



ALAMMAR

قواعد المبرمج

الأستاذ : عبدالرحمن العمار



@AbdurahmanAmmar



@abdulrahman_alammar

Java Program Structure

```
ce History
package alammar_advice;

// comments about the class
public class Alammar_advice ← class header
{
    // comments about the method
    public static void main(String[] args) ← Header of main method
    {
        // Program execution begins here
        System.out.println("Hello World!");
    }
} ← class body
    { ← Main method body
        }
```

Class header	<ul style="list-style-type: none">• يجب مراعاة عدة أمور في الـ class header :• التأكد من كلمتي public و class أنها مكتوبة بأحرف جميعها small.• التأكد من أن أول حرف من اسم الـ class يكون capital.
Class body	<ul style="list-style-type: none">• كل الأوامر التي نريد كتابتها في برنامجنا يجب أن تكون داخل أقواس الـ class body .
Header of main method	<ul style="list-style-type: none">• الـ header الخاص بالـ main method ثابت لا يتغير أبدا يجب كتابته دائما كما هو موضح في الصورة أعلاه.• يمكن أن يكون في البرنامج أكثر من method ولكن لا يمكن أن يعمل البرنامج إن لم تكتب الـ main method وذلك لأن عند تشغيل البرنامج الـ JVM سوف يقوم بالبحث عنها لكي يبدأ بتشغيل البرنامج الذي قمنا بكتابته.• يجب أن تكون جميع كلمات سطر تعريف الـ main مكتوبة بحروف small ما عدا كلمة String يجب أن تبدأ بحرف capital.
Main method body	<ul style="list-style-type: none">• الأوامر التي نريدها أن تنفذ عند تشغيل البرنامج تكتب بداخل أقواس الـ main method bod .



ALAMMAR

@AbdurahmanAmmar

@abdulrahman_alammar

Anatomy of a Java Program

Comments:

كتابة الملاحظات لا تؤثر أبداً على البرنامج عند تشغيله ولكن تستخدم لكي يستطيع الشخص فهم الكود عند الرجوع له بعد فترة .

Reserved words:

عبارة عن كلمات تم حجزها واختيارها من قبل مخترعين لغة Java لكي تستخدم للأغراض معينة في البرمجة ولا يمكن استخدامها لغير الغرض الذي خصصت له .

أمثلة على Reserved words:

int – double – void – if – else – for – while ...etc

Modifiers:

هي جزء من الكلمات المحجوزة تستخدم مع المتغيرات أو الـ Methods أو الـ classes لكي تحدد من يمكن له الوصول لهم.

أمثلة على Modifiers:

public – private – protected – static ...etc

Identifiers:

هي أسماء المتغيرات أو الـ Methods أو الـ classes التي تم تعريفها من قبل المبرمج.

تسمية المتغيرات :

يجب أن لا تبدأ برقم ولا أن تكون من الكلمات المحجوزة. يمكن استخدام (_) أو (\$) ولا يمكن استخدام غيرها من الرموز في تسمية المتغيرات. يمكن أن يكون اسم المتغير مكون من أحرف وأرقام وأحد الرزمين _ أو \$

Statements:

هي كل ما يتم كتابته في البرنامج من أوامر ونهيه ؛

Blocks:

عندما نقوم بفتح قوس من هذا النوع { ونكتب بداخله عدة أوامر ونغلقه نكون قد قمنا بعمل block كأنه صندوق يحتوي على عدة أوامر مثل الـ methods بما أنها تبدأ وتنتهي بهذه الأقواس يمكننا أن نقول أنها block.

Classes:

الـ class هو المكون الأساسي لأي برنامج في لغة Java فلا يمكن القول بأننا قمنا بعمل برنامج في Java دون كتابة class.

Methods:

يمكننا القول بأنها عبارة عن block يحتوي على العديد من الـ statements تنفذ بشكل متسلسل لتنفيذ عملية ما مثل الطباعة على الشاشة .

The main method:

هي نقطة بداية تشغيل أي برنامج وتتحكم في سير تنفيذ البرنامج.



ALAMMAR

@AbdurahmanAmmar

@abdulrahman_alammar

Good Programming Practice

لكي تكون أكوادك مفهومة وواضحة لفيرك من المبرمجين ولك أيضا إذا عدت لها بعد فترة من كتابتها يجب عليك اتباع الآتي:

- كتابة Comments قبل ال class يوضح الغرض من هذا ال class مثال :

//this class represent the file of student

- كتابة Comments قبل أي method وذلك لكتابة الغرض من هذه ال method مثال:


//this is method to sum of two number


- كتابة Comment في نهاية ال statement يوضح عملها مثال :

System.out.println("Hello World!"); // print a string on the screen



ALAMMAR

 @AbdurahmanAmmar

 @abdurahman_alammar

The Java reserved words:

الكلمات المحجوزة في java:

جميع الكلمات المحجوزة في java يجب أن تكتب بـ small letters ، ولا يمكن استخدامها لغير الغرض الذي حجزت له.

الكلمات المحجوزة في java هي :

abstract
assert
boolean
break
byte
case
catch
char
class
const
continue
default
do
double

else
enum
extends
false
final
finally
float
for
goto
if
implements
import
instanceof
int

interface
long
native
new
null
package
private
protected
public
return
short
static
strictfp
super

switch
synchronized
this
throw
throws
transient
true
try
void
volatile
while



Notes

قواعد التسميات في لغة Java :

عند تسمية ال class أو ال method أو ال variable .. لا يمكننا استخدام مسافات لذلك هناك طريقتين في التسمية لكي يكون الاسم مفهوم وسهل القراءة.

- عن طريق استخدام _ underscore مثل :

`float exam1_grade=90;`

- عن طريق استخدام capital letter في بداية كل كلمة مثل :


`float examOneGrade = 90 ;`

ملاحظة :-

لا يمكن استخدام الأرقام ولا أحد الكلمات المحجوزة في بداية اسم ال class أو ال method أو ال variable .



ALAMMAR

 @AbdurahmanAmmar

 @abdurahman_alammar

Notes for variables

طرق تعريف ال variables :

- إسناد مباشر (Initialization)

```
double width = 50;
```

- إسناد غير مباشر (Declaration ثم Initialization)

```
double width;
```

```
width = 50;
```

- إسناد عن طريق معادلة (Assignment Statements)

```
double width = 2;
```

```
double height = 3;
```

```
double area = width*height;
```

Constant Variables Declarations:

```
final type varName = value;
```


```
final double PI = 3.14159;
```

OR

```
public static final double PI= 3.14159;
```



ALAMMAR

 @AbdurahmanAmmar

 @abdurahman_alammar

Notes for variables

Variables Types:

Primitive								Reference
boolean	byte	char	short	int	long	float	double	String
True / false	9	'A'	6	3	4	2.5	10.9	"Alammar"
يمكن أن يخزن هذه القيمتان فقط	لتخزين عدد صحيح	يمكن تخزين حرف واحد فقط	لتخزين عدد صحيح ومساحته أكبر من byte	لتخزين عدد صحيح ومساحته أكبر من short	لتخزين عدد صحيح ومساحته أكبر من int	للقيم العشرية	للقيم العشرية ومساحته أكبر من الـ float	لتخزين سلسلة نصية



Precedence of Operators

عندما يقوم الجهاز بقراءة الكود يتم قراءته من أعلى لأسفل و من اليسار لليمين إلا في حالات وجود عمليات حسابية فإنه يعتمد على أولوية كالاتي :

1- دائما الأولوية للأقواس ()
2- عمليات ال postfix مثال : a++ g a--
3- عمليات ال prefix مثال : --a g ++a
4- عمليات ال Casting
5- عمليات الضرب * والقسمة / والقسمة بالباقي % لهم نفس الأولوية
6- عمليات الجمع + والطرح - لهم نفس الأولوية
7- العمليات المنطقية > و < و <= و >=
8- عمليات المساواة == و !=
9- عملية &&
10 - عملية



Notes for printf

The screenshot shows a Java IDE with a source code editor on the left and an output window on the right. The source code is as follows:

```
1 package alammr_advices;
2
3
4 // comments about the class
5 public class Alammr_advices {
6
7     // comments about the method
8     public static void main(String[] args) {
9
10        // Program execution begins here
11        int number = 1;
12        String course = "Java";
13        System.out.printf("The course is %s(%d)..%s%n", course, number, "Alammr");
14    }
15
16 }
```

The output window, titled "Output - Alammr_advices (run)", shows the following text:

```
run:
The course is Java(1)..Alammr
```

Arrows indicate the mapping between the code and the output. A yellow arrow points from the format string `"The course is %s(%d)..%s%n"` to the output. A blue arrow points from the variable `course` to the first `%s` in the format string. Another blue arrow points from the variable `number` to the `%d` in the format string. A third blue arrow points from the variable `"Alammr"` to the second `%s` in the format string.

في هذا الجزء نضع ما نريد كتابته ك String ثابت بالإضافة إلى ال format specifiers مثل:
%s خاصة بالنصوص
%d خاصة بالأرقام الصحيحة
%f خاصة بالأرقام العشرية
%n تنزل سطر جديد

format

هي القيم التي نريد طباعتها بتنسيق معين عن طريق استخدام ال format specifiers

variable



Scanner class

الفرض من الاستخدام :

نستخدم Scanner class لأخذ قيم من المستخدم عن طريق لوحة المفاتيح .

خطوات استخدام Scanner :

✓ عمل تضمين لـ Scanner

```
import java.util.Scanner;
```

* أمر التضمين يكتب قبل الـ class

✓ إنشاء object من Scanner

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

* كلمة keyboard يمكن تغييرها بأي كلمة بحسب ما يراه المبرمج مناسب

أخذ قيمة من المستخدم من نوع String	أخذ قيمة من المستخدم من نوع double	أخذ قيمة من المستخدم من نوع int
<pre>Scanner keyboard = new Scanner(System.in); String word1 = keyboard.next();</pre> <p>* عند استخدام .next() (يقرأ من المستخدم نص بدون أي مسافات)</p> <pre>String line = keyboard.nextLine();</pre> <p>* عند استخدام .nextLine() (يقرأ من المستخدم سطر واحد فقط)</p>	<pre>Scanner keyboard = new Scanner(System.in); double doubleNum = keyboard.nextDouble();</pre>	<pre>Scanner keyboard = new Scanner(System.in); int intNum = keyboard.nextInt();</pre>



ALAMMAR

@AbdurahmanAmmar

@abdurahman_alammar

Type of Errors in Java



String Methods

Method	Example
<code>indexOf();</code> للبحث عن خانة حرف	<code>int رقم = name.indexOf(حرف);</code>
<code>indexOf();</code> للبحث عن خانة الحرف من خانة البداية	<code>int رقم = name.indexOf(حرف, رقم خانة البداية);</code>
<code>substring();</code> اقتصاص من	<code>String نص = name.substring(رقم خانة البداية);</code>
<code>substring();</code> اقتصاص من - إلى	<code>String نص = name.substring(رقم خانة البداية - 1, رقم خانة البداية);</code>
<code>charAt();</code> ابحث عن الحرف في الخانة	<code>char حرف = name.charAt(الخانة رقم);</code>
<code>parseInt();</code> تحويل من String إلى int	<code>int رقم = Integer.parseInt(string);</code>
<code>compareTo();</code> مقارنة String ب String	<code>int رقم = name.compareTo(string);</code>
<code>equals();</code> مقارنة String ب String	<code>Boolean صح أو خطأ = name.equals(string);</code>
<code>indexOf();</code> ابحث عن رقم خانة أول حرف في String	<code>int رقم = name.indexOf(string);</code>
<code>indexOf();</code> ابحث عن خانة الحرف ابتداء من خانة معينة	<code>int رقم = name.indexOf(string, رقم خانة البداية);</code>
<code>replace();</code> استبدال حرف بحرف	<code>String string = name.replace(الحرف المستبدل, الحرف البديل);</code>



Algorithm & Pseudo Code & Flow charts

Algorithm

الـ Algorithm عبارة عن خطوات يجب أن تكتب بلغة وطريقة يفهمها أي شخص قد لا يدرس الحاسب ولكن يفهم الغرض وماذا ستنفذ هذه الخوارزمية.

Pseudo code

تشبه الـ Algorithm إلى حد ما، حيث يقوم المبرمجين بكتابة خطوات الخوارزمية ولكن بشكل مختصر ويستخدمون بعض الرموز للتسهيل على المبرمج بتحويلها إلى code حقيقي.

Flow charts

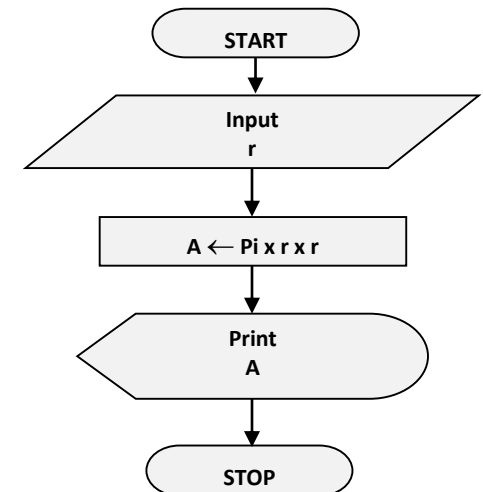
عبارة عن رسم بياني يساعد في توضيح وتسهيل Pseudo Code نستخدم اشكال محددة كل شكل لو وظيفة خاصة فيه.

Algorithm

- Step 1: Input r
- Step 2: $A \leftarrow \pi \times r \times r$
- Step 3: Print A

Pseudocode

- Input the radius (r) of a circle
- Calculate the area (A) :
 $A \leftarrow \pi \times r \times r$
- Print A



ALAMMAR

@AbdurahmanAmmar

@abdurahman_alammar

for Loop

```
Source History
1 package alammr_advices;
2 import java.util.Scanner;
3
4 public class Alammr_advices {
5
6     public static void main(String[] args) {
7
8         Initialization      termination      Update
9         {
10             for (int i = 0; i < 10; i++) {
11                 Body {
12                     System.out.println("we are Good Programmers");
13                 }
14             }
15         }
16     }
17 }
```

Initialization	الرقم الذي سوف يبدأ منه اللوب
termination	الشرط الذي يتم التحقق منه في كل مرة يتم قبل الدخول إلى الـ for إذا لم يتحقق لا يدخل إلى الـ for
Update	العملية المسؤولة عن التحديث للقيمة التي سنبدأ منها
Body of for	الـ statements التي نريد تكرارها

تستخدم الـ for:

إذا كنا نعلم كم مرة علينا الدخول لـ for



@AbdurahmanAmmar

@abdurahman_alammar

while Loop

```
Source History
1 package alammr_advices;
2
3 import java.util.Scanner;
4
5 public class Alammr_advices {
6
7     public static void main(String[] args) {
8
9         int count = 1;
10
11         Expression
12         {
13             while (count < 11) {
14                 System.out.println("we are Good Programmers");
15                 count++;
16             }
17         }
18     }
```

Expression	الرقم الذي سوف يبدأ منه اللوب
Update	العملية المسؤولة عن التحديث للقيمة التي سنبداً منها
Body of while	ال statements التي نريد تكرارها

تستخدم ال while:

إذا كنا لا نعلم كم مرة علينا الدخول لا while



ALAMMAR

@AbdurahmanAmmar

@abdurahman_alammar

do-while Loop

```
Source History
1 package alammar_advices;
2
3 import java.util.Scanner;
4
5 public class Alammar_advices {
6
7     public static void main(String[] args) {
8
9         int count = 1;
10        do {
11            System.out.println("Count is: " + count);
12
13            Update
14            count++;
15        } while (count < 11);
16
17    }
18 }
```

Diagram labels in the code editor:

- Body**: A blue bracket on the left side of the `do` block, spanning lines 10 to 15.
- Update**: A yellow bracket above the `count++` statement on line 14.
- Expression**: A green bracket below the `while (count < 11);` statement on line 16.

Expression	الرقم الذي سوف يبدأ منه اللوب
Update	العملية المسؤولة عن التحديث للقيمة التي سنبدأ منها
Body of while	ال statements التي نريد تكرارها

تستخدم ال do-while :

إذا كنا لا نعلم كم مرة علينا الدخول لـ `while` ، ولكن تنفذ ما بداخلها مرة واحدة على الأقل وتستخدم غالباً لعرض ال menu



ALAMMAR

@AbdurahmanAmmar

@abdurahman_alammar

Conditions statements



```
if( Boolean-Expression ){  
    statement;  
}
```



```
if(Boolean-Expression){  
    statement;  
    statement;  
}  
else {  
    statement;  
    statement;  
}
```



```
if( Boolean-Expression_1 )  
    Statement_1;  
else if( Boolean-Expression_2 )  
    Statement_2;  
else if(Boolean-Expression_N )  
    Statement_3;  
else  
    Statement_4;
```

تستخدم ال **if**:

إذا أردنا التحقق من أي شرط
على أي نوع من القيم أو العمليات



```
switch ( Controlling_Expression )  
{  
    case Label_1:  
        Statement_Sequence_1;  
        break;  
    case Label_2:  
        Statement_Sequence_2;  
        break;  
    case Label_N:  
        Statement_Sequence_N;  
        break;  
    default:  
        Default_Statement Sequence;  
}
```

تستخدم ال **switch**:

إذا أردنا التحقق من قيم محددة إما
من نوع **int** أو **char** أو **String**

