Tutorial 2
CSC 201

Java Programming Concepts

مبادئ البرمجة باستخدام الجافا

Chapter 2

1. Variables

١- المتغيرات

Definition Data types Primitive Data Types
Declaration of a variable Variable names

2. Literals

٢-القيم الحرفية

What is a literal? Where can it be used?
What are possible types of literals?
Looking closer at literals

1. Integer Literals
2. Floating-Point Literals
3. Boolean Literal
4. Character Literals
5. String Literals

3. Expressions and operators

٣-التعابير والمعاملات

Arithmetic Operators Comparison Operators
Increment and decrement operator Logical operators

Variables

المتغير: هو اسم يعطى لمكان في الذاكرة ويكون مرتبطا بعنوانه لتخزين البيانات

- ***Definition:*** it is a name for a location in main memory to store data.

Data types

أنواع البيانات: تحدد القيم التي يمكن ان يأخذها المتغير والعمليات التي يمكن إجراؤها عليه

A variable's data type determines:

- The values that the variable can contain
- The operations that can be performed on it.

For example: **int x;** the integer data type (int). Integers can contain only integral values (both positive and negative). You can perform arithmetic operations, such as addition, on integer variables.

متغير x نوعه عدد صحيح يحتوي قيم صحيحة موجبة او سالبة ويمكن إجراء عمليات حسابية عليه (جمع – طرح – ضرب – قسمة)

Variables

Primitive Data types:

أنواع البيانات الأساسية

Keyword	Description	Size/Format
(integers)		
byte	Byte-length integer	8-bit two's complement
short	Short integer	16-bit two's complement
int	Integer	32-bit two's complement
long	Long integer	64-bit two's complement
(real numbers)		
float	Single-precision floating point	32-bit IEEE 754
double	Double-precision floating point	64-bit IEEE 754
(other types)		
char	A single character	16-bit Unicode character
boolean	A boolean value (true or false)	true or false

Variables

Declaration of a variable:

الإعلان عن متغير: (نوعه – اسمه – قيمته المبدئية)

Example:

```
class Example1
{
        public static void main ( String[] args )
        {
           long payAmount = 123; //a declaration of a variable

           System.out.println("The variable contains: " + payAmount );
        }
}
```

There are several ways to declare variables:

- **dataType variableName;**
- **dataType variableName1, variableName2;**
- **dataType variableName = initialValue ;**
- **dataType variableName1 = initialValue1, variableName2 = initialValue2 ;**

Variables

Variable names: (Identifier)

اسم المتغير – المعرف

It must be:

- only from characters 'a' through 'z', 'A' through 'Z', '0' through '9', character '_', and character '$'.
- a legal identifier: begins with a letter, is unique within its scope.

It should not:

- be Keyword, e.g. public, void, static, main, null
- be Boolean literal (true or false)
- contain the space character.
- start with a digit.

شروط اسم المتغير الصحيح:

١- يتكون فقط من الحروف (a-z, A-Z, 0-9, _, $)

٢- يبدأ بحرف ابجدي

٣- لا يكون من الكلمات الخاصة باللغة

٤- لا يحتوي على مسافات أو حروف غير المذكورة

By Convention: متفق على ان:

- Variable names begin with a lowercase letter اسم المتغير يبدا بحرف صغير
- Class names begin with an uppercase letter. اسم الفئة يبدا بحرف كبير
- If a variable name consists of more than one word, the words are joined together, and each word after the first begins with an uppercase letter, like this: isVisible.
- The underscore character (_) is acceptable anywhere in a name, but by convention is used only to separate words in constants (because constants are all caps by convention and thus cannot be case-delimited).

اسم المتغير المكون من كلمتين تبدا الكلمة الثانية بحرف كبير – القيم الثابتة تكتب اسماؤها بأحرف كبيرة وتستخدم (_) لتفصل كلمتين في اسم الثابت

Variables

أي من تعريفات المتغيرات التالية صحيح؟

Example: Which of the following variable declarations are correct?

1. long good-by ;
2. short shrift = 0;
3. double bubble = 0, toil= 9, trouble = 8
4. byte the bullet ;
5. int double;
6. char thisMustBeTooLong ;
7. int 8ball;
8. int number;
9. float number2;
10. double amount_of_sale;
11. double $amount;
12. int amount of sale;
13. double &amount;
14. char item#;

Variables

في المثال التالي: يمكن تعريف المتغير من نوع حرف وإعطاؤه قيمة مبدئية حرف معين أو كود الحرف

Examples:

In the following example, the character can be declared by number which represents the ASCII Code of the letter or by the letter itself

```
public class Example2
{
    public static void main(String args[])
    {
       char ch1,ch2;
       String s;
       ch1=88; //ASCII Code for 'X'
       ch2='X';
       s=”hello”;
       System.out.println("ch1=" + ch1);
       System.out.println(“ch2 =” + ch2);
       System.out.println(“s =” + s);
    }
}
```

Literals

القيم الحرفية

What is a literal?

رمز أو حرف يمثل قيمة محددة

A letter or symbol that stands for itself as opposed to a feature, function, or entity associated with it.

Examples

10	a literal that specifies an integer
3.14	specifies a floating point value
true	specifies a boolean value
'X'	specifies a character constant
"Hello"	specifies a string

What are possible types of literals?

1. Number Literals
   a. Integer Literals 1 8 24
   b. Floating-Point Literals 2.0 8.2 0.99 3.14 ≡ 314159 E-05
2. Boolean Literals true false don't have numerical representation ≠ 1 , ≠ 0
3. Character Literals 'a' 'z' '1' '?' '@' '\n' '\t' '\' ' '\" '
4. String Literals "abc" "Hello World!"

Literals

مثال آخر قم بتجربته

Now let’s have another example, try it by copying and pasting:

```
class Example3
{
  public static void main ( String[] args )
  {

    long hoursWorked = 40;
    double payRate = 10.0, taxRate = 0.10;

    System.out.println("Hours Worked: " + hoursWorked );
    System.out.println("pay Amount : " + (hoursWorked * payRate) );
    System.out.println("tax Amount : " + (hoursWorked * payRate * taxRate) );
  }
}
```

The program will print out:

Hours Worked: 40
pay Amount : 400.0
tax Amount : 40.0

Expressions and Operators

1- Arithmetic operators

المعاملات الحسابية

Operator	Use	Description
+	op1 + op2	**Adds op1 and op2**
-	op1 - op2	**Subtracts op2 from op1**
*	op1 * op2	**Multiplies op1 by op2**
/	op1 / op2	**Divides op1 by op2**
%	op1 % op2	**computes the remainder of dividing op1 by op2**
+	+op	**Promotes + to int if its a byte,short or char**
-	-op	**Arithmetically negates op**
++	op++	**Increments op by 1; evaluates to value before incrementing**
++	++op	**Increments op by 1; evaluates to value after incrementing**
--	--op	**Decrements op by 1; evaluates to value before decrementing**
--	op--	**Decrements op by 1; evaluates to value after decrementing**

Expressions and Operators

1- Arithmetic operators

- **% modulus**

It can be applied to floating-point types as well as integer types.

Example:

```
class Example4
{
    public static void main(String args[])
    {
        int x = 42;
        double y = 42.3;
        System.out.println("x mod 10 = " + x % 10);
        System.out.println("y mod 10 = " + y % 10);
    }
}
```

When you run this program you will get following output:

x mod 10 = 2

y mod 10 = 2.299999999999997

Expressions and Operators

1- Arithmetic operators

- *** has purpose to multiply one operand by another**

Example:

b = b * 3; // can be written as b *= 3;

- **+ : addition**

if its operand are numbers then it will make normal addition, but if its operands are strings then it will make concatenation.

Example:

```
public class name
{
   public static void main (String args[])
   {
     //here the operands are strings thus concatenation
     //will be done
       System.out.println("Smith" + "Rachel");
    }
}
```

Expressions and Operators

1- Arithmetic operators

Example:

```
class example5
{
  public static void main(String args[])
  {
    System.out.println("Integer Arithmetic");
    int a = 1 + 1;
    int b = a * 3;
    int c = b / 4;
    int d = c - a;
    int e = -d;
    System.out.println("a = " + a);
    System.out.println("b = " + b);
    System.out.println("c = " + c);
    System.out.println("d = " + d);
    System.out.println("e = " + e);
  }
}
```

When u run this program u will see the following output

a = 2
b = 6
c = 1
d = -1
e = 1

Expressions and Operators

1- Arithmetic operators

Example to show how to get the division and quotient and remainder:

```
class div
{
   public static void main(String args[])
   {
      int x=5,y=2,quotient,remainder;
      double a=5,b=2;
      double division;
      quotient=x/y;
      remainder=x%y;
      division=a/b;
      System.out.println("quotient =" + quotient);
      System.out.println("remainder=" + remainder);
      System.out.println("division=" + division);
   }
}
```

2- Comparison Operators

معاملات المقارنة

Operators	Description
==	Equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
!=	Not equal to

Expression	Result
3 + 4 == 7	true
3 + 4 != 7	false
3 + 4 != 2 + 6	true
3 + 4 < 10	true
3 + 4 <= 10	true
3 + 4 == 4 + 4	false
3 + 4 > 10	false
3 + 4 >= 7	true
3 + 4 >= 8	false

3- Increment and Decrement Operators

Increment operators

معاملات زيادة القيمة بـ ١

example expression	equivalent longer expression
a++;	a=a+1;
b=a++;	b=a; a=a+1;
++a;	a=a+1;
b=++a;	a=a+1; b=a;

Decrement operators

معاملات إنقاص القيمة بـ ١

example expression	equivalent longer expression
a--;	a=a-1;
b=a--;	b=a; a=a-1;
--a;	a=a-1;
b=--a;	a=a-1; b=a;

Note: the increment operators Always take a variable never take a numerical value.
Example
++a; //is correct syntax.
++1; //will give an error.

3- Increment and Decrement Operators

```
class Example6
{
   public static void main(String args[])
   {
     int a = 23;
     int b = 5;

       System.out.println("a & b : " + a + " " + b);
       a += 30;
       b *= 5;
       System.out.println("after arithmetic assignment a & b: "+a+" "+b);
       a++;
       b--;
       System.out.println("after increment & decrement a & b: "+a+" "+b);
   }
}
```

The output of this program follows

a & b : 23 5

after arithmetic assignment a & b : 53 25

after increment & decrement a & b : 54 24

4- Logical Operators

المعاملات المنطقية

Operator	Result
&&	AND
&	Logical AND
||	OR
|	Logical OR
!	Logical unary NOT
==	Equals to
!=	Not Equals to
^	XOR

X	Y	X && Y
true	true	true
true	false	false
false	true	false
false	false	false

X	Y	X || Y
true	true	true
true	false	true
false	true	true
false	false	false

X	Y	X ^ Y
true	true	false
true	false	true
false	true	true
false	false	false

X	!X
true	false
false	true

4- Logical Operators

To take a deeper look on how this can affect the code

```
class test
{
    public static void main(String args[])
    {
      int count = 0;
      int total = 345;
      boolean b;
      b=count > 0 && total / count > 80;
      System.out.println("b=" + b);
    }
}
```

There is a problem here.
The method that follows && sets b as false while the other expression should get an error.

Similarly with || and | operators:
Once you know that the first subexpression is true there is no need to go further.
true OR anything is true.

1. To rearrange the code again as follows:

```
class test
{
  public static void main(String args[])
  {
    int count = 0;
    int total = 345;
    boolean b;
    b=total / count > 80&&count > 0 ;
    System.out.println("b=" + b);
  }
}
```

2. Or to use & operator

```
class test
{
    public static void main(String args[])
   {
      int count = 0;
      int total = 345;
      boolean b;
      b=count > 0 & total / count > 80;
      System.out.println("b=" + b);
   }
}
```

4- Logical Operators

Precedence of Logical Operators

Operator	precedence
!	High
&&	Medium
||	Low

For example:

A || B && C	means	A || (B && C)
A && B || C && D	means	(A && B) || (C && D)
A && B && C || D	means	((A && B) && C) || D
!A && B || C	means	((!A) && B) || C

Example

```
class example7
{
    public static void main(String args[])
    {
        boolean b;
        b = (2 > 3) && (3 < 2);
        System.out.println("b = "+b);
        b = false || true ;
        System.out.println("b = "+b);
    }
}
```

The output of the program is shown here.

b = false
b = true

Expressions and Operators

Operator Precedence

أسبقية المعاملات (ترتيب تنفيذها)

Precedence of the most important operators in Java
They are ordered from the highest to the, lowest:

Operators	Comments
()	Anything in parentheses
++ -- !	Unary operators
* / %	Multiplicative operators from left to right
+ -	Additive binary operators from left to right
> >= < <=	Relational comparison operators
== !=	Identity comparison operators
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
&&	Logical AND
||	Logical OR
= += *= -= /=	Assignment