Less is more …

- Effective
- Attractive
- Impactive

Introduction to Java Programming and Data Structures

Presented to:
Amol University of Special Modern Technologies

Vahid.khazaeinezhad@gmail.com

Introduction to Java Programming and Data Structures

فصل دوم

Elementary Programming

زمستان 99

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

```java
/*
 * This program is wrote for educational purpose
 * Advance Programmin Course, AUSMT
 */
package computecirclearea;

/**
 * This program calculates the area of an
 * arbitary circle, at first read its radius
 * and then presents the result *
 *
 * @author Khazaeinezhad_Vahid
 */
public class ComputeCircleArea {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // 1) Read in radius
        // 2) Compute area
        // 3) Dispaly the resault

    }//end of main() method

}//end of class
```

- 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 کامپیلر 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;
23          double area;
24          //1-2)
25          radius = 30;
26          // 2) Compute area
27          area = radius * radius * 3.141596;
28          // 3) Dispaly the resault
29          System.out.println("The area of circle with radius " + radius + " are " + area);
30
31      }//end of main() method
```

Declaration

Assignment

Concatenation

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 st
22          // 1)
23          //1-
24
25          doub
26          Scan
27          //1-2)
28          System.out.println("Enter radius:");
29          radius = input.nextDouble();
30          // 2) Compute area
31          area = radius * radius * 3.141596;
32          // 3) Dispaly the resault
33          System.out.println("The area of circle with radius " + radius + " are " + area);
34
35      }//end of main() method
36
37  }//end of class
```

```
run:
Enter radius:

The area of circle with radius 30.0 are 2827.4364
BUILD SUCCESSFUL (total time: 8 seconds)
```

2-3)نکته های برنامه

- کنسول استاندارد خروجی **مونیتور** 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 باید 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  * arbitary circle, at first read its radius
12  * and then presents the result *
13  *
14  * @author Khazaeinezhad_Vahid
15  */
16 public class ComputeCircleArea {
```

مثال ۱) 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("P
28          number1 = input.nex
29
30          System.out.print("Pl
31          number2 = input.next
32
33          System.out.print("Please enter third number:");
34          number3 = input.nextDouble();
35
36          average = (number1 + number2 + number3) / 3;
37
38          System.out.println("For these 3 [" + number1 + ", " + number2 +
                    ", " + number3 + "] numbers the average are " + average);
40
41      }//end of main() method
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($)
  - 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.

2-5)متغیرها (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:
  - 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.

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

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

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
```

2-7)ثابت ها (CONSTANTS)

- ثابت های نام دار **شناسه** هایی هستند که مقادیر **دائمی / ثابت** را نمایه می کنند.
  - مقدار متغیرها در زمان اجرای برنامه ممکن است تغیر بکند.
  - ثابت های نام دار و یا ساده **هرگز** تغییر نمی کنند.
  - یک ثابت به عنوان Final Variable نیز در جاوا شناخته می شود.
  - ثابت ها **باید** هم زمان با اعلان مقدار دهی اولیه (initialize) بشوند.
  - کلیدواژه تعریف یک ثابت final است.
  - به عنوان یک **قرارداد** ثابت ها با حروف بزرگ (تمامی حروف) نام گذاری می شوند.

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

2-8)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

2-9)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

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

2-9-1)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

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

اپراتورهای عددی تقویت شده
(Augmented)

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

مثال 2) برنامه ای بنویسید که بر اساس عدد دریافتی از ساعت هوشمند محاسبه نماید کاربر چه مدت زمانی مشغول تمرین بوده است. (معمولا 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
    se
    se
    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
```

```
run:
Please enter total msec's: 1257130
Total 1257130ms means it's been working for 0:20:57 130ms (h:m:s ms)

BUILD SUCCESSFUL (total time: 3 seconds)
```

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 کندn 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؟!

2-10)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:
  - افزودن 0b به ابتدای اعداد مبنای دو (Binary)
  - افزودن 0 در ابتدای اعداد مبنای هشت (Octal)
  - افزودن 0x ابتدای اعداد مبنای 16(Hex)

```
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
```

- **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 ازین کار می توانیم انتهای اعداد اعشاری از f یا F استفاده کنیم.

2-10)نماد علمی

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

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

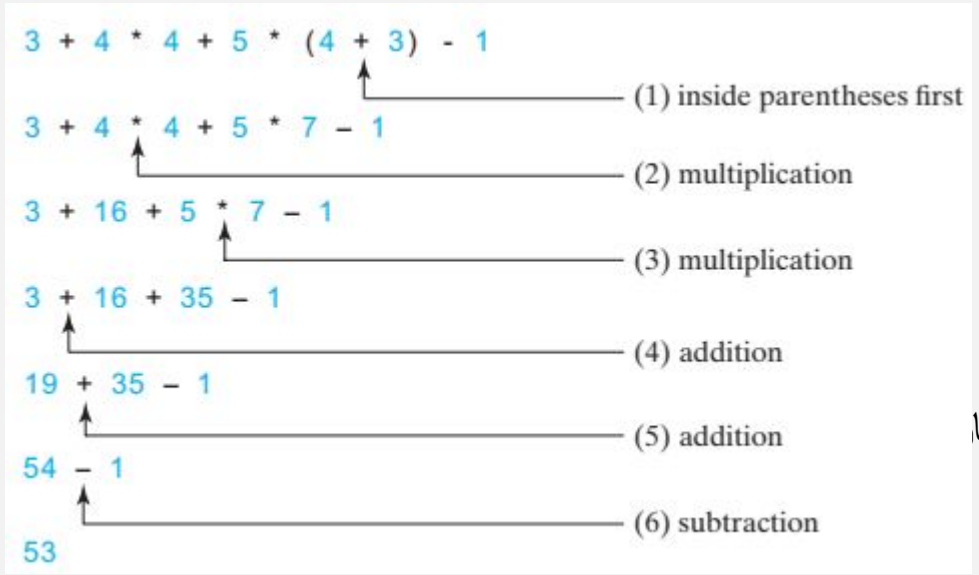
2-11)ارزیابی عبارات و حق تقدم اپراتورها

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

- 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:

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

  - 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 اپراتورهای **تقویت شده** ارزیابی می شوند.
  - اپراتورهای تقویت شده با اپراتور انتساب ترکیب شده اند. (در مرحله انتساب عمل می کنند)
- همه عملوندها از چپ به راست ارزیابی می شوند. در عملوند های باینری اول عملوند (عبا
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 فارانهایت را از کاربر دریافت نموده و به سیلیسیوس تبدیل کند.

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

    double farenheit;
    double ce run:
                Please enter a degree in farenheit: 100
    Scanner i Degree in 100.0 farenheit is equal to 37.77777777777778 celsius
              BUILD SUCCESSFUL (total time: 22 seconds)
    System.out.print("Please enter a degree in farenheit: ");
    farenheit = input.nextDouble();

    celsius = (5.0 / 9) * (farenheit - 32);

    System.out.println("Degree in " + farenheit + " farenheit is equal to "
            + celsius + " celsius");
}//end of main() method
```

مثال 4) بر
دهد.
نمایش h:m:

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

    totalMiliseconds = System.currentTimeMillis();

    miliseconds = totalMiliseconds % 1000; //Computing remained ms
    seconds = totalMiliseconds / 1000;     //Computing seconds

    minutes =
    seconds

    hours = r
    minutes =

    System.out.print(hours + ":" + minutes + ":" + seconds + " ");
    System.out.println(miliseconds +
                       "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
```

```
run:
441251:18:0 600ms (h:m:s ms)  past from UNIX epoch!
d sec
GMT Now: 11:18:0
BUILD SUCCESSFUL (total time: 0 seconds)
```

*2-12 و 2-13 و 2-14 در بخش 2-11 تجمیع گردید.

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 **صراحتا** 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.

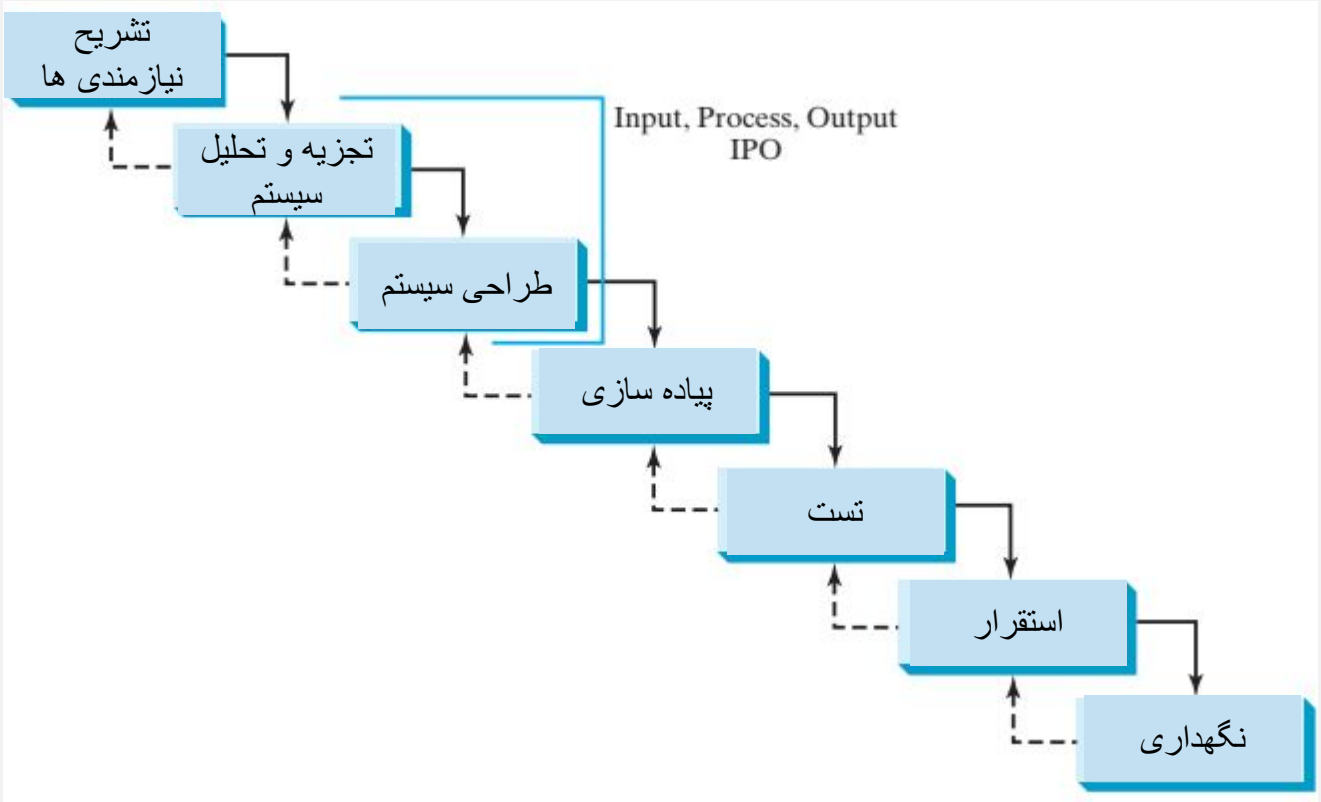
مثال 5) برنامه ای بنویسید که مقدار مالیات بر مصرف کننده (ارزش افزوده ایرانی) را به برای کالای خریداری شده محاسبه نماید. (9%)

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

    double price, tax, total;

    Scanner input = new Scanner(System.in);

    System.out.print("Please enter a purchase amount: ");
    p
    t
    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
```

```
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)
```

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.



2-16)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

2-16)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 بروزرسانی 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 بیاورد 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

مثال 6) ادامه

- مرحله ۳) m design
  1) درخواست از کاربر ب
  2) ورودی کاربر برای نر
     دریافت می شود و ب
     ماهیانه باید نرخ وارد
  3) اقساط ماهیانه را بر
  4) کل مبلغ بازپرداخت
  5) مقدار قسط ماهیانه
- مرحله ۴) mentation

ن نرخ سود سلانه از کاربر
به دست آوردن نرخ سود

ی کنیم.

```
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 anuual rate (eg. 4.5): ");
    annualInterestRate = input.nextDouble();

    System                                                          ");
    number
    loanAm
    //Stag
    monthl
    //Stage 3: 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 resualts
    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
```

```
run:
Please enter your desire anuual rate (eg. 4.5): 4
Please enter number of years as an integer: 7
Please enter your desire loan amount: 50000000
The monthly payment is 683440.31
The total payment is 2.147483647E7
BUILD SUCCESSFUL (total time: 16 seconds)
```

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 بپزیرد.

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.

*2-17 به عنوان تمرین داده شد.

2-18)خطاها

- خطاهای متداول **1**) متغیرهای اعلان نشده، مقداردهی نشده و بلااستفاده
  - معمولا IDE ها به کاربر اخطار می دهند.
- خطاهای متداول **2**) integer overflow
  - جاوا هیچ اخطاری در این زمینه صادر نمی کند. (کاملا 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.

*2-17 به عنوان تمرین داده 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