayment = monthlyPayment * numberOfYears * 12;
```

```
//Stage 5: Dispaling results
```

```
System.out.println("The monthly payment is " +
```

```
    (int) (monthlyPayment * 100) / 100.0);
```

```
System.out.println("The total payment is " +
```

```
    (int) (totalPayment * 100) / 100.0);
```

```
}//end of main() method
```

## مثال (6) ادامه

• مرحله ۳) m design

(1) درخواست از کاربر بر

(2) ورودی کاربر برای نرخ

دریافت می شود و ب

ماهیهانه باید نرخ وارد

(3) اقساط ماهیهانه را بر

(4) کل مبلغ بازپرداخت

(5) مقدار قسط ماهیهانه

• مرحله ۴) nentation

• نرخ سود سالانه از کاربر  
به دست آوردن نرخ سود

ی کنیم.

## مثال (6) ادامه

- مرحله 5 Test: در این مرحله لازم است با دادن ورودی های مختلف درستی خروجی رو مورد بررسی قرار دهید. در برنامه های پیشرفته لازم است طراحی تست برای نرم افزار و تک تک اجزا صورت بپذیرد.

تمرین 4) برنامه ای بنویسید که مبلغ داده شده توسط کاربر  
(کمتر از 100 هزار تومان) را با مبالغ 50000 و 10000 و  
5000 و 2000 و 1000 و 500 تومانی خرد کند.

## خطاها (2-18)

- خطاهای متداول (1) متغیرهای اعلان نشده، مقداردهی نشده و بلااستفاده
  - معمولاً IDE ها به کاربر اخطار می دهند.
- خطاهای متداول (2) integer overflow
  - جاوا هیچ خطاری در این زمینه صادر نمی کند. (کاملاً در خصوص overflow, underflow محتاط باشید)
- خطاهای متداول (3) خطای گرد کردن
  - در مسائله گردن کردن اعداد اعشاری رخ می دهد.
- خطاهای متداول (4) تقسیم های ناخواسته اعداد int
  - اگر دو طرف اپراتور تقسیم مقدار int باشد پاسخ مقدار int خواهد بود و قسم اعشاری احتمالی حذف خواهد شد.

## (2-18) تله های متداول (PITFALLS)

- تله 1) ایجاد اشیا اضافه
- مانند ایجاد چندین شی از کلاس Scanner برای دریافت اطلاعات از کنسول

```
Scanner input = new Scanner(System.in);  
System.out.print("Enter an integer: ");  
int v1 = input.nextInt();
```

```
Scanner input1 = new Scanner(System.in);  
System.out.print("Enter a double value: ");  
double v2 = input1.nextDouble();
```

**BAD CODE**

