

الفصل الثالث

٣

الدوال وإدخال البيانات وعرضها

الفصل الثالث : الدوال وإدخال البيانات وعرضها

الدوال في الجافا

الدالة تعنى الطرق أو المناهج، وفي اللغة الإنجليزية Method أو Function وهي مجموعة من الأوامر والتعليمات المجموعة تحت اسم واحد وفي مكان واحد يتم تنفيذها عندما نقوم باستدعائها. فالتطبيقات و برمجيات الحاسب يتم بناءها بواسطة كتابة مئات الأسطر البرمجية، فكما هو معروف في أي لغة برمجة إذا واجهتنا أي مشكلة كبيرة فإن أفضل طريقة لحلها هي تقسيمها لمجموعة من الأجزاء الصغيرة أو ما يعرف بـ module أو function الدوال بحيث كل منها تؤدي وظيفة معينة ويعرف هذا الأسلوب عادة بما يسمى بـ divide and conquer ، وتقسيم الكود على عدد من الدوال يجعل الكود أسهل في القراءة وأكثر وضوحاً.

وهذه الدوال تعرف في C# ، java بما يسمى بـ Method والمبرمج يستطيع كتابة Method لتعريف مهام معينة ومن ثم استدعائها calling من أي نقطة من البرنامج بمجرد ذكر اسمها لكي يتم تنفيذها عند تلك النقطة.

والمبرمج إما أن يقوم بكتابة الـ method بنفسه عندها تسمى user-defined method ، أو أن يقوم باستخدام دوال معرفة في نفس اللغة build in وكل ما عليه استدعائها عندما يحتاج إليها ليستخدمها في برنامجه

خصائص الدوال

- يمكن استدعائها في أي نقطة في البرنامج فقط بكتابة اسمها
- بعد تنفيذها يتم الرجوع إلى نقطة الاستدعاء في الدالة الرئيسية
- تساعد الدوال في تنظيم وتنسيق البرنامج
- تقسيم البرنامج إلى مجموعة برامج فرعية بحيث يكون لكل منها وظيفة محددة
- تساعد في اختصار الكود وعدم تكرارته
- تساعد في عملية متابعة الكود وتتبعه لتسهيل تصحيح الكود وتحديثه

أنواع الدوال في لغة الجافا Type Of Method In Java

تنقسم الدوال في الجافا إلى نوعين رئيسيين:

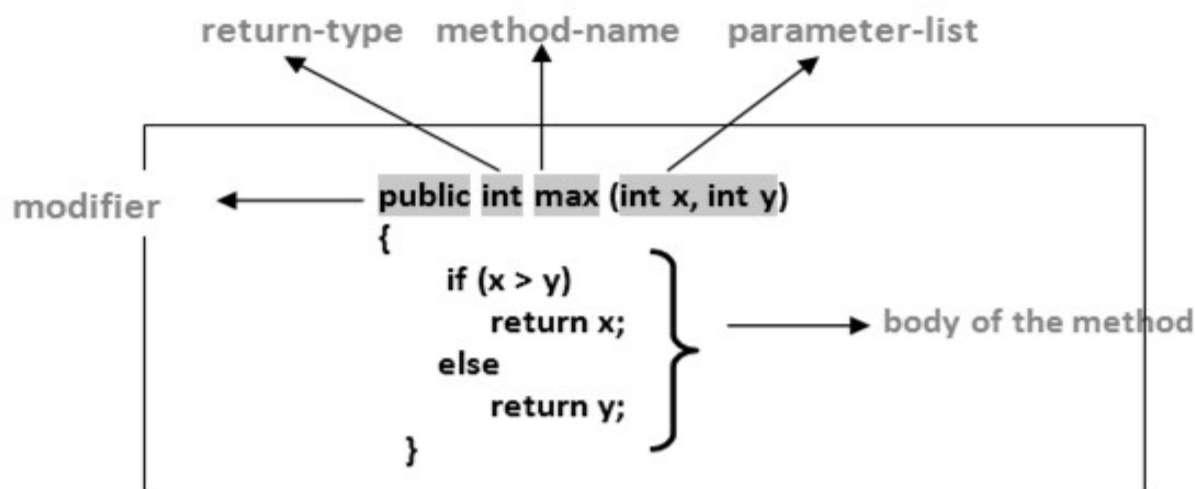
- ١- الدوال الجاهزة Build-In : وهي الدوال المبرمجة مسبقاً ضمن حزمة المترجم الخاص بلغة الجافا (الدوال الرياضية - دوال تتعامل مع النصوص - دوال عامة)
- ٢- دوال مكتوبة بواسطة المبرمج User-Defined : هي الدوال المعرفة من قبل المبرمج لتنفيذ وظيفة معينة وتتكون من جزئين رئيسيين :

```
Method-Header( ) // رأس الدالة
{
Method-Body      // جسم الدالة
}
```

بناء الدوال في الجافا

عند تعريف أي دالة في الجافا يتم إتباع الشكل التالي:

```
access_modifier return -value-Type methodName(Parameter List){  
    // Method Body  
}
```



حيث أن :

Access modifier

يحدد طريقة الوصول للدالة (`public` أو `private` أو `static`)
مثلا عندما يتم كتابة `public` فإن ذلك يعني أن كل الـ `Activities` تستطيع الوصول للدالة أي أنها عامة ،
وعندما يتم كتابة `private` فإن ذلك يعني أن الدالة خاصة يمكن الوصول إليها فقط من داخل `Activity` الحالية.

return -value-Type

يحدد نوع النتيجة التي ستعود بها الدالة عندما تنتهي (إذا كانت الدالة تعود بقيمة فإنها لا تعود بأكثر من قيمة واحدة) أو إذا كانت لن ترجع أي قيمة
ويمكن أن يكون `returnType` أي نوع من أنواع البيانات الموجودة في جافا (`int - double - boolean`)
(`- String`)

الفصل الثالث : الدوال وإدخال البيانات وعرضها

مثلا :

-إذا كانت الدالة تعود بقيمة integer فإننا سنكتب في هذه الخانة integer وهكذا
- أما في حالة إذا كانت الدالة لا تعود بقيمة فإننا نكتب هنا void
(في حالة إذا كانت الدالة لا ترجع أي قيمة يجب وضع الكلمة void مكان الكلمة returnType ، كما يمكن وضع اسم لكلاس معين وهنا يكون القصد أن الدالة ترجع كائن من هذا الكلاس

: methodName

يمثل الاسم الذي نعطيه للدالة و الذي من خلاله يمكننا استدعاءها ، وقواعد كتابة الاسم هنا يتبع نفس شروط كتابة أسماء المتغيرات

: Parameter List

المقصود بها الباراميترات (وضع الباراميترات اختياري) وهي القيم التي نقوم بإرسالها أي ادخالها إلى للدالة.

: Method Body

جسم الدالة وهي الأوامر التي نضعها في الدالة.

معايير تصنيف الدوال المعرفة من قبل المبرمج

١- قابلية الوصول Access Modifier

- عامة Public : يمكن الوصول إليها من أي مكان في المشروع

- خاصة Private : لا يمكن الوصول إليها إلا من نفس الفئة المعرفة فيها (من نفس class)

- محمية Protected : لا يمكن الوصول إليها إلا من خلال الفئة نفسها أو الفئات الوارثة من الفئة

٢- المشاركة بين الكائنات (Static and Non-Static)

بما أن الدوال يتم تعريفها داخل الفئات class إذا يمكن أن نشق منها مجموعة كائنات objects وفي هذه الحالة يوجد نوعان من الدوال:

-دوال مشتركة Static : هذه الدالة مشتركة بين كافة الكائنات المشتقة من الفئة المحتوية علي الدالة وعند الاستدعاء لا تحتاج كائن جديد من الفئة المحتوية علي الدالة

-دوال غير مشتركة None-Static : كل كائن مشتق من الفئة له قيمة خاصة لكل متغيرات الدالة ولاستدعاء هذه الدوال يجب اشتقاق كائن جديد من الفئة المحتوية علي الدالة

تصنيف الدوال من حيث القيمة الراجعة Returned value to calling code

١- دوال ترجع قيمة return value : تسمى (Getter) هذا النوع من الدوال يقوم بتنفيذ تعليمات محددة ثم تقوم بإرجاع القيمة حسب نوع بيانات الدالة الي سطر الاستدعاء بعد انتهاء التنفيذ ويجب ان تحتوي الدالة علي الكلمة المحجوزة return

1	int sum(4,5)
2	{
3	int z=4+5;
4	return z;
5	}
6	عند استدعاء هذه الدالة فأنها ترجع قيمة الجمع //

الفصل الثالث : الدوال وإدخال البيانات وعرضها

٢- دوال لا ترجع قيمة `return no value` : تسمى (setter) وهي دوال من النوع الخالي `void` هذا النوع يقوم بتنفيذ تعليمات محددة دون إرجاع قيمة الي سطر الاستدعاء بعد انتهاء التنفيذ

```
1 public class Test {
2     public static void printName( ){
3         System.out.println("gagacadmey.blogspot.com");
4     }
5     public static void main(String[] args) {
6         // تم استدعاء الدالة مع ملاحظة أنه لم تمرر اليها اي قيمة
7         printName( );
8     }
9 }
10 }
```

- سطر الاستدعاء `calling code` : هو السطر الذي تم عنده استدعاء الدالة في الدالة الرئيسية **Main Method**

طرق تمرير المعاملات الي الدالة

المعاملات هي قيم متغيرات او ثوابت يتم تمريرها الي الدالة أثناء الاستدعاء
- تسمى المعاملات الموجودة في الدالة الرئيسية (المكتوبة امام جملة الاستدعاء) بالمعاملات الفعلية
- تسمى المعاملات المناظرة لها (المكتوبة في رأس الدالة) في الدالة التي تم استدعاؤها بالمعاملات الشكلية او المنسوخة او الصورية

تمرير المعاملات بالقيمة Pass By Value

- في هذه الطريقة يتم إرسال صورة من المعامل الفعلي الي المعامل الصوري او المنسوخ
- المعاملين الفعلي والصوري لا يشتركان في نفس موقع الذاكرة
- المعامل الفعلي لا يتأثر بالتغيرات في المعامل الصوري
- هذا النوع من التمرير يتم تطبيقه تلقائيا عندما يكون نوع المعاملات الفعلية من انواع البيانات البدائية (int -double- float)

تمرير المعاملات بالمرجع Pass By Reference

- يتم إرسال عنوان المعامل الفعلي في الذاكرة الي المعامل الصوري المناظر له في الدالة المستدعية
- المعاملين الفعلي والصوري يشيران الي نفس عنوان الذاكرة
- كل تغير في المعامل الصوري سيتم تطبيقه في المعامل الفعلي
- هذا النوع يتم تطبيقه عندما تكون المعاملات الفعلية من نوع كائن **Object** مثل المصفوفات

الفصل الثالث : الدوال وإدخال البيانات وعرضها

تصنيف الدوال من حيث المعاملات With or Without Parameters

١- دوال تحتاج الي معاملات Have Parameter : وهي دوال تحتاج الي تمرير معاملات اليها عند استدعائها

1	return_type fun_name(param1 , param2)
2	{
3	fun_body
4	}
5	// عند الاستدعاء فان القيمة الراجعة ستكون نتيجة تنفيذ ما بداخل الدالة

حيث أن :

return_type : نوع القيمة الراجعة (int , float , double , char , string)
fun_name : أسم الدالة
parameters : المعاملات
fun_body : محتوى الدالة

٢- دوال تستقبل بارميتر وتعود بقيمة

نريد كتابة دالة وظيفتها أن تعود لنا بقيمة (عبارة عن نتيجة العملية الحسابية) ، أي الدالة تستقبل قيمة عددية integer وتعود بقيمة عددية integer والتي هي الرقم ١٠ مضروباً في القيمة التي تم تمريرها للدالة.

1	public Integer fourthMethod(Integer number) {
2	return number * 10 ;
3	}

حيث أننا في بداية تعريف الدالة في السطر الأول قبل اسم الدالة كتبنا integer معناه أن الدالة ستعود بالقيمة العددية integer وكتبنا (integer number) أي أن الدالة ستستقبل قيمة من النوع integer مخزنة في متغير يسمى number

return 10 * number;

السطر السابق يعنى أن الدالة تعود بالقيمة ١٠ مضروبة في البارميتر المرسل للدالة، والكلمة المفتاحية return تستخدم لكي تعود لنا بقيمة

الفصل الثالث : الدوال وإدخال البيانات وعرضها

وبعد انشاء الدالة يتم مناداتها من داخل الـ onCreate كما يلي:

```
Integer result = fourthMethod(6);
```

٣- دالة تستقبل parameter ولكنها لا تعود بقيمة ، كما يلي:

```
1 public void secondMethod(String name) {  
2     String result;  
3     result = "hello: " + name;  
4     Log.v("second method", result);  
5 }
```

تستقبل الدالة هنا قيمة parameter لذا نكتب بين القوسين اسم المتغير الذي يستقبل هذه القيمة التي ستدخل للدالة وكذلك نكتب نوع هذا المتغير والذي سيكون من نفس نوع القيمة المراد تمريرها للدالة والتي سيحتفظ بها المتغير ، وكما هو واضح فإن الدالة تستقبل القيمة المخزنة في name ثم تضيف العبارة hello قبل الاسم وأخيرا تطبع النتيجة على Log

فقط تبقى مناداة الدالة حتى تنفذ عملها لذا لا بد من ارسال قيمة لهذه الدالة لكي تقوم بعملها وتلك القيمة ستكون من النوع String ، ويتم مناداة الدالة كما يلي:

```
secondMethod("Elsaeed");
```

٤- دوال لا تحتاج الي معاملات Have No Parameter : هي دوال لا تحتاج الي تمرير قيم عن الاستدعاء حيث لا تكتب اي قيم بين قوسين الدالة

```
1 void printmessage()  
2 {  
3     System.out.print("I Am Function Without Any Parameter");  
4 }  
5 // عند الاستدعاء تطبع هذه الدالة ما بداخل جملة الطباعة فقط
```

نلاحظ ما يلي:

- الدالة السابقة لا تحتوي علي الكلمة المحجوزة return لأنها دالة من النوع الخالي أي لا ترجع اي شيء
- عند استدعاء الدوال يجب كتابة أسمها وإرفاق المعاملات بين الاقواس، وإذا كانت لا تحتوي على معاملات تكتب الاقواس فارغة)

```
printmessage( );
```

فائدة وجود الاقواس لإخبار المترجم أن هذه دالة وليست متغير عادي

الفصل الثالث : الدوال وإدخال البيانات وعرضها

٥- دوال لا نمرر لها بارامترات ولكن تعود بقيمة
معنى أن الدالة لا تمرر أو لا تستقبل parameter أنه عند إنشاء الدالة فإن القوسين ستكون فارغة () أي أنه لا توجد قيمة تريد الدالة استقبالها.

ومعنى أن الدالة تعود بقيمة أي عندما ننشئ الدالة سنستخدم keyword وهي return وتكون متبوعة بالقيمة التي تعود بها الدالة ، ولا تنسى أنه يجب أن تكتب في الـ Method header نوع القيمة التي ستعود بها الدالة في خانة الـ ... return_value_type

مثال:

إنشاء دالة لطباعة الجملة التالية "third method was called"

1	public String thirdMethod(String name) {
2	return "third method was called!!";
3	}

هنا الدالة تعود بقيمة من النوع string واسم الدالة thirdMethod ، وتم استخدام الكلمة return متبوعة بالقيمة التي ستعود بها الدالة، ويمكن تنفيذ الدالة من أي نقطة في البرنامج من خلال مناداتها كالتالي:

```
thirdMethod ();
```

لكن الدالة ThirdMethod السابقة تعود بقيمة لذا يجب عند استدعائها أن نخزن القيمة التي تعود منها في متغير وهذا المتغير له نفس نوع الـ return_value_type في الدالة نفسها لذا يتم تعريف متغير من النوع string لكي نخزن فيه نتيجة الدالة ThirdMethod ليصبح استدعاء الدالة صحيحا كالتالي:

```
string result = thirdMethod();
```

٦- دالة لا تعود بقيمة ولا تستقبل أي قيمة "parameter" : يتضح المعنى لها من خلال المثال التالي:

1	public void firstMethod() {
2	Log.v("Method", "FirstMethod was called!!");
3	}

حيث أن :

public : access Modifier أي عامة

- الدالة لا تعود بقيمة لذا نكتب void عند الـ ... return-value-type

- الدالة لا تستقبل أي قيمة لذا نترك فراغ بين القوسين فارغة

- تؤدي عند استدعائها تؤدي وظيفة محددة وهي طباعة الرسالة ! FirstMethod was called !

- بمجرد ذكر اسم الدالة يتم تنفيذها كالتالي:

```
firstMethod();
```

أمثلة حول تعريف دوال جديدة في جافا

مثال :

تعريف دالة إسمها `welcomeMessage` ، نوعها `void` تحتوي على أمر طباعة فقط ، ثم استدعائها في الدالة `main()` حتى يتم تنفيذ أمر الطباعة الموضوع فيها.

1. `public class Main {`
2. تعريف دالة بالاسم `welcomeMessage` عند استدعائها تطبع جملة للترحيب `//`
3. `public static void welcomeMessage() {`
4. `System.out.println("Welcome to Elsaheed.com");`
5. `}`
6. `public static void main(String[] args) {`
7. استدعاء الدالة `welcomeMessage` لطباعة جملة الترحيب الموضوعة فيها `//`
8. `welcomeMessage();`
9. `}`
10. `}`

ناتج التنفيذ:

Welcome to Elsaheed.com

مثال:

تعريف دالة بالاسم `sum` عند استدعائها نعطيها رقمين فترجع ناتج جمع هذين الرقمين ثم استدعائها في الدالة `main()`

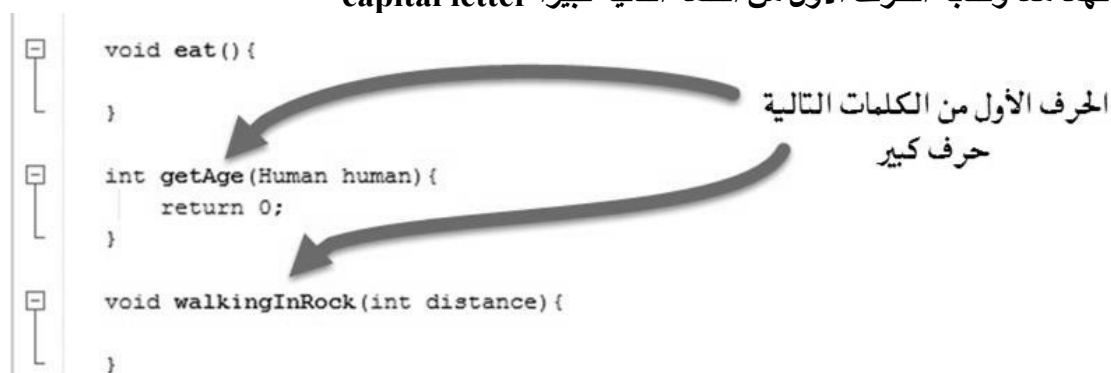
1. `public class Main {`
2. تعريف دالة بالاسم `sum` عند استدعائها نعطيها رقمين فترجع ناتج جمع هذين الرقمين `//`
3. `public static int sum(int a, int b) {`
4. `return a+b;`
5. `}`
6. `public static void main(String[] args) {`
7. استدعاء الدالة `sum` لحساب ناتج جمع الرقمين ١٠ ، ٥ `//`
8. `System.out.println("5 + 10 = " + sum(10, 5));`
9. `}`
10. `}`

ناتج التنفيذ:

5 + 10 = 15

قواعد تسميه الدوال

- عدم استخدام أسماء الكلمات المحجوزة
- أن يبدأ اسم الدالة بحرف صغير small letter ، وعندما يكون اسم الدالة يضم أكثر من كلمة عندئذ يتم إصاقهما معا وكتابة الحرف الأول من الكلمة التالية كبيراً capital letter



تأثير وجود static في الدالة

الكلمة المحجوزة static ذات تأثير مهم على الكود البرمجي وعلى أداء البرنامج وحجمه

فالمفهوم العام لـ static هو أن وجودها بجانب الدالة يعني أن هذه الدالة تتبع للفئة class وليس الكائن Object

فالفئة Class هو الهيكل العام للشيء (الصفات المشتركة بين الكائنات من هذا النوع)

وباستخدام هذه الصفات المشتركة يمكن إنشاء كائنات Objects تتميز عن بعضها البعض في بعض الصفات ولكنها جميعاً تندرج تحت هذه الفئة Class

وعند إنشاء الفئة فإننا ننشئ بعض الدوال methods (تمثل الأفعال التي تستطيع هذه الفئة القيام بها)، وربما نحتاج إلى إنشاء بعض الدوال المساعدة لتنفيذ المهام بالكائن بعد إنشائه

الفصل الثالث : الدوال وإدخال البيانات وعرضها

مثلاً إذا كان الكائن الإنسان مثلما هو موجود بالمثال التالي:

```
1 class Human{
2     private int HumanAge ؛
3         // static method
4     static int convertYearsToDays{()
5         // some logic
6         return 0؛
7     }
8     // non-static mehtod
9     void setAge(int age){
10         HumanAge=age؛
11     }
12 }
```

ملاحظات

-تم إنشاء الدالة convertYearsToDays بهدف تحويل عمر الإنسان من سنين إلى أيام

- وضع الكلمة static أمام الدالة حيث انها من الدوال التي تحتاجها الفئة وليس الكائن، أي بمعنى آخر، هذه الدالة من الممكن استخدامها والاستفادة منها حتى بدون إنشاء الكائن

-لإنشاء الكائن نحتاج إلى تحديد عمر الإنسان، ومالم يُنشئ الكائن فلا فائدة لاستخدام الدالة setAge ، لذلك لم تسبقها الكلمة static

الشكل العام لدالة تستقبل مصفوفة كمعامل ولا ترجع قيمة

Access Modifier return type fun_name (Array_type Array_name[]) { }
--

مثال

```
1 package arrays;
2 public class Test {
3     public static void main(String[] args) {
4         // تعريف مصفوفة
5         int[] numbers = { 10, 20, 30, 40, 50 };
6         // استدعاء الدالة وتمرير المصفوفة لها
7         printArray(numbers);
8         System.out.println();
9         // استدعاء الدالة وإنشاء مصفوفة بداخله
11        printArray(new int[] {23, 43, 54, 34, 90});
11    }
12    // تعريف دالة تطبع محتوى أي مصفوفة نمررها لها كـ argument
13    public static void printArray(int[] array) {
14        for (int i=0; i<array.length; i++) {
15            System.out.print(array[i] + " ");
16        }
17    }
18 }
```

ناتج التنفيذ

```
10 20 30 40 50
23 43 54 34 90
```

الشكل العام لدالة تستقبل مصفوفة كمعامل وترجع مصفوفة من القيم

Access_Modifier return_type[] fun_name (Array_type Array_name[]) {}
--

مثال

1	public class arrayExample {
2	static int MaxNum_Arr(int Arr[]){ // تعريف دالة تستقبل مصفوفة بيانات
	من النوع الصحيح
3	int max = Arr[0];
4	for (int i=0; i < 3; i++) { // المرور علي كل رقم ومقارنته مع القيمة السابقة
5	if(Arr[i] > max);
6	max = Arr[i]; // تبديل الرقم اذا كانت نتيجة المقارنة صحيحة
7	}
8	return max; // ارجاع القيمة المخزنة وهي قيمة اكبر رقم
9	}
11	public static void main(String[] args) {
11	int array1[] = {10,20,30}; // إنشاء المصفوفة
12	int array2[] = {15,25,35}; // إنشاء المصفوفة ثانية بقيم جديدة
13	// استدعاء الدالة وتمرير المصفوفات اليها
14	System.out.println("the max number in array1 : " +
15	MaxNum_Arr(array1));
16	System.out.println("the max number in array2 : " +
	MaxNum_Arr(array2));
	}

ناتج التنفيذ

the max number in array1 : 30

the max number in array2 : 35

التحميل الزائد للدوال Method Overloading

التحميل الزائد هو كتابة أكثر من دالة وتحمل نفس الاسم في نفس الفئة ولكن تختلف في المعاملات

لتعريف أكثر من دالة بنفس الاسم لابد من أن تختلف في ثلاث أشياء:

- عدد المعاملات Number Of Parameters

- نوع المعاملات Data Type Of Parameters

- ترتيب نوع المعاملات Sequence Of Data Type Of Parameters

دالتين مختلفتين في عدد المعاملات

```
public static int max(int x , int y) {  
    // method body  
}
```

```
public static int max(int x , int y , int z) {  
    // method body  
}
```

دالتين مختلفتين في ترتيب نوع المعاملات

```
public static int max(int x , double y) {  
    // method body  
}
```

```
public static int max(double x , int y) {  
    // method body  
}
```

دالتين مختلفتين في نوع المعاملات

```
public static int max(double x , double y) {  
    // method body  
}
```

```
public static int max(int x , int y) {  
    // method body  
}
```

وتحتوى الجافا على مجموعة كبيرة جداً من الدوال الجاهزة التي يمكن استخدامها مباشرة

أمثلة للدوال الجاهزة

دوال الطباعة

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

-الدالة () Print :

دالة تستخدم لعرض أي شيء نضعه داخلها سواء نص أو رقم أو قيمة متغير (طباعة المخرجات بدون الانتقال لسطر جديد)

1	System.out.print("Hello ");
2	System.out.print("World");

-الدالة () Println :

طباعة المخرجات ثم الانتقال لسطر جديد

1	System.out.println("Hello World!");
2	System.out.println("Integer: " + 10 + " Double: " + 3.14 + " Boolean: " + true);

-الدالة () Printf :

لتنسيق الطباعة بسهولة

System.out.printf("pi = %.5f", Math.PI); // => pi = 3.14159

أمثلة على دوال العرض

مثال:

عرض ٣ أشياء باستخدام الدالة print()

```
1 public class Main {
2     public static void main(String[] args) {
3         // عرض نص
4         System.out.print("Welcome to java world");
5         // عرض رقم
6         System.out.print(1000);
7         // تعريف متغير اسمه x ثم عرض قيمته
8         int x = 123;
9         System.out.print(x);
10    }
11 }
```

نتيجة التنفيذ

```
Welcome to java
world1000123
```

كتابة نفس البرنامج السابق لكن باستخدام الدالة println() بدلاً من الدالة print()

```
1 public class Main {
2     public static void main(String[] args) {
3         // عرض نص
4         System.out.println("Welcome to java world");
5         // عرض رقم
6         System.out.println(1000);
7         // تعريف متغير اسمه x ثم عرض قيمته
8         int x = 123;
9         System.out.println(x);
10    }
11 }
```

نتيجة التنفيذ

```
Welcome to java world
1000
123
```

مثال :

دمج النص الموجود في الدالة (printf) مع رقم (يجب وضع فاصلة قبل كل متغير أو قيمة سيتم استبدالها)

1	public class Main {
2	public static void main(String[] args) {
3	// تعريف متغير اسمه x قيمته 10
4	int x = 10;
5	// وضع قيمة المتغير x مكان الـ %d ثم عرض كامل محتوى دالة الطباعة
6	System.out.printf("The value of x is: %d", x);
7	}
8	}
9	

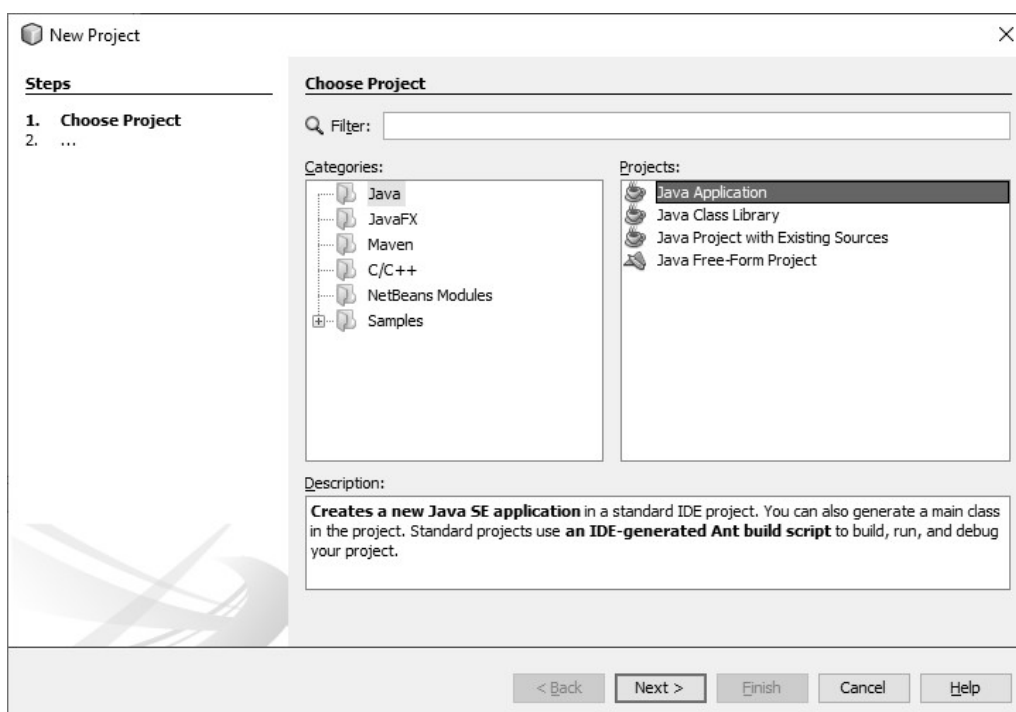
نتيجة التنفيذ

```
y = 10.550000 and x = 10
```

Displaying program output

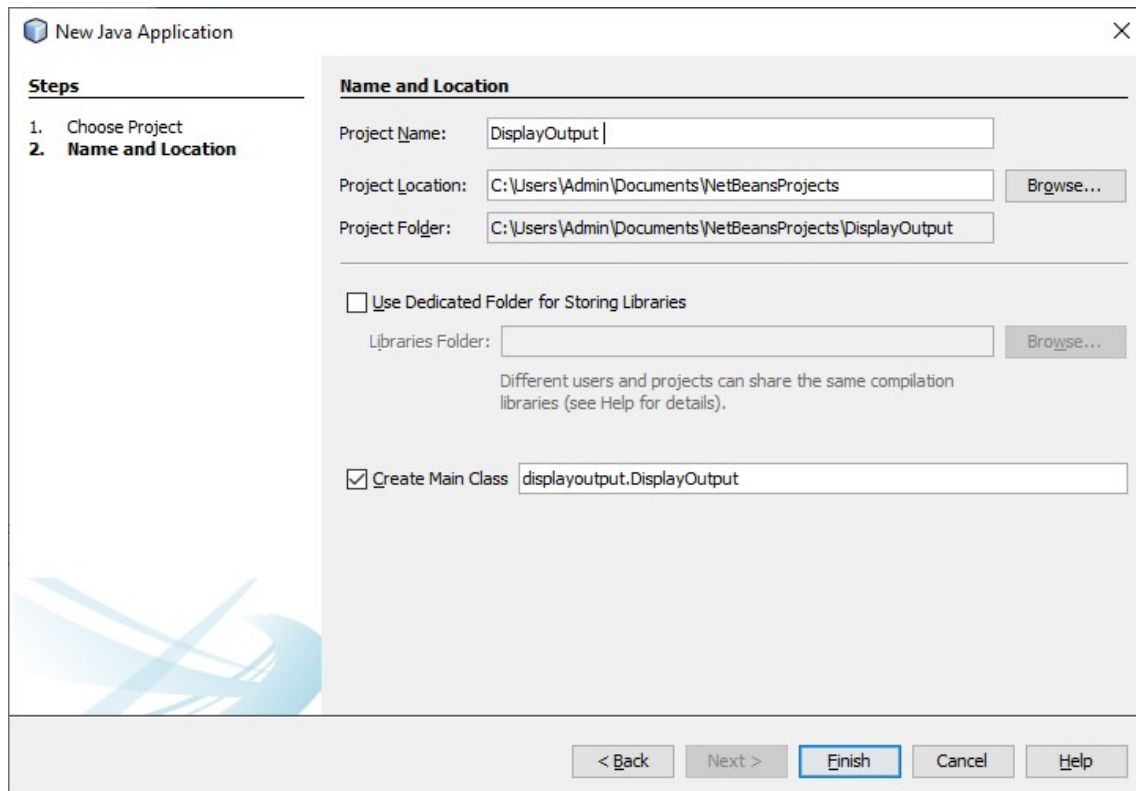
```
System.out.print();  
System.out.println();
```

مثال
نفتح برنامج NetBeans IDE ونفتح قائمة File ونختار new project ثم نختار java/java Application

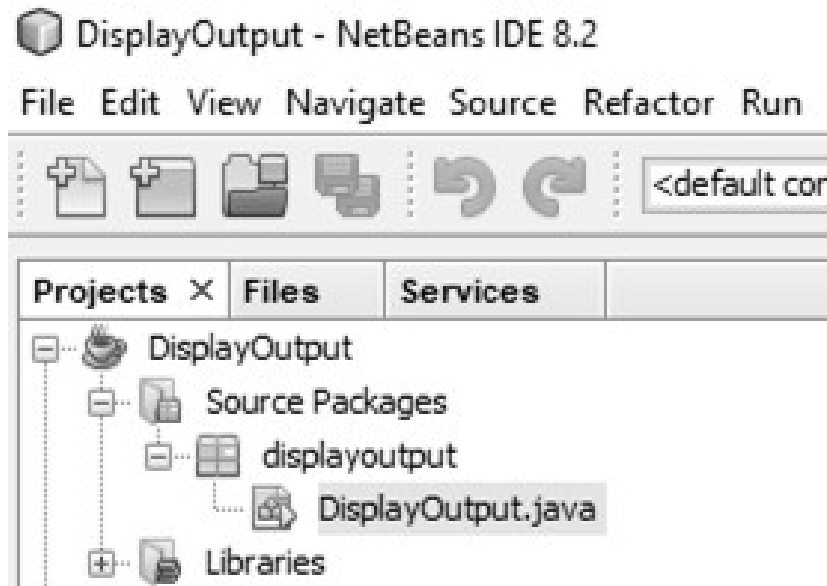


نضغط next عندئذ يظهر مربع حوارى يطلب اسم للمشروع ويمكن تسميته DisplayOutput ، مع ملاحظة تنشيط الخيار Create Main Class

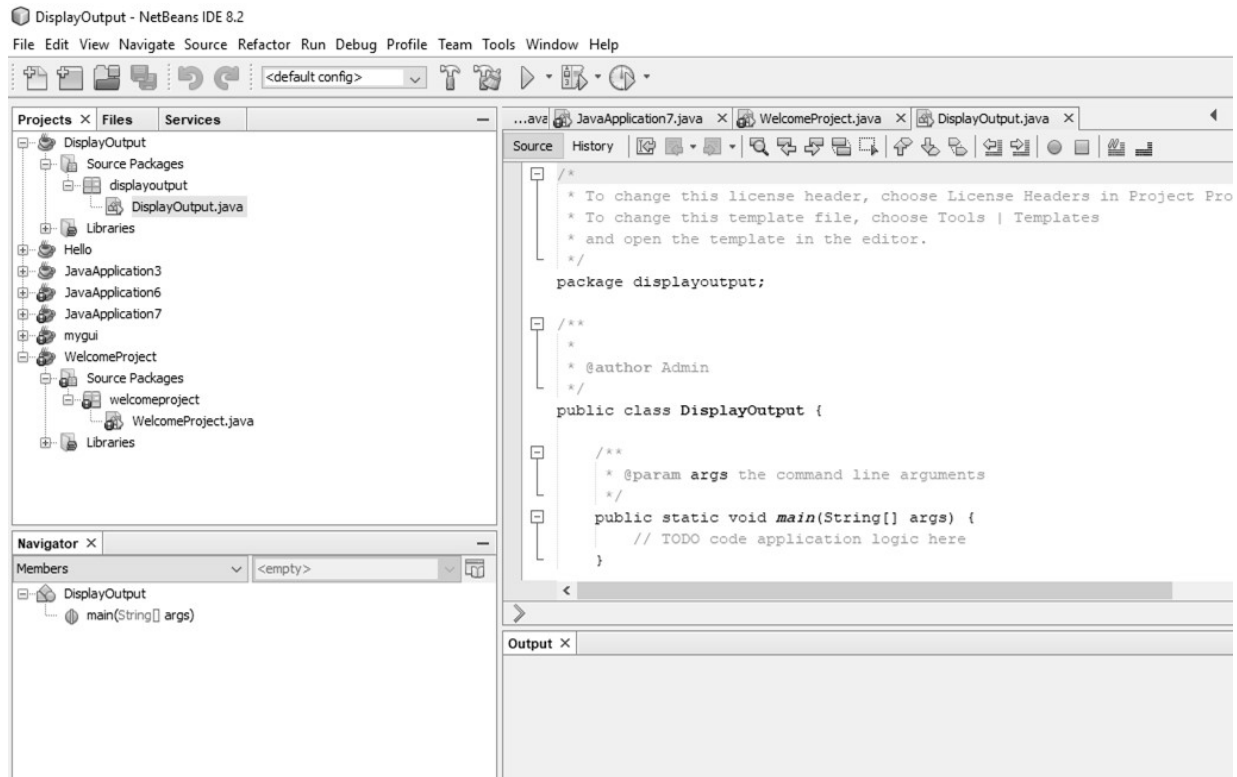
الفصل الثالث : الدوال وإدخال البيانات وعرضها



نضغط الزر Finish



الفصل الثالث : الدوال وإدخال البيانات وعرضها



كتابه Mehod الطباعة التالى:

`System.out.print();`

```
public static void main(String[] args) {
    // TODO code application logic here
    System.out.print("Hello world");
    System.out.print("Iam prof.elsaeed");
}
```

وعند التنفيذ تظهر المخرجات كما يلى:



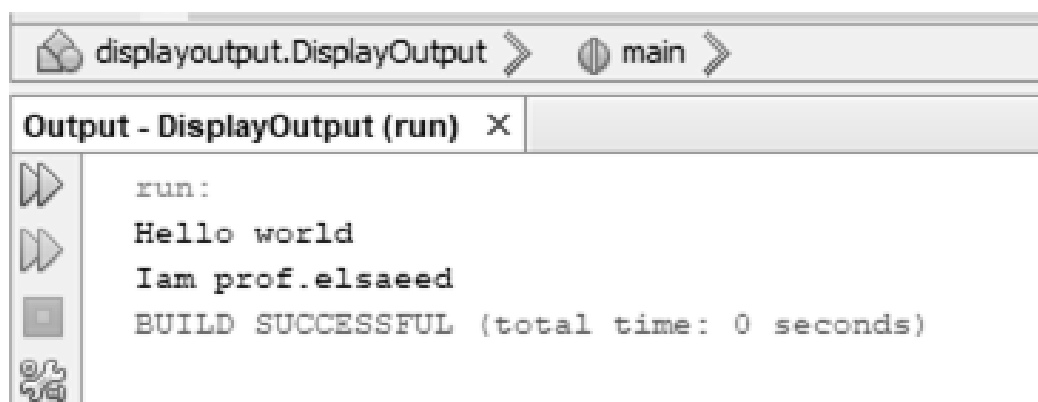
الفصل الثالث : الدوال وإدخال البيانات وعرضها

وبالتالي فإن الميثود `print` يجعل مؤشر الطباعة `Cursor` يقف عند آخر حرف تم طباعته مسبقاً، وبالتالي عند وجود جملة طباعة تالية عندئذ يتم الطباعة عند النقطة التي يقف عندها مؤشر الطباعة

وعند الرغبة في طباعة كل جملة على سطر مستقل يتم استبدال جملة `print` بالجملة `println` والتي تقوم بالطباعة ثم نقل المؤشر للسطر التالي لتبدئ عملية طباعة جديدة

```
System.out.println("Hello world");  
System.out.println("Iam prof.elsaeed");
```

وعند التنفيذ تظهر المخرجات كما يلي:



الاحرف التي تستخدم مع النصوص

Escape Sequence

الرمز \t (Horizontal tab)

يضيف عدة مسافات في مكان وضعها

```
1 public class Main {
2     public static void main(String[] args) {
3         System.out.println( "1\t 2\t 3\t 4" );
4     }
5 }
```

وعند التنفيذ تظهر المخرجات كما يلي:



الرمز \b

يزيل الحرف الموجود قبلها

```
1 public class Main {
2     public static void main(String[] args) {
3         System.out.println("abc\b");
4     }
5 }
```

وعند التنفيذ تظهر المخرجات كما يلي:



الفصل الثالث : الدوال وإدخال البيانات وعرضها

الرمز \n (NewLine)

يجعل المحتوى الذي يأتي بعدها ينزل على سطر جديد

```
1 public class Main {  
2     public static void main(String[] args) {  
3         System.out.println("My\nName\nIs\nelsaeed");  
4     }  
5 }
```

وعند التنفيذ تظهر المخرجات كما يلي:

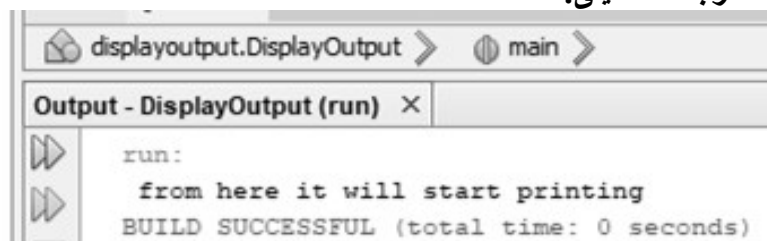


الرمز \r (Carriage return)

يجعل الكود يبدأ في التنفيذ من عندها

```
1 public class Main {  
2     public static void main(String[] args) {  
3         System.out.println("This text will be removed \r from here it will  
start printing");  
4     }  
5 }
```

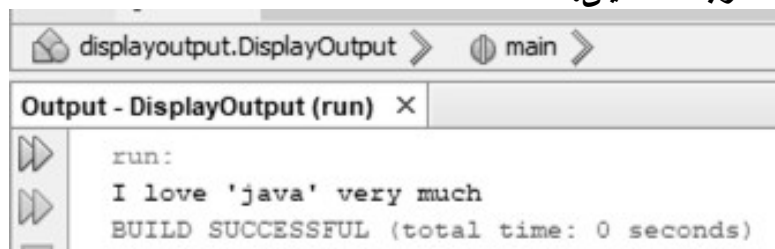
وعند التنفيذ تظهر المخرجات كما يلي:



الرمز \' : لإضافة الرمز ' في مكان وضعها

1	<code>public class Main {</code>
2	<code>public static void main(String[] args) {</code>
3	<code>System.out.println("I love \'java\' very much");</code>
4	<code>}</code>
5	<code>}</code>

وعند التنفيذ تظهر المخرجات كما يلي:



الرمز \" (Double quote) : لإضافة الرمز " في مكان وضعها

إدخال البيانات في الجافا (قبول بيانات من المستخدم)

Accepting Input from User

عندما نريد انشاء برنامج يتفاعل مع المستخدم بحيث عندما يقوم بتشغيله يطلب من المستخدم إدخال بيانات وبعد إدخالها يقوم البرنامج بمعالجتها وفعل شيء معين بها.

فإننا نحتاج لكلاس جاهز في الجافا يسمى Scanner يستخدم بشكل عام لجعل البرنامج يستقبل بيانات من المستخدم بالإضافة إلى إمكانية تحويل نوع هذه البيانات والتعديل عليها وهو يوجد داخل الحزمة Package المسماة Java.util ، والكلاس Scanner كبير جداً حيث ويتكون من 9 constructors ، و أكثر من 50 دالة

أنواع البيانات التي يمكن استقبالها باستخدام الكلاس Scanner

- أعداد صحيحة: أي أعداد لا تتضمن علامة عشرية
- أعداد تقبل العلامة العشرية
- حرف واحد (حرف- رقم- رمز)
- كلمة واحدة
- سطر كامل: أي أكثر من كلمة
- صح أو خطأ: أي إجبار المستخدم على إدخال كلمة واحدة إما true ، أو false

ملحوظة هامة : كل نوع من البيانات السابقة تريد استقباله في البرنامج له دالة خاصة عند التعامل معه

مصطلحات تقنية

Delimiter : تعني الـ Pattern المستخدمة في تحديد شكل المسافات الفارغة White Spaces حيث يمكن التحكم في جعل مترجم جافا يعتبر حرف أو كلمة أو جملة ما تبدو وكأنها مسافة فارغة White Space

Console Application : تعني البرنامج الذي يعمل بدون واجهة مستخدم أي بدون GUI

طريقة جعل البرنامج يستقبل بيانات من المستخدم في الجافا

- في لغة جافا لكي تجعل البرنامج يستقبل معلومات من المستخدم يتم إتباع ثلاث خطوات أساسية:
- استدعاء الكلاس Scanner الذي يتضمن دوال إدخال المعلومات من المستخدمين.
- إنشاء كائن من هذا الكلاس حيث لا يمكن استخدام دوال الكلاس Scanner إلا من خلال الكائن الذي يشير إليه.
- استدعاء إحدى دوال إدخال المعلومات من هذا الكائن.

إذاً سنحتاج إلى الأوامر التالية في كل برنامج (يعتبر Console Application) تطلب فيه من المستخدم إدخال بيانات من لوحة المفاتيح:

١ - استدعاء الكلاس Scanner حتى نستطيع إنشاء كائن منه

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

٢ - إنشاء كائن من الكلاس Scanner مع اعطائه اسم ما وليكن مثلاً input
`Scanner input = new Scanner(System.in);`

٣ - استدعاء إحدى دوال إدخال البيانات من خلال كائن Scanner الذي تم إنشائه سابقاً ، ثم استدعاء الدالة `nextInt()` من هذا الكائن والتي تعني أنه عند تشغيل البرنامج وعندما يأتي لتنفيذ هذا الكود سيقوم بانتظار المستخدم لإدخال عدد صحيح من لوحة المفاتيح ، بعدها سيتم تخزين ما أدخله المستخدم في المتغير `a`

```
int a = input.nextInt();
```

ملحوظة

دائماً نضع دوال استقبال البيانات بداخل الجملة `try` ، وإغلاق كائن Scanner بواسطة الدالة `close()` مباشرة عند الانتهاء منه لضمان عدم حدوث أي أخطاء من شأنها توقف البرنامج

الخلاصة

حتى يمكن استخدام الكلاس Scanner يجب ما يلي:
الخطوة الأولى: استخدام جملة import وهي المسؤولة عن استدعاء تلك الحزمة (كأنه يتم القول للـ Compiler اننا في حاجة لتلك الحزمة package لذلك قم باستدعائها وضمها للبرنامج الحالي) ، لتصبح الصيغة كما يلي :

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

الخطوة الثانية : نحدد مصدر البيانات المدخلة (يمكن للبرنامج الحصول على المدخلات من اكثر من مصدر مثل لوحة المفاتيح key board ، أو من ملف file بالجهاز ، نكتب السطر التالي :

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

الخطوة الثالثة : تخزين المدخلات التي تم الحصول عليها من المستخدم في الذاكرة Memory من خلال المتغيرات Variables

مثال:

إنشاء برنامج يطلب من المستخدم إدخال: الاسم - العمر-الوظيفة

- تعريف ٣ متغيرات (name-age-job) لتخزين القيم التي سيطلب البرنامج من المستخدم إدخالها عند تشغيله

- استخدام الدالة nextLine() لاستقبال اسم المستخدم ووظيفته

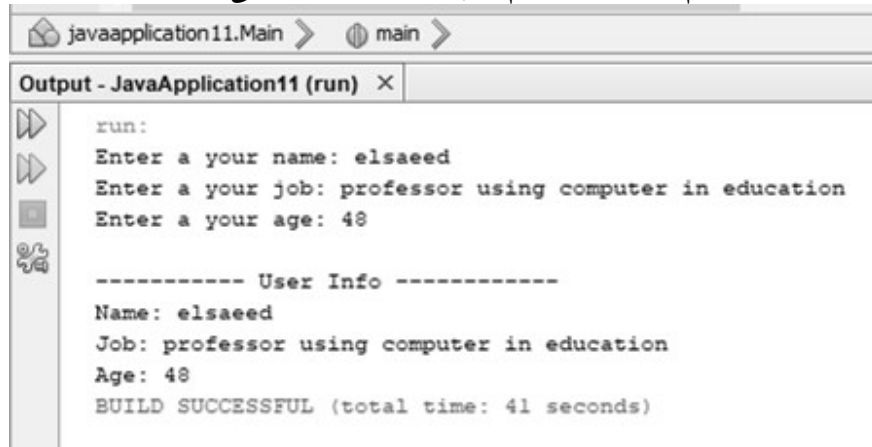
-استخدام الدالة nextInt() لاستقبال عمر المستخدم

الفصل الثالث : الدوال وإدخال البيانات وعرضها

Main.java

```
1 import java.util.Scanner;    // استدعاء الكلاس Scanner
2 public class Main {
3     public static void main(String[] args) {
4         Scanner input = new Scanner(System.in);    // إنشاء كائن من الكلاس Scanner وإسمه input
5         String name;    // استخدام هذا المتغير لحفظ الاسم الذي سيدخله المستخدم
6         String job;    // استخدام هذا المتغير لحفظ الوظيفة التي سيدخلها المستخدم
7         int age;    // استخدام هذا المتغير لحفظ العمر الذي سيدخله المستخدم
8         try {
9             System.out.print("Enter a your name: ");    // نطلب من المستخدم إدخال اسمه
10            name = input.nextLine();    // name سيُدخله المستخدم و تخزينه في المتغير name
11            System.out.print("Enter a your job: ");    // نطلب من المستخدم إدخال الوظيفة
12            job = input.nextLine();    // job سيُدخله المستخدم و تخزينه في المتغير job
13            System.out.print("Enter a your age: ");    // نطلب من المستخدم إدخال عمره
14            age = input.nextInt();    // age سيُدخله المستخدم و تخزينه في المتغير age
15            // استقبال العدد الصحيح الذي سيدخله المستخدم و تخزينه في المتغير age
16            // في النهاية يتم عرض جميع المعلومات التي أدخلها المستخدم إذا لم يحدث أي خطأ أثناء إدخال البيانات
17            System.out.println("\n----- User Info -----");
18            System.out.println("Name: " +name);
19            System.out.println("Job: " +job);
20            System.out.println("Age: " +age);
21        } catch (Exception e) {    // اكتشاف أي خطأ قد يحدث وعرضه
22            System.out.print(e.toString());
23        }
24        finally {    // إذا تم أو لم يتم إدخال البيانات يتم إغلاق الـ Scanner
25            input.close();
26        }
27    }
28 }
```

وعند التنفيذ يطلب من المستخدم ادخال بياناته ثم تظهر المخرجات كما يلي:



```
run:
Enter a your name: elsaeed
Enter a your job: professor using computer in education
Enter a your age: 48

----- User Info -----
Name: elsaeed
Job: professor using computer in education
Age: 48
BUILD SUCCESSFUL (total time: 41 seconds)
```

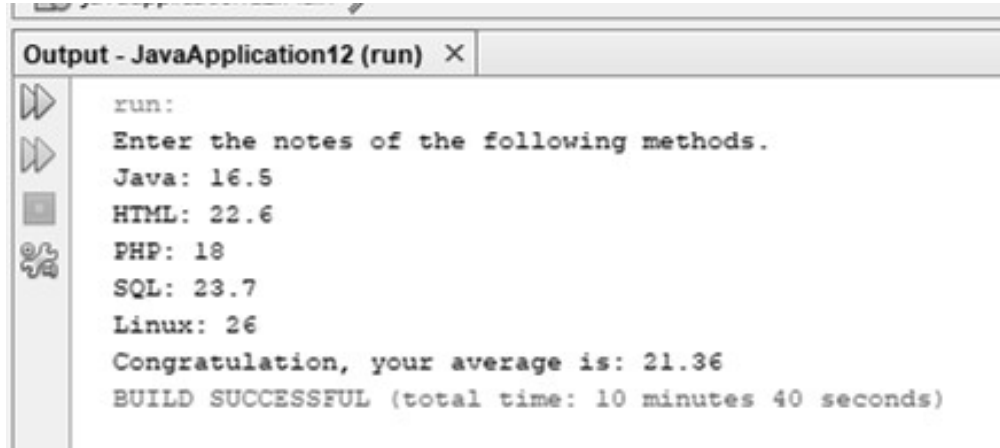
الفصل الثالث : الدوال وإدخال البيانات وعرضها

- مثال : إنشاء برنامج يطلب من المستخدم إدخال درجاته وتخزينها في مصفوفة
- إنشاء مصفوفة اسمها notes نوعها float تتكون من ٥ عناصر لتخزين الدرجات التي يدخلها الطالب
- استخدام الدالة nextFloat() لاستقبال الأرقام التي يدخلها الطالب وقد تتضمن الدرجات علامة عشرية
- بعد إدخال جميع الدرجات التي يطلبها البرنامج من الطالب سيطبع البرنامج متوسط الدرجات

```
1 package javaapplication12;
2 import java.util.Scanner; // استدعاء الكلاس Scanner
3 public class Main {
4     public static void main(String[] args) {
5         Scanner input = new Scanner(System.in); // إنشاء كائن من الكلاس Scanner اسمه input
6         float[] notes = new float[5]; // استخدام هذه المصفوفة لتخزين جميع درجات الطالب
7         float avg; // استخدام هذا المتغير لتخزين المتوسط الحسابي
8         System.out.println("Enter the notes of the following methods.");
9         try {
10             System.out.print("Java: "); // نطلب من المستخدم إدخال درجة المادة الأولى
11             notes[0] = input.nextFloat(); // تخزين الدرجة التي أدخلها في العنصر الأول في المصفوفة
12
13             System.out.print("HTML: "); // نطلب من المستخدم إدخال درجة المادة الثانية
14             notes[1] = input.nextFloat(); // تخزين الدرجة التي أدخلها في العنصر الثاني في المصفوفة
15
16             System.out.print("PHP: "); // نطلب من المستخدم إدخال درجة المادة الثالثة
17             notes[2] = input.nextFloat(); // تخزين الدرجة التي أدخلها في العنصر الثالث في المصفوفة
18
19             System.out.print("SQL: "); // نطلب من المستخدم إدخال درجة المادة الرابعة
20             notes[3] = input.nextFloat(); // تخزين الدرجة التي أدخلها في العنصر الرابع في المصفوفة
21
22             System.out.print("Linux: "); // نطلب من المستخدم إدخال درجة المادة الخامسة
23             notes[4] = input.nextFloat(); // تخزين الدرجة التي أدخلها في العنصر الخامس في المصفوفة
24             avg = (notes[0] + notes[1] + notes[2] + notes[3] + notes[4]) / 5; // تخزين المتوسط الحسابي في المتغير avg
25             if (avg >= 10) { // إذا كان المتوسط أكبر أو يساوي ١٠ سيطبع الجملة التالية
26                 System.out.println("Congratulation, your average is: " + avg);
27             }
28             else { // إذا كان المعدل أقل من ١٠ سيطبع الجملة التالية
29                 System.out.println("Sorry, you fail! your average is: " + avg);
30             }
31         } catch (Exception e) { // اكتشاف أي خطأ قد يحدث وعرضه
32             System.out.print(e.toString());
33         } finally { // إذا تم أو لم يتم إدخال البيانات سيتم إغلاق الـ Scanner
34             input.close();
35         }
36     }
37 }
```

الفصل الثالث : الدوال وإدخال البيانات وعرضها

وعند التنفيذ يطلب من المستخدم ادخال درجاته ثم تظهر النتائج كما يلي:



```
Output - JavaApplication12 (run) X
run:
Enter the notes of the following methods.
Java: 16.5
HTML: 22.6
PHP: 18
SQL: 23.7
Linux: 26
Congratulation, your average is: 21.36
BUILD SUCCESSFUL (total time: 10 minutes 40 seconds)
```