البرمجة باستخدام لغة الجافا

JAVA

البرمجة باستخدام لغة الجافا
Java Language

مقدمة :

تعتبر لغة الجافا من اللغات القوية جداً في مجال إنشاء التطبيقات المختلفة سواء كانت هذه التطبيقات تعمل منفردة على أجهزة الكمبيوتر الشخصي أو تطبيقات الإنترنت أو التطبيقات المختلفة للأجهزة المحمولة ، مثل الموبايل والمفكرات الإلكترونية وهكذا.

ولقد قامت شركة صن (Sun Microsystems) بأختراع وتطوير هذه اللغة. وأصبحت شركةصن مملوكة لشركة أوراكل وبالتالي انتقلت ملكية الجافا لأوراكل . وكان الهدف عند اختراع لغة الجافا هو عمل لغة **قادرة على** برمجة نظم التشغيل لجميع الأجهزة من حاسبات عملاقة (mainframes) إلى الأجهزة الصغيرة مثل مشغلات MP3 ولقد اختارت الشركة صورة فنجان القهوة لتمثيل هذه اللغة .

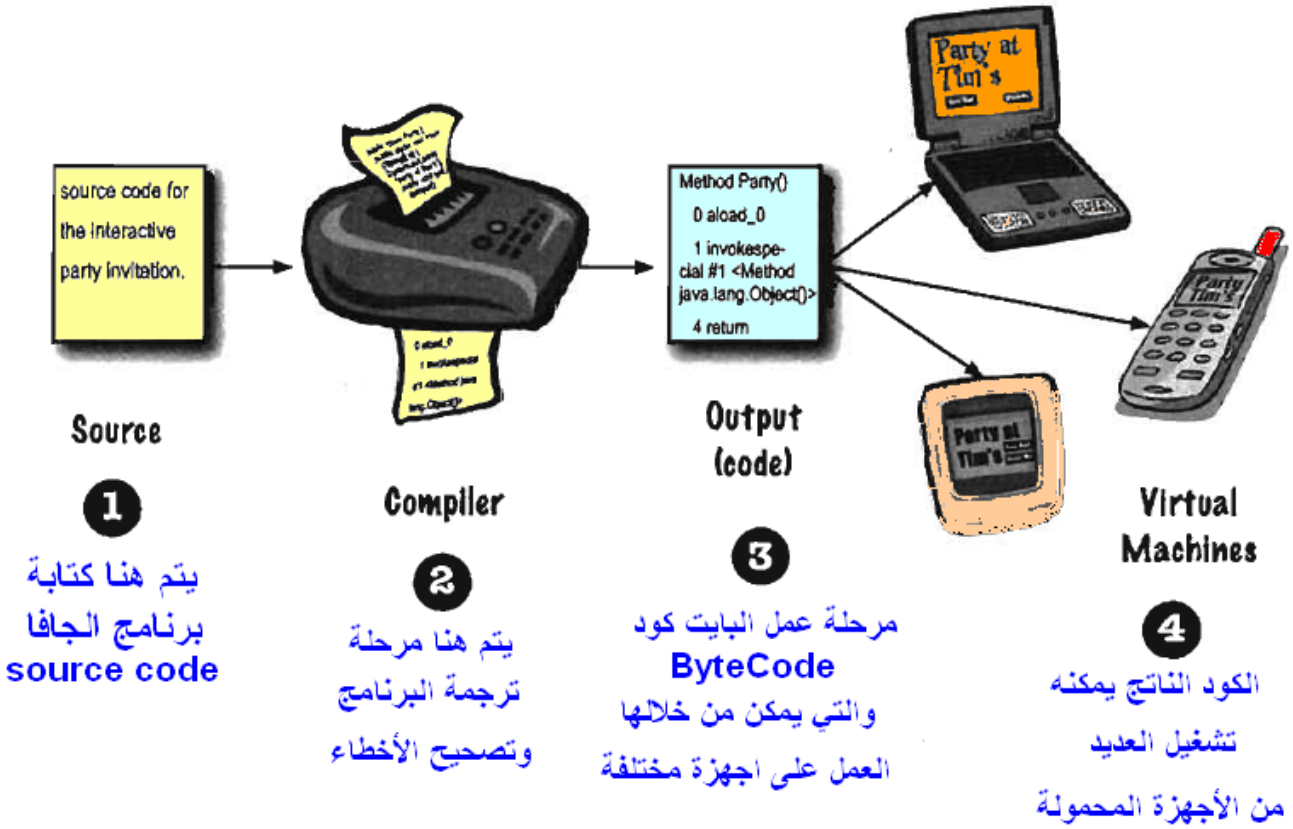
١ـ١ أسس البرمجة باستخدام لغة الجافا

قبل البدء في عملية البرمجة (أي كتابة البرنامج المطلوب تنفيذه) بلغة الجافا لابد من توافر العدة اللازمة (Tool Kit) . وهذه العدة عبارة عن البرامج اللازمة لعملية كتابة البرنامج نفسه ونقول أننا كتبنا برنامج بلغة الجافا . بعد ذلك تأتي عملية الترجمة لهذا البرنامج وهي ما نطلق عليها عملية الترجمة (compiling).

والحقيقة فأنه يوجد أكثر من طريقة لكتابة برامج الجافا وترجمتها نوجز منها :

١- استعمال المكتبة (**JDK**) وهي اختصار JAVA DEVELOPMENTE KIT من انتاج شركة صن مع أي محرر نصوص وليكن برنامج NotePad الموجود في الويندوز.
٢- استعمال برامج وسيطة تسهل عملية الكتابة والترجمة وتصحيح الأخطاء مثل برنامج (NetBeans – Jcreator ...).

وسوف نتناول في الجزء الخاص بالمعمل كيفية تثبيت هذه البرامج على جهاز الحاسب وكيفية التعامل معها. والشكل (١-١) يبين كيفية عمل لغة الجافا .



شكل (١-١)

ويوجد عدة نسخ للغة الجافا هي :-

- Java SE: هي اختصار ل Java Standard Edition يتم من خلالها دراسة اللغة وإنشاء التطبيقات المختلفة لتشغيلها على جهاز الحاسب (Desktop Application). وسوف تكون هي موضوع دراستنا في هذا الكتاب.
- Java EE: هي اختصار ل Java Enterprise Edition وهي تزودنا بالتطبيقات الكبيرة على مستوى الشركات الكبيرة.
- Java ME: هي اختصار ل Java Micro Edition فهي تخص الأجهزة اللاسلكية (wireless devices) بشكل عام يعني على أجهزة المحمول وغيرها.

١-١-١مميزات لغة الجافا

١- لغة الجافا غير مرتبطة بأنظمة التشغيل المختلفة Java Is Platform Independent.

٢- تعتمد على أسلوب برمجة الأهداف Object Oriented Programming .

٣-إنشاء برامج ذات واجهة مستخدم .

٤- تصميم برمجيات تستفيد من كل مميزات الإنترنت Java Applet .

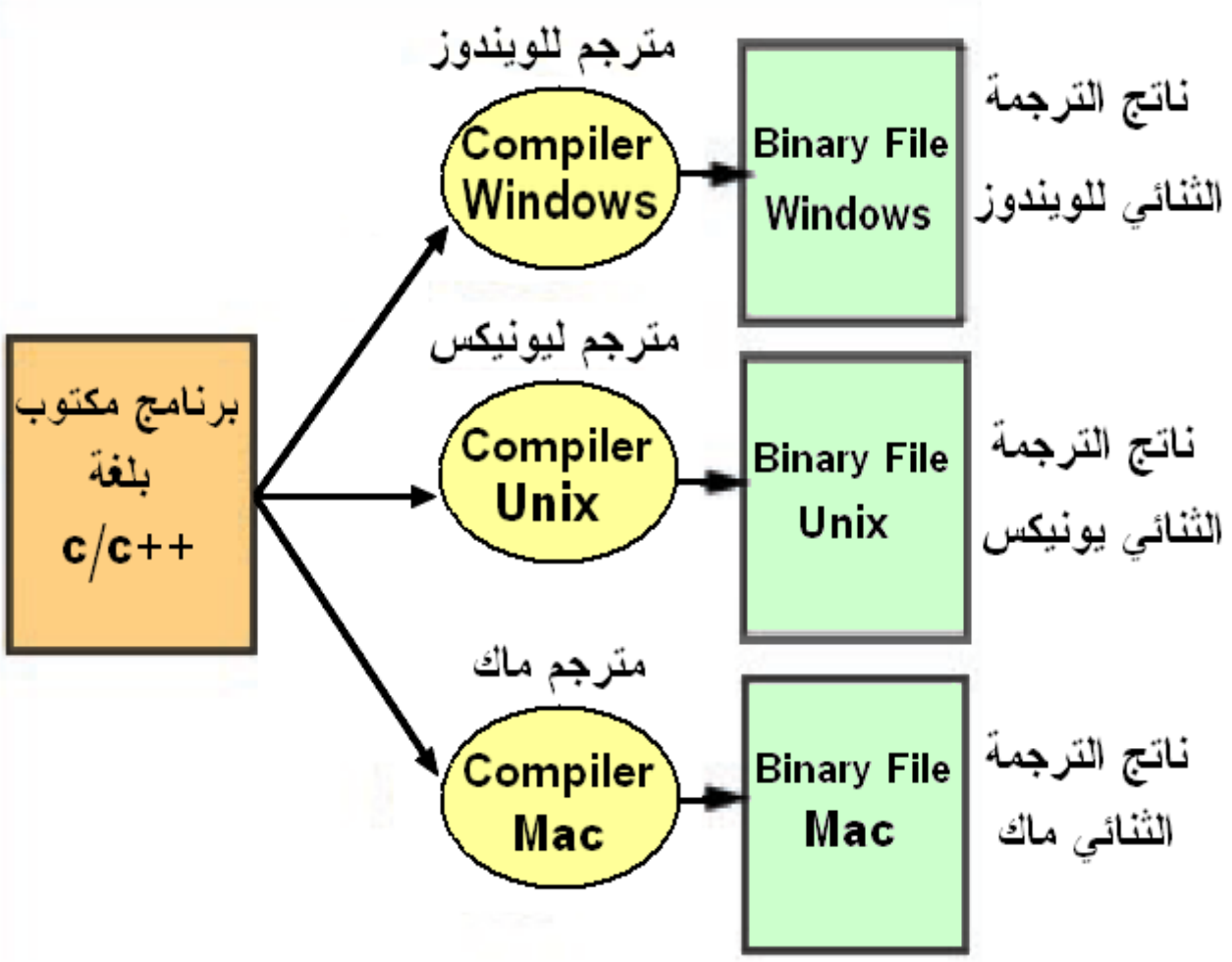
وفيما يلي شرح لأهم مميزات لغة الجافا كما ذكرناها في النقاط السابقة :

١- لغة الجافا غير مرتبطة بأنظمة التشغيل المختلفة Java Is Platform Independent

ومعنى ذلك إنه يمكن نقل البرامج (المكتوبة بلغة الجافا) بسهولة من نظام تشغيل إلى آخر.

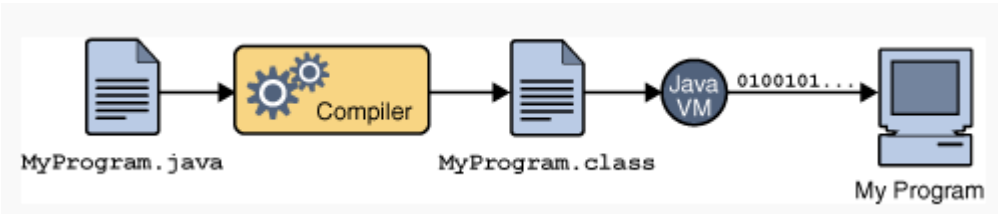
وفي المقابل يمكن القول إنه **لا** يمكن تشغيل برنامج WORD مثلا والخاص بنظام تشغيل ويندوز(WINDOWS) على جهاز حاسب آخر يعمل بنظام تشغيل مختلف مثل يونيكس (UNIX) أو نظام تشغيل لينيكس (LINUX) أو أي نظام تشغيل آخر غير نظام WINDOWS والمستخدم مع أجهزة الحاسبات المختلفة. ويرجع ذلك لأن برنامج WORD بشكل عام مكتوب بلغة (C++/C) والتي تعطي ملف من نوع EXE خلال عملية تسمى عملية الترجمة COMPILATION وبذلك يكون الملف الناتج مرتبطاً ارتباطاً كلياً بنظام التشغيل.

أما بالنسبة للغة الجافا فالوضع مختلف حيث يوجد وسيط بين البرنامج وبين نظام التشغيل وهذا الوسيط يسمى (Byte Code Interpreter) أي الترجمة على مستوى البايت. وكذلك يمكن تسميته بالآلة التخيلية للجافا (Java Virtual Machine). ويوضح الشكل (٢-١) خطوات تشغيل برنامج مكتوب بلغة (c أو c++) .

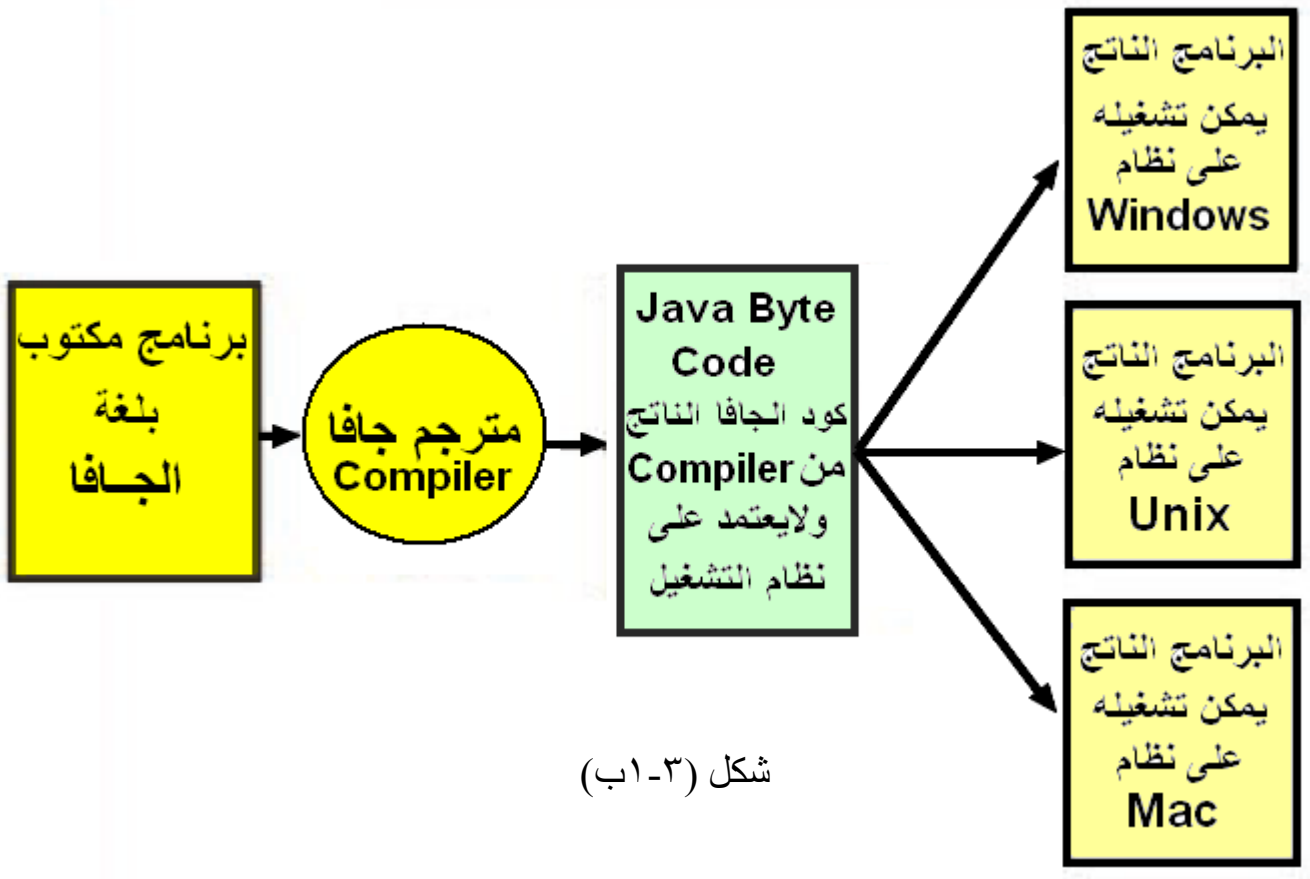


الشكل (٢-١)

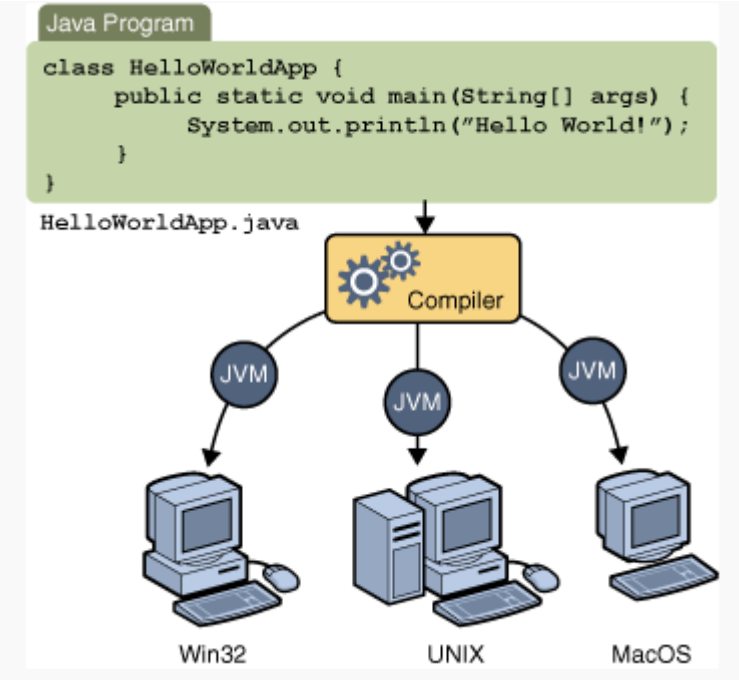
كما يوضح شكل (٣-١أ) و شكل (٣-١ب) و شكل (٣-١ج)خطوات تشغيل برنامج مكتوب بلغة الجافا .



شكل (٣-١أ)



شكل (٣-١ب)



شكل (٣-١ج)

٢- تعتمد لغة الجافا على أسلوب برمجة الأهداف Object Oriented Programming

حيث وفرت كثيرا من الجهد الذي كان يبذل باستخدام البرمجة التقليدية . فقد كانت البرمجة التقليدية توفر للمبرمج مكتبة من الدوال إضافة إلى تركيب تقليدي للبرنامج وعلى المبرمج أن يستعمل الدوال مع تركيب البرنامج لإنشاء التطبيقات المختلفة مما يضطره لكتابة السطور الكثيرة أكثر من مرة؛ و لقد كانت وحدة بناء البرنامج هي الدالة function. في حين أتت البرمجة بواسطة الأهداف بفكرة جديدة هي إنشاء عناصر متكاملة تحتوي على بيانات ودوال هي أساس إنشاء البرنامج. وبالتالي أصبحت وحدة بناء البرنامج وحدة كبيرة هي الفصيلة أو الفئة Class أو العنصر Object مما سهل واختصر الكثير من الوقت والجهد.

وسوف نتحدث على هذه النقطة بالتفصيل في الباب الثاني .

٣-إنشاء برامج ذات واجهة مستخدم رسومية .

يعتبر بناء واجهة المستخدم الرسومية من الأجزاء الهامة في البرنامج . حيث أن هذه الواجهات تعطي البرنامج شكلا معيناً ، كما أن استخدام مفاهيم وأجزاء موحدة في بناء الواجهات للعديد من البرامج المختلفة يعطي المستخدم قدرا كبيرا من الراحة اثناء استخدام البرامج ، كما أنه يقلل كثيرا من الوقت المستخدم لتعلمها . وقد تعرفنا في السنوات السابقة وأثناء استخدامنا للحاسب على واجهات رسومية كثيرة . مثل واجهات الويندوز والمستكشف للإنترنت وغيرها .

إن الأجزاء الرسومية الموجودة في لغة الجافا مرتبطة مباشرة مع الإمكانيات الرسومية للجهاز الذي يعمل عليه البرنامج . وبذلك فإن الواجهات الرسومية الموجودة في الجافا سوف تظهر بأشكال متباينة على الأجهزة المختلفة . أي أننا عندما نقوم بكتابة برنامج يقوم بعمل زر على نظام الويندوز فأن هذا الزر يأخذ نفس شكل الزر المستخدم في نظام الويندوز . ولكن اذا تم كتابته في نظام تشغيل آخر فإنه يأخذ شكل يتناسب مع نظام التشغيل المستخدم .

١-١-٢الشكل العام لبرنامج الجافا

البرنامج الآتي يبين الشكل العام لبرنامج الجافا ولا يهمنا هنا فهم كل جزئية في البرنامج فهذا سوف يتم في الدروس التالية :

ويقوم هذا البرنامج بطباعة جملة (Welcome to my World). وعند حفظ هذا البرنامج كما سنعرف لاحقا لابد وأن يتم تسمية الملف باسم Welcome.java. وكذلك يجب ان نراعي جيدا أن لغة الجافا هي لغة حساسة بالنسبة للأحرف فمثلا حرف (A) لا يساوي حرف (a). ويمكن تمثيل الهيكل العام لبرنامج الجافا بالشكل(٤-١) التالي :

شكل (٤-١)

١-٢ أنواع البيانات والمعاملات

١-٢-١ أولا :حروف لغة الجافا

تتألف حروف لغة الجافا مما يلي :

١- الحروف الأبجدية (Letters) وهي الحروف الكبيرة (Capital Letters) من A إلى Z وكذلك الحروف الصغيرة (Small Letters) من a إلى z.

٢- الأرقام العددية (Digits) من ٠ إلى ٩.

٣- الحروف الخاصة (Special Characters) وهي تلك الحروف التي ليست بأعداد أو بحروف أبجدية ولكنها تكون على هيئة رموز كالآتي:

(+,-,/,//,<,>,$,#,%,(),||,!,[],!=,,;;,”,.....) وتعد هذه الرموز بأنواعها المادة الخام التي تتكون منها مفردات لغة الجافا.

١-٢-٢ ثانيا الثوابت والمتغيرات Constants & variables:

أولا : الثوابت Constants:

وهى عبارة عن قيم ثابتة يراد الاحتفاظ بها طوال البرنامج ولا تتغير قيمتها أبدًا.

وتنقسم الثوابت في لغة الجافا إلى:-

١- ثوابت عددية Numeric Constants

٢- ثوابت رمزية Non-numeric Constants

١- الثوابت العددية:

يمكن تمثيل الثوابت العددية في لغة الجافا كالآتي:-

(أ)-الثابت العددي الصحيح integer :

- هو عبارة عن عدد مكون من الأرقام من (٠ إلى ٩).

- لا يحتوي على فاصلة عشرية.
- يمكن أن يحوى الإشارة (+ أو -).

ومن أمثلة الثابت العددي الصحيح (٠، ١٢، ١٠٠٠، ٢٠-،.....).

كما يمكن تصنيف الأعداد الصحيحة في لغة الجافا حسب طولها والسعة التخزينية لها في الذاكرة كما يلي :-

- الثوابت الصحيحة (١٩٦٧٩، ٤٠٠٠٠) تسمى ثوابت صحيحة طويلة long int.
- الثوابت (١٦-، ٩٠ ، ٥٥) تسمى ثوابت صحيحة قصيرة short int.
- الثوابت (٢٠٠٠٠ ، ٩٦٧) تسمى ثوابت صحيحة بدون إشارة unsigned int.

والفرق بين الثوابت الطويلة والقصيرة هو في عدد الوحدات التخزينية المطلوبة لكل نوع في الذاكرة، فالثوابت الطويلة تأخذ حيزًا أكبر، والقصيرة توفر عدد الوحدات التخزينية المستعملة، أما الثوابت الصحيحة بدون إشارة فإن استعمالها يوفر وحدة تخزينية تستعمل للإشارة.

(ب)-الثابت العددي العشري Floating Constant

- هو عدد مكون من الأرقام من (٠ إلى ٩)
- يجب أن يحتوي على فاصلة عشرية
- يمكن أن يحوي الإشارة (+، -)

ومن أمثلة الثوابت العددية الحقيقية

(٤٢١,٥، ١٠,٥٥، ٦٧,٩٩-، ٠٠٠٠٠٠)

٢- الثوابت الرمزية Non-Numeric:

وهى عبارة عن رموز اللغة وتتكون من الحروف والأرقام وتكون بين علامتي تنصيص أو اقتباس.

ومن الأمثلة على الثوابت الرمزية ما يلي:-

("name" - "Khaled" - "12345" - "30+40 ")

ونلاحظ أن هناك ثوابت رمزية تحتوي على أرقام وهى في الحقيقة قيم حسابية لا يمكن إجراء أي عملية حسابية عليها بل يتم طبع الأرقام أو الرموز كما هي.

وإذا أردنا أن نضع قيمة سوف تظل ثابتة داخل البرنامج في مكان في الذاكرة فأننا نستخدم العبارة final للإعلان أن هذه القيمة ستظل ثابتة طوال تنفيذ البرنامج مثل:

```
final int TABLE_SIZE = 41;

final float PI = 3.14159;
```

ويجب مراعاة أن اسم الثوابت constants يكون بالأحرف الكبيرة كاملاً و يفصل بين الكلمات كما يتم في المتغيرات مع ملاحظة أن الثوابت يتم تعريفها على أنها final .

مثال لأسماء الثوابت :

EXIT_ON_CLOSE

LEFT MY_NAME

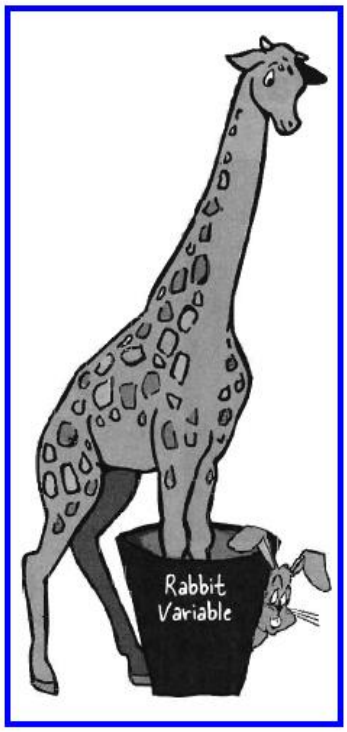
CENTER CLASS_VERSION

BOTTOM MATH_PI

TOP

ثانيا : المتغيرات Variables:

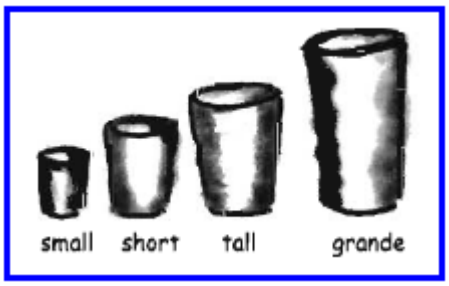
وهي عبارة عن أسماء تحجز مواقع في الذاكرة حتى يتمكن البرنامج من تخزين البيانات فيها. أو بمعنى آخر يمكن القول أن المتغيرات هي عبارة عن وعاء يمكن تحميله بقيمة وهذا الوعاء يتغير حجمه حسب القيمة التي سوف توضع فيه. أي أن هذه الأوعية متغيرة الحجم حسب قيمة ما تحمله والشكل (٥-١) يوضح هذه الفكرة: ونلاحظ في هذا الشكل أنه إذا كان هناك وعاء على حجم الأرنب فأنه لا يستطيع بأي حال من الأحوال أن يحمل شيء كبير بحجم الزرافة مثلا . لذا لابد من عمل وعاء كبير يستطيع حمل الزرافة. ومن هذه الفكرة نستطيع القول أننا يجب أن نحدد حجم مخزن الذاكرة بالسعة المناسبة حسب حجم المتغير الذي سيتم تحميله بها. وعلى هذا الأساس نستطيع القول بأنه هناك أحجام مختلفة من الأوعية كما يظهرها الشكل (٥-١).



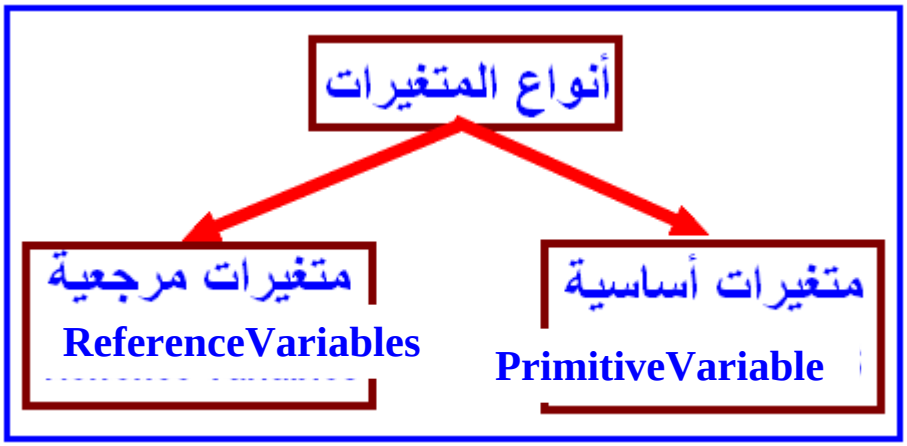
شكل (٥-١)

قواعد تسمية المتغيرات:

- يمكن أن يبدأ الاسم بالحرف أو الشرطة السفلية (_) under score أو علامة الدولار ($) ولايسمح ببدء التسمية برقم ولكن يمكن أن نضع رقم بعد الحرف.
- لا يمكن تسمية المتغير بإحدى الكلمات المحجوزة للغة الجافا والجدول شكل (٦-١) يبين هذه الكلمات .
- اسم المتغير variable يكون بالأحرف الصغيرة لكل الأحرف ويلاحظ عدم وجود أقواس.



ويمكن تقسيم المتغيرات بصفة عامة إلى نوعين أساسين:

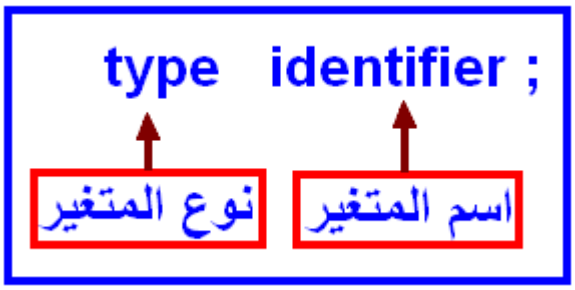


وتنقسم المتغيرات الأساسية (Primitive Variable) إلى نوعين:-

١- متغيرات رمزية (حرفية).

٢- متغيرات عددية.

ولابد قبل استخدام المتغير من الإعلان عنه ويتم ذلك كالتالي:



أ- المتغيرات الحرفية Char:

وتتضمن الحروف بكافة أشكالها والرموز والفراغات (مسافة فارغة) مثل:

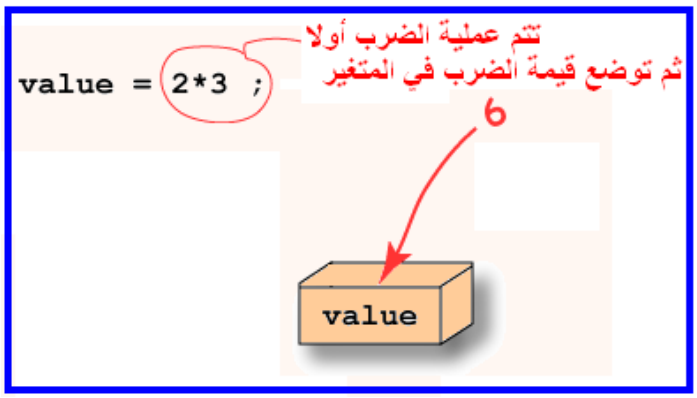
```
char a,b;
a= 'a';    char var1;
b=' ';      var1=' ' ;
```

ب- المتغيرات العددية Numeric Variables:

أ) المتغيرات العددية الصحيحة Integer:

تتضمن قيم عددية صحيحة يمكن أن تأخذ قيمة تصل إلى ٣٢٧٦٧ وتكتب على الشكل التالي:

```
int a;     a=100;
int value ;  value=2*3;
```



ب) المتغيرات العددية الحقيقية Floating Point:

تتضمن جميع الأعداد الحقيقية وتكتب على الشكل التالي:

```
float x;      x=5.2;
```

كما يجوز تعريف المتغير وتخصيص قيمة له في نفس السطر كالتالي: `float x= 5.2`

وهنا يجب علينا الانتباه لجملة الإعلان والتخصيص السابقة float x= 5.2 ، فهنا رغم أننا عرفنا المتغير على أنه متغير من نوع float حقيقي أي يقوم الحاسب بحجز مكان له في الذكرة مقداره ٣٢ بت حسب الجدول شكل (٧-١) إلا أن الحاسب يعتبره من النوع double أي يحجز له مكان ٦٤ بت. والمبرمج الذكي هو الذي يجعل برنامجه يحتل أقل مساحة في الذاكرة.

وللتغلب على المشكلة السابقة يتم الإعلان والتخصيص كالتالي:

```
float x= 5.2 f
```

أي يتم وضع حرف (f) بعد الرقم لكي يتم حجز مكان له في الذاكرة مقداره ٣٢ بت وبذلك نكون قد وفرنا في الذاكرة المستخدمة.

ج) المتغيرات العددية الحقيقية الطويلة Double:

هي نفس المتغيرات العددية الحقيقية ولكن يمكن تمثيلها في خمسة عشرة خانة وتكتب على الشكل التالي:

```
double x;
```

ويجب الإعلان عن المتغيرات في بداية البرنامج .

والجدول شكل(٦-١) الآتي يوضح تطبيق بعض قواعد تسمية المتغيرات والإعلان عنها:

```
int myPay, yourPay; // OK
long good-by ;  // اسم متغير خطأ لأنه يحتوي على "-"الشرطة
short shrift = 0;  // OK
double bubble = 0, toil= 9, trouble = 8 // خطأ لأنه لاينتهي بـ (;)
byte the bullet ;   // تعريف خطأ يحتوي على مسافات
int  double;   // double تعريف خطأ يحتوي على كلمة محجوزة
char thisMustBeTooLong ;  //تعريف صحيح ولكن اسم المتغير طويل جدا

int  8ball;   // تعريف خطأ اسم المتغير يبدأ برقم
float a=12.3; b=67.5; c= -45.44; //تعريف خطأ لأنه يستخدم الفاصلة المنقوطة
```

الجدول شكل(٦-١)

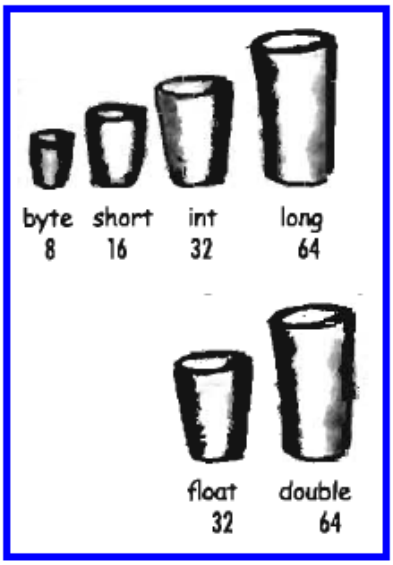
والجدول شكل (٧-١) الآتي يبين أنواع البيانات والمتغيرات في لغة الجافا :

◄ انواع المتغيرات :

نوع المتغير	المتغير	الحجم
المتغيرات الصحيحة	byte	8 bit
	short	16 bit
	int	32 bit
	long	64 bit
المتغيرات الكسرية	float	32 bit
	double	64 bit
المتغيرات النصية	char	16 bit
	String	-
المتغير المنطقي	boolean	1 bit

الجدول شكل (٧-١)

ومن الجدول السابق يمكن تمثيل أحجام البيانات في الذاكرة كما في الشكل (١-٨):



الشكل (١-٨)

ملاحظة:

كل أنواع المتغيرات التي تم شرحها أعلاه تسمى متغيرات أساسية primitive type وسوف نتعرف فيما بعد على نوع آخر يسمى المتغيرات المرجعية Reference type .

١-٢-٣ العمليات الحسابية والمنطقية في لغة الجافا

❖ الجدول شكل(٩-١) التالي يبين أهم العمليات الحسابية في لغة الجافا :

◄ المعاملات الحسابية :

المعامل	الوصف	مثال بلغة جافا
+	جمع	X = A + B
-	طرح	X = A - B
*	ضرب	X = A * B
/	قسمة	X = A / B
+=	جمع ثم إسناد	(A+=B) = (A = A + B)
-=	طرح ثم إسناد	(A-=B) = (A = A - B)
*=	ضرب ثم إسناد	(A*=B) = (A = A * B)
/=	قسمة ثم إسناد	(A/=B) = (A = A / B)
%=	باقي القسمة	(A%=B) = (A = A % B)
++	زيادة بمقدار واحد	(A++) = (A = A + 1)
--	نقصان بمقدار واحد	(A--) = (A = A - 1)
%	باقي القسمة	X = A%B

الجدول شكل (٩-١)

وهذه المعاملات قد تمت دراستها باستفاضة في منهج الصف الثاني.

والجدول شكل (١٠-١) التالي يبين العمليات المنطقية:

◄ المعاملات المنطقية :

المعاملات بالشكل الرياضي	شكل العمليات في لغة الجافا	مثال على الشرط	معنى الشرط
=	=	x==y	x تساوي y
≠	!=	x!=y	x لاتساوي y
>	>	x>y	x اكبر من y
<	<	x<y	x اصغر من y
≥	>=	x>=y	x اكبر من او تساوي y
≤	<=	x<=y	x اصغر من او تساوي y
And	&& او &	if (x ==1 & y== 1)	اذا تحقق كلا الشرطين
Or	|| او |	if (x = =1 | y== 1)	اذا تحقق احد الشرطين
Xor	^	if (x == 1 ^ y== 1)	-

الجدول شكل (١٠-١)

❖ **أما الجدول التالي شكل (١١-١) فهو يبين الكلمات المحجوزة في لغة الجافا :-**

الكلمات المحجوزة في الجافا

public	abstract	finally	boolean	return
float	break	short	for	Static
if	byte	case	implements	Super
catch	switch	import	char	int
synchronized	instanceof	this	class	Throw
continue	interface	default	long	else
throws	native	true	transient	New
do	package	double	null	try
private	extends	while	protected	Final
volatile	void	false	--------	--------

الجدول شكل (١١-١)

ومعنى الكلمات المحجوزة أي أنها هي الكلمات والأوامر التي تعبر وتستخدم في لغة الجافا ولا يجوز استخدامها في غير ذلك كأسماء لمتغيرات مثلا ولذلك فهي محجوزة لمفردات اللغة فقط .

١-٢-٤ دالة الإخراج في لغة الجافا

System.out.print

وهي من الدوال الهامة في لغة الجافا وهي تقوم بطباعة المخرجات سواء كانت عددية أو حرفية .

ولتوضيح عمل هذه الدالة سوف يتم دراسة بعض الأمثلة :

مثال(١)المطلوب عمل برنامجا يقوم بطباعة العبارة HelloEgypt.

```
public class HelloEgypt {

   public static void main ( String s[ ] ) {

            System.out.print("Hello Egypt ") ;

   }
}
```

شرح البرنامج

- **السطر الأول**

```
public class HelloEgypt {
```

وهذا هو السطر الأول في البرنامج وهو يتكون من:

public وهي دائما تسبق تعريف الفئة أو الفصيلة Class وهي تعني أن هذه الفئة أو الفصيلة عامة أي يمكن لأي فئة أخرى في البرنامج استخدام عناصر هذه الفئة .لأن برنامج الجافا قد يتكون من أكثر من فئة class.

class وهنا يتم بداية الفصيلة

HelloEgypt وهذا هو اسم الفصيلة ولقد تم تسميته هنا بالاسم الذي نريده .

☚ ولابد هنا أن نشير إلى نقطة هامة جدًا وهي أنه عند حفظ ملف الجافا لابد أن يتم حفظه بنفس اسم الفصيلة وبنفس شكل الحروف والمسافات وفي مثالنا هذا سيكون الاسم HelloWorld.java.

{ وهذا هو قوس بداية تعريف الفصيلة .

- **السطر الثاني**

```
Public static void main ( String s[ ] ) {
```

وهنا سوف يتم شرح الكلمات المطلوبة حاليا والباقي سوف نتناولها فيما بعد:

void ومعناها أن الدالة بعد تنفيذ البرنامج لن تعود بأي قيم.

main وهي تعتبر نقطة البداية لوظيفة الفصيلة main method .

`( String s[ ] )` الجملة الموجودة داخل قوسي البداية للدالة main وهي **String s[ ]** تعني مصفوفة من النوع الحرفي وتسمى s لتخزين جملة الطباعة في البرنامج .

وكما قلنا من قبل أن لغة الجافا هي لغة حساسة لحالة الأحرف لذلك يجب ملاحظة أن حرف **S** في كلمة String يجب أن يكون حرفاً كبيراً (capital letter) وإلا سيعطى البرنامج خطأ عند الترجمة .

- **السطر الثالث** `System.out.print ("Hello Egypt") ;`

`System.out.print` وهذا هو أمر الطباعة في لغة الجافا وسوف نتناوله بالتفصيل في الأمثلة القادمة .

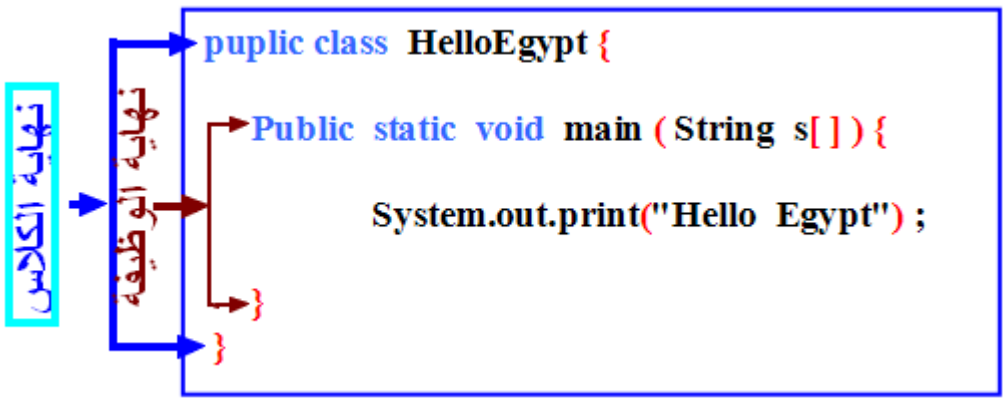
وهنا يجب أن نلاحظ أن حرف S يجب أن يكون كبير (Capital letter)

`("Hello Egypt")` وهذا هو النص المراد طباعته ويجب أن يوضع بين علامتي تنصيص (" ") وقوسين

`;` لابد وأن تنتهي كل جملة (سطر أو أمر) بعلامة (;)

```
}
}
```

بعد ذلك يتم إنهاء البرنامج بقوسي النهاية حيث يمثل القوس الأول نهاية الوظيفة method للدالة main والقوس الآخر يمثل نهاية الفصيلة class . والشكل الآتي يبين ذلك :



وكما ذكرنا سابقاً ان وظيفة هذا البرنامج هو طباعة الجملة Hello Egypt .

ولتشغيل هذا البرنامج يجب أن يتم له عملية ترجمة أولاً.

كما ذكرنا سابقا يجب تسمية البرنامج بالاسم الموجود أمام جملة class وهو Hello World .

خطوات تنفيذ البرنامج

١- نجري له عملية ترجمة كالآتي:

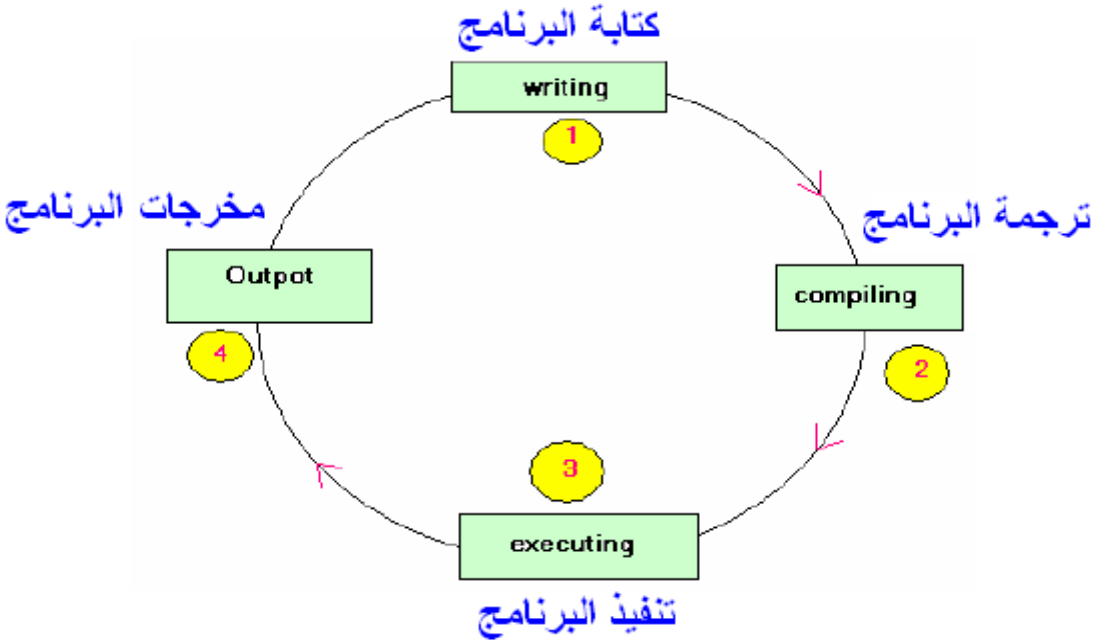
إذا لم يكن هناك أخطاء لا تظهر أي رسالة ومعنى ذلك أن البرنامج صحيح لغويا وهنا يتم عمل ملف كلاس أي Hello**Egypt**.class .

٢- بعد ذلك نقوم بعملية تشغيل البرنامج كالآتي:

فتظهر على الشاشة عبارة **HelloEgypt**

وبذلك يكون قد تم تنفيذ البرنامج.

وكما هو معروف فأن خطوات كتابة وتنفيذ أي برنامج يمكن أن تكون كما بالشكل (١-١٢)



شكل(١٢-١)

ويمكن إضافة متسلسلات الهروب مع جملة الطباعة للحصول على عدة أشكال من المخرجات

كما هو موضح بالجدول شكل (١٣-١) التالي:

الحرف الخاص	الوصف
\n	سطر جديد. يضع المؤشر في بداية السطر التالي
\t	مسافة أفقية. تحريك المؤشر مسافة معينة إلى النقطة التالية في السطر
\r	carriage return. يضع المؤشر في بداية السطر الحالي ولا يتقدم إلى السطر التالي ، وأي حرف يطبع يتم طباعته على حرف سابق تم كتابته في نفس السطر
\\	شرطة خلفية. إظهار "\" في الخرج
\"	علامة تنصيص مزدوجة. إظهار علامة التنصيص المزدوجة

شكل (١٣-١)

أمثلة على جملة الطباعة

مثال(٢): ماهي مخرجات البرنامج التالي :

```
public class  Welcome {

   public  static  void  main ( String  s[ ] ) {

      3    System.out.print("Welcome to") ;

      4     System.out.print("World") ;

   }
}
```

نلاحظ أنه عند تنفيذ هذا البرنامج ستظهر العبارة (**Welcome to World**) على سطر واحد وذلك تبعا لعبارتي الطباعة في السطر (٣،٤)

☚ أما اذا اردنا أن تكون المخرجات على سطرين مختلفين فيتم اضافة حرفي (ln) على العبارة print ومعناها الانتقال إلى سطر جديد (new line) ويتم ذلك في السطر الثالث كالآتي :

```
3   System.out.println ("Welcome to") ;

4     System.out.print("World") ;
```

فتكون مخرجات البرنامج كالتالي:

```
Welcome to
World
```

☚ ويمكن تنفيذ نفس شكل المخرجات السابقة بسطر واحد وذلك عن طريق اضافة (\n) وتعني الانتقال إلى سطر جديد ويكون شكل البرنامج كالتالي :

3 **System.out.print ("Welcome to\n World") ;**

فتكون شكل المخرجات كالشكل السابق:

```
Welcome to
World
```

وفي هذه الحالة يتم الغاء السطر رقم (٤)

☚ أما اذا أردنا طباعة عدة أسطر متتالية بأمر طباعة واحد فيتم ذلك بتكرار (\n) كالتالي:

System.out.print ("One\n Two \n Three \n Four") ;

فتكون المخرجات كالتالي:

```
One
Two
Three
Four
```

☚ أما اذا أردنا أن تكون المخرجات على مسافات أفقية متساوية فأننا نستخدم (\t) كالتالي:

```
System.out.print ("One\t Two \t Three \t Four") ;
```

وتكون المخرجات كالتالي:

```
One    Two    Three    Four
```

مثال (٣): أكتب برنامج يقوم بجمع العددين (١٦+٥)

ويتم ذلك بكتابة الأرقام المراد جمعها داخل أقواس جملة print ولكن من دون علامتي تنصيص لأن علامتي التنصيص تكون دائما لطباعة الحروف وحتى اذا تم كتابة ارقام داخل علامتي التنصيص فأنها تعامل معاملة الحروف أي لا يمكن اجراء أي عمليات حسابية عليها .

ويكون شكل عبارة print كالتالي:

```
System.out.print (5+16) ;
```

ويكون الناتج (٢١) .

١-٢-٥ التعليقات Comment

إن أي مبرمج يحتاج في بعض الأحيان إلى اضافة بعض التعليقات والملاحظات الخاصة به و التي لا يتم تنفيذها في البرنامج ولكن فقط تذكره بالغرض من الأوامر التي يقوم بكتابتها .

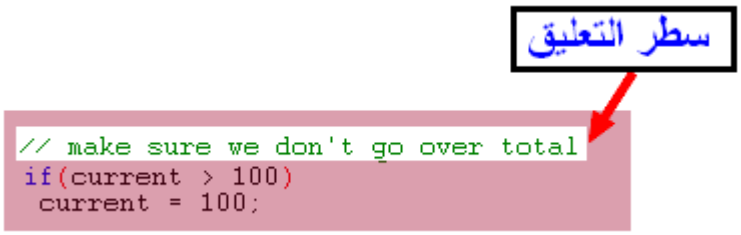
ويمكن تعريف التعليقات كالآتي :

أنها الأسطر التي يتجاهلها مترجم الجافا، و لكنها تجعل البرنامج أسهل قراءة للمبرمج نفسه. بعبارة أخرى، أنها مجموعة الملاحظات التي يضعها المبرمج في برنامجه لتسهيل قراءته.

ومن أنواع التعليقات في الجافا :

١- التعليق بسطر واحد

ويكون هذا السطر مسبوقا بعلامتي (//) كالآتي :



أو يمكن كتابة التعليقات بجانب أسطر البرنامج كما يلي :

```
g.fillArc(0, 0,      // start
capWidth, height,    // size
90, 180);            // angle
```

٢- التعليق بعدة أسطر

وفي هذه الحالة يمكن كتابة تعليق مكون من عدة أسطر كما يلي:
ويكون التعليق بين علامتي (/* التعليق */)

```
/*
   see if we are serializing or deserializing.
   The ability to deserialize or serialize allows
   us to see the bidirectional readability and writeability
 */

if (args.length == 1) {
    if (args[0].equals("-d")) {
deserialize = true;
    } else if (args[0].equals("-s")) {
```

مثال (٤): أكتب برنامج يقوم بجمع عددين أحدهما صحيح والآخر حقيقي.

```
// this programe add two numbers
puplic class  Addition {
    Public  static  void  main ( String  s[ ] ) {
    int  a=15 ;     // first number
     float  b=12 ;  // seconde number
      float  c ;
     c= a+b ;
            System.out.print("The Result =" + c ) ;
    }
}
```


ويلاحظ في السطر الأخير للبرنامج أنه تم كتابة (+c) . وذلك لطباعة محتويات المخزن (c) أمام علامة (=) وعند تنفيذ البرنامج سوف تكون المخرجات كالتالي:

The Result = 27

مثال (٥) ما هو ناتج مخرجات البرنامج التالي:

في هذا المثال تم استخدام عدة أشياء منها :

- الطرق المختلفة للإعلان عن المتغيرات.
- العمليات الحسابية المختلفة.
- جملة الطباعة

```
public class ArithOper
{
        public static void main(String arg[])
        {
            int a=15;
            int b=4;
            int x,y,z,v,u;          // جمل الأعلان والتخصيص

            x=a+b;
            y=a-b;                  // جمل المعاملات والمؤثرات الحسابية
            z=a*b;
            v=a/b;
            u=a%b;

            System.out.println("a+b="+x);
            System.out.println("a-b="+y);
            System.out.println("a*b="+z);
            System.out.println("a/b="+v);
            System.out.println("a%b="+u);
        }
}
```

/* جمل طباعة المخرجات */

وتكون مخرجات البرنامج على الشكل التالي:

```
a + b = 19
a  - b  =11
a  * b = 06
a / b   = 3
a %b = 3
```

مثال (٦) ماهي مخرجات البرنامج التالي :

هذا البرنامج تطبيق على المؤثرات الأحادية

```
public class UnaryOper
{
    public static void main(String arg[])
    {
        int a,b,i,j;
        i=j=5;
        a=i++ * 3; // الزيادة بعد العملية الحسابية
        b=++j *3; // الزيادة قبل العملية الحسابية
        System.out.println("a = "+a+"\n"+"b = "+b);
    }
}
```

وتكون مخرجات البرنامج على الشكل:

```
A=15
B=18
```

مثال (٧) ماهي مخرجات البرنامج التالي :

```
public class UnaryOper1
{
    public static void main(String arg[])
    {
        int x1,x2,z=10;
        x1=z--;   // x1=10 هنا مازالت قيمة
        System.out.println("x1 = "+x1);
        x2=--z;  // x2=8 هنا أصبحت قيمة
        System.out.println("x2 = "+x2);

    }
}
```

ونلاحظ في هذا المثال أن قيمة x1 لازالت تساوي ١٠ ولا يتم انقاص الواحد منها إلا بعد الخروج من هذه الخطوة وتصبح قيمة z الجديدة z=9

ويكون ناتج البرنامج كالتالي:

```
X1=10
X2=8
```

١-٢-٦ دالة الإدخال Input

بطبيعة الحال لا يخلو أي برنامج ذو فائدة من جملة الإدخال ، فهي الجملة التي تربط البرنامج بالعالم الخارجي وهي الوسيلة الوحيدة التي يستطيع فيها المستخدم إدخال القيم عن طريق لوحة المفاتيح للحاسب ، حتى يقوم بمعالجة هذه القيم سواء كان البرنامج (برنامج حسابات – قاعدة بيانات .....) .

والحقيقة أن لغة الجافا تحتوي على أكثر من طريقة لإدخال البيانات منها ما هو مناسب لتطبيقات الويندوز ومنها ما هو مناسب لبرامج الدوس (Console Application).

وعبارة الإدخال التي سوف نستخدمها هنا هي العبارة (Scanner) وهي عبارة عن فصيلة من فصائل لغة الجافا وهي موجودة في مكتبة تسمى (java.util) ولابد لاستخدام عبارة الإدخال (Scanner) أن نستدعيها من مكتبات لغة الجافا ويتم ذلك كالتالي :

```
Import.java.util ;
```

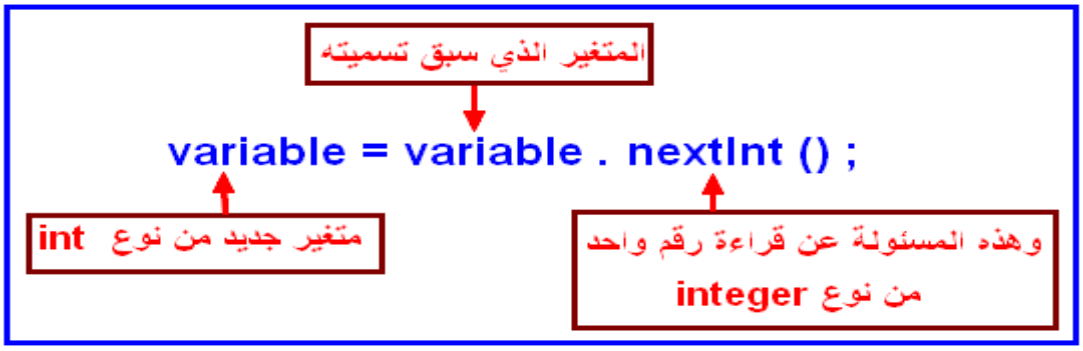
وبذلك يمكننا استخدام عبارة الإدخال (Scanner) في البرنامج المطلوب .

ويكون ذلك بحجز مخزن لمتغير في الذاكرة ليحتوي الرمز المدخل عن طريق لوحة المفاتيح كالتالي:

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

اي اسم متغير
(a , b , sum .....)

ثم بعد ذلك يتم كتابة العبارة التالية :



والمثال التالي يوضح طريقة عمل عبارة الإدخال (Scanner)

مثال (٨)
أكتب برنامج لجمع رقمين على أن يتم إدخال الرقمين من لوحة المفاتيح ثم طباعة الناتج على الشاشة.

```
1 import java.util.Scanner  ;   ← استدعاء المكتبة التي تحتوي على جملة الأدخال

2   public class magdy{

3  public static void main(String s[] ){
4  int  a,b,sum;   ← الأعلان عن ثلاث متغيرات

5  Scanner Keyboard=new Scanner(System.in);   ← تهيئة البرنامج لعملية الأدخال

6   System.out.println("Enter first number");
7   a=Keyboard.nextInt();   ← هنا يتم ادخال الرقم الأول وتخزينه في المتغير a

8   System.out.println("enter seconde number");
9   b=Keyboard.nextInt();   ← هنا يتم ادخال الرقم الثاني وتخزينه في المتغير b

10  sum=a+b;
11  System.out.println("the sum is="+sum);
    }
}
```

شرح البرنامج

١- في السطر **الأول** تم استدعاء المكتبة التي تحتوي على جملة الإدخال Scanner

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

٢- أما في السطر **الرابع** فقد تم الإعلان عن ثلاث متغيرات من النوع integer وهي المتغير a لتخزين الرقم الأول والمتغير b لتخزين الرقم الثاني والمتغير sum لتخزين ناتج عملية الجمع.

```
4 int  a,b,sum;
```

٣- أما السطر الخامس فهو يعمل على تهيئة الحاسب لاستقبال مدخلات من لوحة المفاتيح ولقد تم تسمية مخزن مؤقت تم تسميته Keyboard أو يمكن تسميته بأي اسم ويتم فيه تخزين القيمة المدخلة مؤقتا تمهيدا لنقلها لمتغير اخر سيكون هنا a أو b ولاحظ كلمة (System.in) أصبح بجوارها كلمة in دلالة على عملية الإدخال

```
5 Scanner Keyboard=new Scanner(System.in);
```

٤-أما السطر السادس فوظيفته هي طبع رسالة على الشاشة تخبر المستخدم بإدخال الرقم الأول.

٥- أما في السطر السابع فيتم إدخال الرقم الأول ثم يخزن مؤقتاً في المخزن Keyboard ثم ننقل أو نخصص القيمة الموجودة في المخزن Keyboard وهي هنا الرقم الأول ونضعها في المخزن a.

```
7 a=Keyboard.nextInt();
```

٦- أما في السطر الثامن فهو يكرر العملية لطلب الرقم الثاني.

٧- وفي السطر التاسع يتم إدخال الرقم الثاني كما سبق ولكن يتم تخزينه هذه المرة في المخزن b.

٨- أما السطر العاشر فيتم فيه عملية الجمع ووضع الناتج في المخزن sum.

٩- وفي السطر الحادي عشر يتم طباعة قيمة الجمع على الشاشة.

ملاحظات هامة عن البرنامج

١- يجب الملاحظة جيدا أن هناك كلمات لابد وأن يكتب الحرف الأول منها بحروف كبيرة Capital letter مثل الكلمات في هذا المثال (System , Scanner , nextInt) .

٢- يجب أن تكون اسماء المتغيرات واضحة حتى يتم فهم البرنامج جيدا.

٣- بالنسبة للأرقام المدخلة يجب أن تكون من النوع الصحيح فقط integer و هذا يكون في مثالنا فقط لأننا طلبنا منه ذلك في برنامجنا وذلك في السطر السابع عن طريق عبارة

(**nextInt**) فالحروف الثلاثة ذات اللون الأحمر (**Int**) والتي جاءت بعد كلمة next هي المسئولة عن المدخلات يجب أن تكون من النوع الأرقام الصحيحة وهي لها عدة حالات:

حالات العبارة (next) :

والجدول شكل (١٣-١) التالي يوضح الحالات المختلفة للعبارة (**next**)

شكل عبارة next	مثال EXAMPLE
Int_Variable = Object_Name.nextInt()	int number; ادخال عدد صحيح number = keyboard.nextInt();
Double_Variable = Object_Name.nextDouble()	double cost; ادخال عدد ذو دقة مضاعفة cost = keyboard.nextDouble();
String_Variable = Object_Name.next()	String word; ادخال حرفيات word = keyboard.next();
String_Variable = Object_Name.nextLine()	String line; ادخال حرفيات line = keyboard.nextLine();

الجدول شكل (١٣-١)

تشغيل البرنامج السابق

عند تشغيل البرنامج السابق يظهر الآتي :

```
Enter first number
20
enter second number
30
The sum is = 50
```

١- السطر الأول يطلب منك إدخال الرقم الأول
وهنا تم إدخال العدد ٢٠.

٢- السطر الثاني يطلب منك إدخال الرقم الثاني
وهنا يتم إدخال العدد ٣٠.

٣- أما السطر الأخير فيظهر النتيجة وهي حاصل الجمع ٥٠.

ويجب ملاحظة أنه عند إدخال الرقم الثاني يجب الضغط على مفتاح Enter أو ترك مسافة واحدة.

مثال (٩)
أكتب برنامج تقوم من خلاله بإدخال اسمك فيطبع عبارة ترحيب بك

```
import java.util.Scanner;
   public class  Fahad
   public static void main(String s[] ){

 Scanner Keyboard=new Scanner(System.in);
 System.out.println("Enter your name");

6  String a=Keyboard.next();   ← أنظر الجدول السابق next

 System.out.println("Welcome\t"+a);

     }
 }
```

ونلاحظ هنا أن عبارة الإدخال لم تتغير كثيرا عن البرنامج السابق والذي تم فيه إدخال الأرقام، إلا اختلافا بسيطاً في السطر السادس.
وقد تم عمل متغير حرفي من نوع String هو المتغير a والذي يتم فيه تخزين الحروف المدخلة من لوحة المفاتيح كما نلاحظ تغير العبارة next ولقد كتبت منفردة بدون أي اضافات (أنظر الجدول السابق الذي يوضح وظائف next) .

تشغيل البرنامج

عند تشغيل البرنامج يطلب منك إدخال أسمك فنقوم بإدخال الاسم من لوحة المفاتيح فيقوم بعد ذلك بطباعة عبارة الترحيب كالتالي:

```
Enter your name
Fahad
Welcome Fahad
```

١-٣ جمل الاختيار Selection Statements

ويطلق عليها أيضًا جمل التحكم أو جمل اتخاذ القرار.

١-٣-١ جملة الشرط if statement

- **الصيغة الأولى لجملة if**

تأخذ الجملة if الصيغة العامة التالية :

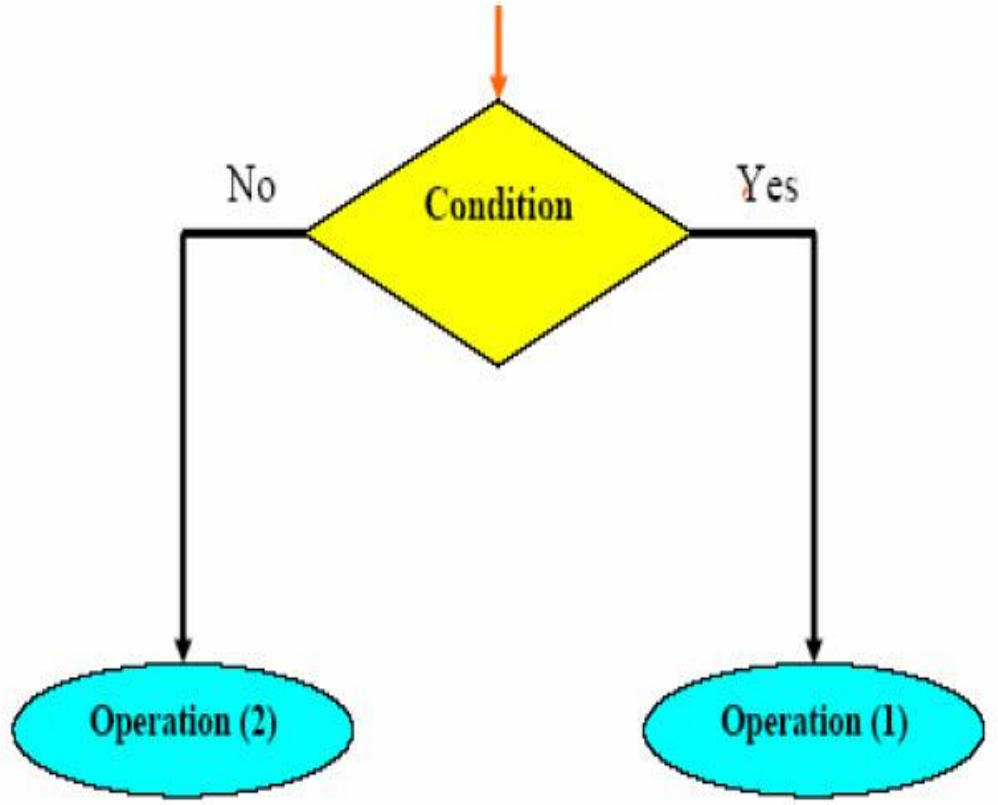
```
الشرط
if (expression) {
            statement1; // جملة1
            statment2;  // جملة2
              }
```

ومعنى هذه العبارة أنه اذا كان الشرط الذي تقوم الجملة (if) باختباره صحيحا فقم بتنفيذ الجملة التي بين القوسين . وفي حالة عدم صحة الاختبار فلا تقم بتنفيذ جملة (if) وإنما استمر في تنفيذ بقية جمل البرنامج من بعد تخطي جملة (if) .

وفي حالة تنفيذ جملة واحدة فقط بعد جملة (if) فإنه يمكن الاستغناء عن الأقواس وفي هذه الحالة تنتهي جملة الشرط بالفاصلة المنقوطة (;) كما يلي :

```
الشرط
if (expression)
            statement1;
تنتهي جملة الشرط عند الفاصلة المنقوطة
```

ويمكن تمثيل عبارة if بالشكل (١٤-١) التالي :



شكل (١٤-١)

وكما نعرف دائما فإن الشرط يجب أن يكون شرطا منطقيا ولفهم طريقة عمل جملة if ندرس الأمثلة التالية:

مثال (١٠)

المطلوب كتابة برنامج يقوم بفحص رقم يتم ادخاله من لوحة المفاتيح وليكن (x) بحيث اذا كان موجبا يقوم بطباعة العبارة (x is positive) .

```
import java.util.Scanner;
public class magdy{

public static void main(String s[] ){
int x;
  Scanner Keyboard=new Scanner(System.in);
  System.out.println("Enter Number X");
   x=Keyboard.nextInt();
if(x>0)
        System.out.println("X is positive");
      }
   }
```

هنا يتم ادخال الرقم X

هنا تتم عملية المقارنة X

ونلاحظ هنا أن جملة if انتهت بأول فاصلة منقوطة قابلتها أي بعد عبارة الطباعة ونلاحظ هنا أننا لم نستخدم الأقواس في جملة if لأننا لم نكتب غير سطر واحد فقط بعد عبارة if وهي جملة الطباعة أما لو كتبنا أكثر من جملة يجب تنفيذها عند تحقق الشرط في هذه الحالة يجب اضافة الأقواس

ويتم تنفيذ البرنامج كما يلي :

```
Enter Number X
5
X is positive
```

مثال (١١)

كرر نفس المثال السابق مع استخدام أكثر من سطر في جملة if كالآتي:

```
import java.util.Scanner;
public class magdy{

public static void main(String s[] ){
int x;
  Scanner Keyboard=new Scanner(System.in);
  System.out.println("Enter Number X");
   x=Keyboard.nextInt();
```

هنا يتم ادخال الرقم X

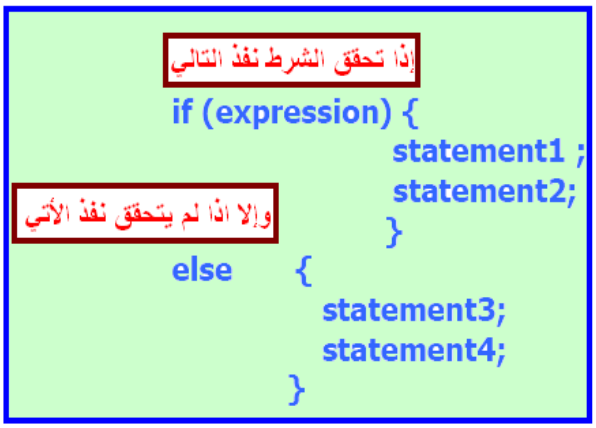
```
if(x>0){
        System.out.println("X is positive");
        System.out.println("X is not negative"); }
   }
}
```

نلاحظ أن جملة الشرط احتوت على أكثر من سطر ولذلك وضعنا الأقواس

وتكون مخرجات البرنامج كالتالي:

```
Enter Number X
10
X is positive
X is not negative
```

- **الصيغة الثانية لجملة if**



مثال (١٢)

نفذ البرنامج السابق بحيث يطبع عبارة (X is positive) في حالة إذا كانت موجبة وإلا يطبع عبارة (X is negative) في حالة إذا كانت X سالبة.

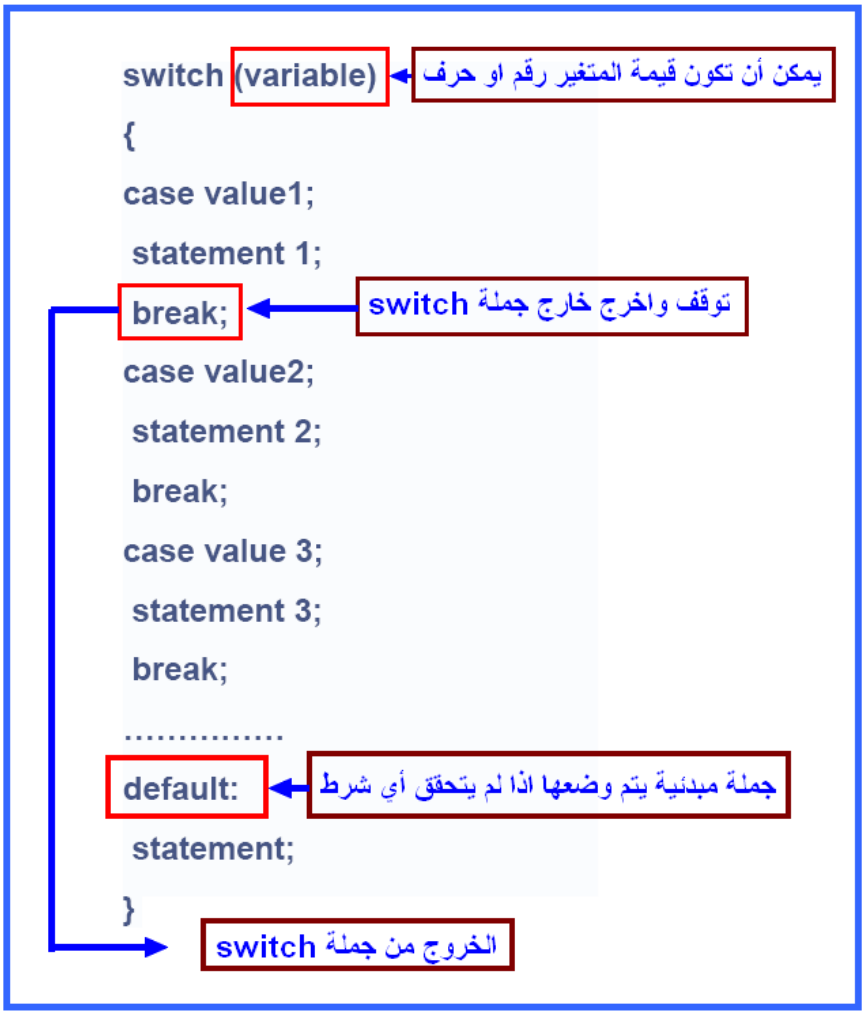
ويتم ذلك باستخدام عبارة IF الكاملة كالآتي:

```
import java.util.Scanner;
public class magdy{

public static void main(String s[] ){
int x;
  Scanner Keyboard=new Scanner(System.in);
  System.out.println("Enter Number X");
   x=Keyboard.nextInt();  <-- هنا يتم ادخال الرقم X

   if(x>0){
          System.out.println("X is positive"); }
else  {
       System.out.println("X is negative");  }

        }
       }
```

و يمكننا في البرنامج السابق حذف الأقواس الموجودة بعد جملة if ، وكذلك حذف الأقواس الموجودة بعد else وذلك لوجود جملة طباعة واحدة بعد كلاً منهما.

٢-٣-١ جملة switch

تستخدم عبارة if إذا كان جواب الشرط عبارة عن احتمالين أو ثلاثة احتمالات على الأكثر، أما إذا زاد عدد الاحتمالات على ذلك فمن الأفضل استخدام عبارة switch وصيغتها العامة كالآتي:



ولكي نفهم كيفية عمل جملة switch نجري المثال التالي:

مثال (١٣)

لنفرض أننا نريد ان يطبع الحاسب جملة ترحيب معينة إذا تم الضغط على أحد الأرقام في لوحة المفاتيح .

خطوات البرنامج :

```
import java.util.Scanner;
public class magdy{

public static void main(String s[] ){
int a;
Scanner Keyboard=new Scanner(System.in);
System.out.println("Enter a Number ");
 a=Keyboard.nextInt();

switch(a)
            {
   case 1:    في حالة اذا كان الرقم 1
        System.out.println("welcome");
    break;
    case 2:    في حالة اذا كان الرقم 2
         System.out.println("how are you");
    break;

               في حالة اذا كان الرقم 3
    case 3:
          System.out.println("good morning");
      break;
   default:

        System.out.println("good by"); }
     }
 }
```

وعند تنفيذ هذا البرنامج فأنه تحدث إحدى الحالات الآتية:

١- إذا تم إدخال الرقم (١) فإنه يطبع العبارة الأولى Welcome ثم يجد عبارة break فيخرج خارج جملة switch وينتهي البرنامج .

٢- إذا تم إدخال الرقم (٢) فإنه يطبع العبارة الثانية how are you ثم يجد عبارة التوقف break فيخرج خارج جملة switch وينتهي البرنامج .

٣- وهكذا في حالة إدخال الرقم (٣) فإنه يطبع الجملة الثالثة ثم break ثم يخرج .

٤- أما في حالة إدخال أي رقم غير موجود في البرنامج وليكن (٤) مثلا، فإن البرنامج يطبع العبارة الموجودة في جملة default ثم ينتهي البرنامج .

٥- يجب مراعاة أن جملة switch لها قوسي بداية ونهاية.

٦- يجب دائماً أن تنتهي كل حالة case من حالات switch بالعبارة break.

☺ حاول أن تقوم بإلغاء هذه العبارة وتجربة البرنامج ... ماذا تجد ؟؟

☺ جرب إدخال حرف بدلا من الرقم ماذا تجد ؟؟

١-٤ جمل الدوران

كثيرا ما نحتاج في البرنامج إلى تكرار أمر معين موجه إلى الحاسب عدداً من المرات ، وتوفر لغة الجافا عدة وسائل تمكن المبرمج من أداء هذا التكرار. وعادة ما تسمى هذه الوسائل بالحلقات التكرارية ويوجد العديد من الحلقات التكرارية التي سوف نتناولها بالشرح وهي:

١- الحلقة (for Loop)

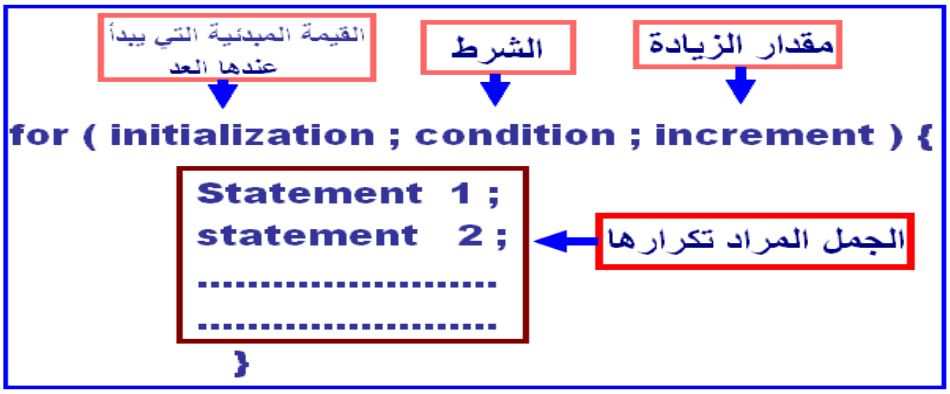
٢- الحلقة (while loop)

٣- الحلقة (do-while Loop)

١ - ٤ - ١ الحلقة (for loop)

تستخدم الحلقة for لتكرار أمر معين (أو مجموعة من الأوامر) عددا من المرات.

والصيغة العامة لهذه الحلقة كالتالي :



نلاحظ أن هذه الحلقة تتكون من ثلاث أقسام هي:

١ - القيمة الابتدائية initialization .

نضع في هذا الجزء متغير ونعطيه القيمة الابتدائية التي يبدأ منها التكرار.

٢ - الشرط Condition

هنا نضع الشرط الذي يتوقف عنده العد.

٣ - مقدار الخطوة increment

هنا نضع مقدار الزيادة في حالة العد التصاعدي أو النقصان في حالة العد التنازلي.

ونلاحظ هنا أن جملة for لها قوس بداية وقوس نهاية ويتم وضع الأقواس في حالة تكرار أكثر من جملة كما يمكن الاستغناء عن هذه الأقواس في حالة تكرار جملة واحدة.

مثال (١٤)

نفذ برنامج بلغة الجافا يقوم بالعد من (١ إلى ٢٠) .

خطوات البرنامج كالتالي :

```
import java.util.Scanner;
public class magdy{

public static void main(String s[] ){
 int a;
      for ( a=1;a<=20;++a)

           System.out.println(a) ;
        }
  }
```

جملة for

في هذا البرنامج تم استخدام متغير a من نوع integer فيتم زيادة قيمته كل مرة بمقدار واحد.
والقيمة الابتدائية له داخل الحلقة a=1 حتى يصل العد إلى ٢٠ . تنتهي الحلقة وينتهي البرنامج
و في كل خطوة زيادة يتم طباعة قيمتها على الشاشة عن طريق أمر الطباعة.
وتكون الأرقام في شريط تحت بعضها على الشاشة. لماذا ؟؟
ثم فكر كيف يمكن طباعة المخرجات متجاورة أو على سطر واحد بينها مسافات متساوية.

مثال (١٥)
المطلوب عمل عداد تصاعدي يبدأ العد من القيمة (١) حتى القيمة (x) على أن يتم إدخال قيمة نهاية العد من لوحة المفاتيح .

وفي هذا البرنامج استخدمنا عبارة الإدخال كما استخدمناها في البرامج السابقة، وعن طريقها تم إدخال رقم نهاية العد وتم وضعه في المتغير (x) ثم وضعناه في الجزء الخاص بالشرط في الحلقة .
ونلاحظ هنا أن المتغير (a) قد تم تعريفه واعطاؤه قيمة ابتدائية داخل الحلقة (int a=1).
خطوات البرنامج :

```
import java.util.Scanner;
public class magdy{

public static void main(String s[] ){
 int x;
 Scanner Keyboard=new Scanner(System.in);
 System.out.println("Enter a Number ");
  x=Keyboard.nextInt();
    for (int a=1;a<=x;++a)

          System.out.println(a) ;
       }
  }
```

هنا يتم ادخال رقم نهاية العد

for جملة

مثال (١٦)

عمل برنامج يوضح تكرار أكثر من جملة داخل الحلقة for.

```
import java.util.Scanner;
public class magdy{

public static void main(String s[] ){
 int a;
      for ( a=1;a<=5;++a){
            System.out.println(a) ;
              System.out.println("\t"+a*10) ;
              }

         }
   }
```

طباعة اكثر من جملة داخل الحلقة

وفي هذا البرنامج تم استخدام أكثر من جملة يراد تكرارها داخل الحلقة ولذلك تم استخدام قوسي بداية ونهاية للحلقة وهذا البرنامج يقوم بطباعة المخرجات كالتالي :

```
ناتج البرنامج
1
10
2
20
3
30
4
40
5
50
```

١-٤-٢-الحلقة (while loop)

في هذه الحلقة التكرارية نحتاج إلى الشرط فقط وطالما كان هذا الشرط متحققا استمرت الحلقة في التكرار والصيغة العامة لها كالآتي:

ونلاحظ هنا أن الشرط يأتي أولاً قبل تنفيذ الحلقة.

مثال (١٧)

أكتب برنامج يقوم بعملية العد من (٠ إلى ١٠) باستخدام الحلقة while loop مع طباعة النتائج على الشاشة.

```
import java.util.Scanner;
public class magdy{

public static void main(String s[] ){
int a=0;        القيمة المبدئية للعداد

    while ( a<=10){
        System.out.println(a) ;
          ++a;
                }
     }
 }
```

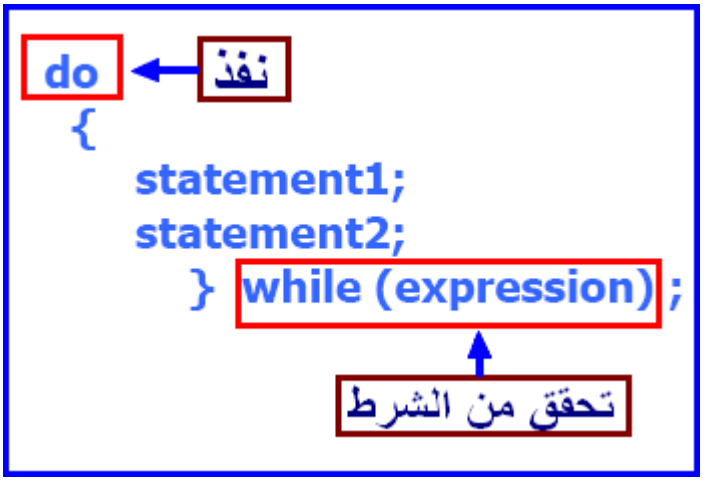
جملة while

ونلاحظ في هذا البرنامج انه لابد من إعطاء قيمة ابتدائية للعداد (int a=0) . ونلاحظ كذلك أنه في جملة while لابد من وجود الأقواس، لأن الجملة بطبيعتها تتكون من أكثر من سطر.

كذلك يتم زيادة قيمة (a) بمقدار واحد عن طريق الصيغة (++a) وبعد أن يتم زيادة قيمة a بمقدار واحد يتم التحقق من الشرط كل مرة وستكون نتيجة البرنامج طباعة الأعداد من (٠ إلى ١٠) .

١-٤-٣-الحلقة (do – while)

تختلف هذه الحلقة عن الحلقات السابقة في مكان وضع الشرط، حيث يكتب الشرط بعد العبارات المطلوب تكرارها وتكون صيغتها العامة كالتالي:



بإمكاننا القول أن الحلقة (do-while) تعني قم بالدخول في الكتلة do وقم بتنفيذ الأوامر. وفي حالة الانتهاء قم باختبار التعبير الشرطي الموجود في آخر الكتلة، وفي حالة صحة التعبير قم بالرجوع مرة أخرى إلى مكان الكلمة do.

مثال (١٨)

يمكن تطبيق نفس المثال السابق في حلقة while وهو البرنامج الذي يقوم بالعد من (٠ إلى ١٠) ولكن هذه المرة باستخدام الحلقة (do-while) كالآتي :

```
import java.util.Scanner;
public class magdy{

public static void main(String s[] ){
 int a=0;

    do{
        System.out.println(a) ;
         ++a;
                  }while ( a<=10);

    }
}
```

تحقق من الشرط

ونلاحظ هنا أنه في كل مرة يقوم البرنامج بالتحقق من الشرط في نهاية الحلقة، وهذا يعني أنه إذا لم يتحقق الشرط فسوف يتم تكرار الحلقة مرة واحدة فقط.

١-٥ الدوال methods

وهي عبارة عن طرق ودوال من تعريف (تصميم) المبرمج أو تكون جاهزة في البرنامج. والغرض منها هو تسهيل عملية البرمجة في الأشياء التي تتكرر أكثر من مرة في البرنامج.

❖ **الهدف من الدوال**

في حالة تكرار مجموعة من سطور الأوامر أكثر من مرة في مواضع مختلفة في البرنامج فإن أوامر التكرار لن تكون ذات منفعة. ولذلك يتم كتابة هذه الجمل منفصلة عن البرنامج الرئيسي .

❖ **مزايا استخدام الدوال**

١- عدم الحاجة إلى تكرار التعليمات داخل البرنامج حيث يتم إنشاء الدالة مرة واحدة ويمكن استدعائها أكثر من مرة عند الحاجة اليها .

٢- باستخدام الدوال يصبح البرنامج أكثر وضوحا.

٣- باستخدام الدوال الجاهزة يمكن توفير الكثير من الوقت والجهد.

❖ **هناك نوعان من الدوال يمكن استخدامهما:**

١- دوال جاهزة يمكن أن توفرها لغة الجافا .

٢- دوال يمكن تعريفها عن طريق المستخدم.

١-٥-١ دوال جاهزة يمكن ان توفرها لغة الجافا .

مثل الدوال الرياضية بأنواعها والجدول شكل (١٥-١) الآتي يبين الدوال الحسابية الجاهزة في لغة الجافا :

اسم الدالة	وصف الدالة	مثال
abs(x)	القيمة المطلقة لـ x.	Math.abs(6.2) → 6.2 Math.abs(-2.4) → 2.4
ceil(x)	تقرّب x إلى أقل عدد صحيح ليس اقل من x.	Math.ceil(5.1) → 6 Math.ceil(-5.1) → -5
floor(x)	تقرب x إلى أكبر عدد صحيح ليس أكبر من x.	Math.floor(5.1) → 5 Math.floor(-5.1) →-6
max(x,y)	أكبر قيمة من x و y.	Math.max(7,6) → 7
min(x,y)	أقل قيمة من x و y.	Math.min(-7,-8) → -8
pow(x,y)	x مرفوعة للأس y.	Math.pow(6,2) → 6^2 → 36
sqrt(x)	الجذر التربيعي لـ x.	Math.sqrt(9) → $\sqrt{9}$ → 3
random()	تكوّن رقم عشوائي بين الصفر والواحد.	Math.random() → 0.23121

شكل (١٥-١)

الجدول شكل (١٦-١) يبين المكتبات الخاصة بالجافا وما تقدمه هذه المكتبات من خدمات .

المكتبة	الخدمات / الفئات	التطبيقات
java.util	تحتوي هذه الباقة على فئات تمثل هياكل بيانات عامة الاستعمال مثل الصفوف و المجموعات و غيرها.	البرمجة الخوارزمية العادية.
java.io	تحتوي هذه الباقة على فئات تتصرف في عمليات تصدير و توريد البيانات.	البرامج التي تتطلب معاملة فورية مع المستخدم.
java.lang	تحتوي هذه الباقة على الفئات المتعلقة بتنفيذ البرنامج و مراقبته، بما فيها الفئات التي تعالج أخطاء التنفيذ و بعض الفئات العامة. نظرا لأهميتها، فإن هذه الباقة يقع توريدها ضمنيا في كل برنامج.	كل البرامج/ كل التطبيقات.
java.math	تحتوي هذه الباقة على فئات تقوم بعمليات حسابية، بأي دقة يطلبها المستخدم.	التطبيقات الهندسية و تطبيقات الرياضيات.
java.sql	تحتوي هذه الباقة على فئات تختص في عمليات على قواعد البيانات.	تطبيقات التصرف التي تتطلب قواعد بيانات.
java.awt	تحتوي هذه الباقة على فئات تختص في الرسم و في إنجاز واجهات رسومية.	تطبيقات تتطلب واجهات رسومية مع المستخدم.
java.swing	تمدد هذه الباقة إمكانيات و قدرات الباقة السابقة.	تطبيقات تتطلب واجهات رسومية مع المستخدم.
java.security	تحتوي هذه الباقة على فئات تختص في تنفيذ إجراءات أمنية في البرنامج، مثل مراقبة المستخدمين و صيانة الوارد و غير ذلك.	تطبيقات تتطلب إجراءات أمنية.

شكل (١٦-١)

ويمكن استدعاء الدوال بكتابة اسم الفصيلة(الفئة) متبوعاً بنقطة بعدها اسم الطريقة ثم قائمة المعاملات داخل أقواس دائرية كما يلي:

```
Class_Name.method_Name(Argument List)
```

فمثلا إذا اردنا الحصول على الجذر التربيعي للعدد(٢٥) فيمكن كتابة الصيغة كالتالي:

```
System.out.print(Math.sqrt (25.0))
```

تقوم هذه الجملة باستدعاء الدالة (sqrt) الموجودة في الفصيلة (Math) والتي تأخذ معامل واحد من نوع (Double) ونتيجة تنفيذ هذه الجملة سيكون طباعة (٥,٠) .

مثال (١٩) :

المطلوب عمل برنامج يستقبل قيمة من لوحة المفاتيح ثم يقوم بإيجاد الجذر التربيعي ومربع هذا الرقم وذلك باستخدام الدوال الجاهزة في لغة الجافا .

خطوات البرنامج كالآتي :

```
import java.util.Scanner;
public class Math{

public static void main(String s[] ){
double number;

Scanner Number=new Scanner(System.in);
System.out.println("Enter a Number");

number=Number.nextDouble();

System.out.println("The square Root ="+Math.sqrt(number));
System.out.println("The Squar number is ="+Math.pow(number,2));

}}
```

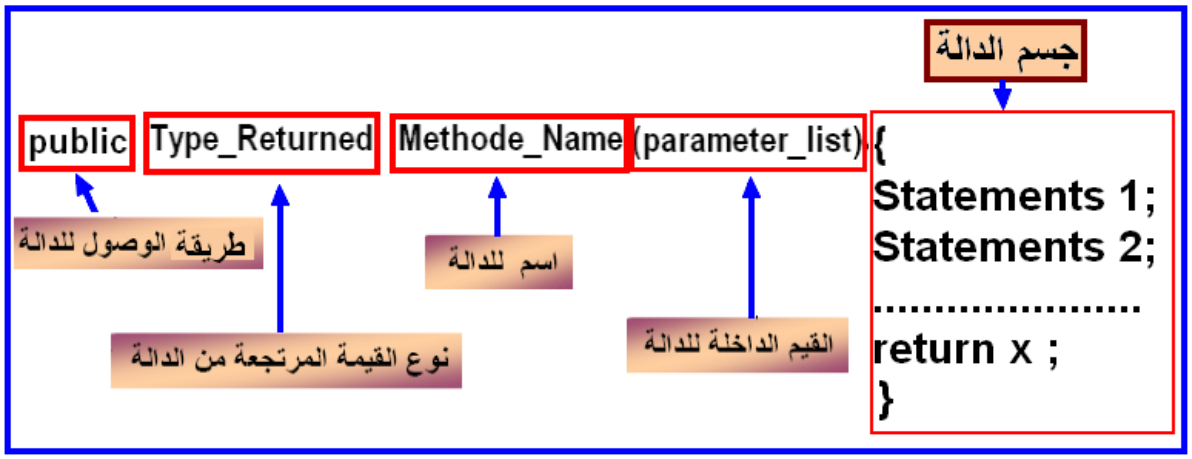
جمل طباعة الدوال الرياضية

في هذا البرنامج تم الإعلان عن متغير من نوع double (double number).
ثم يتم إدخال الرقم من لوحة المفاتيح وفي سطري الطباعة. السطر الأول يقوم بطباعة الجذر التربيعي أما السطر الثاني فيقوم بطباعة مربع هذا الرقم .
وعند تنفيذ البرنامج تظهر المخرجات كالتالي :

```
Enter a Number
9
The square Root is = 3.0
The Square number is =81.0
```

١-٥-٢- الدوال يمكن تعريفها عن طريق المستخدم :

الشكل العام للدالة



وفيما يلي شرح الشكل العام للدالة:

✓ **طريقة الوصول للدالة**
ويوجد ثلاث طرق نذكرها كالتالي:

١- public :أي عامة أي تستطيع الوصول اليها من خارج الفصيلة ومن خارج البرنامج أيضا.

٢- private : أي خاصة فتستطيع الوصول للدالة من داخل الفصيلة فقط، ولا يمكن ان تصل اليها من خارج الفصيلة.

٣-protected : أي محمي، أي أنك تستطيع الوصول للدالة من داخل الفصيلة أو من خارج الفصيلة (وهذا يدعم موضوع الوراثة).

أما عبارة static التي نجدها في معظم البرامج فهي من أجل إخبار المترجم أن هذه الدالة من نوع ثابت أي يتعرف عليها المترجم قبل الدخول للدالة الرئيسية.

✓ **وهناك نوعان من الدوال كالآتي** :

١- نوع يعود بقيمة .

وفي هذا النوع لابد من استخدام العبارة return كالشكل العام الذي رأيناه سابقا.

وكمثال على الدالة التي تعود بقيمة:

```
public int getDay( )
{
        return  day;
}
```

وهنا نرى أن الدالة المعرفة تعود بالتاريخ وهو قيمة ولذلك تم وضع عبارة return.

٢- نوع لا يعود بقيم void method.

ويكون تعريف هذا النوع كالآتي :

```
public void  method_ name(parameter_list)
   {
       <list of statements>
    }
```

ونلاحظ أنه في هذا النوع لم يتم استخدام عبارة return.

وكمثال لهذا النوع:

```
public void writeoutput(  )
  {
       System.out.println(month +" "+day + "   " + year);
}
```

وهنا نلاحظ أن عبارة الطباعة لا تعود بأي قيم للبرنامج الرئيسي ولذلك تم استخدام void أي دالة لا تعود بقيم وكذلك لم يتم استخدام العبارة return .

✓ **العبارة return**

وتوجد في نهاية الدالة وهي تجعل البرنامج يعود في مساره بعد انتهاء تنفيذ الدالة.

والصيغة العامة لهذه الدالة كالتالي :

```
return Expression ;
```

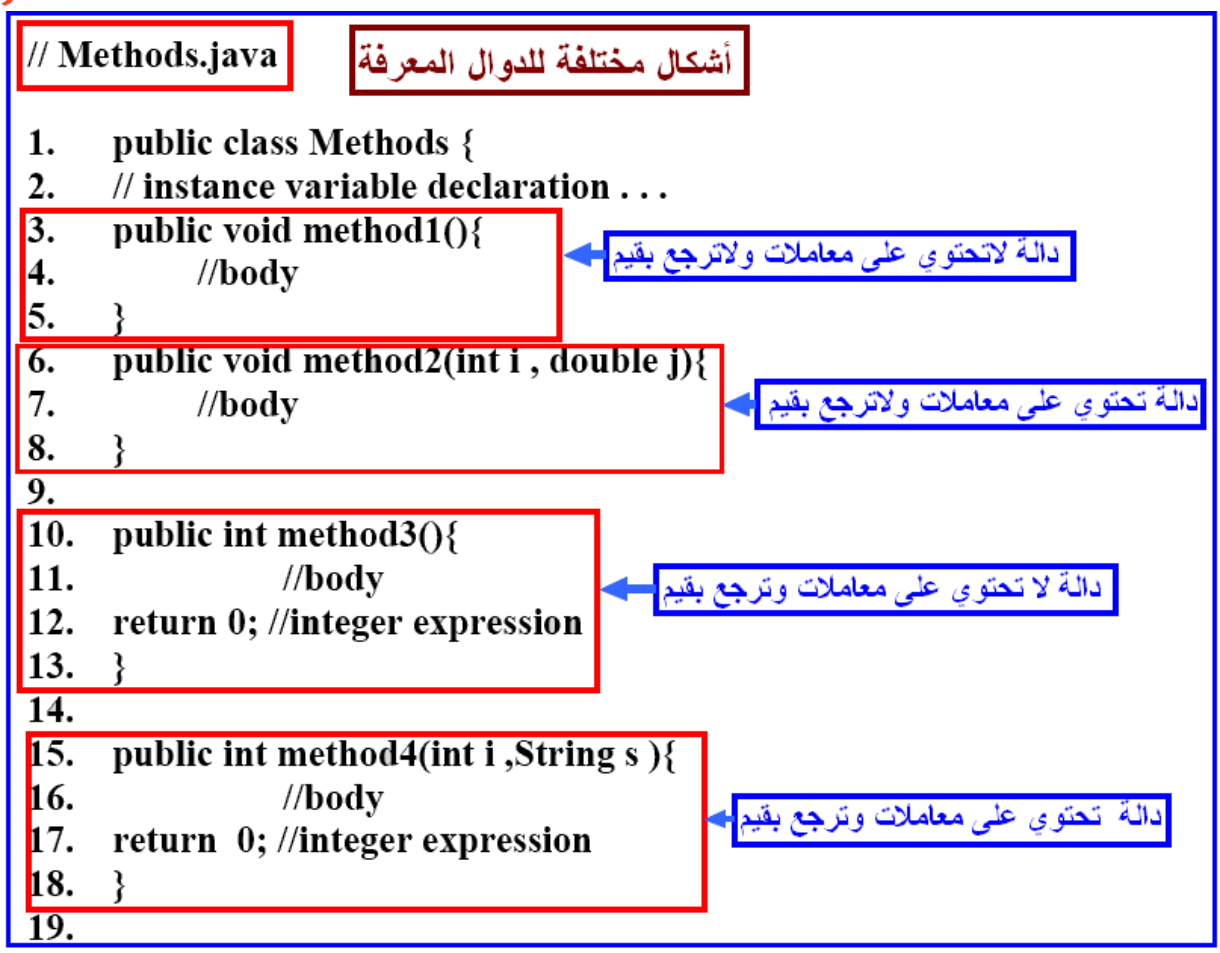
وكمثال على هذه العبارة

```
public int getYear(  )
 {
    return year ;
  }
```

☝ **لاحظ**

استخدام العبارة return بدون أي أقواس في الدوال من نوع void يمكن أن يتسبب في إنهاء البرنامج في الحال .

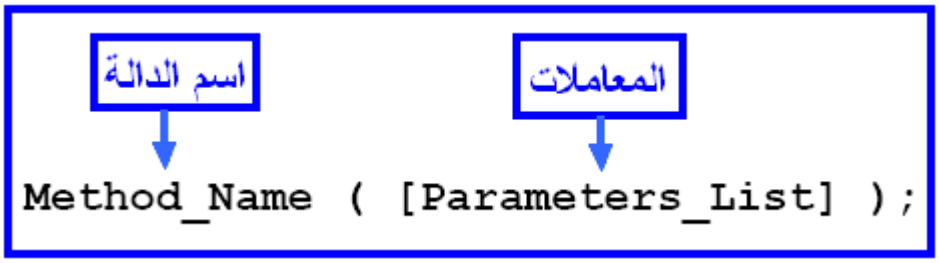
والشكل التالي (١٧-١) يوضح أشكال الدوال التي يمكن تعريفها :



شكل(١٧-١)

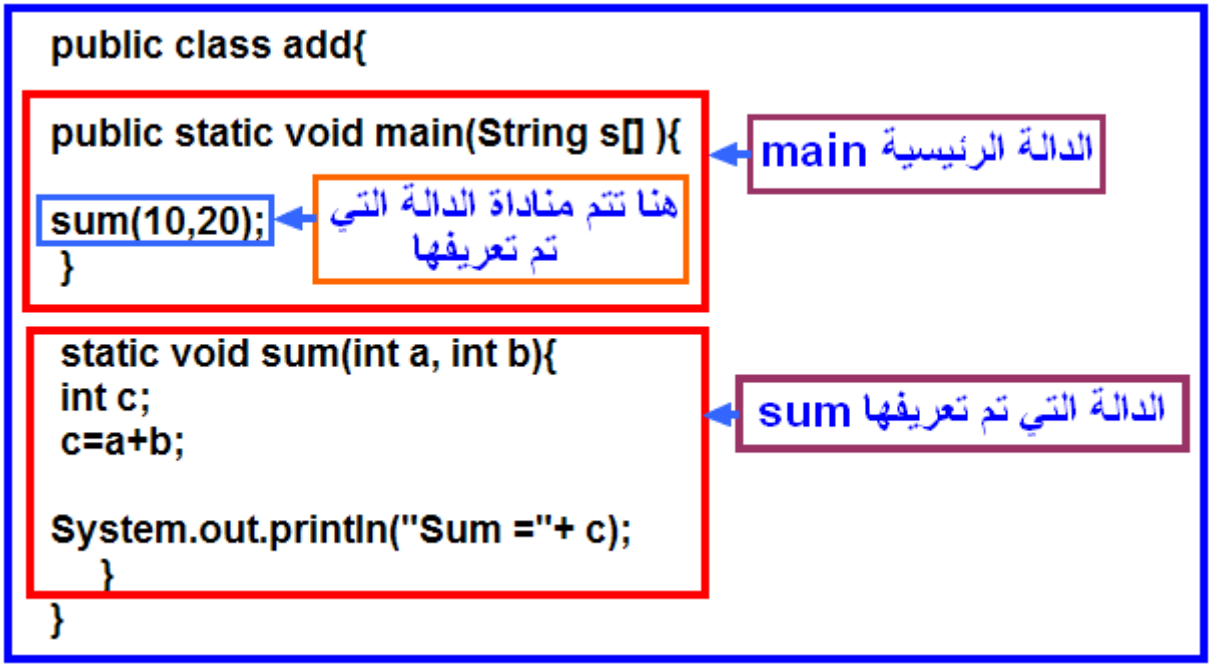
كيفية استدعاء الدوال داخل البرنامج

يمكن استدعاء الدالة داخل أي مكان في البرنامج عن طريق كتابة اسمها وارسال قيم المعاملات إن وجدت . والصيغة العامة لاستدعاء الدالة كالآتي :



مثال (٢٠) :

كيفية كتابة دالة بسيطة تقوم بعملية الجمع وطريقة استدعائها.



ونلاحظ أنه تم مناداة دالة الجمع sum في الدالة الرئيسية مع اسناد معاملات لها .

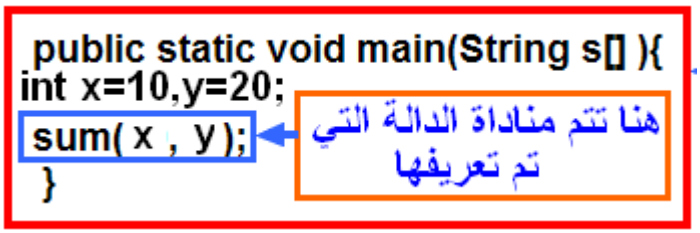
Sum(10,20)

ونلاحظ في جسم الدالة الفرعية التي تقوم بعملية الجمع أنها من نوع void. أي لا تعود للدالة الرئيسية بأي قيم، بل ينتهي دورها بمجرد طباعة ناتج الجمع. ونلاحظ كذلك أنها لا تنتهي بعبارة return . لماذا؟

ويلاحظ أن معاملات الدالة المستدعاة عبارة عن متغيرين (a,b) من النوع int كما تم تعريف متغير آخر داخل جسم الدالة الفرعية وهو (c) ليتم تخزين ناتج الجمع به .

وعند تشغيل البرنامج يتم طباعة ناتج الجمع وهو هنا (٣٠) .

كما يمكن تغيير القيم الداخلة إلى الدالة الفرعية بمتغيرات (x,y) كالتالي :



ولا يتم تغيير شيء في الدالة الفرعية .

مثال (٢١):

استخدام دالة تعود بقيم .

ولذلك لم نستخدم void وتم استخدام العبارة return .

```
public class add{

public static void main(String s[] ){
int x=10,y=20,w;
w=sum(x,y);        <-- استدعاء الدالة sum
 System.out.println("Sum ="+ w);
}

 static int sum(int a, int b){
 int c=0;
 c=a+b;

return c;          <-- للعودة بقيمة الجمع

    }
}
```

ومن المؤكد عند تنفيذ البرنامج سوف يتم طباعة حاصل الجمع (٣٠) .

١-٦-المصفوفات (المنظومات) Array .

في الحقيقة وقبل أن نبدأ في شرح المصفوفات نسأل أنفسنا أولاً: لماذا استخدمت طريقة المصفوفات ؟

وللإجابة على هذا السؤال نرجع إلى تعريف المتغيرات .

فالمتغير كما هو معروف يستخدم في تخزين البيانات سواء كانت هذه البيانات حروفًا أم أرقامًا.

فمثلا لو افترضنا أن هناك متغيرا من النوع الصحيح يسمى (a) وبه قيمة معينة فأننا كنا نعلن عنه هكذا:

int a=3;

ولكن ماذا لوكنا سنتحدث مثلا عن درجات خمس طلاب وكل طالب له درجة معينة ففي هذه الحالة سوف نحتاج خمس متغيرات .ولو فرضنا أن الدرجات من النوع الصحيح فأننا سنعلن عن هذه المتغيرات ونعطيها قيماً كالتالي :

```
int a1=80;
int a2=90;
int a3=60;
int a4=50;
int a5=45;
```

فهنا يمكننا فعلاً الإعلان عن خمس متغيرات واعطائهم القيمة المطلوبة. ولكن ماذا نفعل لو أن هناك مائة طالب أو الف طالب مثلا ؟ هل سنعلن عن كل هذه المتغيرات في البرنامج ؟ فيمكن ان نتخيل حجم البرنامج وكيفية فهمه وتصحيحه اذا تم الإعلان بالطريقة العادية. ولذلك كله تم الاستعانة بالمصفوفات .

والمصفوفات تعتبر من نوع المتغيرات المرجعية Reference variables .

- **تعريف المصفوفة**

المصفوفة هي عبارة عن مخزن يحمل عدد محدد من القيم Values لمتغيرات Variables من نفس النوع type. وهذا النوع يمكن ان يكون (int , float , string ,....) ويتحدد سعة هذا المخزن (المصفوفة) عند الإعلان عنها وبعد الإعلان عن المصفوفة وتحديد طولها (عدد المتغيرات التي ستخزنها) فإن هذا الطول يظل ثابتا ولا يمكن تحميل المصفوفة بعناصر أكثر من سعتها .

وكل عنصر في المصفوفة array يسمى element ويمكن الوصول لهذا العنصر في المصفوفة عن طريق فهرس رقمي index .

- **أنواع المصفوفات :**

يوجد نوعان من المصفوفات :

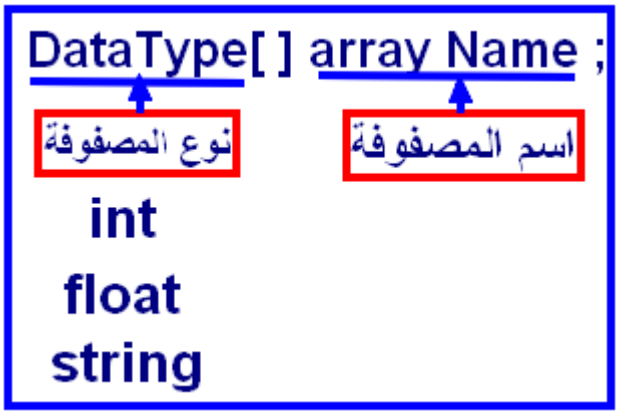
١- المصفوفة الأحادية: وهي مكونة من بعد واحد فقط.

٢- المصفوفة متعددة الأبعاد: وهي مكونة من عدد من الصفوف والأعمدة (ليس شرطا ان تكون بعدين) .

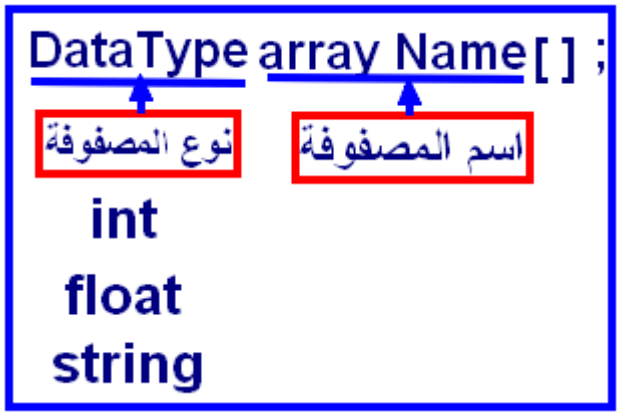
وسوف نتناول بالشرح ، المصفوفة ذات البعد الواحد ، والمصفوفة ذات البعدين .

١-٦-١ المصفوفة ذات البعد الواحد.

والصيغة العامة للإعلان عن المصفوفة ذات البعد الواحد كالتالي:



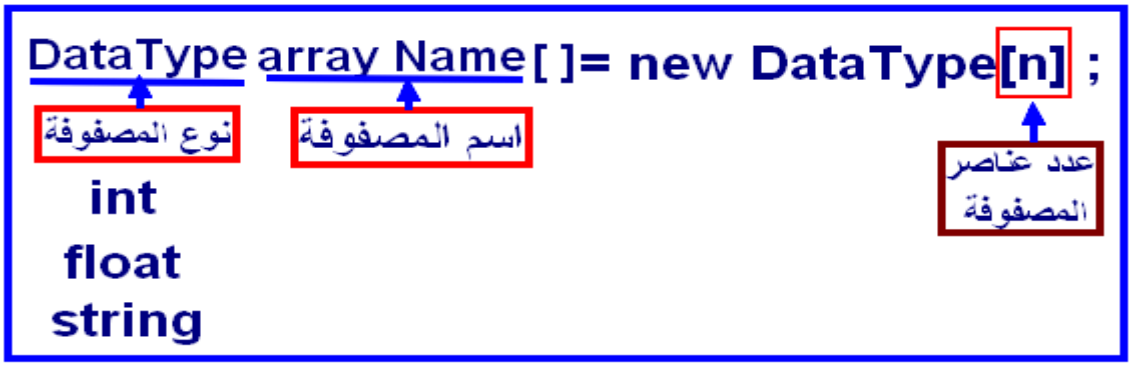
أو يمكن الإعلان عنها بنفس الصيغة السابقة مع وضع الأقواس بعد اسم المصفوفة كالتالي:



فمثلا يمكن الإعلان عن المصفوفة ذات البعد الواحد كالتالي :

وسوف نستخدم الصيغة الثانية (int degree[ ] ; float degree[] ; String name[])

وبعد الإعلان عن المصفوفة لابد من تحديد عدد عناصرها ويتم ذلك كالتالي:



فمثلا لعمل مصفوفة رقمية من النوع int خاصة بدرجات عشرة طلاب مثلاً يتم ذلك كالتالي:

```
int degree [] ;
degree[ ]= new int[10];
```

أو يمكن الإعلان عن المصفوفة وتحديد عدد عناصرها في سطر واحد كالتالي:

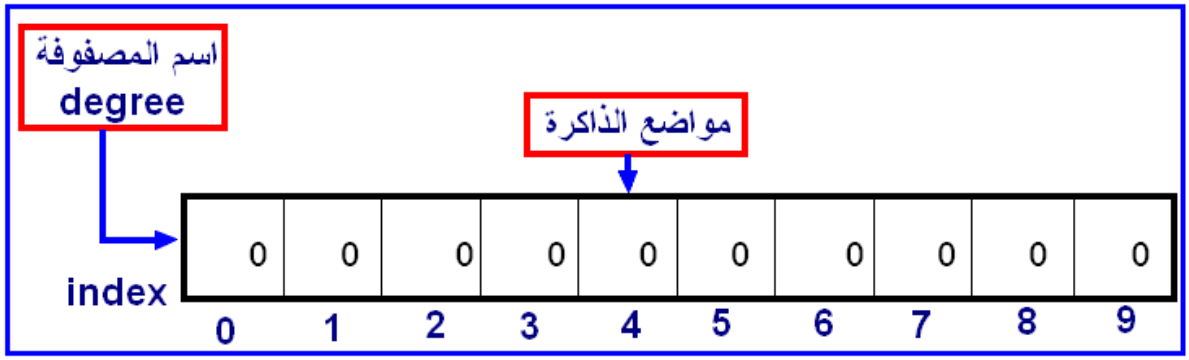
```
int degree[ ]=new int[10]
```

وهذا السطر يخبر الحاسب بحجز عشرة أماكن لمصفوفة ذات بعد واحد من النوع int وتسمى degree .

وكما قلنا سابقًا يمكن كتابة الصيغة السابقة كالتالي:

```
int[ ] degree=new int[10]
```

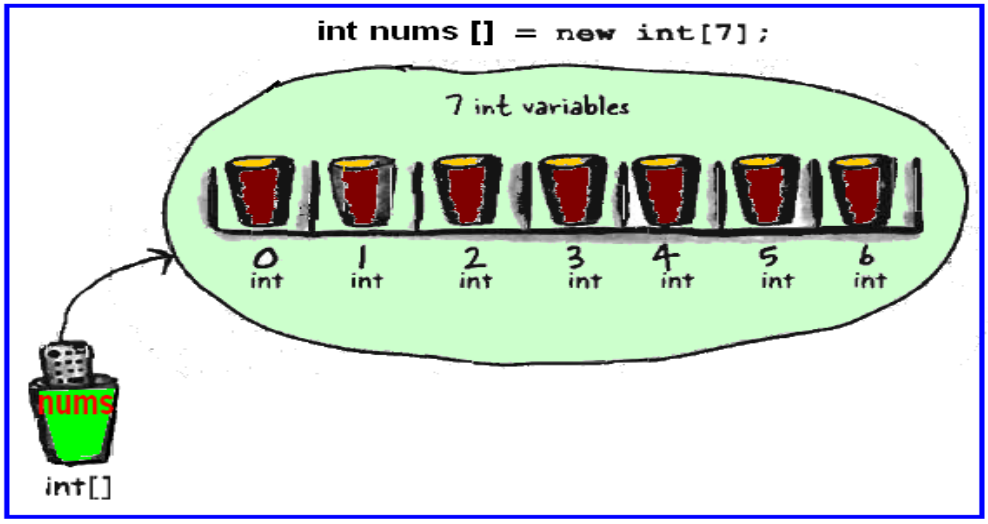
والحقيقة أنه بعد تحديد عدد عناصر المصفوفة يتم حجز ١٠ مواضع في الذاكرة لتخزين الأرقام الصحيحة التي سيتم إدخالها ويبدأ الترقيم في الذاكرة من الصفر كالتالي:



وكما قلنا ان عناصر المصفوفة عبارة عن متغيرات يتم تخزينها في الذاكرة ، وأن المتغير عبارة عن وعاء يتم تخزين القيم به والمثال التالي يوضح هذا المفهوم :

```
int nums[]=new int [7]
```

وهذا معناه حجز عدد (٧) أماكن (أوعية) في الذاكرة تمهيداً لتخزين قيماً بها كالآتي:



و يمكن تخزين قيماً (أعداد صحيحة) في هذه الأوعية كالتالي:

```
nums[0] = 6;
nums[1] = 19;
nums[2] = 44;
nums[3] = 42;
nums[4] = 10;
nums[5] = 20;
nums[6] = 1;
```

فمثلاً العنصر رقم (٠) يمكن اعطاؤه الرقم (٦) ، ورقم (٥) يمكن اعطاؤه القيمة (٢٠) ، وهكذا ....

ومن المؤكد أن المصفوفات لا تتعامل فقط مع الأرقام بل يمكنها تخزين الحروف والكلمات.

والبرنامج الآتي يبين ذلك:

نفرض أننا نريد تخزين عدد (٥) أسماء ثم طباعتهم فيتم عمل ذلك كالآتي :

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

String name[]=new String[5] ;
name[0]="Hassn";
name[1]="Magdy";
name[2]="Mohamed";
name[3]="Ahmed";
name[4]="amr";

for(int i=0;i<5;i++){

System.out.println(name[i]);

 }
   }
     }
```

الأعلان عن المصفوفة واضافة الأسماء

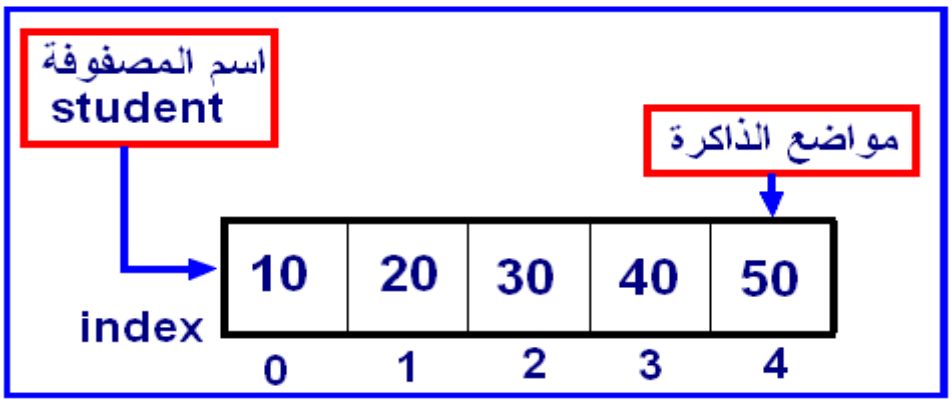
طباعة المصفوفة

ونلاحظ هنا أننا استخدمنا الحلقة التكرارية (for) لطباعة عناصر المصفوفة وتكون المخرجات كالتالي:

ويمكن اعطاء المصفوفة قيمًا ابتدائية كالتالي :

```
int student[]={10,20,30,40,50} ;
```

فهذه المصفوفة تم اعطاؤها ٥ قيم مسبقة ويتم تخزينها في الذاكرة كالتالي :



ولطبع الرقم ٣٠ الموجود في الخانة ٢ يتم كتابة الأمر التالي:

```
System.out.println(student[2]);
```

ونلاحظ مما سبق أنه إذا لم نحدد قيمًا ابتدائية للمصفوفة فيجب أن نستخدم كلمة (new) لحجز مواقع للمصفوفة كما أوضحنا سابقا.

مثال (٢٢)

المطلوب عمل مصفوفة ذات بعد واحد تحتوي على درجات خمس طلاب وطباعة الناتج على الشاشة.

نفترض أن درجات الطلاب (١٠, ٢٠, ٣٠, ٤٠, ٥٠) .و البرنامج كالتالي:

```
import java.util.Scanner;
public class array{

public static void main(String s[] ){
    int student[]={10,20,30,40,50};

        for(int i=0;i<=4;++i)
              System.out.println(student[i]) ;

    }
}
```

الأعلان عن المصفوفة واسناد قيم لها

حلقة تكرارية لقراءة محتويات المصفوفة

ونلاحظ هنا أنه تم عمل حلقة بجملة for لقراءة محتويات المصفوفة وطباعتها على الشاشة.

ونلاحظ كذلك أن نهاية العداد هو العدد ٤ على الرغم من كونهم ٥ عناصر. لماذا ؟

وضح ماذا يحدث لو جعلنا نهاية العد إلى الرقم ٥ ؟

وتكون مخرجات البرنامج كالتالي:

```
١٠
٢٠
٣٠
٤٠
٥٠
```

مثال (٢٣)

مطلوب كتابة برنامج يقوم بعمل مصفوفة حروف تقوم بطبع أيام الأسبوع على الشاشة كالآتي:

```
import java.util.Scanner;
public class array{

public static void main(String s[] ){

String student[]= {"saturday","sunday","monday","tuseday",
                   "wedensday","thrusday","friday"};

for(int i=0;i<=6;++i)
      System.out.println(student[i]) ;

    }
}
```

مصفوفة حرفية String

حلقة تكرارية لقراءة محتويات المصفوفة

ونلاحظ في هذا البرنامج أنه تم الإعلان عن مصفوفة من النوع الحرفي String type لأن عناصر المصفوفة عبارة عن حروف.

ويجب أن نلاحظ أن كلمة String يجب أن يكتب أول حرف فيها بحرف كبير capital.ثم يتم تنفيذ أمر الطباعة داخل الحلقة لطباعة أيام الأسبوع. وهنا يتبادر إلى ذهننا سؤالاً:

ماذا لوقلت قيمة نهاية العد عن ٦ ؟ ماذا لو اصبحت ٤ مثلا ؟

وماذا لو زادت هذه القيمة عن ٦ ؟ ماذا لو اصبحت ٨ مثلا ؟

ونترك لك عزيزي الطالب التفكير واستخلاص النتائج.

١-٦-٢- المصفوفة متعددة الأبعاد (ذات البعدين) Multidimensional array

ويمكن القول بأن المصفوفة ذات البعدين هي عبارة عن جدول يحتوي على صفوف وأعمدة .
والصيغة العامة لهذه المصفوفة كالتالي :

```
DataType arrayName[][] = new DataType[n1][n2]
```

نوع المصفوفة: int, float, string

اسم المصفوفة

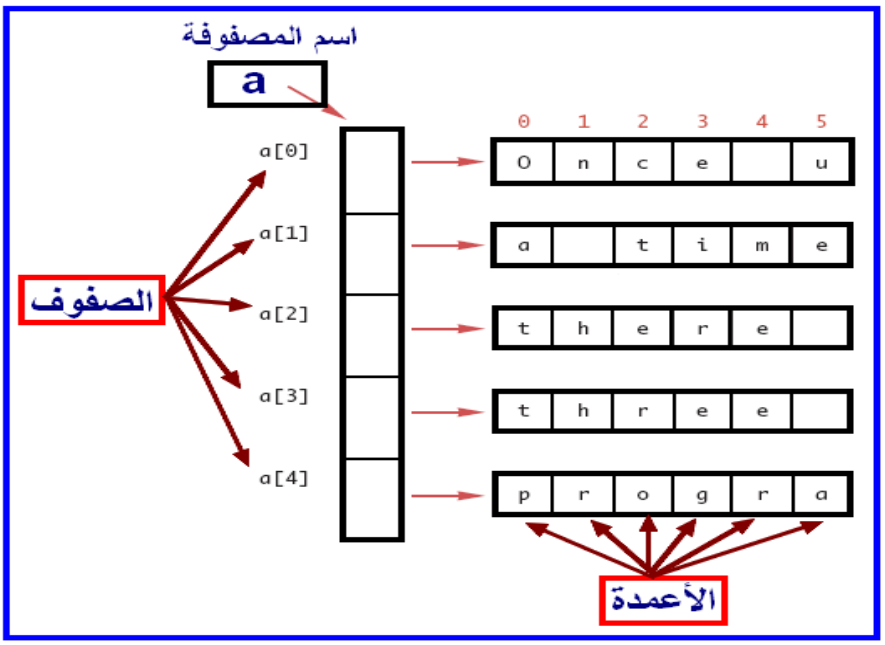
عدد الصفوف

عدد الأعمدة

فمثلاً اذا كان هناك مصفوفة تم تعريفها كالآتي :

ولفهم طريقة ترتيب العناصر في هذه المصفوفة نفرض أن لدينا مصفوفة(a) حروف كالآتي:

char a[][]=new char[5][6] ;



فإذا أردنا مثلاً أن نعرف محتويات المصفوفة في الموقع (a[١][٢]) فنجد أنه حرف (t).

وكذلك الموقع (a[٢][٣]) فنجد أنه الحرف (r) وهكذا.

وبطبيعة الحال يمكن اعطاء هذا النوع من المصفوفات قيماً ابتدائية كما سبق ورأينا في المصفوفة ذات البعد الواحد ، ولكننا هنا في المصفوفة ذات البعدين سوف نتعرف على كيفية إدخال القيم من لوحة المفاتيح .

✓ كيفية إدخال العناصر للمصفوفة

لنفرض أن هناك مصفوفة ذات بعدين يراد فيها إدخال درجات ٦ طلاب عن طريق لوحة المفاتيح يتم ذلك كالتالي:

سنقوم بتسمية المصفوفة student وسوف نستخدم دالة الإدخال Scanner لإدخال قيماً صحيحة إلى هذه المصفوفة وهذا هو شكل البرنامج:

```
import java.util.Scanner;
public class array{

public static void main(String s[ ] ){
1  int student[ ][ ]=new int[3][2];        ← الأعلان عن مصفوف ذات بعدين
2  Scanner Keyboard=new Scanner(System.in);
3  for(int row=0;row<3;row++){             ← يتم استخدام حلقتين واحدة للصف
4   for(int column=0;column<2;column++)        والأخرى للعمود
5  student[row][column]=Keyboard.nextInt() ;  ← يتم هنا ادخال عناصر المصفوفة
        }
    }
}
```

١- في السطر الأول يتم الإعلان عن مصفوفة ذات بعدين من النوع int وعدد عناصرها ٦ عناصر.

٢- السطر الثاني سبق وتم شرحه في جملة الإدخال.

٣- السطر الثالث والرابع تم عمل حلقتين ، الحلقة الأولى الخارجية للإشارة إلى رقم الصف row والحلقة الثانية للإشارة إلى رقم العمود column. وطبعا لاحظ ان الحلقة الخارجية قد تم إضافة أقواس لها . لماذا ؟

٤- أما السطر الخامس فيتم استخدام دالة الإدخال كما شرحناها سابقًا. ولكننا هنا استخدمنا المصفوفة studentكمتغير يتم تخزين عناصر المصفوفة المدخلة فيه.

وعند تشغيل البرنامج لن تظهر لك أي رسالة . لأننا لم نضف رسالة للإدخال يمكنك أنت اضافتها

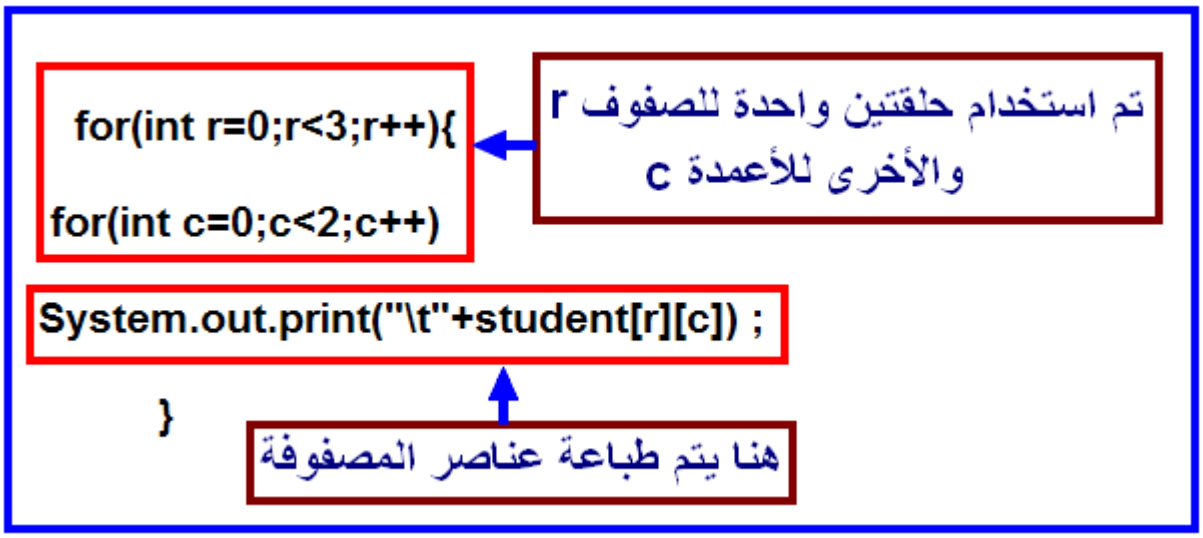
ولكن، سيظهر المؤشر في أقصى يسار الشاشة منتظرا إدخال قيم عناصر المصفوفة

فيتم إدخال ٦ عناصر والضغط على مفتاح الإدخال كل مرة.

وبعد تمام الإدخال للستة عناصر تظهر علامة المحث للدوس دلالة على انتهاء الإدخال.

✓ كيفية قراءة عناصر المصفوفة

بعد أن يتم إدخال عناصر المصفوفة واجراء أي عمليات عليها كالعمليات الحسابية مثلاً، يهمنا أن نرى المخرجات على الشاشة. وفي هذا البرنامج سوف نتعرف على كيفية قراءة عناصر المصفوفة وطباعة هذه العناصر كما هي على الشاشة دون أي تغيير، ويتم ذلك عن طريق البرنامج التالي:



ونلاحظ أننا استبدلنا جملة الطباعة print بجملة الإدخال في البرنامج السابق. ونلاحظ أننا أضفنا علامة (\t) وذلك لتنسيق الطباعة على الشاشة. فتظهر عناصر المصفوفة على سطر واحد لماذا ؟ وبينها مسافات متساوية لماذا؟

مما سبق يتبين لنا أنه لابد من جمع البرنامج الأول (إدخال عناصر المصفوفة) والبرنامج الثاني وهو طباعة عناصر المصفوفة مع بعضهما ليظهر برنامج واحد متكامل للإدخال والإخراج كالتالي

```
import java.util.Scanner;
public class magdy{
public static void main(String s[ ] ){

int student[ ][ ]=new int[3][2];
 Scanner Keyboard=new Scanner(System.in);
   for(int row=0;row<3;row++){

  for(int column=0;column<2;column++)

     student[row][column]=Keyboard.nextInt() ;

        }

   for(int r=0;r<3;r++){
         for(int c=0;c<2;c++)

 System.out.print("\t"+student[r][c]) ;

            }
       }
   }
```

هنا يتم ادخال عناصر المصفوفة

هنا يتم طباعة عناصر المصفوفة

تطبيقات الباب الأول

١- ماهي الإصدارات المختلفة للغة الجافا ؟
٢- ماهي مميزات لغة الجافا ؟
٣- وضح مع الرسم كيف أن لغة الجافا لا تعتمد على نظام التشغيل في الأجهزة المختلفة .

٤- قم بتظليل الإجابات الصحيحة فقط مما يأتي :

٥ – تكتب رأس الدالة الرئيسية للبرنامج كالآتي:

```
○ A. public static void main(string[] args)
○ B. public static void Main(String[] args)
○ C. public static void main(String[] args)
○ D. public static main(String[] args)
○ E. public void main(String[] args)
```

٦- أي العبارات الآتية صحيحة :

○ A كل سطر في البرنامج يجب أن ينتهي بفاصلة منقوطة
○ B كل جملة في البرنامج يجب أن تنتهي بفاصلة منقوطة
○ C كل سطر ملاحظة يجب أن ينتهي بفاصلة منقوطة
○ D كل دالة يجب أن تنتهي بفاصلة منقوطة
○ E كل كلاس (فئة) يجب أن تنتهي بفاصلة منقوطة

٧ - أي العبارات الآتية تقوم بطباعة العبارة (Welcome to Java) .يمكن أن تختار أكثر من اختيار.

```
☐ A. System.out.println('Welcome to Java');
☐ B. System.out.println("Welcome to Java");
☐ C. System.println('Welcome to Java');
☐ D. System.out.print('Welcome to Java');
☐ E. System.out.print("Welcome to Java");
```

٧- أذا أردنا ترجمة الملف المسمى (Test.java) فأننا نقوم بكتابة الآتي في سطر الأوامر:

- A. java Test
- B. java Test.java
- C. javac Test.java
- D. javac Test
- E. JAVAC Test.java

٨- إذا افترضنا أن هناك فصيلة تمت تسميتها كما يلي :

```
public class Test {

}
```

فإنه بعد عملية الترجمة ينتج ملف باسم:

- A. Test.class
- B. Test.doc
- C. Test.txt
- D. Test.java
- E. أي اسم له امتداد Java

٩- أي السطور الآتية لا تعتبر سطور ملاحظات comment يمكن اختيار أكثر من اجابة :

- A. /** comments */
- B. // comments
- C. -- comments
- D. /* comments */
- E. ** comments **

١٠- أي من الكلمات الآتية تعتبر من الكلمات المحجوزة في لغة الجافا (يمكنك اختيار أكثر من اجابة) ؟

- A. public
- B. static
- C. void
- D. class

١١- كل عبارات لغة الجافا يجب أن تكتب بحروف صغيرة :

- ○ A. true
- ○ B. false

١٢- أي اسماء المتغيرات الآتية صحيحًا. يمكن أن تختار أكثر من اجابة :

- ☐ A. radius
- ☐ B. Radius
- ☐ C. RADIUS
- ☐ D. findArea
- ☐ E. FindArea

١٣- أي الطرق الآتية تستخدم في الإعلان عن المتغيرات (يمكن اختيار أكثر من اجابة) ؟

- ☐ A. int length; int width;
- ☐ B. int length, width;
- ☐ C. int length; width;
- ☐ D. int length, int width;

١٤- بفرض ان x=1 ما هي قيمة x بعد تنفيذ x+=2 :

- ○ A. 0
- ○ B. 1
- ○ C. 2
- ○ D. 3
- ○ E. 4

١٥- ماهي قيمة X بعد تنفيذ العملية الآتية ؟ إذا كانت :

```
int x = 1;
x *= x + 1;
```

- ○ A. x is 1;
- ○ B. x is 2;
- ○ C. x is 3;
- ○ D. x is 4;

١٦- ماهي نتيجة تنفيذ البرنامج التالي ؟

```
public class Test1 {
   public static void main(String[] args) {
     int x = 1;
     int y = x = x + 1;
     System.out.println("y is " + y);
   }
}
```

- y is 0
- y is 1 لأن x ساوت y أولا .
- y is 2 لأن x +1 ساوت x أولا ثم بعد ذلك تم مساواة x=y.
- المترجم سوف يعطي خطأ عند الترجمة لأن x تم اعادة تخصيصها في العبارة y=x=x+1 .

١٧- ماهي النتيجة التي سوف يتم طبعها على الشاشة ؟

```
public class Test {
   public static void main(String[] args) {
     int x = 1;
     int y = x + x++;
     System.out.println("y is " + y);
   }
}
```

- y is 1.
- y is 2.
- y is 3.
- y is 4.

١٨- أي العبارات الآتية تقوم بطباعة الآتي (Ahmed\exam1\test.txt):

- `System.out.println("Ahmed\exam1\test.txt");`
- `System.out.println("Ahmed\\exam1\\test.txt");`
- `System.out.println("Ahmed\"exam1\"test.txt");`
- `System.out.println("Ahmed"\exam1"\test.txt");`

١٩- بفرض أننا نريد إدخال قيمة عدد صحيح من لوحة المفاتيح عن طريق استخدام العبارة الآتية
Scanner input = new Scanner(System.in);
ماهي الطريقة المستخدمة فيما يلي لقراءة العدد الصحيح :

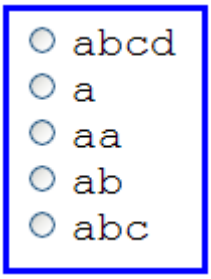
```
○ input.nextInt();
○ input.nextInteger();
○ input.int();
○ input.integer();
```

٢٠- أكتب برنامج يقوم بطباعة الشكل التالي :

```
*****
****
***
**
*
*
**
***
****
*****
```

٢١ – ما هو ناتج تنفيذ البرنامج التالي ؟

```
char ch = 'a';

switch (ch) {
  case 'a':
  case 'A':
    System.out.print(ch); break;
  case 'b':
  case 'B':
    System.out.print(ch); break;
  case 'c':
  case 'C':
    System.out.print(ch); break;
  case 'd':
  case 'D':
    System.out.print(ch);
}
```



٢٢- أكتب برنامج يقوم بطباعة الأعداد الفردية على الشاشة في صف واحد بدءًا من (١ إلى ٥٠).

٢٣- أكتب برنامج يقوم بعمل مقارنة بين مصفوفتين من النوع char، إذا كانت كلاً من المصفوفتين تحتوي على القيم الآتية:

{'d','h','r','f'}.

٢٤- قم بحساب قيمة المضروب لعدد صحيح يتم إدخاله من لوحة المفاتيح.

٢٥- قم بحساب مجموع القيم التالية باستخدام مصفوفة من النوع int :

(٥٥, ٣٤, ٥٧, ٩٠, ١٠)