الفصل الأول

في هذا الفصل:

سنتعلم ما هو برنامج جافا **Learn What is Java Program**

نكتشف العناصر الأساسية لبرنامج جافا **Basic Elements of Java Program**

نتعلم خطوات تنفيذ برنامج جافا **Learn the steps to implement java program**

طرق الإدخال والإخراج بلغة جافا **Input and Output in java**

أولا : ما هو برنامج جافا What is Java Program

هناك نوعان من برامج جافا وهي:

- تطبيقات **Applications**
- وبريمجات **Applets**

والنوعان متشابهان من ناحية البرمجة. فالبرامج التطبيقية هي برامج قائمة بذاتها ويمكن أن تنفذ على حاسوبك.

أما البريمج applet فينفذ من متصفح الوب web browser أو برنامج لعرض البريمجات حيث تجعل الوب تتفاعل وتستجيب فتكون ممتعة للإستخدام. من أشهر المتصفحات هي Netscape و Internet Explorer حيث يمكن تنفيذ البريمجات على أي منهما.

ستكون كل برامجنا من النوع الأول أي التطبيقات Applications والذي هو عبارة عن تجمع من الفئات، أو على الأقل فئة واحدة تسمى الفئة الرئيسية تحوي على الأقل دالة واحدة هي الدالة الرئيسية main والفئة عبارة عن تجمع من الدوال والأعضاء البيانية.

تعتبر لغة جافا من اللغات الأكثر شعبية لعدة سنوات.

- فهي لغة كيانية نقية pure object oriented language .
- قواعد كتابتها مشابهة لقواعد لغة c / c++

- لايتوفر بلغة جافا بعض خصائص كالمؤشرات.
- أي برنامج لجافا لابد ان يكتب بصيغة الفئة (class).
- تستخدم لغة جافا بكل انواع التطبيقات، كتطبيقات الموبايل (اندرويد)، تطبيقات سطح المكتب، تطبيقات الوب، تطبيقات client server، و تطبيقات enterprise وغيرها كثير.

معالجة برنامج جافا Processing a java Program

لتنفيذ برنامج تطبيقي Application Program مكتوب بلغة جافا يجب إتباع الخطوات التالية :

1- يجب استخدام **محرر** مثل Notepad أو أي **محرر لبرامج جافا**، لخلق برنامج جافا يسمى البرنامج المصدر source program،(تم استخدام NetBeans) حيث يخزن البرنامج في ملف نصي يسمى باسم الفئة المعرفة وبالامتداد (.java)، أي **ClassName.java** فالفئة الرئيسية˚ والتي تكون الدالة الرئيسية main داخلها والتي يجب أن تكون عامة أي publicتوضع في ملف باسمها مع الامتداد (.java) فمثلا إذا كانت الفئة الرئيسية اسمهاwelcome فإن البرنامج يجب أن يخزن بملف اسمه **welcome.java** وإلا فلن ينفذ البرنامج وستحصل على رسالة خطأ.

2- **البرنامج المصدر** هو برنامج مكتوب بلغة عالية المستوى (**جافا**) ويجب ان يتطابق مع قواعد كتابة البرنامج بتلك اللغة. يعمل المترجم على ترجمة برنامج جافا إلى برنامج مكتوب بلغة وسطية اسمها **bytecode**. بعد أن يفحص المترجم البرنامج المصدر من أي أخطاء بقواعد اللغة **syntax error** فإذا لم يجد أي خطأ فيه يترجمه إلى برنامج جديد مكتوب بـلغة **bytecode** حيث يخزن البرنامج الجديد تحت نفس الاسم السابق وهو اسم الفئة الرئيسية مع الامتداد .**class** فمثلا بعد ترجمة التطبيق **welcome.java** تتكون نسخة جديدة بملف جديد اسمه **welcome.class**. طبعا لغة **bytecode** لاعلاقة لها بلغة الماكنة الخاصة بحاسوبك machine independent code.

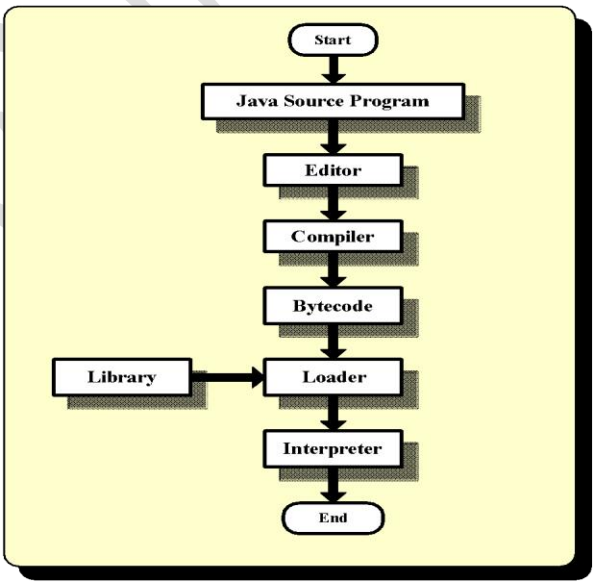
3- لتنفيذ برنامج جافا تطبيقي، يجب تحميل البرنامج الذي امتداده .**class** في ذاكرة الحاسوب الرئيسية. البرامج التي تكتب بلغة جافا يتم تطويرها باستخدام عدة تطوير البرمجيات (SDK). هذه العدة تحوي على العديد من البرامج المفيدة لخلق برنامجك، كالكود الضروري لعرض نتائج البرنامج والعديد من الدوال الرياضية الجاهزة لتسهيل عمل المبرمج نوعا ما، لهذا يفضل استخدام هذه الدوال المتوفرة والجاهزة بدلا من قيامك بكتابتها بنفسك من الصفر، علاوة على ذلك يمكنك تطوير مكتبات البرامج الخاصة بك (تسمى الحزم **packages** بلغة جافا) سنتطرق لها في المحاضرات القادمة. بصورة عامة يقسم برنامج جافا إلى أقسام مختلفة تسمى فئات **classes**

ونموذجيا توضع كل فئة في ملف منفصل ويتم ترجمته على انفراد، ولهذا لتنفيذ برنامج جافا بنجاح يجب ربط **bytecode** كل الفئات المستخدمة في البرنامج، والبرنامج الذي يقوم بهذه المهمة بلغة جافا يسمى المحمل.**loader**

4- الخطوة التالية هي تنفيذ برنامج جافا. فبالإضافة إلى ربط **bytecode** الفئات المختلفة يقوم البرنامج المحمل **loader** بتحميل **bytecode** البرنامج في الذاكرة الرئيسية. بعد تحميل الفئات في الذاكرة الرئيسية يقوم محقق لغة البايت كود **verifier bytecode** بالتحقق من إن البايت كود **bytecode** للفئات لا تفسد شروط **أمنية جافا**. وأخيرا يعمل برنامج يسمى المفسر **interpreter** لتفسير كل تعليمة من تعليمات البايت كود bytecode إلى لغة الماكنة الخاصة بحاسوبك، ثم ينفذها ولا يحتفظ بنسخة مفسرة من البرنامج بامتداد معين بل يفسر التعليمة وينفذها ولا يحتفظ بها فعند الرغبة باعادة تنفيذ نفس البرنامج مرة أخرى ببيانات جديدة تبدأ عملية التنفيذ من خطوة إعادة تحميل البايت كود في الذاكرة الرئيسية أي نسخة البرنامج بالإمتدلد (.class) يفسرها وينفذها عبارة بعد أخرى إلى نهاية البرنامج.

المفسر interpreter : هو برنامج يقرأ ويفسر كل تعليمة من البايت كود bytecode إلى لغة الماكينة الخاصة بحاسوبك ثم ينفذها. حيث يقوم المفسر بتفسير وتنفيذ تعليمة واحدة فقط كل مرة.

يبين الشكل التالي خطوات تنفيذ برنامج جافا.

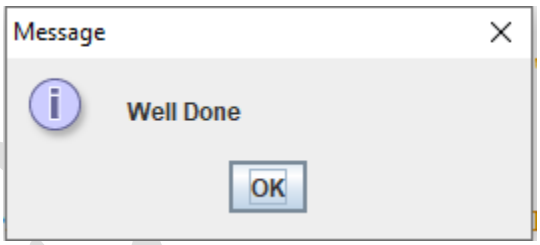


ثانيا : العناصر الأساسية لبرنامج جافا **Basic Elements of Java Program**

لمعرفة المكونات الأساسية في أي برنامج جافا سنعطي مثالا لبرنامج تطبيقي application program بسيط ثم نتفحصه. فيما يلي برنامج جافا بسيط يحتوي على ثلاث عبارات طباعة وصندوق رسائل مفرد:

```
import javax.swing.JOptionPane;
public class AsimpleJavaProgram {
public static void main(String [ ] a)
 {  System.out.println("My first Java program.");
   System.out.println("The sum of 2 and 3 = "  + 5);
   System.out.println("7 + 8 = " + (7 + 8));
   JOptionPane.showMessageDialog(null," Well Done" );
}
}
```

```
run:
My first Java program.
The sum of 2 and 3 = 5
7 + 8 = 15
```



الجملة الأولى ;**import javax.swing.JOptionPane** وهي ضرورية في أعلى كل برنامج يستخدم الواجهات الرسومية كصناديق الحوار، حيث الفئة JOptionPane الموجودة في الحزمة javax.swing والتي سنستخدم بعض دوالها التي تعطينا المركبات الرسومية التي نحتاجها.

يبدأ برنامج جافا بفئة **class** لأن لغة جافا هي لغة كيانيه نقية **pure object oriented language** فلهذا لا يمكن كتابة برنامج جافا بدون تعريف فئة، حتى أصغر برنامج لجافا لابد أن يتكون من فئة واحدة على الأقل (حيث عادة يتكون من مجموعة من الفئات) وهي الفئة الرئيسية كما في برنامجنا السابق، والتي تحوي على الدالة الرئيسية. تفصيل البرنامج كما يلي :

هذا البرنامج عبارة عن فئة واحدة تم ذكر اسمها **AsimpleJavaProgram** بعد ذكر public class في الجملة public class AsimpleJavaProgram حيث لا بد أن تكون الفئة الرئيسية public. ويفضل أن يبدأ اسم أي فئة بلغة جافا بحرف كبير كتقليد، ثم القوس المعقوف { الذي يحدد بداية الفئة.

السطر public static void main(String [ ] a) :

هو رأس الدالة الرئيسية main في البرنامج حيث هذا هو الشكل العام لأي دالة رئيسية main والتي تبدأ بقوس معقوف مفتوح { وتنتهي بقوس معقوف أيضا } ليغلق القوس الأول. البرنامج البسيط السابق يتكون من فئة رئيسية فقط وعادة أي برنامج جافا لابد أن يكون له فئة رئيسية تحتوي على الدالة الرئيسية main. وفي البرنامج السابق هذه الدالة تحتوي على العبارات :

```
System.out.println("My first Java program.");

System.out.println("The sum of 2 and 3 = "  + 5);

System.out.println("7 + 8 = " + (7 + 8));
```

وهذه العبارات هي مثال على عبارات الأخراج بلغة جافا. فهي تسبب حساب ماهو محصور بين قوسين وتعرضه على الشاشة. وكما هو موجود في معظم اللغات فإن مايتم حصره بين علامات التنصيص المزدوجة العليا يعتبر سلسلة string ويظهر في الخرج تماما كما هو دون أي تغيير فالعبارة الأولى:

```
System.out.println("My first Java program.");
```

ستسبب إخراج الجملة : My first Java program

لنتأمل العبارة :

```
System.out.println("The sum of 2 and 3 = "  + 5);
```

ستسبب إخراج The sum of 2 and 3 = والمحصورة بين علامات تنصيص مزدوجة كما هي تماما أما علامة الجمع + التي تليها تفيد بدمج رقم 5 الذي يلي هذه السلسلة معها وجعلهما سلسلة واحدة فيكون الخرج النهائي:

The sum of 2 and 3 = 5

أما العبارة

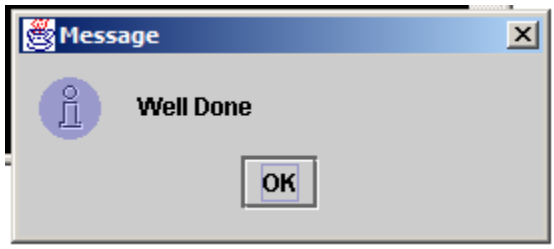
```
System.out.println("7 + 8 = " + (7 + 8));
```

فتسبب إخراج النص المحصور بين علامات تنصيص مزدوجة "7 + 8 =" ثم يتم إيجاد قيمة (7 + 8) وتدمج هذه النتيجة والتي هي 15 مع السلسلة السابقة فيكون خرج هذا السطر: 7 + 8 = 15

وأخيرا العبارة :

```
JOptionPane.showMessageDialog(null," Well Done" );
```

فهي التي تسبب ظهور صندوق الرسائل كما في الشكل التالي وقد استخدم هنا لإبقاء نافذة المخرجات على الشاشة لحين كبس الزر ok وسنأتي على شرح ذلك بالتفصيل في فصل الواجهات الرسومية GUI:



وأخيرا الأقواس{{ في آخر البرنامج أحدهما لإغلاق الدالة الرئيسية main والآخر لإغلاق الفئة الرئيسية AsimpleJavaProgram وبهذا نكون قد أغلقنا كل الأقواس التي تم فتحها وأنهينا البرنامج.

مفهوم المتغيرات

متغير: تعني variable في اللغة الإنجليزية، أنها فقط أماكن يتم حجزها في الذاكرة لتخزين بيانات أثناء تشغيل البرنامج. النوع الذي نعطيه للمتغير يجعل نظام التشغيل يحدد نوع البيانات الذي يمكن تخزينه في المساحة المحجوزة لهذا المتغير في الذاكرة.

البيانات في جافا نوعين:

Primitive Data Types

Reference/Object Data Types

النوع الأول :هناك ثمانية أنواع بدائية في جافا و هي:

byte – short – int – long – float – double – boolean – char.

byte : هذا النوع يمثل عدد صحيح يتألف من bit.8 أقل قيمة يمكن تخزينها فيه هي -2^{7} و هذا يساوي -128. أكثر قيمة يمكن تخزينها فيه هي 2^{7-1} و هذا يساوي +127. إذا لم نضع أي قيمة، توضع القيمة 0 كقيمة إفتراضية. النوع byte يستخدم لتخزين عدد صغير الحجم لا يحتوي على فاصلة عشرية، أي لتخزين عدد صحيح. مثال byte a = 123; و byte b = -70;

short: هذا النوع يمثل عدد صحيح يتألف من bit.16 أقل قيمة يمكن تخزينها فيه هي -2^{15} و هذا يساوي −32,768. أكثر قيمة يمكن تخزينها فيه هي2^{15-1} و هذا يساوي +32,767. إذا لم نضع أي قيمة، توضع القيمة 0 كقيمة إفتراضية. النوع short يستخدم لتخزين عدد متوسط الحجم لا يحتوي على فاصلة عشرية، أي لتخزين عدد صحيح. مثال short a = 12345; و short b = −7000;

int : هذا النوع يمثل عدد صحيح يتألف من bit.32 أقل قيمة يمكن تخزينها فيه هي -2^{31} و هذا يساوي −2,147,483,647. أكثر قيمة يمكن تخزينها فيه هي 2^{31-1} و هذا يساوي +2,147,483,646. إذا لم نضع أي قيمة، توضع القيمة 0 كقيمة إفتراضية. النوع int يستخدم لتخزين عدد كبير لا يحتوي على فاصلة عشرية، أي لتخزين عدد صحيح. مثال int a = 1234567; و int b = −700000 ؛

long : هذا النوع يمثل عدد صحيح يتألف من bit.64 إذا لم نضع أي قيمة، توضع القيمة 0 Lكقيمة إفتراضية. النوع long يستخدم لتخزين عدد كبير جداً لا يحتوي على فاصلة عشرية، أي لتخزين عدد صحيح حجمه كبير جداً.

float :هذا النوع يمثل عدد بفاصلة عشرية يتألف من bit.32 إذا لم نضع أي قيمة، توضع القيمة 0.0 f كقيمة إفتراضية. النوع float يستخدم لتخزين عدد كبير بفاصلة عشرية. مثال float a = 12.05f;

float b = −8.123f;

double: هذا النوع يمثل عدد بفاصلة عشرية يتألف من 64 bit. إذا لم نضع أي قيمة، توضع القيمة 0.0 d كقيمة إفتراضية. النوع double يستخدم لتخزين عدد كبير جداً بفاصلة عشرية.

Boolean :هذا النوع يمثل معلومة تتألف من bit.1 يستطيع أن يحتوي إما على القيمة true أو على القيمة.false إذا لم نضع أي قيمة، توضع القيمة false كقيمة إفتراضية. النوع boolean يستخدم في الشروط.

char :هذا النوع يمثل معلومة تتألف من bit.16 أقل قيمة يمكن تخزينها فيه هي 0. أكثر قيمة يمكن تخزينها فيه هي 65,535. و يستطيع أن يحتوي على أي حرف أو رمز كقيمة. إذا لم نضع أي قيمة، يوضع اليونيكود '\u0000' كقيمة إفتراضية. و هذا اليونيكود يمثل أصغر قيمة يمكن وضعها في النوع char. النوع char يستخدم لتخزين حرف واحد، و يستخدم في الشروط.

اما النواع الثاني البيانات المرجعية

أي نوع أصله كائن من كلاس يعتبر من البيانات المرجعية.

أي نوع نضع الكلمة new عندما نقوم بتعريفه، يعتبر من البيانات المرجعية.

جميع أنواع المصفوفات في جافا تعتبر من البيانات المرجعية.

إذا لم نضع أي قيمة، توضع القيمة null كقيمة إفتراضية، و التي تعني فارغ.

طرق الإدخال والإخراج بلغة جافا Input and Output in java

عادة ما يتكون أي برنامج من ثلاث أجزاء مهمة، وهي جزء **إدخال البيانات** وجزء **معالجة البيانات** والجزء الأخير **إخراج النتائج**، ولذلك لابد أن نتعلم كيفية إدخال البيانات وإخراجها بلغة جافا نظرا لأهمية هذا الجزء من البرنامج لهذا سنحاول شرحه ولو بصورة مبسطة ونعطي بعض الأمثلة لتسهيل الفهم.

توفر لغة جافا دعم واسع لعمليات الإدخال والإخراج بتجهيز عدد من الفئات الأساسية لعمليات الإدخال والإخراج كالفئة **Scanner**. سنوضح الطرق الثلاث للإدخال والإخراج بلغة جافا.

سنوضح الطرق الثلاث للإدخال والإخراج بلغة جافا.

- **الطريقة الأولى هي الإدخال عن طريق لوحة المفاتيح باستخدام الفئة** : Scanner
  Using Class Scanner for I/O from Keyboard

مثال بسيط يوضح كيفية كتابة برنامج ليقرأ البيانات المدخلة عن طريق لوحة المفاتيح ثم نشرحه لزيادة التوضيح. البرنامج التالي يطلب إدخال ساعات العمل الأسبوعية.

```
package weeklywages;
import java.util.*;
public class WeeklyWages {
   static Scanner console = new Scanner(System.in);
   public static void main(String[] args) {
     double  hours;
     String name;
     System.out.print("Enter the name : ");
     name =console.next();
     System.out.print("Enter the working hours : ");
     hours = console.nextDouble();
     System.out.println();

      }
}
```

لاحظ بأننا عرفنا كائن console من نوع الفئة Scanner في الجملة :

static Scanner console = new Scanner(System.in);

لاستخدامه في عملية الإدخال مع الدالة **console.next()** ؛ لإدخال القيم التي هي من نوع السلسلة **String** كما في الجملة **name = console.next()** .

والدالة () **console.nextDouble** لإدخال القيم التي هي من نوع **double** في الجملة :

hours = console.nextDouble();

وهناك دوال خاصة للأنواع الأخرى من البيانات كالنوع الصحيح **nextInt ( )** والقصير **nextShort ()** وهكذا. وعند تنفيذ البرنامج سيظهر السطر التالي حيث يطلب إدخال عدد ساعات العمل الأسبوعية:

Enter the name:

وينتظر إدخال الأسم والضغط على مفتاح الرجوع Enter فيظهر السطر:

Enter the working hours :

- **الطريقة الثانية باستخدام صناديق الحوار للإدخال والإخراج**

Using Dialog Boxes for I/O

والتي تسمح للمبرمج استخدام بعض المركبات الرسومية بحيث تجعل عملية الإدخال والإخراج أكثر كفاءة والبرنامج أكثر تفاعليا مع المستخدم وأكثر جاذبية.

توجد الفئة **JOptionPane** في الحزمة **javax.swing**، وتحتوي على دالتين هما:

showInputDialog و **showMessageDialog**. الدالة **showInputDialog** تسمح للمستخدم إدخال سلسلة من لوحة المفاتيح وصيغتها كما يلي:

String Str;

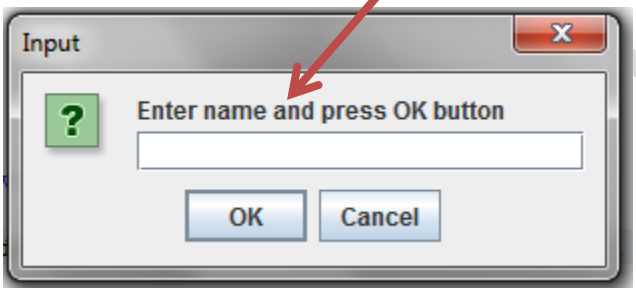
Str = JOptionPane.showInputDialog(stringExpression);

Str هو متغير من نوع السلسلة String و **stringExpression** هو أي تعبير يعطي قيمة من نوع السلسلة ويستخدم عادة لإرشاد المستخدم عن ما هو مطلوب عمله. عند تنفيذ العبارة السابقة يظهر صندوق حوار على الشاشة يحتوي على stringExpression وهو عبارة عن رسالة لحث المستخدم على إدخال بيانات حيث

يتم تحويل البيانات إلى نوع السلسلة وتسند إلى المتغير Str. لنفرض بأن لدينا متغير name من نوع السلسلة String، فعند تنفيذ العبارة التالية:

Name = JOptionPane.showInputDialog("Enter name and press OK button");

سيظهر صندوق الحوار كما في الشكل التالي:



وبعد إدخال الاسم والضغط على زر OK يختفي صندوق الحوار ويتم إسناد الاسم المدخل إلى المتغير name.

أما الدالة **showMessageDialog** تسمح للمستخدم من عرض النتائج وصيغتها :

JOptionPane.showMessageDialog(parentComponent,messageStringExpression,boxTitleString, messageType);

حيث **parentComponent** تعطى القيمة null كي يظهر صندوق الحوار في وسط الشاشة. **null** هي من الكلمات المحجوزة في لغة جافا.

messageStringExpression : هي الرسالة التي ستظهر على صندوق الحوار.

boxTitleString: عبارة عن عنوان صندوق الحوار حيث يظهر كل ما يوضع هنا على شريط عنوان الصندوق.

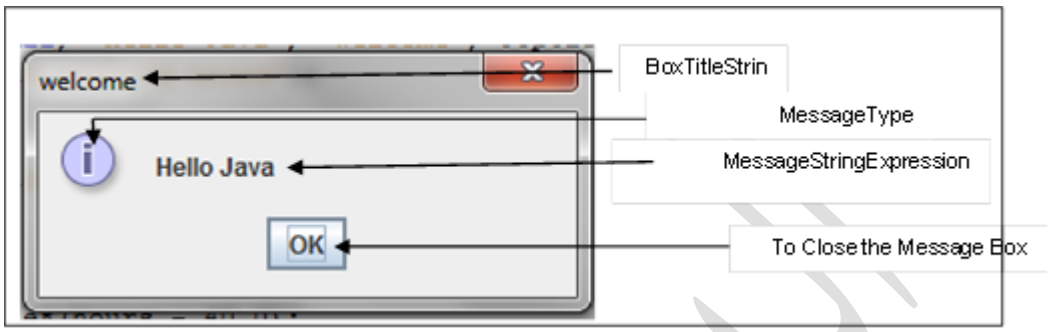
messageType : عبارة عن قيمة صحيحة تمثل نوع الأيقونة التي ستظهر في صندوق الحوار أو يمكن استخدام أسماء الأيقونات وكما يلي:

- **JOptionPane.INFORMATION_MESSAGE**

لإظهار أيقونة المعلومات في صندوق الحوار فالعبارة التالية :

JOptionPane.showMessageDialog(null,"Hello Java", "welcome", JOptionPane.INFORMATION_MESSAGE);

ستؤدي إلى ظهور صندوق الحوار كما في الشكل:

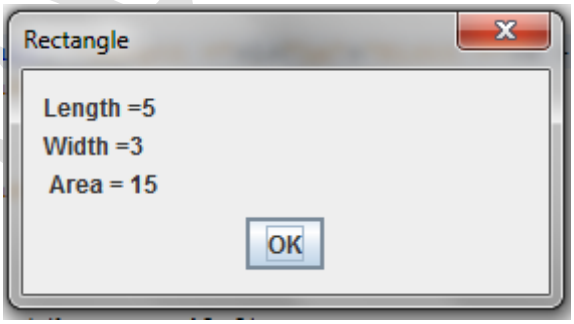


- **JOptionPane.PLAIN_MESSAGE**

لا تظهر أي أيقونة على صندوق الحوار.

العبارة التالية تعرض صندوق حوار عليه العنوان Rectangle والرسالة هي طول وعرض المستطيل ومساحته وهو بلا أيقونة حيث تم اختيار PLAIN_MESSAGE:

JOptionPane.showMessageDialog(null, "Length ="+L+"\n"+"Width ="+W +"\n Area = "+L*W, "Rectangle", JOptionPane.PLAIN_MESSAGE



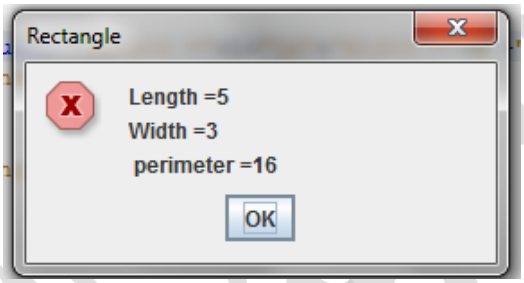
- **JOptionPane.ERROR_MESSAGE**

لإظهار أيقونة الخطأ في صندوق الحوار.

العبارة التالية تعرض صندوق حوار عليه العنوان Rectangle والرسالة هي طول وعرض المستطيل ومحيطه وأيقونة الخطأ حيث تم اختيار :ERROR _MESSAGE

```
JOptionPane.showMessageDialog(null, "Length ="+L+"\n"+"Width ="+W
+"\nperimeter="+(L+W)*2,"Rectangle",JOptionPane.ERROR_MESSAGE);
```

شكل صندوق الحوار كما يلي:

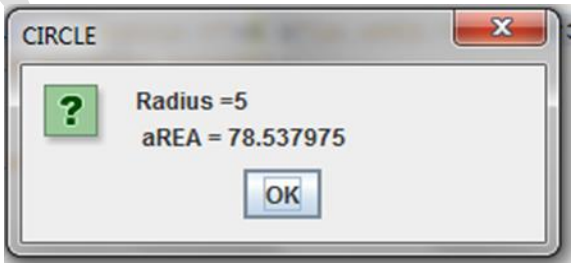


- **JOptionPane.QUESTION_MESSAGE**

لإظهار أيقونة الاستفهام في صندوق الحوار.

العبارة التالية تعرض صندوق حوار شكل بعنوان CIRCLE والرسالة هي نصف قطر الدائرة ومساحتها وأيقونة الاستفهام حيث تم اختيار QUESTION_MESSAGE

```
JOptionPane.showMessageDialog(null, "Radius ="+R +"\n Area =
"+R*R*3.141519, "CIRCLE", JOptionPane.QUESTION_MESSAGE);
```

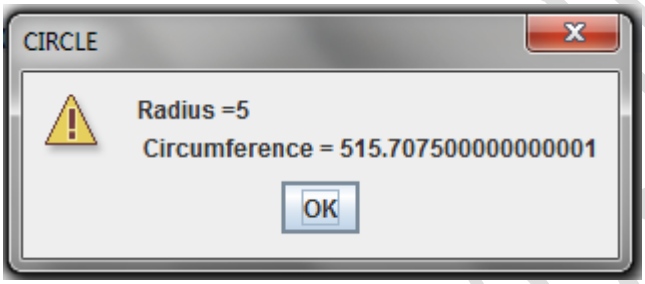


- **JOptionPane.WARNING_MESSAGE**

لإظهار أيقونة التحذير في صندوق الحوار.

العبارة التالية تعرض صندوق حوار شكل بعنوان CIRCLE والرسالة هي نصف قطر الدائرة ومحيطها وأيقونة التحذير حيث تم اختيار WARNING_MESSAGE

```
JOptionPane.showMessageDialog(null, "Radius ="+R +"\n Circumference =
"+R+R*3.1415, "CIRCLE", JOptionPane.WARNING_MESSAGE);
```



ومن الجدير بالذكر انه إذا أردنا استخدام صناديق الحوار في البرنامج فلابد من تضمين عبارة :

import javax.swing.JOptionPane;

أو عبارة import javax.swing.*;

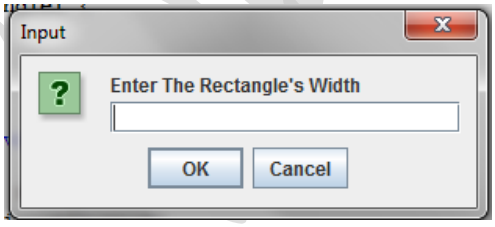
في بداية البرنامج لأن الفئة JOptionPane موجودة في الحزمة javax.swing

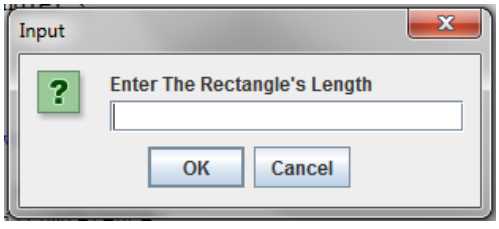
وكذلك يجب إضافة عبارة : ( ) System.exit؛ في نهاية البرنامج كي يتم إنهاء تنفيذ البرنامج

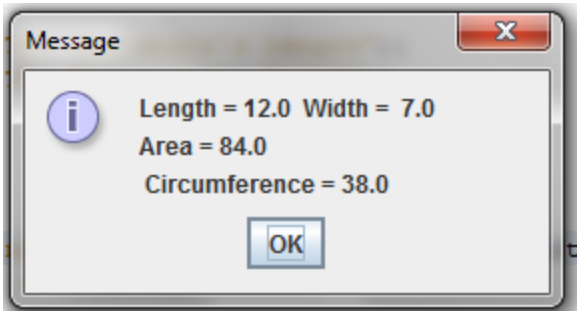
بصورة صحيحة، حيث تستخدم هذه العبارة في البرامج التي تستخدم الواجهات الرسومية GUI مثل صناديق الحوار للإدخال والإخراج.

لنأخذ مثال آخر لاستخدام صناديق الحوار للإدخال والإخراج، المثال التالي الذي يحسب محيط ومساحة مستطيل. لاحظ إن المدخلات والمخرجات من نوع **String** لهذا فإذا أردنا إدخال أرقام عن طريق صناديق الإدخال فلابد من تخزينها بمتغير من نوع السلسلة أولا ثم يتم تحويلها إلى متغير من النوع الملائم من الأعداد باستخدام دوال خاصة لهذا الغرض وحسب نوع الرقم كما موضح في البرنامج التالي:

```
package rectangle;
import javax.swing.JOptionPane;
public class Rectangle {
public static void main(String[] args) {
    double Length, Width;
    double Area, Circumference;
    String L, W;
     L = JOptionPane.showInputDialog("Enter The Rectangle's Length");
     W = JOptionPane.showInputDialog("Enter The Rectangle's Width");
     Length = Double.parseDouble(L);
     Width = Double.parseDouble(W);
     Area = Length * Width;
     Circumference = (Length + Width)*2;
     JOptionPane.showMessageDialog(null, "Length = "+Length + "  Width =
     "+Width+"\n"+"Area = "+Area +"\n Circumference = "+Circumference);
    } }
```







لاحظ إن كل من Length و Width عبارة عن متغيرين من نوع Double ولكن تم إدخالهما أولا على شكل سلسلة تم تخزينهما في المتغيرين L و W واللذان تم تعريفهما من نوع السلسلة String ثم تم تحويل السلسلتين إلى أعداد باستخدام الدالة **Double.parseDouble** وتوجد في جافا دوال أخرى للأنواع الأخرى من الأعداد. البرنامج السابق يحسب مساحة ومحيط مستطيل بعد أن يتم إدخال الطول Length والعرضًWidth عن طريق صناديق الإدخال.

- **الطريقة الثالثة باستخدام الملفات Using Files for I/O**

ملخص سريع

هناك نوعان من برامج جافا وهي تطبيقات Applications وبريمجات Applets والنوعان متشابهان من ناحية البرمجة.

البرامج التطبيقية تنفذ على حاسوبك أما البريمج applet فينفذ من متصفح الوب web browser أو برنامج لعرض البريمجات حيث تجعل الوب تتفاعل وتستجيب فتكون ممتعة للإستخدام.

يتكون برنامج جافا من فئة واحدة على الأقل تسمى الفئة الرئيسية وتحتوي على الدالة الرئيسية main.

لكتابة برنامج جافا تحتاج لمحرر مثل Notepad أو أي محرر آخر لخلق برنامج جافا يسمى البرنامج المصدر source program، حيث يخزن البرنامج في ملف نصي يسمى باسم الفئة المعرفة وبالامتداد . java.

يعمل المترجم على تحويل البرنامج إلى آخر بلغة وسطية تسمى bytecode ويخزن بنفس اسمه السابق مع الامتداد .class الخطوة التالية هي تنفيذ برنامج جافا فبالإضافة إلى ربط bytecode الفئات المختلفة يقوم برنامج يسمى المحملloader بتحميل bytecode البرنامج في الذاكرة الرئيسية. وأخيرا يعمل برنامج يسمى المفسر interpreter يقرأ ويفسر كل تعليمة من البايت كودbytecode إلى لغة الماكينة الخاصة بحاسوبك ثم ينفذها.

هناك ثلاث طرق لإدخال البيانات لبرنامج جافا وهي: الإدخال عن طريق لوحة المفاتيح باستخدام الفئة Scanner، باستخدام صناديق الحوار للإدخال والإخراج Dialog Boxes for I/O و الطريقة الثالثة باستخدام الملفات .Files for I/O