جامعة طرابلس كلية تقنية المعلومات

# البرمجة الشيئية

# Object Oriented Programming (with Java)

## ITGS211

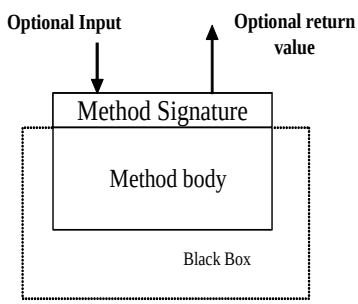
المحاضرة السادسة

الفصل الدراسي: ربيع 2022

## الدوال Methods

الدالة هي عبارة عن مجموعة من التعليمات تم تجميعها معاً لتؤدي وظيفة معينة. يتم استدعاؤها في البرنامج بواسطة اسمها.

تُسمى الدوال في Java بـ methods




**Method Abstraction**: يمكن التفكير في متن الدالة (method body) على أنه صندوق أسود يحوي تفاصيل تنفيذ الدالة.

2

## الهدف من استخدام الدوال Methods

- زيادة تنظيم البرنامج.
- سهولة تتبع البرنامج.
- سهولة اكتشاف الأخطاء في حالة حدوثها.
- توفير في كتابة الجمل البرمجية (يتم كتابة الجمل البرمجية مرة واحدة وتوضع في مكان عام ويتم استدعاء هذه الجمل البرمجية وتنفيذها من أكثر من مكان في البرنامج ويُعـــاد استخدامها عدة مرات وفي أي وقت).
- إخفاء المعلومات (Information hiding) حيث تُخفى آلية التنفيذ (implementation)عن المستخدم.
- التقليل من تعقيد البرنامج.

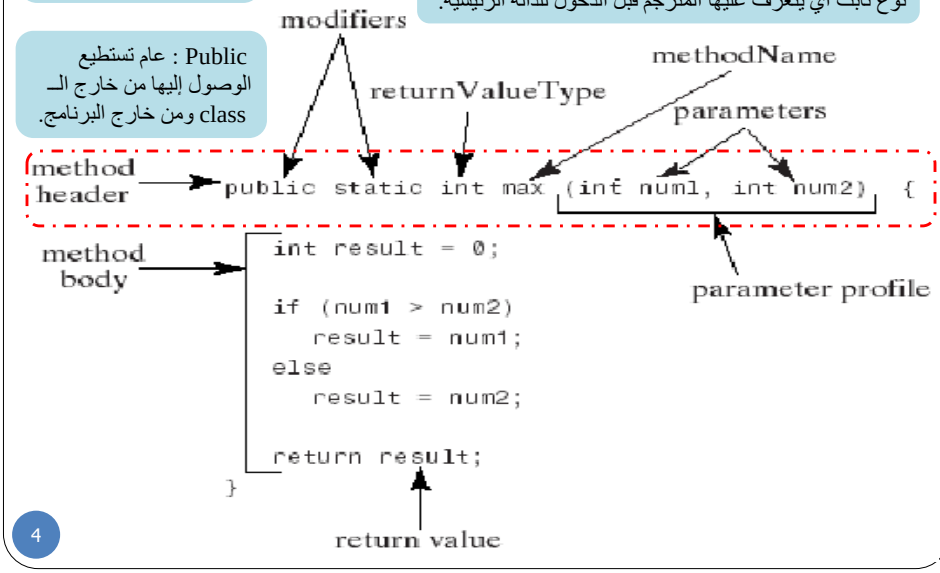
3

## تركيب الدالة Method

Private: خاصة فتستطيع الوصول للدالة من داخل الـــ class فقط.

• Static: وهي عبارة من أجل إخبار المترجم أن هذه الدالة من نوع ثابت أي يتعرف عليها المترجم قبل الدخول للدالة الرئيسية.

Public : عام تستطيع الوصول إليها من خارج الـ class ومن خارج البرنامج.




4

- ***parameter profile:*** هو وصف مختصر يشير إلى نوع وترتيب وعدد معاملات الـ method.
- ***method signature:*** أي توقيع الـmethod وهو مركب من اسم الـ method و parameter profiles .
- المعاملات المُعرفة في method header وتعرف بـ ***formal parameters***. أي معاملات رسمية.
- عندما يتم استدعاء الـmethod ، فإنه يتم استبدال معاملاتها الرسمية بالمتغيرات (variables) أو البيانات (data)، والتي يُطلق عليها المعاملات الفعلية (***actual parameters***).

5

## Method Declaration اعلان الدالة

```
Type methodName (parameterList)
{
  ,,,, body of method
}
```

**مثال:**

```
public static int max(int num1, int num2)
{
      int result=0;
      if (num1 > num2)
            result = num1;
      else
            result = num2;
      return result;
}
```

6

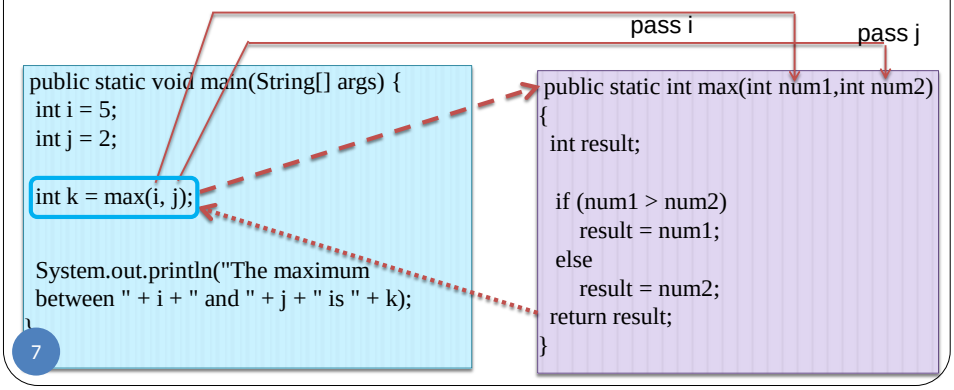
# Calling Methods استدعاء الدالة

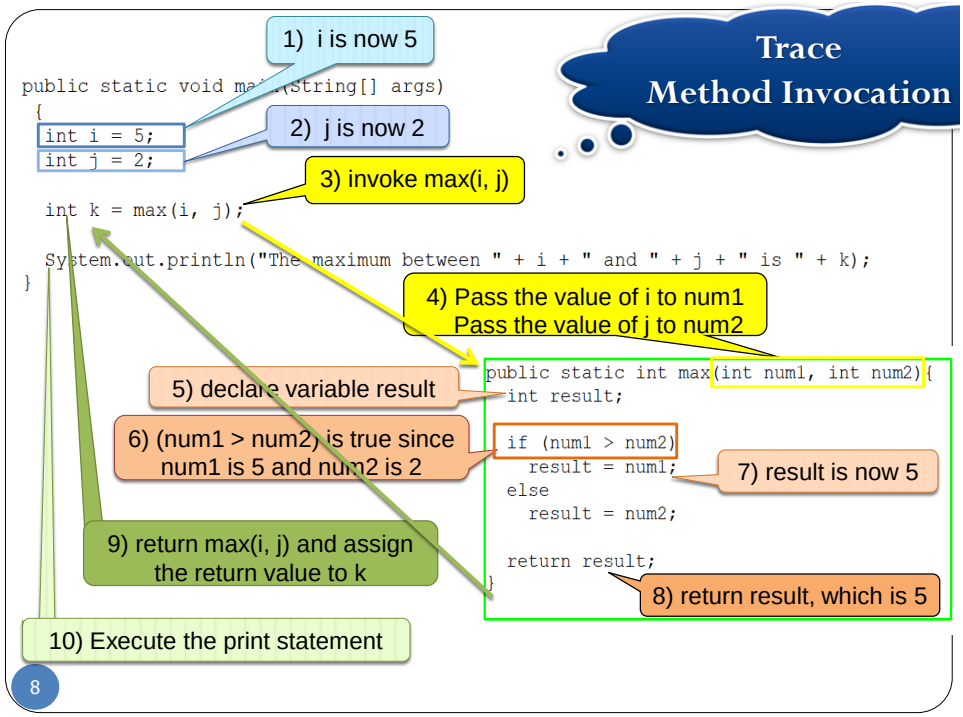
يمكن استدعاء الدالة داخل أي مكان في البرنامج عن طريق كتابة اسمها وإرسال قيم المعاملات إن وجدت، والصيغة العامة كالآتي:-

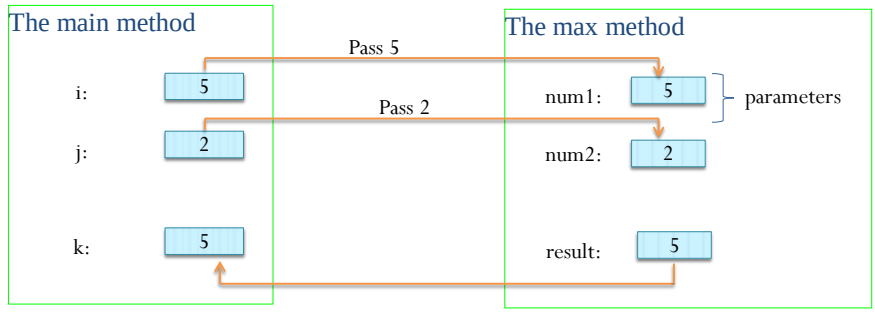
اسم المتغير = اسم الدالة (المدخلات)

**Example:** Testing the `max` method

البرنامج يوضح استدعاء الدالة max لتُرجع أكبر قيمة بين قيمتين صحيحين.






9

## Passing Parameters تمرير المُعاملات

```
import java.util.Scanner;
public class PassingPrm {
    public static void main(String[] args) {
        Scanner in=new Scanner(System.in);
        System.out.println("أدخل السلسلة المراد تكرارها");
        String m =in.next();
        System.out.println("ادخل عدد مرات التكرار");
        int n=in.nextInt();
        nPrintln(m,n);
    }
    public static void nPrintln(String message,int n)
    {
        for (int i = 0; i < n; i++)
            System.out.println(message);
    }
}
```

```
Output - PassingPrm (run)
run:
أدخل السلسلة المراد تكرارها
ITGS211_S2016
ادخل عدد مرات التكرار
2
ITGS211_S2016
ITGS211_S2016
BUILD SUCCESSFUL (total time: 18 seconds)
```

10

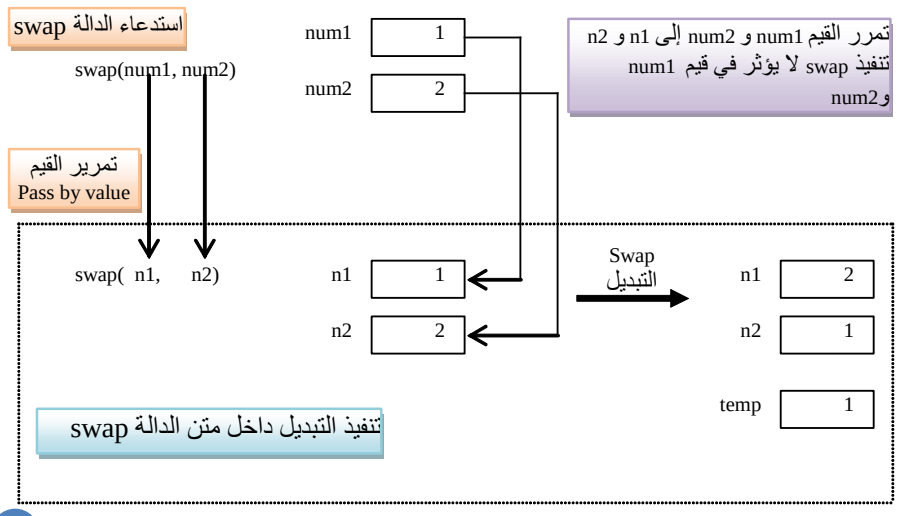
# Pass by Value التمرير بالقيمة

```
public class TestPassByValue
{
  public static void main(String[] args) {
        int num1 = 1;
        int num2 = 2;
        System.out.println("Before, num1 is " +  num1 + " and num2 is " + num2);
        swap(num1, num2);
        System.out.println("After, num1 is " + num1 + "  and num2 is " + num2);
  }

  public static void swap(int n1, int n2) {
        System.out.println("\t Inside the swap method");
        System.out.println("\t\t Before swapping n1 is " + n1 + " n2 is " + n2);
        int temp = n1;
        n1 = n2;
        n2 = temp;
        System.out.println("\t\t After swapping n1 is " + n1 + " n2 is " + n2);
  }
```

11

# Pass by Value التمرير بالقيمة




12

## نتيجة تنفيذ البرنامج :-

```
Before, num1 is 1 and num2 is 2
    Inside the swap method
            Before swapping n1 is 1 n2 is 2
            After swapping n1 is 2 n2 is 1
After, num1 is 1  and num2 is 2
```

13

## دالة تقوم بعملية الجمع واستدعائها

```
public class Method1 {
    public static void main(String[] args) {
    sum(10,15);
  }
  public static void sum(int a, int b)
    {
     int c;
        c=a+b;
        System.out.println(" Sum= "+c);
            }
}
```

استدعاء لدالة sum

Sum = 25

14

## نفس الدالة السابقة ولكن دالة تعود بقيمة

```
public class Method1 {
     public static void main(String[] args) {
    result = sum(30,25);
    System.out.println(" Sum= "+result);
  }
  public static int sum(int a, int b) {
      int c;
         c=a+b;
         return c;
               }
}
```

استدعاء لدالة sum

Sum = 55

15

## تتبع الجمل البرمجية التالية :-

```
public class Method1 {
     public static void main(String[] args) {
     int m=77
    System.out.println(" Result is : " + result(m));
  }
  public static boolean  result (int a) {
        boolean c;
       if(a>=50)
          c= true;
      else
          c=false;
       return c;}
 }
```

استدعاء لدالة result

Result is : true

16

## تتبع الجمل البرمجية التالية :-

```
public class Method1 {
    public static void main(String[] args) {
        System.out.println(" Area =" + area(20,40));
    }
    public static float area (int a , int b)
        {
         int x;
         x = a*b;
         return x;
        }
}
```

استدعاء لدالة area

Area = 800.0

17

**مثال**: اكتب برنامج بلغة جافا يقوم بتعريف المصفوفة list ذات البعد الواحد وطولها 5 كما يستدعي الدوال التالية:
1- readArray : تقوم بالقراءة (استقبال عناصر المصفوفة من المستخدم).
2- avgArray: تقوم بإرجاع متوسط عناصر المصفوفة.

18

**مثال**: اكتب برنامج بلغة جافا يقوم بتعريف المصفوفة list ذات البعد الواحد وطولها 5 كما يستدعي الدوال التالية:
1- readArray : تقوم بالقراءة (استقبال عناصر المصفوفة من المستخدم).
2- avgArray: تقوم بإرجاع متوسط عناصر المصفوفة.

```
import java.util.Scanner;    // Needed for the Scanner class
public class Array
{
  public static void main(String[] args)
  {
    double[] list = new double[5];
    System.out.println("** Elements of Array ***");
    readArray (list,5);
    System.out.println(" Average = " +avgArray(list) );
  }
```

استدعاء لدالة القراءة، معاملات الدالة: المصفوفة list وطولها 5

استدعاء لدالة حساب المتوسط، معامل الدالة: المصفوفة list

19

```
public static void readArray(double[] ar,int l)
  {  Scanner in = new Scanner(System.in);
     for(int i=0;i<l;i++)
     { System.out.print("\n list["+i+"]=");
       ar[i]=in.nextDouble();  }
   }

   public static double avgArray (double[] ar)
  {
   double sum1=0;
   for(int i=0; i<ar.length ;i++)
      sum1 += ar[i];
   return sum1/ ar.length ;
  }
}
```

دالة القراءة

دالة حساب المتوسط

20

اكتب برنامج باستخدام الدوال يقوم بإستدعاء دالة لا تعود بقيمة ولكن من خلالها يبدأ المستخدم بإدخال درجات الطالب ويتم طباعة رقم الطالب ودرجاته ثم يقوم بطباعة نتيجة الطالب راسب أو ناجح وذلك من خلال الاستدعاء لدالة أخرى ويطبع عدد الطلاب الناجحين وعدد الطلاب الراسبين؟

اكتب برنامج باستخدام الدوال يقوم بإنشاء دالة average1 تقوم بإدخال اسم الطالب ومعدله في الدالة الرئيسية ثم ترجع الدالة average1 الجهة الذي سيذهب إليها الطالب على الأسس التالية:
المعدل >= 87 يكون الجهة (القسم العلمي)
المعدل >= 77.2 يكون الجهة (القسم الأدبي).
المعدل >= 61.4 يكون الجهة (معهد متوسط).
المعدل >= 50 يكون الجهة (معهد صناعي).
غير ذلك يكون راسب

21

# Overloading Methods

Overloading التحميل الفائض للدالة: ويعني الاعلان على عدد من الدوال Methods تحمل نفس الاسم ولكن بمعاملات *parameters* مختلفة من حيث النوع والعدد.

Example Overloading the `max` Method

```
public static double max(double num1,double num2)
 {
   if (num1 > num2)
     return num1;
   else
     return num2;
 }
```

22

```
public class TestMethodOverloading {
public static void main(String[ ] args) {
   System.out.println("The max between 3 and 4 is " + max(3, 4) );
   System.out.println("The max between 3.0, 5.4, and 10.14 is " + max(3.0, 5.4, 10.14) );
   System.out.println("The max between 3.0 and 5.4 is " + max(3.0, 5.4) );
}
      public static int max(int num1, int num2) {
          if (num1 > num2)
                    return num1;
          else
                    return num2;
       }
      public static double max(double  num1, double  num2) {
          if (num1 > num2)
                    return num1;
          else
                    return num2;
       }
      public static double max(double num1, double num2, double num3) {
          return max(max(num1, num2), num3) ;
       }
}
```

23

# Overloading Methods, cont.

Example:

```
public class Calculation{
   public  static void  sum(int a, int b) {
           System.out.println(a + b);
   }
   public static void  sum(int a, int b, int c) {
           System.out.println(a + b + c);
   }

   public static void main(String args[]) {
           sum(10,10,10);
           sum(20,20);
   }
}
```

24

- The following methods are incorrectly overloaded; the compiler generates an error:
- Example (1): The following methods are incorrectly overloaded because they have the same method name and same formal parameter lists:
  - public void methodABC (int x, double y)
  - public int methodABC (int x, double y)
- Example (2): Changing the names of the formal parameters, does not allow overloading of the previous counter-example:
  - public void methodABC (int x, double y)
  - public int methodABC (int num1, double num2)
- Counter-example (3): Adding the modifier static does not allow overloading of the previous example:
  - public static void methodABC (int x, double y)
  - public int methodABC (int num1, double num2)

Note that the method type and modifiers are not part of the overloading rules

25

## Ambiguous Invocation

Sometimes there may be two or more possible matches for an invocation of a method, but the compiler cannot determine the most specific match. This is referred to as *ambiguous invocation*. Ambiguous invocation is a compilation error.

.

26

## Ambiguous Invocation الاستدعاء الغامض

```
public class AmbiguousOverloading  {
      public static void main(String[] args)  {
      System.out.println( max(1, 2) );
  }

  public static double max(int num1, double num2) {
      if (num1 > num2)
            return num1;
      else
            return num2;
}
  public static double max(double num1, int num2) {
      if (num1 > num2)
            return num1;
      else
            return num2;
  }
}
```

27

## Ambiguous Invocation

```
   public static void main(String[] args) {
       System.out.println( max(1, 2) );
```

reference to max is ambiguous
both method max(int,double) in OverLod and method max(double,int) in OverLod match
---
(Alt-Enter shows hints)

```
  public static double max(int num1, double num2) {
        if (num1 > num2)
              return num1;
        else
              return num2;
  }
  public static double max(double num1, int num2) {
        if (num1 > num2)
              return num1;
        else
              return num2;
  }
}
```

28

```
    public static void main(String[] args) {
        max(2,3.2);
    }

  public static double max(int num1, double num2) {
        if (num1 > num2)
            return num1;
        else
            return num2;
}
  public static double max(double num1, int num2) {
        if (num1 > num2)
            return num1;
        else
            return num2;
  }
```

29

```
    public static void main(String[] args) {
        max(2.5,3);
    }

  public static double max(int num1, double num2) {
        if (num1 > num2)
            return num1;
        else
            return num2;
}
  public static double max(double num1, int num2) {
        if (num1 > num2)
            return num1;
        else
            return num2;
  }
}
```

30

## Scope of Local Variables

A **local variable**: a variable defined inside a method.

Scope: the part of the program where the variable can be referenced.

The scope of a local variable starts from its declaration and continues to the end of the block that contains the variable.

A local variable must be declared before it can be used.

متغير المحلي : متغير تعريف داخل اللدالة.

النطاق: جزء من البرنامج حيث المتغير يمكن الرجوع إليها.

نطاق متغير محلي يبدأ من إعلانه ويستمر إلى نهاية الكتلة التي تحوي المتغير، و يجب أن يتم تعريف المتغير المحلي قبل أن تتمكن من استخدامه.

31

## Scope of Local Variables, cont.

You can declare a local variable with the same name multiple times in different non-nesting blocks in a method,

but you cannot declare a local variable twice in nested blocks. Thus, the following code is correct.

يمكنك تعريف متغير محلي بنفس الاسم عدة مرات في الدالة، ولكن لا يمكنك تعريف متغير محلي مرتين في كتل متداخلة و هكذا

32

## Scope of Local Variables, cont.

```
// Fine with no errors
public static void correctMethod()
{
          int x = 1;
          int y = 1;
          // i is declared
          for (int i = 1; i < 10; i++) {
               x += i;
          }
          // i is declared again
          for (int i = 1; i < 10; i++) {
               y += i;
          }
}
```

33

## Scope of Local Variables, cont.

```
// With no errors
public static void incorrectMethod() {
          int x = 1;
          int y = 1;
          for (int i = 1; i < 10; i++)
          {
            int x = 0;
            x += i;
          }
}
```

34

ويوجد عدد كبير من الدوال الجاهزة المعرفة في اللغة جاهزة وما على المستخدم الا استدعاؤها ومنها:

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

35

## The `Math` Class

- Class constants:
  - `PI`
  - `E`
- Class methods:
  - Trigonometric Methods المثلثية
  - Exponent Methods الأسية
  - Rounding Methods التقريبية
  - min, max, abs, and random Methods

  دوال الحد الأعلى، الحد الأدنى، القيمة المطلقة، القيمة العشوائية

36

## Trigonometric Methods الدوال المثلثية

- `sin(double a)`
- `cos(double a)`
- `tan(double a)`
- `acos(double a)`
- `asin(double a)`
- `atan(double a)`

37

## Exponent Methods الدوال الأسية

- `exp(double a)`
  Returns `e` raised to the power of `a`.
- `log(double a)`
  Returns the natural logarithm of `a`.
- `pow(double a, double b)`
  Returns `a` raised to the power of `b`.
- `sqrt(double a)`
  Returns the square root of `a`.

38

## Rounding Methods

- `double ceil(double x)`
  `x rounded up to its nearest integer. This integer is returned as a double value.`
- `double floor(double x)`
  `x is rounded down to its nearest integer. This integer is returned as a double value.`
- `double rint(double x)`
  `x is rounded to its nearest integer. If x is equally close to two integers, the even one is returned as a double.`
- `int round(float x)`
  `Return (int)Math.floor(x+0.5).`
- `long round(double x)`
  `Return (long)Math.floor(x+0.5).`

39

## min, max, abs, and random

- `max(a, b)` and `min(a, b)`
  Returns the maximum or minimum of two parameters.
- `abs(a)`
  Returns the absolute value of the parameter.
- `random()`
  Returns a random `double` value in the range [0.0, 1.0).

40

# The Math Class

**مثال:** حساب طول ضلع الوتر في مثلث قائم الزاوية باستخدام نظرية فيثاغورس ؟

```
public class Math1 {
   public static void main(String[] args) {
      double z,x,result;
      z=12;
      x=16;
      result=(z*z)+(x*x);
      System.out.println("The result is : "+Math.sqrt(result));
        }
   }
```

The result is : 20.0

41

**مثال:** اكتب برنامج بلغة جافا يقوم بقراءة المصفوفة ذات البعدين matrix وطباعة مربع كل عنصر من عناصر المثلث السفلي لها.

```
import java.util.Scanner;
public class Array {
    public static void main(String[] args) {
        Scanner in= new Scanner (System.in);
        int m=in.nextInt();
        double[][] matrix = new double[m][m];
        for(int k=0; k<m;k++)
            for(int j=0; j<m;j++)
                matrix[k][j]=in.nextDouble();
        for(int k=0; k<m;k++){
            for(int j=0; j<m;j++){
                if (k>j)
                    System.out.print(Math.pow(matrix[k][j],2) + "  ");
                }
            System.out.println();
        }
    }
}
```

42

```java
public static void main(String[] args) {
    System.out.println(Math.max(10, 30));
    //أكبر قيمة بين العددين
    System.out.println(Math.min(10, 30));
    // أصغر قيمة بين العددين
    System.out.println(Math.abs(-10));
    // القيمة المطلقة للعدد
    System.out.println(Math.exp(3.0));
    // القيمة الأسية للعدد
    System.out.println(Math.sqrt(144));
    // الجذر التربيعي للعدد
    System.out.println(Math.pow(2,8));
    //العدد الأول مرفوع للأس العدد الثاني
    System.out.println(Math.ceil(9.5));
    //  تقرب إلى أصغر قيمة أكبر من القيمة المعطاة
    System.out.println(Math.ceil(-9.5));
    //  تقرب إلى أصغر قيمة أكبر من القيمة المعطاة
    System.out.println(Math.floor(5.2));
    //5.2 أكبر قيمة صحيحة أقل من أو تساوي
    System.out.println(Math.floor(-5.2));
    //5.2 أكبر قيمة صحيحة أقل من أو تساوي
}
```

```
Output - JavaApplication1 (run)
run:
30
10
10
20.085536923187668
12.0
256.0
10.0
-9.0
5.0
-6.0
BUILD SUCCESSFUL (to
```

43

# انتهت المحاضرة
# أي أسئلة؟

اسئل ولا تتردد ...

فالفائدة تبدأ بالسؤال ثم النقاش ..

44