المؤسسة العامة للتدريب التقني والمهني
Technical and Vocational Training Corporation

# أساسيات برمجة الحاسب

أ. ساجده المكي

أ. أمجاد المطيري


بسم الله الرحمن الرحيم

# جدول المحتويات

## تمهيد

نعيش اليوم في عالم رقمي متجدد و متغير، فقد أصبح خلف جميع ما يحيط بنا من أعمال و تجارة و تسوق و علوم و اختراعات و صحة و طيران برمجيات تديرها و تتحكم بها، لذا دعت الحاجة إلى التفكير بعمق في تعلم البرمجيات و التفكير الخوارزمي (الحسابي) بما يعود على وطننا وشعبنا وأمتنا بالفائدة والنفع الكثير .

تعلم البرمجة و التفكير الحسابي ينمي مجموعة من المهارات المعرفية وعمليات حل المشكلات التي تشمل :

- التنظيم المنطقي وتحليل البيانات.
- تحليل المشكلة إلى أجزاء أصغر.
- حل المشكلة باستخدام أساليب التفكير البرمجي مثل التكرار، والتمثيل الرمزي، والعمليات المنطقية.
- إعادة صياغة المشكلة من أجل أن تحل باستخدام سلسلة من الخطوات ( الخوارزميات)
- تحديد وتحليل وتنفيذ الحلول الممكنة بهدف تحقيق الحل الأكثر كفاءة وفعالية من الخطوات

في هذه الحقيبة بإذن الله سنتعلم أساسيات البرمجة بإحدى لغات البرمجة المعروفة والمشهورة في هذه الأيام وهي لغة الجافا لما تتمتع به من القدرة على العمل على مختلف الأجهزة وإمكانية كتابة البرامج البسيطة والمعقدة بها .

# مقدمة إلى الجافا

## برنامج الحاسب :

البرنامج هو عبارة عن مجموعة من التعليمات تعطى للحاسب للقيام بعمل ما مثل حساب مجموع قيم مختلفة ، حساب المتوسط الحسابي ، حساب مضروب عدد معين ...الخ وتنقسم برامج الحاسب إلى ثلاثة أنواع رئيسية :

## ١.أنظمة التشغيل :

هي مجموعة من البرمجيات الأساسية التي تقوم بإدارة الحاسب وتتحكم في كافة الأعمال والمهام التي يقوم بها وتيسر هذه البرمجيات على المستخدم الاستفادة من الأجهزة التي يتكون منها الحاسب والملحقات التابعة له مثل الطابعة والفأرة وغيرها كما تمكن المستخدم من الاستفادة من البرمجيات التطبيقية المختلفة للحاسب كبرمجيات الطباعة للرسائل أو إجراء الأعمال الحسابية أو غير ذلك.
**أمثلة** : Windows , Unix ,Mac OS

## ٢.برامج التطبيقات :

برامج التطبيقات هي برامج تم تصميمها لأداء مهام محددة للمستخدم ويوجد العديد من تلك البرامج لخدمة الاحتياجات المتنوعة للمستخدمين مثل البرامج الخاصة بكتابة الوثائق وأخرى لإعداد الحسابات المالية والميزانيات وبرامج لإرسال البريد الإلكتروني وغيرها الكثير.
**أمثلة** : برنامج معالجة النصوص WORD – برنامج الجداول الالكترونية EXCEL - برنامج التعديل على الصور PHOTOSHOP وغيرها .

## ٣.لغات البرمجة :

البرمجة هي طريقة للتخاطب بين الانسان والحاسب فالحاسب لا يفهم لغة الانسان لذا تم انشاء لغات البرمجة و هناك ثلاثة أنواع رئيسية من لغات البرمجة هي :

**أ- لغة الآلة**

- هي اللغة الوحيدة التي يفهمها الحاسب و تكتب اوامرها باستخدام لغة الحاسب الثنائية ( ١,٠ ) لذلك تسمى بلغة الآلة Machine Language
- تتخاطب مع Hardware مباشرة
- من اللغات صعبة التعلم حتى بالنسبة للمبرمجين أنفسهم.

**ب- لغة التجميع**

- لغة تستخدم اختصارات معبرة من اللغة الإنجليزية للتعبير عن العمليات البسيطة مثل الأوامر التالية :

***Add*** *للتعبير عن الجمع*
***Sub*** *للتعبير عن الطرح*

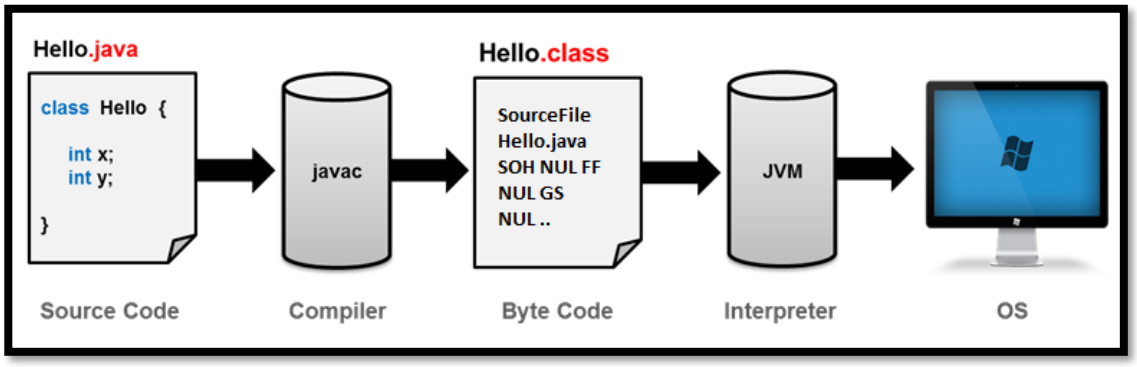
- تحتاج محول assembler لتحويل لغة التجميع إلى لغة الآلة ليتم تنفيذ المطلوب.

**ج- اللغات عالية المستوى**

- تكتب بأوامر شبيهة بلغة الانسان مثل (.. open , if.. else write)
- تحتاج إلى مترجم compiler أو مفسر Interpreter لتحويل الأوامر إلى لغة الآلة
- الأمثلة على لغات عالية المستوى : ***Visual Basic • Java • C++ • C# • Delphi***

## الإعداد للجافا :

تعتبر لغة الجافا كائنية التوجه وهي لغة التطوير الأكثر استخداما في العالم اليوم وحيث أن الأوامر التي نكتبها على جهاز الحاسب لا تعمل بشكل مباشر بل تمر بعدة مراحل تباعاً حتى تعمل تماماً كما في الصورة التالية:

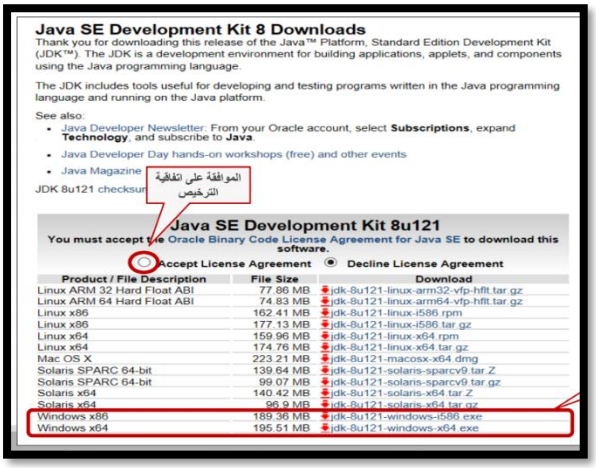



والسبب الرئيسي في ذلك أن نظام التشغيل ليس مكتوباً بلغة جافا لذلك وجب علينا تهيئة البيئة للجافا حتى يتعرف نظام التشغيل الذي نستخدمه على اللغة عن طريق :

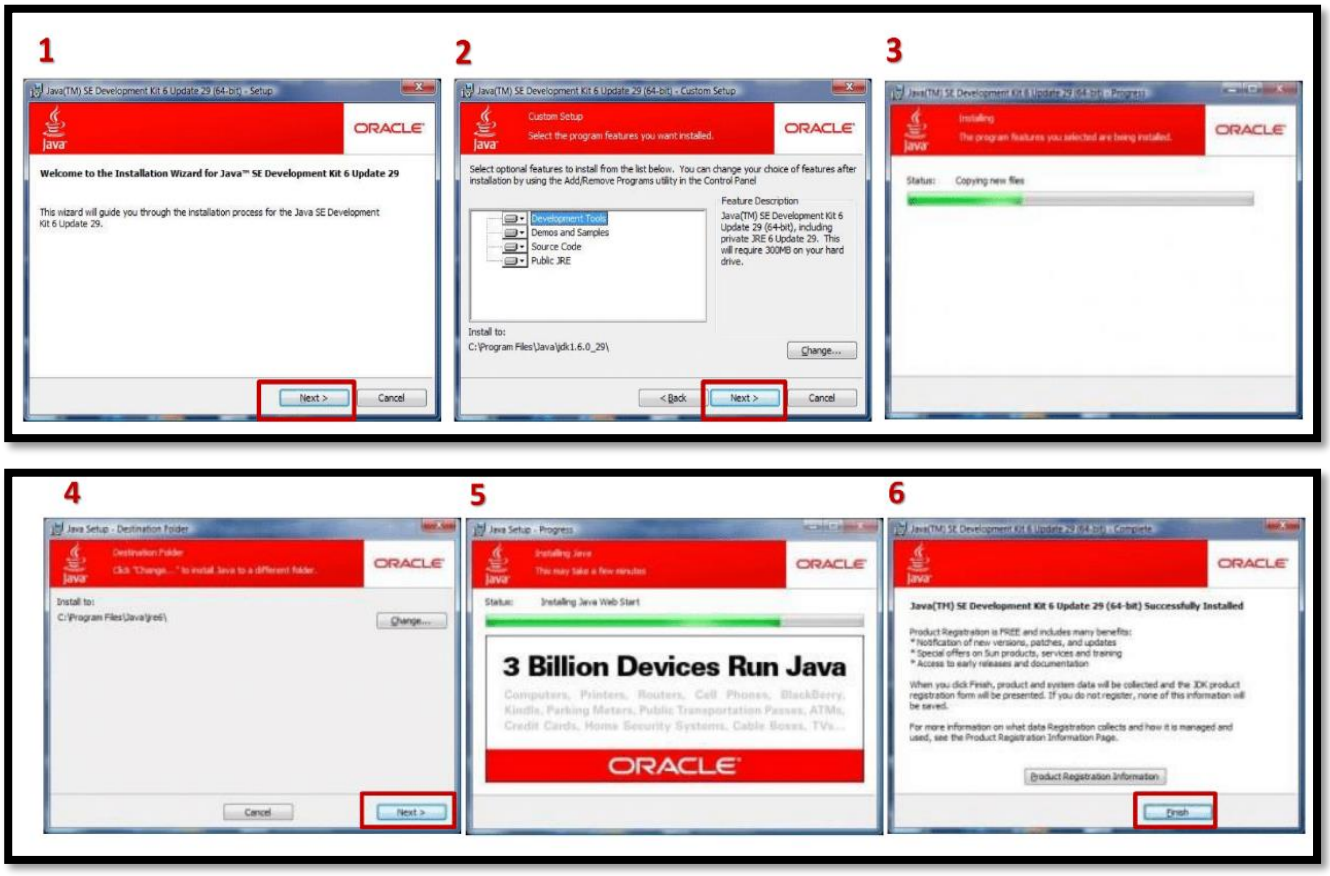
١- تحميل و تنصيب حزمة تطوير جافا ( JDK)
٢- برنامج خاص لكتابة الأكواد البرمجية مثل Eclipse

## تحميل و تنصيب حزمة تطوير جافا ( JDK)

١- تحميل JDK من موقع أوراكل



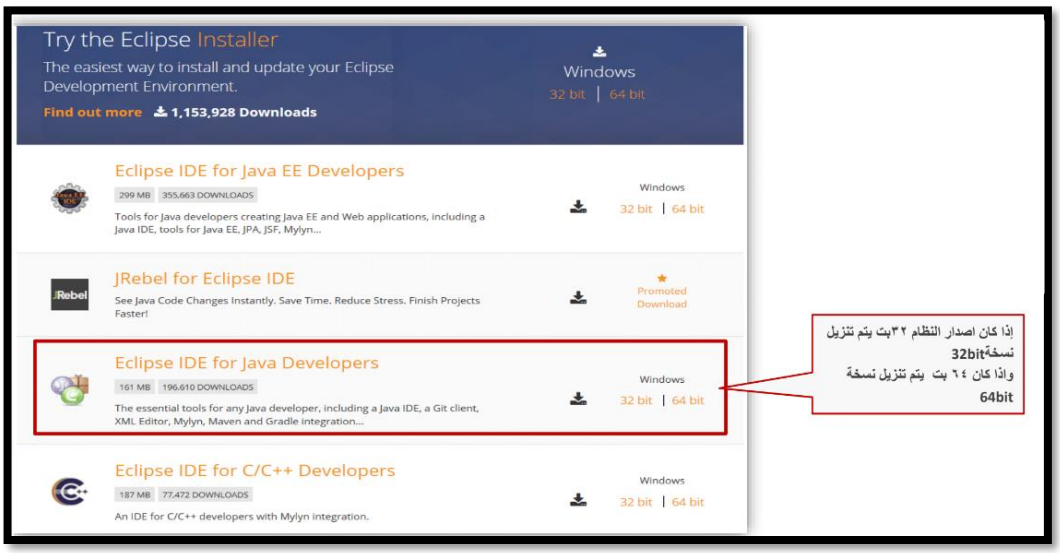

٢- تثبيت JDK



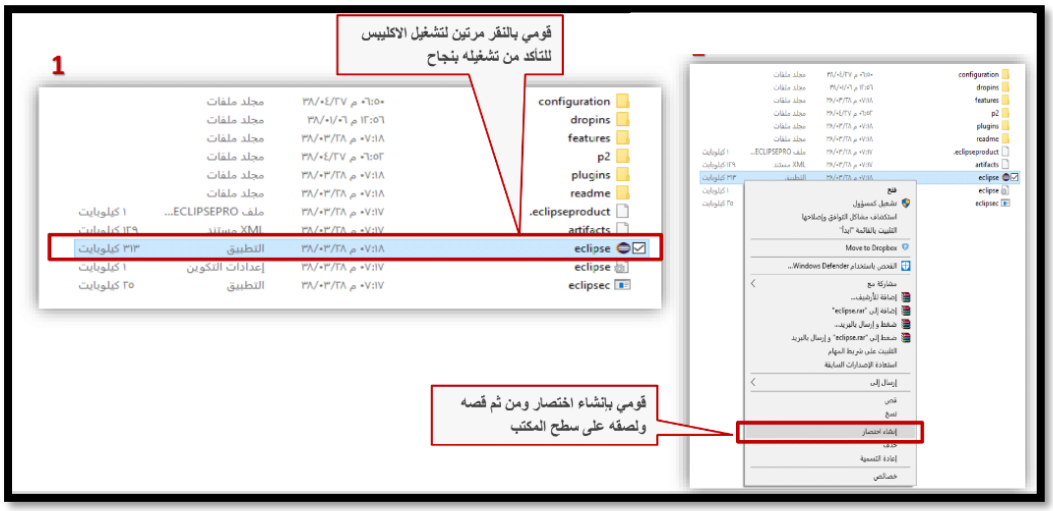
## تحميل و تنصيب برنامج الاكليبس ( Eclipse)

برنامج Eclipse هو عبارة عن بيئة تطوير متكاملة من أجل تطوير وتحديث البرمجيات المكتوبة بلغة الجافا ويعتبر برنامج مجاني .الخطوات التالية تبين طريقة تحميل وتثبيت البرنامج :

١- تحميل Eclipse من الموقع الرسمي للبرنامج



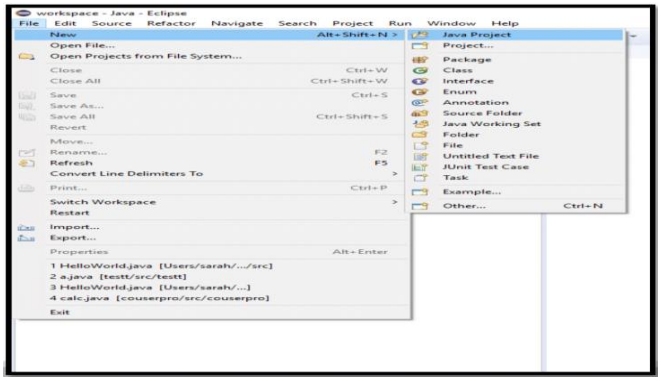

٢- نقوم بفك الضغط عن الملف الذي تم تنزيله ، سيتم انشاء مجلد باسم Eclipse بداخله مجموعة من الملفات والمجلدات ويمكن تشغيل البرنامج من خلال أيقونة Eclipse الموجودة داخل المجلد .



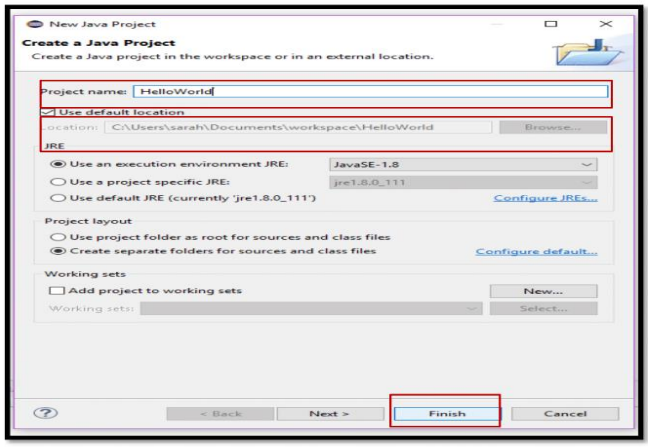
## انشاء مشروع جافا ( Project ) :

بعد تهيئة البيئة لتطوير برامج الجافا وتنصيب البرامج المطلوبة سنقوم بإنشاء اول مشروع جافا بإتباع الخطوات التالية :

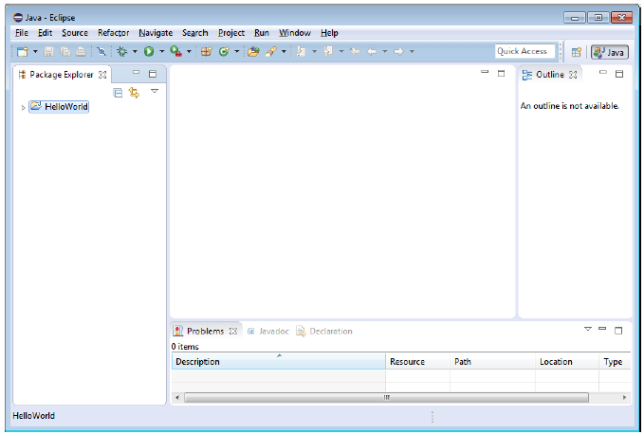
١- في برنامج Eclipse من قائمة File نختار New ومن ثم نختار Java project



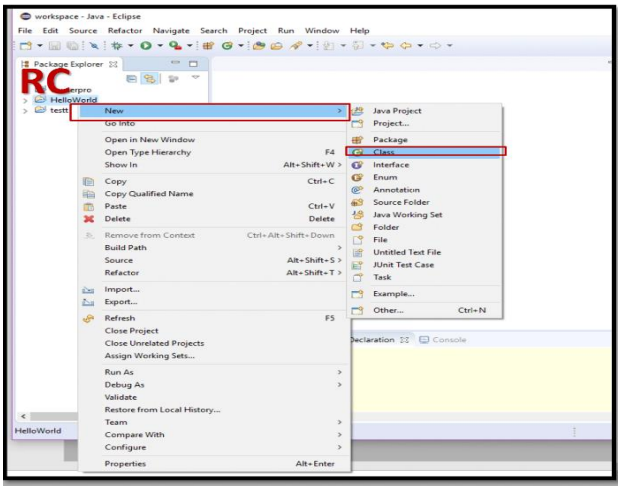

٢- سيتم عرض النافذة التالية ، ندخل اسم المشروع وبقية الخيارات كما هو موضح في الصورة التالية



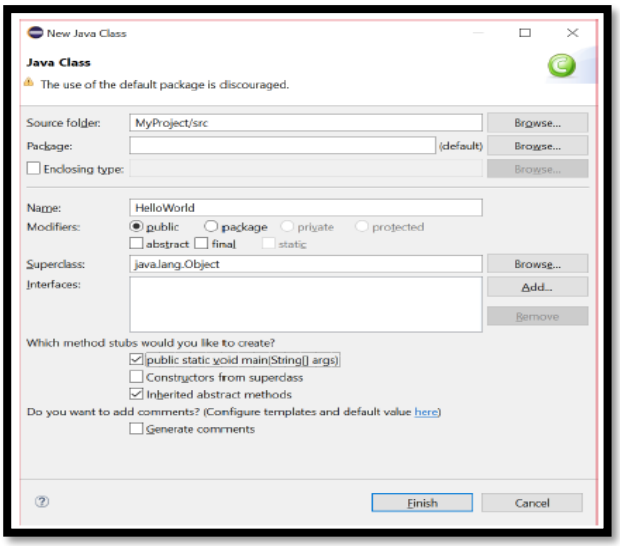

٣- مبروك لقد أنشأت أول مشروع جافا لكِ .



٤- الان نقوم بإنشاء كلاس ( Class) ثم يتم النقر بالزر الأيمن على اسم المشروع ثم نختار New ثم Class




٥- سيتم عرض النافذة التالية ، ندخل اسم الكلاس وبقية الخيارات كما هو موضح في الصورة التالية:




٦- لقد تم إنشاء كلاس جديد ونستطيع الآن كتابة الأوامر البرمجية

```
Package test;
```
هنا اسم الملف أو المجلد الذي يحتوي على البرنامج
```
Public class Test {
```
هنا اسم الكلاس
```
Public ststic void main (String[] args)  {
```
في هذه المنطقة تكون الدالة الأساسية
```
System.out.println("Hello") ;
```
في هذه المنطقة يتم كتابة الأكواد الخاصة بنا
```
}          }
```
أقواس اغلاق الكلاس والدالة الأساسية

## عرض البيانات في الجافا

في الجافا يوجد دالتين يمكنك استخدامهم للطباعة:

١- دالة *System.out.print()*
دالة تستخدم لعرض أي شيء نضعه في داخلها سواء نص . رقم أو قيمة متغير في نفس السطر .

٢- دالة *System.out.println()*
نفس الدالة السابقة ، الفرق بينها وبين الدالة السابقة أنها تطبع البيانات على سطر جديد .المثال التالي يوضح الفرق بينهما .

```
Public class Test {
Public static void main (String[] args)  {

System.out.print("Hello ") ;
System.out.print("world ") ;
 } }
```

الناتج من تشغيل البرنامج الأول :

```
Hello World
```

```
Public class Test {
Public static void main (String[] args)  {

System.out.println("Hello ") ;
System.out.println("world ") ;
 } }
```

الناتج من تشغيل البرنامج الثاني :

```
Hello
World
```

# أنواع البيانات والمتغيرات

## مكونات لغة البرمجة ( الجافا ) :

تتكون لغات البرمجة عامة ولغة الجافا خاصة من مكونات أساسية هي :


قبل التعرف على المكونات الأساسية وطريقة تعريفها والتعامل معها يجب علينا التعرف على أنواع البيانات التي سيتم التعامل معا .

## أنواع البيانات :

| النوع | الطول | البيانات |
|---|---|---|
| Boolean | ١ بت | True / false |
| Byte | ٨ بت | اعداد صحيحة |
| Short | ١٦ بت | اعداد صحيحة |
| Int | ٣٢ بت | اعداد صحيحة |
| Long | ٦٤ بت | اعداد صحيحة ذات حجم كبير جدا |
| Float | ٣٢ بت | اعداد بفاصلة عشرية |
| Double | ٦٤ بت | اعداد كبيرة بفاصلة عشرية |
| Char | ١٦ بت | الأحرف المنفردة |

## ١- المتغيرات :

هي عبارة عن أماكن في الذاكرة يتم حجزها لتخزين القيم فيها ويمكن تغيير القيم بعد تخزينها لذلك سميت بالمتغيرات

**طرق تعريف المتغيرات**

هناك طريقتان لتعريف المتغيرات وإسناد القيم لها موضحه في الجدول التالي مع الأمثلة :

| الطريقة | الصيغة | مثال |
|---|---|---|
| ١ | DataType VariableName = Value ; | int Age = 25; |
| ٢ | DataType VariableName ;<br>VariableName = Value ; | int Age ;<br>Age = 25; |

**طريقة طباعة المتغيرات**

يتم طباعة المتغيرات من خلال الصيغة التالية : System.out.print( VariableName);

**مثال :**

الناتج من تشغيل البرنامج :

```
Public class Test {
Public static void main (String[] args)  {

int Age = 22;
System.out.print( Age ) ;
} }
```

```
22
```

## ٢- الثوابت :

هي متغيرات لا يمكن تغيير قيمتها بعد تعريفها واسناد قيمة لها .

**طريقة تعريف الثوابت**

يتم تعريف الثوابت وإسناد القيم لها من خلال الصيغة التالية : final DataType ConstantName  = Value ;

**مثال :**

```
Public class Test {
Public static void main (String[] args)  {

final  Boolean Result = true ;

} }
```

**طريقة طباعة الثوابت :**

يتم طباعة الثوابت بنفس طريقة طباعة المتغيرات تماما .

## ٣- علامات الترقيم

من المعروف عند كتابة أي لغة استخدام ما يعرف بعلامات الترقيم بهدف :

- تحديد بداية أو نهاية بعض الجمل
- فصل الجمل عن بعضها حتى وإن كتبت في سطر واحد .

تحتوي لغة الجافا على مجموعة من علامات الترقيم والتي يكون لها معنى خاص أحيانا ، وهي موضحة في الجدول التالي :

| العلامة | اسمها | استخدامها |
|---|---|---|
| ; | الفاصلة المنقوطة | تستخدم لتحديد نهاية الأوامر أو التعليمات البرمجية |
| { } | اقواس الكتل البرمجية | تعبر عن بداية ونهاية مجموعة من الجمل المترابطة منطقيا لأداء وظيفة معينة . |
| | الفراغات | تستخدم لفصل المفردات في نفس الجملة وهذه ضرورية |
| ( ) | أقواس القيم الممررة | تستخدم لاستقبال قيم معينة للدوال |
| " " | علامة التنصيص المزدوجة | تستخدم لكتابة النصوص |
| ' ' | علامة التنصيص المفردة | تستخدم لكتابة الحروف |

## ٤- الكلمات المحجوزة

لكل لغة برمجة مجموعة من الكلمات المفتاحية التي تُسمى الكلمات المحجوزة **(Keywords)** ، حيث تكون هذه الكلمات مُخصصة لتنفيذ أوامر معينة و يتوجب على المُبرمج تجنب استخدامها أثناء عملية كتابة الكود البرمجي**(codes)** ، ولا يمكن استخدامها كأسماء متغيرات أو دوال أثناء عملية كتابة البرنامج و الجدول التالي يحتوي على أمثلة من الكلمات المحجوزة .

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

## ٥- العمليات

### أ- العمليات الحسابية

في لغة الجافا يمكن تنفيذ العمليات الحسابية الأساسية بالطريقة التالية :

| اسم العملية | رمز العملية | التعبير الجبري | التعبير بالجافا |
|---|---|---|---|
| جمع | + | A+5 | A+5 |
| طرح | - | A-5 | A-5 |
| ضرب | * | AB | A*B |
| قسمة | / | $\frac{A}{B}$ | A/B |
| Modules<br>حساب باقي القسمة | % | A mod S | A % S |

### ب- العمليات الإسنادية

تستخدم هذه العمليات لإسناد قيمة معينة لمتغير بعد تعريفه وتتم عملية الاسناد بالشكل التالي :

Variable name = value ;

**مثال :**

```
A=3 ;
Ch='B' ;
Name =" Sara" ;
```

كما يمكن كتابة عمليات الإسناد والعمليات الحسابية بشكل مختصر كما هو موضح في الجدول التالي :

| العملية | مثال | ما يقابله دون اختصار |
|---|---|---|
| += | A += 7 | A = A + 7 |
| -= | B -= 4 | B = B - 4 |
| *= | C *= 3 | C = C* 3 |
| /= | D /= 5 | D = D / 5 |
| %= | F %= 9 | F = F % 9 |

البرنامج التالي هو مثال على تنفيذ العمليات الحسابية وعمليات الاسناد معا :

```
Public class Test {
Public static void main (String[] args) {
int Z = 2;
Z*=3;
System.out.print( Z ) ;
} }
```

الناتج من تشغيل البرنامج :

```
6
```

**ج- عمليات الزيادة أو النقصان**

| العملية | مثال | توضيح |
|---|---|---|
| ++ | ++c | يتم زيادة المتغير c بمقدار ١ ثم تستخدم القيمة الجديدة للمتغير c في التعبير المتواجد فيه . |
| ++ | c++ | تستخدم القيمة الاولى للمتغير c في التعبير المتواجد فيه ثم يتم زيادة المتغير c بمقدار ١ |
| -- | --c | يتم إنقاص المتغير c بمقدار ١ ثم تستخدم القيمة الجديدة للمتغير c في التعبير المتواجد فيه . |
| -- | c-- | تستخدم القيمة الاولى للمتغير c في التعبير المتواجد فيه ثم يتم إنقاص المتغير c بمقدار ١ |

البرنامج التالي يوضح الفرق بين عمليتي الزيادة والنقصان .

```
Public class Test {
Public static void main (String[] args)
{
int C =5 ;
System.out.print( C ) ;
System.out.print( C++ ) ;
System.out.print( C ) ;

System.out.println( ) ;

C=5;
System.out.print( C ) ;
System.out.print( ++C ) ;
System.out.print( C ) ;
} }
```

الناتج من تشغيل البرنامج :

```
5
5
6

5
6
6
```

**د- عمليات المقارنة**

هي العمليات التي تقوم بالمقارنة بين القيم العددية ويكون الناتج من المقارنة قيمة منطقية ( **true / false** )

| اسم العملية | رمزها | مثال | توضيح |
|---|---|---|---|
| يساوي | == | **(a==b)** | هل قيمة **a** تساوي قيمة **b** ؟ إذا كان الجواب نعم فإنها ترجع **true** . |
| لا يساوي | != | **(a !=b)** | هل قيمة **a** لا تساوي قيمة **b** ؟ إذا كان الجواب نعم فإنها ترجع **true** . |
| أكبر من | > | **(a > b)** | هل قيمة **a** أكبر من قيمة **b** ؟ إذا كان الجواب نعم فإنها ترجع **true** . |
| أصغر من | < | **(a < b)** | هل قيمة **a** أصغر من قيمة **b** ؟ إذا كان الجواب نعم فإنها ترجع **true** . |
| أكبر من أو يساوي | >= | **(a >= b)** | هل قيمة **a** أكبر من أو تساوي قيمة **b** ؟ إذا كان الجواب نعم فإنها ترجع **true** . |
| أصغر من أو يساوي | <= | **(a<= b)** | هل قيمة **a** أصغر من أو تساوي قيمة **b** ؟ إذا كان الجواب نعم فإنها ترجع **true** . |

عند الحاجة للمقارنة بين السلاسل النصية من حيث التطابق فلا ينبغي استخدام عملية == في المقارنة لانها قد تؤدي إلى نتائج خاطئة و بدلا من ذلك يجب استخدام دالة **.equals()** المثال التالي يوضح الفرق بين استخدام عملية == و دالة **.equals()**

```
Public class Test {
Public static void main (String[] args)  {

String s1="sara ";
String s2="Mohammad";
String s3=s1+s2;

System.out.println(s3==(s1+s2));
System.out.println(s3.equals(s1+s2));
}}
```

الناتج من تشغيل البرنامج :

```
false
true
```

**هـ -العمليات المنطقية**

هي العمليات التي تقوم بالمقارنة بين القيم المنطقية ويكون الناتج من المقارنة قيمة منطقية أيضا ( **true / false**) لذلك سميت بهذا الاسم . الجدول التالي يوضح العمليات المنطقية ونواتجها .

| اسم المعامل | رمزه | مثال | AND-OR | | | |
|---|---|---|---|---|---|---|
| | | | A&&B | A\|\|B | B | A |
| **AND** | && | (a&&b) | True | True | True | True |
| **OR** | \|\| | (a\|\|b) | False | True | False | True |
| | | | False | True | False | True |
| **NOT** | ! | !a | False | False | False | False |

البرنامج التالي هو مثال على تنفيذ العمليات المنطقية وعمليات المقارنة معا.

```
Public class Test {
Public static void main (String[] args)  {

Int a = 10;
Int b = 20;

/*
الشرط التالي يعني أنه إذا كانت قيمة المتغير a تساوي ١٠ وقيمة المتغير b تساوي ٢٠ سيتم تنفيذ أمر الطباعة
*/
If (a==10 && b==20 )
{
System.out.println(" The first and the second conditions return true");
} } }
```

الناتج من تشغيل البرنامج :

```
The first and the second conditions return true
```

# جمل التحكم في سير التنفيذ

## جمل الاختبار الشرطية

جمل الاختبار هي جمل برمجية تحاكي عمليات الاختيار بين أمرين أو أكثر ويمكن تسميتها أيضا بجمل الاختيار الشرطية وهناك عدة أنواع من جمل الاختبار الشرطية و هي :

١. If
٢. If..else
٣. Switch

### ١. جملة الاختبار if

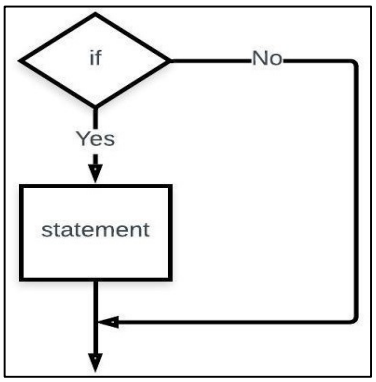
if في اللغة العربية تعني " إذا ", و هي تستخدم في حال كنا نريد تنفيذ امر معين عند تحقق شرط معين وفي حال عدم التحقق لا يتم تنفيذ أي أمر . تتكون جملة if الشرطية من ثلاثة أجزاء :

- اداة الشرط ( if ) .
- الشرط .
- جواب الشرط أو الجملة المراد تنفيذها .

يتم كتابة الأمر الخاص بجملة الشرط if على النحو التالي :

```
if ( الشرط ) {
الجملة أو الجمل المراد تنفيذها عند تحقق الشرط
}
```

الصورة التالية توضح مكونات جملة if الشرطية و طريقة عملها .




**مثال :**

```
Public class Test {
Public static void main (String[] args)  {

    int S = 30;

     if( S > 5 ) {
    System.out.print("S is bigger than 5");

    } } }
```

الناتج من تشغيل البرنامج :

```
S is bigger than 5
```

## ٢. جملة الاختبار if..else

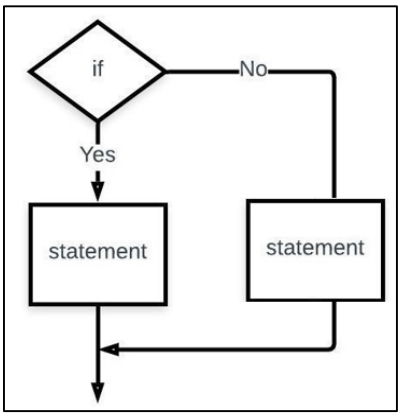
هي جملة شرطية تستخدم في حال كنا نريد تنفيذ امر معين عند تحقق الشرط وتنفيذ أمر آخرفي حال عدم التحقق . تتكون جملة if…else الشرطية من أربعة أجزاء :

- اداة الشرط ( if..else ) .
- الشرط .
- جواب الشرط أو الجملة المراد تنفيذها عند تحقق الشرط .
- الجملة المراد تنفيذها في حال لم يتحقق الشرط.

يتم كتابة الأمر الخاص بجملة الشرط if..else على النحو التالي :

```
if ( الشرط ) {
الجملة أو الجمل المراد تنفيذها عند تحقق الشرط
} else {
الجملة أو الجمل المراد تنفيذها عند عدم تحقق الشرط
}
```

الصورة التالية توضح مكونات جملة if..else الشرطية و طريقة عملها .




**مثال :**

```
Public class Test {
Public static void main (String[] args)  {

        int S = 20;

        if ( S == 5 ) {
            System.out.print("S is equal to 5");
        }
        else {
            System.out.print("S is not equal to 5");
        }
}}
```

الناتج من تشغيل البرنامج :

```
S is not equal to 5
```

## ٣. جملة الاختبار Switch

هي عبارة عن شكل من أشكال جمل الاختباروتعتمد على وجود خيارات مساوية للقيمة التي يتم اختبارها ويمكننا من خلالها الاختيار بين الخيارات الموجودة . تتكون جملة Switch الشرطية من أربعة أجزاء :

- switch : تحديد المتغير المطلوب اختبار قيمته
- Case : مقارنة قيمة متغير switch
- Break : تستخدم كأخر جملة في كل قائمة من قوائم جمل case وتعمل على نقل التحكم إلى نهاية جملة switch
- Default : في حالة عدم تطابق المدخلات مع الحالات تطبق جملة default وهي اختيارية .

يتم كتابة الأمر الخاص بجملة الشرط Switch على النحو التالي :

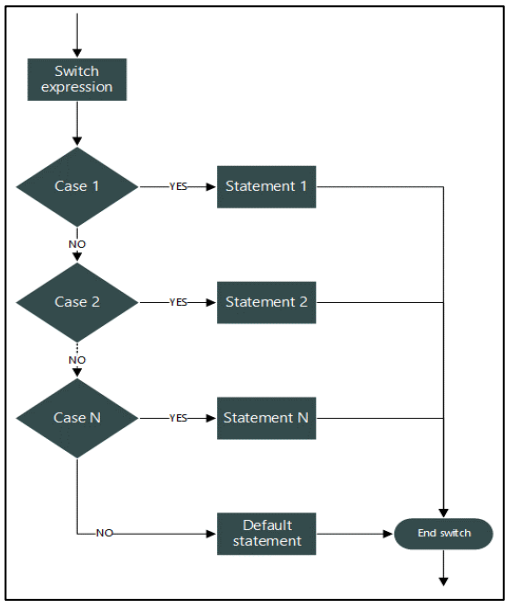
```
switch ( x )  {
case 1:
جمل برمجية تنفذ في حال كانت x = الحالة ١
break;

case 2:
جمل برمجية تنفذ في حال كانت x = الحالة ٢
break;

case 3:
جمل برمجية تنفذ في حال كانت x = الحالة ١
break;
.
.
default:
جمل برمجية تنفذ في حال كانت x لا تساوي أي حالة من الحالات السابقة
break;
  }
```

**المتغير x قد يكون من النوع :**
Char
Short
Byte
Int
String

الصورة التالية توضح مكونات جملة Switch الشرطية و طريقة عملها .



**مثال :**

```
Public class Test {
Public static void main (String[] args)  {

      int x = 2;

      switch( x ) {   // x اختبر قيمة المتغير

         case 1:   // في حال كانت تساوي ١ سيتم تنفيذ أمر الطباعة الموضوع فيها
            System.out.println("x contain 1");
            break;

         case 2:   // في حال كانت تساوي ٢ سيتم تنفيذ أمر الطباعة الموضوع فيها
            System.out.println("x contain 2");
            break;

         case 3:   // في حال كانت تساوي ٣ سيتم تنفيذ أمر الطباعة الموضوع فيها
            System.out.println("x contain 3");
            break;

         default:   // في حال كانت لا تساوي أي قيمة من القيم الموضوعة سيتم تنفيذ أمر الطباعة الموضوع فيها
            System.out.println("x contain a different value");

      }

   }

}
```

الناتج من تشغيل البرنامج :

```
x contain 2
```

## جمل التكرار

الدوران أو التكرار هو تنفيذ أمر أو مجموعة من الأوامر ككتلة واحدة بناء على شرط معين لعدد من المرات . وهناك عدة أنواع من جمل التكرار وهي :

١. while
٢. do..while
٣. for

### ١. جملة التكرار while

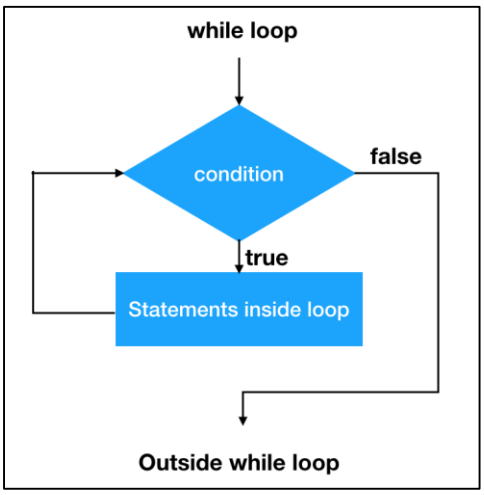
هي جملة تقوم بتنفيذ مجموعة من الأوامر بعدد غير معلوم من المرات بحيث يتم تنفيذ الأمر طالما كان الشرط صحيحا وهي سلسلة اختبار سابق أي تختبر الشرط أولا ثم تنفذ الأمر اعتماداً على نتيجة الشرط . تتكون جملة while من أربعة أجزاء :

- اداة التكرار: ( while ) .
- تعريف وتهيئة العداد : يحدد بداية الانطلاق للتكرار (من الكبير للصغير أو العكس )
- شرط : يستخدم لاستمرارية التكرار ويكون مرتبط بالعداد .
- تحديث العداد : يتم من خلاله التعديل على قيمة العداد

يتم كتابة الأمر الخاص بجملة التكرار while على النحو التالي :

```
تعريف وتهيئة العداد
While ( شرط ){
الجملة أو الجمل المراد تنفيذها بشكل متكرر
تحديث العداد }
```

الصورة التالية توضح مكونات جملة التكرار while و طريقة عملها .



**مثال :**

```
Public class Test {
Public static void main (String[] args)  {

 // هنا قمنا بتعريف المتغير الذي استخدمناه كعداد في الحلقة
        int i=1;

/*
  هنا أنشأنا حلقة while حيث تظل تنفذ الأوامر الموضوعة فيها طالما
                                أن قيمة العدد لا تزال أصغر أو تساوي ١٠
*/
        while( i<=10 )
        {
            // في كل دورة سيتم طباعة قيمة العداد ثم إضافة ١ عليها
            System.out.println( i );
            i++;
          }
    } }
```

الناتج من تشغيل البرنامج :

```
1
2
3
4
5
6
7
8
9
10
```

## ٢. جملة التكرار do..while

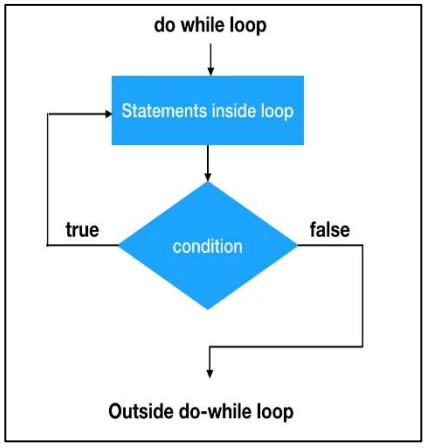
هي جملة تقوم بتكرار تنفيذ مجموعة من الجمل أو الأوامر عدد من المرات بحيث يتم تنفيذ الأمر طالما كان الشرط صحيحا ولكن يتم فحص الشرط بعد تنفيذ الأمر فهي سلسلة اختبار لاحق . تتكون جملة do..while من أربعة أجزاء :

- اداة التكرار: ( do..while) .
- تعريف وتهيئة العداد : يحدد بداية الانطلاق للتكرار (من الكبير للصغير أو العكس )
- شرط : يستخدم لاستمرارية التكرار ويكون مرتبط بالعداد .
- تحديث العداد : يتم من خلاله التعديل على قيمة العداد

يتم كتابة الأمر الخاص بجملة التكرار do..while على النحو التالي :

```
تعريف وتهيئة العداد
do {
الجملة أو الجمل  المراد تنفيذها بشكل متكرر
تحديث العداد
  } While  (شرط  ) ;
```

الصورة التالية توضح مكونات جملة التكرار do..while و طريقة عملها .



**مثال :**

```
Public class Test {
Public static void main (String[] args)  {

// هنا قمنا بتعريف المتغير الذي استخدمناه كعداد في الحلقة
        int x =1;

/*
   هنا أنشأنا حلقة do..while حيث تظل تنفذ الأوامر الموضوعة فيها
                طالما أن قيمة العدد لا تزال أصغر أو تساوي ١٠
*/
        do
        {
          // في كل دورة سيتم طباعة قيمة العداد ثم إضافة ١ عليها
            System.out.println( x );
            x++;
        } while( i<=10 );
  }}
```

الناتج من تشغيل البرنامج :

```
1
2
3
4
5
6
7
8
9
10
```

## ٣. جملة التكرار for

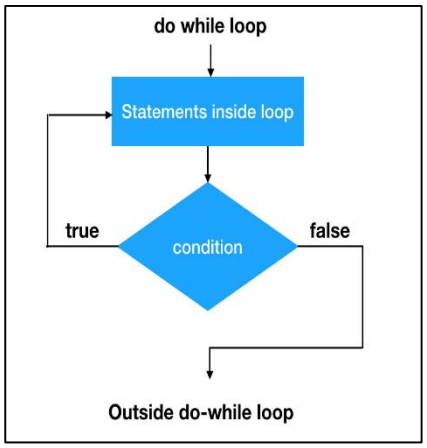
هي جملة تقوم بتنفيذ مجموعة من الأوامر بعدد معلوم من المرات بحيث يتم تنفيذ الأمر طالما كان الشرط صحيحا وهي سلسلة اختبار سابق . تتكون جملة for من أربعة أجزاء :

- اداة التكرار: ( for) .
- تعريف وتهيئة العداد : يحدد بداية الانطلاق للتكرار (من الكبير للصغير أو العكس )
- شرط : يستخدم لاستمرارية التكرار ويكون مرتبط بالعداد .
- تحديث العداد : يتم من خلاله التعديل على قيمة العداد

يتم كتابة الأمر الخاص بجملة التكرار for على النحو التالي :

```
for (تحديث العداد ؛ شرط ؛  تعريف وتهيئة العداد){
الجملة أو الجمل  المراد تنفيذها بشكل متكرر
}
```

الصورة التالية توضح مكونات جملة التكرار for وطريقة عملها .



**مثال :**

```
Public class Test {
Public static void main (String[] args)  {


// هنا قمنا بإنشاء حلقة for تتألف من ١٠ دورات.
// في كل دورة تطبع قيمة العداد المستخدم فيها
      for( int i=1; i<=10; i++ )
      {
         System.out.println( i );
      }
}}
```

الناتج من تشغيل البرنامج :

```
1
2
3
4
5
6
7
8
9
10
```

## جمل التحكم في التكرار

نستخدم جمل التحكم ( Control Statements ) للتحكم في سير تنفيذ الحلقات و مع جملة الشرط switch فيمكن استخدامها مثلا لإيقاف تنفيذ جمل التكرار قبل أن يصبح الشرط خاطئا .هناك نوعان من جمل التحكم هما :

- break
- continue

### ١- جملة التحكم break

جملة برمجية تستخدم لإجبار المترجم على الخروج من التكرار حتى و إن لم يتم عدد التكرارات المخصصة له سلفا ويمكن استخدامها أيضا مع الجملة الشرطية switch .

يتم كتابة الأمر الخاص بجملة التحكم break على سطر منفرد. على النحو التالي :

```
break ;
```

الصورة التالية توضح طريقة عمل جملة التحكم break .

**مثال :**

```
Public class Test {
Public static void main (String[] args)  {

// هنا قمنا بإنشاء حلقة for تتألف من ١٠ دورات
// في كل دورة تطبع قيمة العداد المستخدم فيها
      for( int i=1; i<=10; i++ )
      {

// في كل دورة سيتم فحص قيمة العداد و بمجرد أن تصبح مساوية للعدد ٦
// سيتم إيقاف الحلقة نهائيا
          if( i == 6 ) {
             break;
          }
          System.out.println( i );
      }
}}
```

الناتج من تشغيل البرنامج :

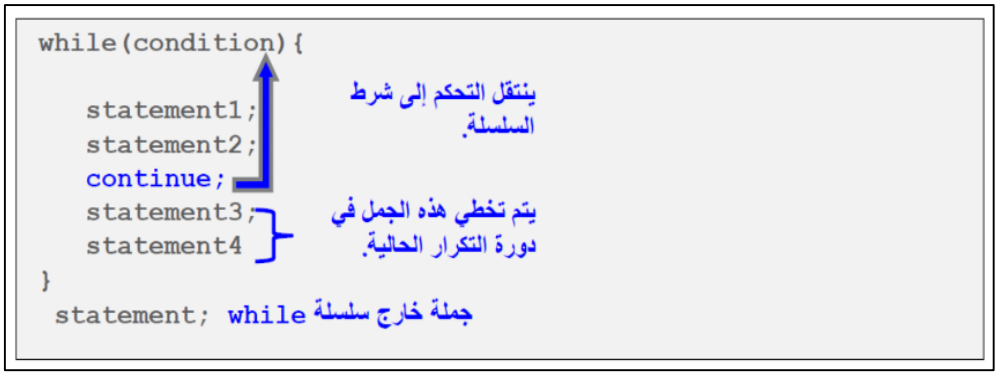
```
1
2
3
4
5
```

## ٢- جملة التحكم continue

جملة برمجية تستخدم لإجبار المترجم على إنهاء الدورة الحالية وبدء دورة جديدة بالعودة إلى رأس الدوران.ويتم كتابة الأمر الخاص بجملة التحكم continue على سطر منفرد. على النحو التالي :

```
continue;
```

الصورة التالية توضح طريقة عمل جملة التحكم continue.



**مثال :**

```
Public class Test {
Public static void main (String[] args)  {

// هنا قمنا بإنشاء حلقة for تتألف من ١٠ دورات
// في كل دورة تطبع قيمة العداد المستخدم فيها
        for( int i=1; i<=10; i++ )
        {

// في كل دورة سيتم فحص قيمة العداد و بمجرد أن تصبح مساوية للعدد ٣
// سيتم تخطي الدورة الحالية والانتقال للدورة التالية
            if( i == 3 ) {
               continue;
            }
            System.out.println( i );
        }
}}
```

الناتج من تشغيل البرنامج :

```
1
2
4
5
6
7
8
9
10
```

# الدوال

## تعريف الدوال :

الدالة تعني Method او Function في اللغة الإنجليزية، و هي عبارة عن مجموعة أوامر مجمعة في مكان واحد و تتنفذ عندما نقوم باستدعائها.
كما أن الجافا تحتوي على مجموعة كبيرة جداً من الدوال الجاهزة التي يمكنك استعمالها مباشرةً. كما يمكننا من إنشاء دوال خاصة تؤدي وظائف محددة.

**الهدف من كتابة الدوال:**

١. عدم الحاجة إلى تكرار التعليمات داخل البرنامج حيث يتم إنشاء الدالة مرة واحدة ويمكن استدعائها أكثر من مرة عند الحاجة إليها.
٢. باستخدام الدوال يصبح البرنامج أكثر وضوحا.
٣. باستخدام الدوال الجاهزة يمكن توفير الكثير من الوقت والجهد.

## أنواع الدوال:

**١. دوال جاهزة يمكن أن توفرها لغة الجافا:(Built In)**

وهي دوال تكون مكتوبة وموجودة مسبقا ويقوم المبرمج باستخدامها فقط، وتكون موجودة في ما يعرف بالمكتبات Libraries الخاصة بلغة الجافا. **مثل**: الدوال الرياضية، دوال التعامل مع النصوص، الدوال العامةو أيضا دوال الطباعة التالية :

```
System.out.print();
System.out.println();
System.out.printf();
```

**٢. دوال يمكن تعريفها عن طريق المستخدم:(User-defined)**

وهي مجموعة الدوال التي يتم انشاؤها من قبل المبرمج لأداء وظيفة معينة.

كما يمكن تقسيم الدوال إلى أنواع أخرى حسب عدة عوامل أهمها :

## أنواع الدوال حسب المشاركة بين الكائنات (Static, non static):

الدوال يتم تعريفها داخل الفئات والتي يمكن ان نشتق منها مجموعة من الكائنات وفي هذه الحالة توجد هنالك نوعين من الدوال:

- **مشتركة Static (Class member):**

أي ان هذه الدالة مشتركة (لها موقع واحد في الذاكرة) بين كافة الكائنات المشتقة من الفئة المحتوية على الدالة وعند استدعاء هذا النوع من الدوال لا نحتاج الى اشتقاق كائن من الفئة المحتوية على الدالة.

- **غير مشتركة Non Static (instance member):**

أي انه لكل كائن مشتق من الفئة قيما خاصة لكافة متغيرات الدالة وفي مواقع مختلفة من الدالة ولاستدعاء هذه الدالة يجب أولا اشتقاق كائن (Object) من الفئة المحتوية على الدالة.

## أنواع الدوال حسب القيمة المرجعة لسطر الاستدعاء Retured Value to Calling Code:

**سطر الاستدعاء (Calling Code):** هو السطر الذي تم عنده استدعاء الدالة في الدالة الرئيسية (Main Method).

- **دوال ترجع قيمة Return Value (Getter):**

هذا النوع من الدوال يقوم بتنفيذ تعليمات محددة ويقوم بإرجاع قيمة (يتم تحديد نوعها اثناء تعريف الدالة) الى سطر الاستدعاء بعد انتهاء التنفيذ.

❖ **دوال لا ترجع قيمة (Setter) Return no Value (void):**

هذا النوع من الدوال تقوم بتنفيذ تعليمات محددة دون ان تقوم بإرجاع قيمة الى سطر الاستدعاء (Calling code) بعد انتهاء التنفيذ.

## أنواع الدوال حسب احتوائها على المعاملات With or without Parameters:

**المعاملات:** هى عبارة عن قيم (متغيرات او ثوابت) يتم تمريرها الى الدالة اثناء استدعائها من خلال كتابة قيم او متغيرات مناظرة للمتغيرات المعرفة في راس الدالة في جملة الاستدعاء.

❖ **دوال لا تحتاج الى تمرير معاملات Have no Parameter:**

وهي دوال لا تحتاج الى تمرير قيم اثناء استدعائها حيث لا يتم كتابة أي قيم بين قوسي الدالة

❖ **دوال تحتاج الى تمرير معاملات Have Parameter:**

وهي دوال تحتاج الى تمرير (ارسال) معاملات (ثوابت او متغيرات) اثناء استدعائها.

**يتم كتابة المعاملات كقيم ثابتة او متغيرات تحمل قيما بين قوسين امام اسم الدالة اثناء استدعاء الدالة.**

## بناء الدوال في الجافا:

عند تعريف أي دالة في جافا عليك إتباع الشكل التالي:

```
modifier returnType methodName (Parameter List) {

    //Method Body
}
```

**modifier:** يحدد طريقة الوصول للدالة، ستفهم معنى هذه الكلمة في دروس لاحقة.
**returnType:** يحدد النوع الذي سترجعه الدالة عندما تنتهي أو إذا كانت لن ترجع أي قيمة.
**methodName:** يمثل الاسم الذي نعطيه للدالة، و الذي من خلاله يمكننا استدعاءها.
**Parameter List:** المقصود بها البارامترات ( وضع البارامترات اختياري ).
**Method Body:** تعني جسم الدالة، و المقصود بها الأوامر التي نضعها في الدالة.
الـ **returnType** في الدالة يمكن أن يكون أي نوع من أنواع البيانات الموجودة في جافا ( int - double - boolean - String إلخ.. ). ويمكن وضع اسم لكلاس معين، وهنا يكون القصد أن الدالة ترجع كائن من هذا الكلاس. في حال كانت الدالة لا ترجع أي قيمة، يجب وضع الكلمة void مكان الكلمة returnType كما في المثال التالي :

```
public void welcome (){

System.out.println("welcome");

}
```

## أمثلة حول تعريف دوال جديدة في جافا :

في المثال التالي قمنا بتعريف دالة اسمها **welcomeMessage**، نوعها void، و تحتوي على أمر طباعة فقط.
بعدها قمنا باستدعائها في الدالة main() حتى يتم تنفيذ أمر الطباعة الموضوع فيها.

```
public class Main {

// welcomeMessage هنا قمنا بتعريف دالة إسمها
 عند إستدعاءها تطبع جملة للترحيب

public static void welcomeMessage() {
System.out.println("Hello World");
}

public static void main(String[] args) {

// welcomeMessage هنا قمنا باستدعاء الدالة
لطباعة جملة الترحيب الموضوعة فيها

welcomeMessage();
}
```

الناتج من تشغيل البرنامج :

```
Hello World
```

في المثال التالي قمنا بتعريف دالة اسمها sum، عند استدعاءها نعطيها عددين فترجع ناتج جمع هذين العددين. بعدها قمنا باستدعائها في الدالة main().

```
public class Main {

//هنا قمنا بتعريف دالة إسمها sum عند إستدعاءها نعطيها
 عددين فترجع ناتج جمع هذين العددين

public static int sum(int a, int b) {
return a+b;
}

public static void main(String[] args) {

//هنا قمنا باستدعاء الدالة sum لحساب ناتج جمع
العددين ٥ و ١٠

System.out.println( "5 + 10 = " + sum(10, 5) );
```

الناتج من تشغيل البرنامج :

```
10 + 5 =15
```

# المصفوفات

## مفهوم المصفوفات في جافا:

**مصفوفة:** تعني Array في البرمجة. والمصفوفة عبارة عن كائن يحتوي على مجموعة عناصر من نفس النوع تتخزن بجوار بعضها في الذاكرة. بمعنى آخر المصفوفة عبارة عن كائن يمكنه تخزين عدة قيم من نفس النوع.
عناصر المصفوفة تتميز عن بعضها من خلال رقم محدد يعطى لكل عنصر يسمى index. أول عنصر فيها يتم تخزينه في الـ index رقم ٠ والعنصر الأخير فيها يحمل index رقم length-١.
عدد عناصر المصفوفة ثابت، أي بمجرد أن قمت بتحديده لا يمكنك تغييره من جديد، لكنك تستطيع تغيير قيم هذه العناصر متى شئت.

## أهمية المصفوفات في البرمجة:

١. تقليل عدد المتغيرات المتشابهة، فمثلاً إذا كنا نريد تعريف ١٠ متغيرات نوعهم int، نقوم بتعريف مصفوفة واحدة تتألف من ١٠ عناصر.
٢. تطوير الكود، إذا قمت بتخزين المعلومات داخل مصفوفة، تستطيع تعديلهم، مقارنتهم أو جلبهم كلهم دفعة واحدة بكود صغير جداً باستخدام الحلقات.
٣. تستطيع الوصول لأي عنصر من خلال index.

## أنواع المصفوفات:

١. ذات بعد واحد: أي One dimensional array
٢. ذات بعدين: أي Two dimensional array
٣. ذات عدة ابعاد: أي Multidimensional array

### ١. المصفوفة ذات البعد الواحد One dimensional array:

عبارة عن مصفوفة تتألف من سطر واحد فقط في المثال التالي يوجد مصفوفة تحتوي على سطر واحد فقط يتألف من ٥ أعمدة

**A1**

| | | | | |
|---|---|---|---|---|
| 10 | 2 | 5 | 13 | -4 |

### ٢. المصفوفة ذات البعدين Two dimensional array:

عبارة عن مصفوفة تتألف من أسطر وأعمدة في المثال التالي يوجد مصفوفة تحتوي على ٣ أسطر وكل سطر يتألف من ٥ أعمدة

**A2**

| | | | | |
|---|---|---|---|---|
| 10 | 2 | 5 | 13 | -4 |
| 6 | 7 | -1 | 0 | 3 |
| 8 | 4 | 9 | 21 | 5 |

## تعريف المصفوفات برمجيا :

يتم تعريف المصفوفات بالشكل التالي :

```
المصفوفة ذات البعد الواحد
Datatype[ ]  arrayName = new datatype[size1];

المصفوفة ذات البعدين
Datatype [ ] [ ]  arrayName = new datatype[size1] [size2];
```

**datatype**: هو نوع القيم الأولية التي سيتم توليدها.
**arrayName** : هو اسم المصفوفة التي يجب أن تكون معرفة سابقاً.
**new**: تقوم بتوليد قيم أولية لجميع عناصر المصفوفة، تعطيهم القيمة صفر كقيمة أولية.
**Size1**: هو عدد عناصر المصفوفة( الصفوف )
**Size2**: هو عدد عناصر المصفوفة( الأعمدة )

**أمثلة على تعريف المصفوفات**

مثال على تعريف مصفوفة ذات بعد واحد:

```
int[] OneDimentionalArray = new int[5];
```

مثال على تعريف مصفوفة ذات بعدين :

```
int[] TwoDimentionalArray = new int[5] [4] ;
```

## إسناد القيم لعناصر المصفوفة :

يتم إسناد قيم للمصفوفات بالشكل التالي :

```
المصفوفة ذات البعد الواحد
 arrayName[ k] = value ;

المصفوفة ذات البعدين
arrayName [k] [m]= value ;
```

value : القيمة المراد تخزينها

K : رقم الصف للعنصر

M : رقم العمود للعنصر

**أمثلة على إسناد والقيم للمصفوفات**

**إسناد قيم لعناصر مصفوفة ذات بعد واحد**

```
OneDiementionalArray[0] = 7;
OneDiementionalArray[1] = 40;
```

**إسناد قيم لعناصر مصفوفة ذات بعدين**

```
TwoDiementionalArray[0][0] = 6;
TwoDiementionalArray[1][2] = -3;
```

كما يمكن إنشاء مصفوفة وإعطائها قيم أولية مباشرةً عند إنشائها كما يلي :

```
DataType [ ] arrayName = { value0, value1, …, valuek };
```

**arrayName :** هو اسم المصفوفة

**value0 و value1 و valuek:** عبارة عن القيم التى نعطيها للمصفوفة.

## مفهوم الخاصية length:

عبارة عن ثابت يمكنك اعتباره متغير عادي تملكه كل مصفوفة يتم تعريفها بشكل تلقائي، تستخدم هذه الخاصية لمعرفة عدد عناصر المصفوفة، او كما يقال لمعرفة حجم المصفوفة بشكل عام، نحتاج استخدامها عند بناء كود للتعامل مع المصفوفة مهما كان حجمها.

**طريقة استخدام الخاصية length مع المصفوفات:**

لاستخدام الخاصية length الموجودة في أي مصفوفة نضع اسم المصفوفة، ثم نقطة، ثم الكلمة length كالتالي:

```
arrayName.length;
```

# الكائنات و الكلاسات

## الكلاسات classes

**مفهومها**
الكلاس عبارة عن حاوية كبيرة تستطيع أن تحتوي على كل الكود من متغيرات ودوال وكائنات إلخ..

**طريقة تعريفها في الجافا**

```
Class classname {
Attributes
methods
}
```

Classname : اسم الكلاس المراد انشائه
Attributes : الخصائص أو المتغيرات العامة
Methods : الدوال

**مثال**

```
Class Person {
Int x=3;
Public void printInfo () { }
}
```

أي متغيرات يتم تعريفها بداخل كلاس وخارج أي دالة تسمى خصائص (Attributes). وهذا يعني أن أي كائن من هذا الكلاس سيكون عنده هذه الخصائص. تستطيع التعامل مع هذه الخصائص من الكائن مباشرةً، بينما المتغيرات العادية لا يمكنك التعامل معها من الكائن. المتغيرات التي يتم وضعها كبارامترات أو التي يتم تعريفها بداخل الدوال تسمى متغيرات عادية.

## الكائنات objects

**مفهومها**
الكائن عبارة عن نسخة مطابقة لكلاس معين. و بما أن الكائن عبارة عن نسخة من الكلاس، يمكننا القول إنه لا يمكن إنشاء كائن إذا لم يكن هناك كلاس.
إذاً في مفهوم برمجة الكائنات نقوم بإنشاء كلاس معين يسمونه **blue print** أي (النسخة الخام أو النسخة الأصلية)، و بعدها ننشئ نسخة أو أكثر من هذا الكلاس و نفعل بها ما نريد بدون أن نغير محتويات الكلاس الأساسي و هكذا نكون حافظنا على كودات الكلاس الأساسي لأننا نعدل على النسخ و ليس عليه مباشرةً.

**طريقة تعريفها في الجافا**
بما أن الكائن عبارة عن نسخة من الكلاس. لتعريف كائن من كلاس معين يجب وضع اسم الكلاس ثم وضع اسم للكائن.

```
ClassName ObjectName = new ClassName();
```

Classname : اسم الكلاس المراد انشاء كائن منه .
ObjectName : اسم الكائن المراد إنشائه .

**مثال**

```
Person ahmad = new Person();
```

هنا قمنا بتعريف كائن من الكلاس Person اسمه ahmad. إذاً الكائن ahmad سيكون عنده نسخة خاصة فيه من خصائص الكلاس Person.
**ملاحظة:** الكود new Person() هو الذي يقوم فعلياً بتوليد كائن من الكلاس. وهو يعطي قيم أولية للخصائص الموجودة فيه وسيتم شرح ذلك لاحقاً.

**كيفية استدعاء عناصر الكائن**

١. نضع اسم الكائن.
٢. ثم نقطة.
٣. ثم الشيء الذي نريد الوصول إليه (سواء اسم متغير أو دالة).

**مثال :**

```
A.x= 15 ;
A.printInfo() ;
```

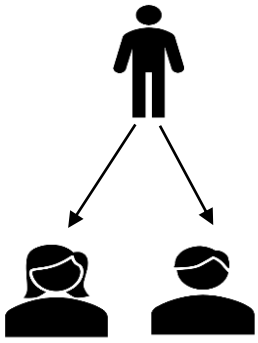
**نصائح علينا إتباعها عند إنشاء الكلاسات والكائنات :**

١. يفضل إنشاء كل كلاس في ملف جافا خاص.
٢. إبدا اسم الكلاس دائماً بحرف كبير.
٣. إبدا اسم الكائن دائماً بحرف صغير.

## علاقة الكلاس والكائن في الجافا

الكائنات تساعد المبرمج كثيراً، فمثلاً إذا كنت تنوي إنشاء برنامج بسيط لحفظ معلومات أشخاص، هل ستنشئ كلاس لكل شخص ؟! طبعاً لا، بل تنشئ كلاس واحد فقط يمثل شخص، وتضع فيه الأشياء الأساسية التي تريدها أن تكون موجودة عند كل شخص. ثم تنشئ منه كائنات قدر ما شئت، وعندها يصبح كل كائن من هذا الكلاس عبارة عن شخص له معلوماته الخاصة.

| أنا كلاس اسمي Person عبارة عن إنسان واملك الخصائص التالية | | | |
|---|---|---|---|
| الاسم | الجنس | الوظيفة | العمر |



| نحن كائنات من الكلاس Person | |
|---|---|
| الاسم: محمد | الاسم: روز |
| الجنس: ذكر | الجنس: انثى |
| الوظيفة: مبرمج | الوظيفة: معلمة |
| العمر: ٢١ | العمر: ٢٢ |

كما نلاحظ قمنا بإنشاء كلاس يحتوي على المعلومات الأساسية التي نريد تعبئتها لكل شخص.
بعدها قمنا بإنشاء كائنين (أي شخصين )، ثم قمنا بإدخال معلومات خاصة لكل كائن فيهم.
الآن في حال قمت بإضافة أي متغير أو دالة جديدة في الكلاس Person، فإن أي كائن من هذا الكلاس سيملك نسخة من الشيء الجديد الذي أضفته.
وفي حال قمت بتعديل كود معين في الكلاس Person، فأيضاً سيتم تعديل هذا الكود عند جميع الكائنات من هذا الكلاس.

**مثال :**

**Person.java:**

```
public class Person {

// هنا قمنا بتعريف ٤ خصائص
String name;
String gender;
String job;
int age;

// هنا قمنا بتعريف دالة تطبع محتوى كل خاصية عندما يتم استدعاءها
void printInfo() {
System.out.println("Name: " +name);
System.out.println("Gender: " + gender);
System.out.println("Job: " +job);
System.out.println("Age: " +age);
System.out.println();
}

}
```

**Main.java:**

```
public class Main {

public static void main(String[] args) {

// Person هنا قمنا بإنشاء كائنات من الكلاس
Person p1 = new Person(); // الكائن p1 سيمثل محمد
Person p2 = new Person(); // الكائن p2 سيمثل روز

// p1 هنا قمنا بتحديد خصائص الكائن
p1.name = "Mhamad";
p1.sex = "Male";
p1.job = "Programmer";
p1.age = 21;

// p2 هنا قمنا بتحديد خصائص الكائن
p2.name = "Rose";
p2.sex = "Female";
p2.job = "Teacher";
p2.age = 22;

// هنا قمنا بعرض خصائص كل كائن
p1.printInfo();
p2.printInfo();
}
}
```

النتيجة من تشغيل البرنامج :

```
Name: Mhamad
Sex: Male
Job: Programmer
Age: 21

Name: Rose
Sex: Female
Job: Teacher
Age: 22
```

## الكونستركتور constructor

**مفهوم الكونستروكتور :**

من أهم الأشياء التي عليك التفكير بها بعد إنشاء كلاس جديد، هي تسهيل طريقة خلق كائنات من هذا الكلاس.
من هنا جائت فكرة الكونستركتور و الذي هو عبارة عن دالة لها نوع خاص، يتم إستدعائها أثناء إنشاء كائن لتوليد قيم أولية للخصائص الموجودة فيه.
بما أنه لا يمكن إنشاء كائن من كلاس إلا من خلال كونستركتور، سيقوم مترجم جافا بتوليد كونستركتور إفتراضي فارغ عنك إذا وجد أن الكلاس الذي قمت بتعريفه لا يحتوي على أي كونستركتور.

**نقاط مهمة حول الكونستركتور:**

- ✓ كل كلاس يتم إنشاؤه، يحتوي على كونستركتور واحد على الأقل. وحتى إن لم تقم بتعريف أي كونستركتور، سيقوم المترجم بإنشاء واحد افتراضي عنك.
- ✓ في كل مرة يتم إنشاء كائن جديد، يجب استدعاء كونستركتور حتى يتم إنشاء هذا الكائن.
- ✓ القاعدة الأساسية عند تعريف كونستركتور هي أنه يجب أن يحمل نفس اسم الكلاس ويكون نوعه public.
- ✓ في حال قمت بتعريف كونستركتور، لن يقوم المترجم بإنشاء واحد افتراضي، أي لن يعود هناك كونستركتور افتراضي.
- ✓ يمكنك تعريف أكثر من كونستركتور. ويمكنك دائماً إنشاء كونستركتور فارغ، حتى تستخدمه إن كنت لا تريد إعطاء قيم أولية محددة للخصائص عند إنشاء كائن.

الآن سنرجع إلى الكلاس Person، و سنضيف فيه ٢ كونستركتور، واحد فارغ ( أي مثل الافتراضي )، و آخر يمكننا من خلاله إدخال قيم مباشرةً في الخصائص الموجودة في الكائن بدل استدعاء كل خاصية موجودة فيه.

**مثال :**

**Person.java:**

```
public class Person {

// هنا قمنا بتعريف ٤ خصائص
String name;
String gender;
String job;
int age;

   // إفتراضي constructor فارغ، أي كأننا قمنا بتعريف constructor هنا قمنا بتعريف
   public Person() {          }

   // هنا قمنا بتعريف constructor ثاني، الهدف منه إعطاء قيم لجميع الخصائص الموجودة في الكائن عند إنشاءه مباشرةً
   // عند استدعاء هذا الـ constructor عليك إدخال ٤ قيم من نفس النوع و بالترتيب الموضوع
   public Person(String n, String s, String j, int a) {
      name = n;        // name سيتم وضعه كقيمة للخاصية n الذي سيتم تخزينه في String الـ
      gender = s;      // gender سيتم وضعه كقيمة للخاصية s الذي سيتم تخزينه في String الـ
      job = j;         // job سيتم وضعه كقيمة للخاصية j الذي سيتم تخزينه في String الـ
      age = a;         // age سيتم وضعه كقيمة للخاصية a الذي سيتم تخزينه في int الـ
   }

// هنا قمنا بتعريف دالة تطبع محتوى كل خاصية عندما يتم استدعاءها
void printInfo() {
System.out.println("Name: " +name);
System.out.println("Gender: " + gender);
System.out.println("Job: " +job);
System.out.println("Age: " +age);
System.out.println();
} }
```

**Main.java:**

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

// Person هنا قمنا بإنشاء كائنات من الكلاس
      Person p1 = new Person("Mhamad", "Male", "Programmer", 21);
 // الكائن p1 يمثل الشخص محمد مع تحديد كامل خصائصه
      Person p2 = new Person("Rose", "Female", "Teacher", 22);
 // الكائن p2 يمثل الشخص روز  مع تحديد كامل خصائصه
 // هنا قمنا بعرض خصائص كل كائن
p1.printInfo();
p2.printInfo();
}
}
```

## أنواع المتغيرات في الكلاس

| نوع المتغير | التوضيح |
|---|---|
| **Local Variables** | هي المتغيرات التي يتم تعريفها بداخل أي دالة، كونستركتور، أو بداخل block (مثل الحلقات، الجملة switch إلخ.. ). |
| **Instance Variables** | هي المتغيرات التي يتم تعريفها بداخل الكلاس وخارج حدود أي دالة، كونستركتور، أو block.<br>تسمى أيضاً Global Variables. |
| **Class Variables** | هي المتغيرات التي يتم تعريفها كـ static بداخل الكلاس وخارج حدود أي دالة، كونستركتور، أو block. |

**مثال :**

```
class VariablesTypes {

   // block المتغيرات ( a, b, c, d ) تعتبر Instance Variables لأنه تم تعريفهم بداخل الكلاس و خارج أي دالة أو
   // ستفهم معناها في الدرس التالي، لكننا وضعناها فقط لتفهم الأسماء المستخدمة ( public, protected, private )
  الكلمات
   int a;
   public int b;
   protected int c;
   private int d;

   // static المتغير e يعتبر Class Variable لأن نوعه
   static int e;

  // المتغيرات ( x, y, z ) تعتبر Local Variables لأنه تم تعريفها بداخل الدالة
   public int sum(int x, int y) {
      int z = x + y;
      return z;
   }
}
```

## الكلمة this في جافا:

الكلمة this هي كلمة محجوزة في لغة جافا، وهي تستخدم للإشارة إلى الـ Global Variables، وتستخدم أيضاً للإشارة إلى الكائن الحالي. ويمكن استخدامها في أماكن عديدة .
في هذا الدرس سنستخدمها للتفرقة بين المتغيرات التي تم تعريفها بداخل الدوال Local Variables وبين المتغيرات التي تم تعريفها بداخل الكلاس وخارج الدوال Global Variables.
سنرجع إلى الكلاس Person وسنقوم باستخدام الكلمة this عدة مرات لمعرفة تأثيرها على الكود.
في هذا المثال لم نغير أي كود كان موجود، لكننا أضفنا كلمة this في كل مكان كنا نقصد فيه أننا نريد الوصول للخصائص.

```
public class Person {

    String name;
    String gender;
    String job;
    int age;

    public Person() {

    }

    // هنا لا يوجد داعي لاستخدام الكلمة this لأن أسماء الباراميترات الموضوعة ليست نفسها أسماء الخصائص
    public Person(String n, String s, String j, int a) {
        this.name = n;
  // القيمة التي سيتم إدخالها في المتغير n الموجود في الدالة، سيتم وضعها في الخاصية name الموجودة في الكلاس
        this.gender = s;
  // القيمة التي سيتم إدخالها في المتغير s الموجود في الدالة، سيتم وضعها في الخاصية gender الموجودة في الكلاس
        this.job  = j;
   // القيمة التي سيتم إدخالها في المتغير j الموجود في الدالة، سيتم وضعها في الخاصية job الموجودة في الكلاس
        this.age  = a;
  // القيمة التي سيتم إدخالها في المتغير a الموجود في الدالة، سيتم وضعها في الخاصية age الموجودة في الكلاس
    }

// هنا لا يوجد داعي لاستخدام الكلمة this لأن الدالة لا تحتوي على باراميترات
// بالتالي سيفهم المترجم أنك تقصد عرض قيم الخصائص الموجودة في الكائن حتى لو لم تستخدمها
    void printInfo() {
        System.out.println("Name: " + this.name);
        System.out.println("Gender: "  + this.gender);
        System.out.println("Job: "  +t his.job);
        System.out.println("Age: "  + this.age);
        System.out.println();
    }

}
```

| المراجع | |
|---|---|
| ١ | الحقيبة التدريبية اساسيات برمجة الحاسب الآلي للمهندس عبدالمجيد العتيبي |
| ٢ | موقع **Harmash** لتعليم البرمجة باللغة العربية<br>**/https://harmash.com/java/java-overview** |
| ٣ | العروض التقديمية الخاصة بالمدربة ساره ثابت |


تمت بحمد الله