



ALZEEKA Tutorial

Programming 1 – CSCE 101



053 359 7191



<https://alzeeka.github.io/alzeeka/>

FREE

ملخص 2 (Declaration & Assignment) lecturer

*الكلمات الي تحتها خط ركزوا عليها

❖ Java Program Structure هيكلية برامج الجافا

```
Class
```

```
{
```

```
    Method
```

```
    {
```

```
        instructions;
```

```
    }
```

```
}
```



053 359 7191



```
// comments about the class
```

```
public class MyProgram
```

```
{
```

class header

class body

Comments can be placed almost anywhere

```
}
```

```
// comments about the class
```

```
public class MyProgram
```

```
{
```

```
    // comments about the method
```

```
    public static void main (String[] args)
```

```
    {
```

method header

method body

```
    }
```

```
}
```



❖ Greetings.java مثال لبرنامج

```
// This program displays a greeting message to the user.
public class Greetings
{
    // Displays a greeting message to the user
    public static void main(String[] args)
    {
        System.out.println("Welcome to Java.");
    }
}
```

Welcome to Java.

❖ مخرج هذا البرنامج

```
public class MyFirstProgram
{
    public static void main(String[] args)
    {
        System.out.println("Welcome to Java.");
        System.out.println("Java program sample.");
        int answer;
        answer = 2 + 2;
        System.out.println("2 Plus 2 is:" + answer);
    }
}
```

Welcome to Java.
Java program sample.
2 Plus 2 is: 4

❖ مخرج هذا البرنامج



053 359 7191



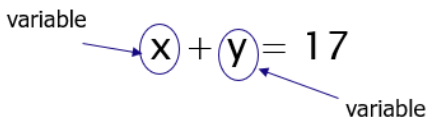
❖ Objects and Methods

الكائنات والدوال

- **Objects:** Perform actions.
- **Methods:** is the action.
- **الكائنات:** تنفيذ الإجراءات.
- **الدوال:** هي الإجراءات.
- Java programs work by having things called **objects** which perform **actions**
- **System.out:** an **object** used for **sending output** to the screen
 - تعمل برامج جافا عن طريق وجود أشياء تسمى الكائنات التي تنفذ الإجراءات
 - **System.out** كائن يستخدم لإرسال الإخراج إلى الشاشة
- The actions performed by an object are called **methods**
- **println:** the **method** or action that the **System.out** object performs
 - الإجراءات التي ينفذها الكائن تسمى أساليب
 - **println:** الدالة أو الإجراء الذي ينفذه كائن System.out

❖ Variables

المتغيرات

- In mathematics:

- In Computer Science: a **variable** is an **identifier** (usually a **letter**, **word**, or **phrase**) that is **linked to a value stored in the computer's memory**
 - في علوم الحاسوب: المتغير هو معرف (عادةً حرف أو كلمة أو عبارة) مرتبط بقيمة مخزنة في ذاكرة الحاسوب.

variable → $x = 5;$



053 359 7191



❖ Variable Declaration

تعريف المتغير

- Every variable in a Java program must be declared before it is used Simply give the type of the variable followed by its name and a semicolon
- كل متغير في برنامج الجافا يجب ان يتم الإعلان عنها قبل استخدامها ما عليك سوى إعطاء نوع المتغير متبوعًا باسمه وفاصلة منقوطة

int answer;

- A variable declaration tells the compiler what kind of data (type) will be stored in the variable
- يخبر تعريف المتغير المترجم بنوع البيانات (النوع) التي سيتم تخزينها في المتغير
- **Basic types** in Java are called primitive types
- تسمى الأنواع الأساسية في Java بالأنواع البدائية
- The type of the variable is followed by one or more variable names separated by commas, and terminated with a semicolon
- نوع المتغير يتبعه اسم متغير واحد أو أكثر مفصولة بفواصل، وتنتهي بفاصلة منقوطة مثل:

int x , y , z;

- Variables are typically declared just before they are used or at the start of a block (indicated by an opening brace {)
- نوع تعلن المتغيرات عادة قبل استخدامها أو في بداية الكتلة (المشار إليها بفتحة الأقواس { }).

Public void info (int x){ }

- Other Examples :

int numberOfBeans;

double oneWeight, totalWeight;



❖ Primitive Data Types

أنواع البيانات الأساسية

Display 1.2 Primitive Types

TYPE NAME	KIND OF VALUE	MEMORY USED	SIZE RANGE
boolean	true or false	1 byte	not applicable
char	single character (Unicode)	2 bytes	all Unicode characters
byte	integer	1 byte	-128 to 127
short	integer	2 bytes	-32768 to 32767
int	integer	4 bytes	-2147483648 to 2147483647
long	integer	8 bytes	-9223372036854775808 to 9223372036854775807
float	floating-point number	4 bytes	$-3.40282347 \times 10^{+38}$ to $-1.40239846 \times 10^{-45}$
double	floating-point number	8 bytes	$\pm 1.76769313486231570 \times 10^{+308}$ to $\pm 4.94065645841246544 \times 10^{-324}$

❖ قواعد تسمية المتغيرات (Identifiers) Rules of naming variables

*مهمة

- **No spaces** in variable names
- **No special symbols** in variable names such as !@#%^&*+/-
- Variable names **can only contain letters, numbers, underscore (_), and dollar sign**
- Variable names can **not start with numbers**, only **letters, underscore, or dollar signs**.
- It is **not allowed to use reserved words** as variable names

- لا توجد مسافات في أسماء المتغيرات
- لا توجد رموز خاصة في أسماء المتغيرات مثل !@#%^&*+/-
- يمكن أن تحتوي أسماء المتغيرات فقط على الحروف والأرقام وشرطة سفلية (_) وعلامة الدولار.
- لا يمكن أن تبدأ أسماء المتغيرات بأرقام، فقط بحروف، أو شرطة سفلية، أو علامات الدولار.
- لا يُسمح باستخدام الكلمات المحجوزة كأسماء متغيرة.



053 359 7191



- Java is a **case-sensitive language**: Sum, sum, and SUM are names of three different variables

• جافا هي لغة **حساسة لحالة الأحرف**: Sum و sum و SUM هي أسماء لثلاثة متغيرات مختلفة.

- Choose meaningful names that describe what the variable is being used for.
- برضو لازم نهتم بالتسميات الصحيحة للمتغيرات مثل هذا المثال : .

X = a + b;

Sum = number1 + number2;

❖ Task: Identifiers

- Which of the following is a valid identifier?

• أي من هذه المعرفات يعتبر صحيح :

- IntRate (A: Valid, B: Not Valid)
- Five_speed (A: Valid, B: Not Valid)
- 5_speed (A: Valid, B: Not Valid)
- _5 speed (A: Valid, B: Not Valid)
- MercedesSL500 (A: Valid, B: Not Valid)
- Time.and.space (A: Valid, B: Not Valid)
- Less<5 (A: Valid, B: Not Valid)

لانه بدأ برقم

لوجود علامة .

لوجود علامة <

Valid	Not Valid
MyVariable	My Variable
MYVARIABLE	a+c
x	O'Reilly
\$myvariable	testing1-2-3
_9pins	9pins
MAIN	main
myvariable	Reilly&_Associates



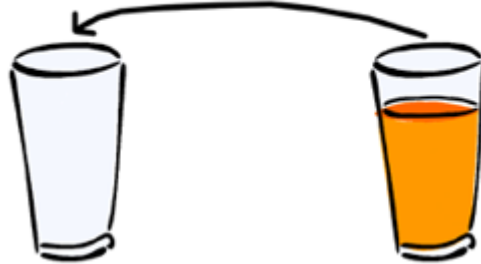
❖ Declaring Variables

- `int x;` `// Declare x to be an integer variable;`
- `double radius;` `// Declare radius to be a double variable;`
- `char a;` `// Declare a to be a character variable;`

❖ Assignment Statements "="

- In Java, the assignment statement is used to change the value of a variable
 - في جافا، يتم استخدام عبارة الإسناد لتغيير القيمة متغير
- An assignment statement consists of a variable on the left side of the operator, and an expression on the right side of the operator
- تتكون عبارة الإسناد من متغير على الجانب الأيسر من العامل، وتعبير على الجانب الأيمن من العامل.

Variable = Expression;



- Example
- `temperature = 98.6;`
- `x = 5 + 3;`
- `count = numberOfBeans;`



❖ Assignment Statements With Primitive Types

- In When an assignment statement is executed, the expression is first evaluated, and then the variable on the left-hand side of the equal sign is set equal to the value of the expression
- في عند تنفيذ عبارة الإسناد، يتم تقييم التعبير أولاً، ثم يتم تعيين المتغير على الجانب الأيسر من علامة اليساوي ليكون مساوياً لقيمة التعبير.

distance = rate * time;

- Note that a variable can occur on both sides of the assignment operator
- لاحظ أن المتغير يمكن أن يحدث على كلا الجانبين من عامل التسديد

count = count + 2;

- The assignment operator is automatically executed from right-to-left, so assignment statements can be chained
- المشغل المسند يتم تنفيذه تلقائياً من اليمين إلى اليسار ، لذا يمكن ربط عبارات التعيين معاً.

number2 = number1 = 3;

❖ Initialize Variables

تهيئة المتغيرات

- A variable that has been declared but that has not yet been given a value by some means is said to be **uninitialized**
- In certain cases, an uninitialized variable is given a **default value**
- المتغير يُقال إن المتغير الذي تم الإعلان عنه ولكن لم يتم إعطاؤه قيمة بعد بطريقة ما **غير مهياً** في بعض الحالات، يتم إعطاء متغير غير مهياً **قيمة افتراضية** ، لكن من الأفضل عدم الإعتماد على القيم الافتراضية ، وتهيئة المتغيرات بقيم ابتدائية عشان يكون البرنامج واضح



- The **declaration** of a variable can be combined with its **initialization** via an **assignment** statement

• يمكن دمج إعلان المتغير مع تهيئته عن طريق عبارة **التعيين**

```
int count = 0;
```

```
double distance = 55 * 0.5;
```

```
char grade = 'A';
```

- Note that some variables can be initialized, and others can remain uninitialized in the same declaration

• يمكن توجد بعض المتغيرات التي يمكن تهيئتها. ويمكن أن يبقى الآخرون غير مهيأين في نفس الإعلان

```
int initialCount = 50, finalCount;
```

```
number2 = number1 = 3;
```

❖ Default Variable Initializations

تهيئات المتغير الافتراضي

- **Instance variables** are automatically initialized in Java
- **المتغيرات الفردية** تتم تهيئتها تلقائيًا في جافا.
- **Boolean** types are initialized to a **false** value
- **Other primitives** are initialized to the **zero** of their type
- **Class types** are initialized to **null**

• أنواع البوليان تهيئًا بقيمة **false**

• الأنواع الأساسية الأخرى تهيئًا بصفر

• أنواع الكلاسات او String تهيئًا بقيمة **null**



❖ Default values

القيم الافتراضية (*مهم)

Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object or class type)	null
boolean	false

❖ Shorthand Assignment Statements

- The general form is

Variable Op= Expression

- which is equivalent to

Variable = Variable Op (Expression)

- حيث **Variable** هو اسم المتغير
- حيث **OP** هو العملية مثل الجمع الطرح الخ
- حيث **Expression** هو التعبير الرياضي او معادلة مثلا



Example	Equivalent To
<code>count += 2;</code>	<code>count = count + 2;</code>
<code>sum -= discount;</code>	<code>sum = sum - discount;</code>
<code>bonus *= 2;</code>	<code>bonus = bonus * 2;</code>
<code>time /= rushFactor;</code>	<code>time = time / rushFactor;</code>
<code>change %= 100;</code>	<code>change = change % 100;</code>
<code>amount *= count1 + count2;</code>	<code>amount = amount * (count1 + count2);</code>

• حدد هل valid او Not valid

- `int x;` **Valid**
- `int x , y;` **Valid**
- `int x=9 ; y;` **Not valid** هنا خطأ لانه فصلنا بين المتغيرين بالفاصلة المنقوطة
- `int x=9; int y;` **Valid**

❖ Assignment Compatibility

توافق الإسناد

- In general, the value of one type cannot be stored in a variable of another type
- في العام، لا يمكن تخزين قيمة نوع واحد في متغير من نوع آخر.

`int intValue = 2.99; //Illegal`

- المثال أعلاه يؤدي إلى عدم تطابق الأنواع لأنه لا يمكن تخزين قيمة عشرية **double** في متغير صحيح **int**

- However, there are exceptions to this
- ومع ذلك، هناك استثناءات لهذا.

`double doubleVariable = 2; ; //legal`

- على سبيل المثال ، يمكن تخزين قيمة int في نوع double
- More generally, a value of any type in the following list can be assigned to a variable of any type that appears to the right of it
- أكثر عموماً، يمكن تعيين قيمة من أي نوع في القائمة التالية إلى متغير من أي نوع يظهر على يمينه

***مهم**

byte → short → int → long → float → double

- يعني يمكن تخزين قيمة نوع صغير داخل نوع كبير

- int x = 5.8; **error**
- double y = 6; **ok**
- short i = 14.3; **error**
- float s = 9.2; **ok**

❖ Type Casting

تحويل الأنواع

- A **type cast** takes a value of one type and produces a value of another type with an "equivalent" value
- يأخذ ال casting قيمة من نوع واحد وينتج قيمة من نوع آخر بقيمة "مكافئة"

int x = (int) 2.9;

- When type casting from a floating-point to an integer type, the number is truncated, not rounded:
-
- صباغة من النقطة العائمة إلى نوع صحيح، يتم **قص** العدد (يعني راح ناخذ العدد الي قبل الفاصلة) وليس **تقريبه** لذلك سيكون ناتج العملية السابقة هو 2 وليس 3



053 359 7191



Arithmetic Operators

العمليات الرياضية

- + addition
- - subtraction
- * multiplication
- / division
- % modulo, remainder تسمى باقي القسمة

❖ Arithmetic Operators and Expressions العمليات و التعبيرات الرياضية

- If an arithmetic operator is combined with int operands, then the resulting type is int
- If an arithmetic operator is combined with one or two double operands, then the resulting type is double

***مهم**

- إذا كانت العمليات الحسابية تشمل أعدادًا صحيحة، فإن النوع الناتج هو عدد صحيح.
- إذا تم دمج مشغل حسابي مع عاملين أو عامل واحد من الأعداد العشرية، فإن النوع الناتج هو عدد عشري

❖ The % (modulo) Operator

- يوجد **3 طرق** لإيجاد باقي القسمة

الطريقة الاولى :

$17 \% 5$

• راج نقسم ال 17 على 5 بطريقة بدائية بهذا الشكل

• بعد ما قسمنا على 5 بشكل متساوي صار معنا باقي خانتين ، يعني الباقي 2

$17 \% 5 = 2$

• الطريقة الثانية باستخدام التفكير يلزمك تكون حافظ جدول الضرب :

$$31 \% 4$$

• شلون نجيبها طيب نحتاج رقم نظربه في ٤ ويكون ناتجه اقل او تساوي ٣١ مثلا $8 * 4$ تساوي ٣٢ يكون ٣٢ اكبر من ٣١ نحاول نشوف رقم اصغر من ٨ مثلا $7 * 4$ يصير ٢٨ هذا تمام لانه اقل من ٣١ يعني باقي القسمة يساوي ٣

$$31 \% 4 = 3$$

• الطريقة الثالثة باستخدام الآلة الحاسبة:

$$64 \% 3$$

$$\frac{64}{3} = 21.333333333333333$$

• شلون نجيبها اولاً نسوي عملية القسمة $64 / 3$ يكون الناتج 21.333

اخذ الرقم الصحيح الي ما قبل الفاصلة العشرية واضربه في ٣ $21 \times 3 = 63$

يكون الناتج ٦٣ خلاص بالالاخير نطرح الناتج من ٦٤ $64 - 63 = 1$ يعني باقي القسمة يساوي ١

$$64 \% 3 = 1$$

- $15/2 = 7$
- $15\%2 = 1$
- $55\%4=3$
- $3\%6 = 3$
- $23\%3=2$
- $36\%5=1$
- $14\%2=0$ even
- $15\%2=1$ odd

* اذا كان العدد الأول اقل من العدد الثاني الناتج هو الرقم الأصفر

***مهم**

في الدروس المتقدمة لما نحتاج نفحص ان العدد زوجي او فردي راج نحتاج هالصيغتين

- باقي قسمة أي عدد على 2 يساوي **صفر** هذا يعتبر عدد زوجي او even $14\%2=0$
- باقي قسمة أي عدد على 2 يساوي **واحد** هذا يعتبر عدد فردي او odd $15\%2=1$



053 359 7191



- **The % operator** is used with operands of type int to recover the information lost after performing integer division
- يتم استخدام عامل النسبة المئوية مع المشغلات من نوع int لإستعادة المعلومات المفقودة بعد أداء القسمة الصحيحة مثلا:

- **15/2** evaluates to the quotient **7**
int/ int = int
- **15%2** evaluates to the remainder **1**

- الآن لما قسمنا 15 على 2 كان الناتج 7 هو المفروض تطلع زي حسبنا 7 وكسور , طيب كيف راح نجيب الباقي , ببساطة نستخدم باقي القسمة % طلع ان 15 تقبل القسمة على 7 من 2 وكان الباقي هو 1

❖ Integer and Floating-Point Division قسمة الأعداد الصحيحة والعشرية

- *** باختصار**
- اذا كان كلا العددين القاسم والمقسوم من نوع **int** الناتج يكون **int**
- اذا كان كلا او احد العددين القاسم والمقسوم من نوع **double** او **float** يعني عدد عشري الناتج يكون **double** او عدد عشري

```
public class MathOperations1 {
    public static void main(String[] args) {
int total;

        int val1, val2;
        val1 = 2;
        val2 = 5;
        total = val1 + val2;
        System.out.println("The addition is: " + total);
    }}

```

- **Output:** The addition is: 7



❖ Parentheses and Precedence Rules

قواعد الأقواس والأسبقيات

- An expression can be fully parenthesized in order to specify exactly what subexpressions are combined with each operator
- If some or all of the parentheses in an expression are omitted, Java will follow precedence rules to determine, in effect, where to place them
- However, it's best (and sometimes necessary) to include them

• *** باختصار**

• يمكن وضع التعبير الرياضي او expression داخل الأقواس او Parentheses للتحكم بالأولوية

- If some or all the parentheses in an expression are omitted, Java will follow precedence rules to determine, in effect, where to place them.

• إذا تم حذف بعض أو كل الأقواس في تعبير، ستتبع جافا قواعد الأسبقية

High precedence



First: The unary operators: +, -, ++, --, and !

Second: The binary arithmetic operators: *, /, and %

Third: The binary arithmetic operators: + and -

Low precedence

*** دائما الأسبقية للأقواس وبعد تكون ل**

- المعاملات الأحادية مثلا سالب العدد او موجب العدد او الزيادة والنقصان او علامة النفي
- عمليات * / %
- عمليات الجمع والطرح

*** إذا صادفت نفس الأسبقيات مثلا الجمع والطرح او الضرب والقسمة قم بالحل من اليسار الى اليمين**



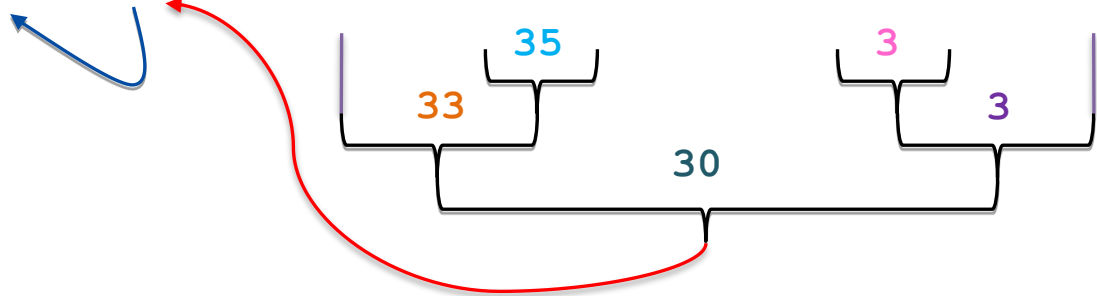
053 359 7191



e.g. ➔ `int x = y = -2 + 5 * 7 - 7 / 2 % 5;`

This will be evaluated as..

`int x = (y = ((-2 + (5 * 7)) - ((7 / 2) % 5))) ;`



❖ Task

```
public class calc1
{
    public static void main(String [] args)
    {
        int x= 5;
        int y= 2;
        System.out.println("The Sum =" + x * (x + y));
    }
}
```

- **Output:** The Sum =35



```
public class ArithmeticOperators
{
    public static void main(String []arg) {
        int a=5; int b=4;
        int x,y,z,v,u;
        float f,c=4.0f;
        x=a+b;          y=a-b;
        z=a*b;          v=a/b;
        f=a/c;          u=a%b;
        System.out.println("a+b="+x);
        System.out.println("a-b="+y);
        System.out.println("a*b="+z);
        System.out.println("a/b="+v+"\t"+"a/c="+f);
        System.out.println("a%b="+u);
    }
}
```

- **Output :**

a-b=1

a*b=20

a/b=1 a/c=1.25

a%b=1



❖ Increment and Decrement Operators

الزيادة والنقصان

- When either operator precedes its variable, and is part of an expression, then the expression is evaluated using the changed value of the variable

• إذا كانت علامة الزيادة او النقصان اتت قبل المتغير سيتم تغيير قيمته بحسب العلامة , مثل هذا المثال اذا كانت ال $n=2$ وكانت علامة الزيادة تسبق المتغير لذلك قمنا بزيادة ال n بمقدار واحد في نفس السطر ثم ضربنا في 2 واصبح الناتج 6

If n is equal to 2, then $2*(++n)$ evaluates to 6

- When either operator follows its variable, and is part of an expression, then the expression is evaluated using the original value of the variable, and only then is the variable value changed

• إذا كانت علامة الزيادة او النقصان اتت بعد المتغير سيتم اعطاء المتغير نفس قيمه الأصلية وسيتم تغيير قيمته لاحقا بحسب العلامة , مثل هذا المثال اذا كانت ال $n=2$ وكانت علامة الزيادة بعد المتغير , فستكون قيمة ال n مثل ماهي بدون تغيير فنحسب المعادلة بشكل طبيعي $2*2$ فسيكون الناتج 4 اما بالنسبة للزيادة فسيتم الزيادة في السطر الذي يلي هذا الكود

If n is equal to 2, then $2*(n++)$ evaluates to 4

suffix \rightarrow $x++;$ // Same as $x = x + 1;$
prefix \rightarrow $++x;$ // Same as $x = x + 1;$
suffix \rightarrow $x--;$ // Same as $x = x - 1;$
prefix \rightarrow $--x;$ // Same as $x = x - 1;$

- لما نكتب كذا $x++$ او $++x$ هي نفسها كذا $x = x + 1$
- لما نكتب كذا $x--$ او $--x$ هي نفسها كذا $x = x - 1$



`int i=10;`
`int newNum = 10*i++;`

Equivalent to

`int newNum = 10*i;`
`i = i + 1;`

- في هذا المثال اولاً نحسب المعادلة ومن ثم نقوم بزيادة الـ `i`

`int i=10;`
`int newNum = 10*(++i);`

Equivalent to

`i = i + 1;`
`int newNum = 10*i;`

- وفي هذا المثال اولاً نقوم بزيادة الـ `i` و ثم نحسب المعادلة

```

public class Increment {
    public static void main(String[] args) {
        int c;
        c = 5;
        System.out.println( c ); // print 5
        System.out.println( c++ ); // print 5
        System.out.println( c ); // print 6
        System.out.println(); // skip a line
        c = 5;
        System.out.println( c ); // print 5
        System.out.println( ++c ); // print 6
        System.out.println( c ); // print 6
    }
}

```

Output

5
5
6

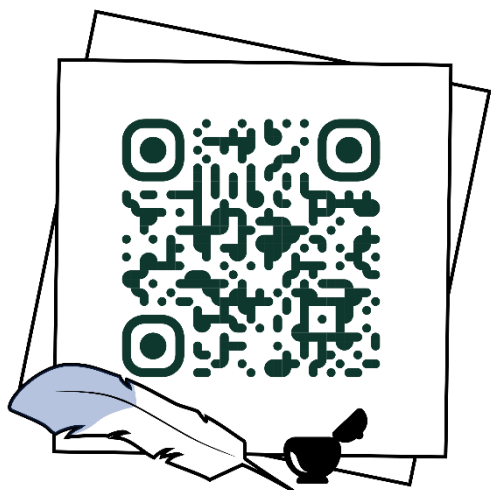
5
6
6



ALZEEKA Tutorial

شروحات - مشاريع - خدمات - تصميم

إنضم الآن



053 359 7191

