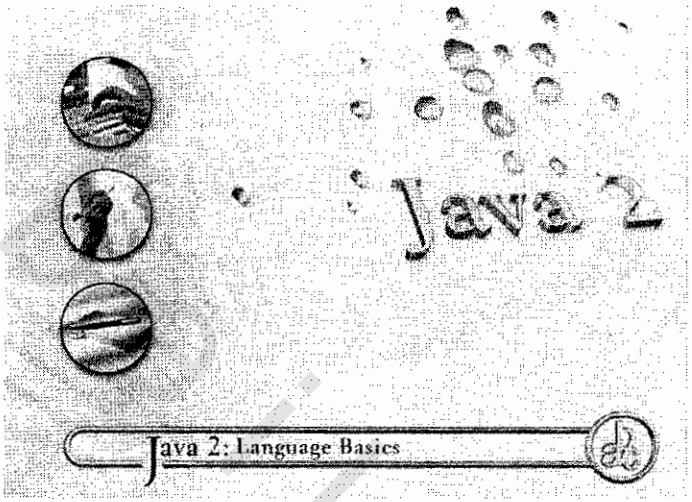
الفصل الثاني



أنواع البيانات في لغة الجافا
Data Types in Java

فى هذا الفصل سوف نستعرض أساسيات اللغة حيث نتناول موضوع أنواع المتغيرات والثوابت والمؤثرات Operators وذلك من خلال النقاط التالية:

- المتغيرات Variables.
- أنواع البيانات Data Types.
- البيانات المعرفة فى أصل اللغة Primitive Data Types.
- الإعلان عن متغير Variable Declaration.
- الجمل Statements.
- التعبيرات Expressions.
- التعامل مع الأنواع المختلفة للبيانات.
- التعليقات Comments.
- المؤثرات Operators.
- أنواع المؤثرات Operators Types.
- أسبقية التعامل مع المؤثرات Operator Precedence.
- التحوير Casting.

openkan.com

obeikan.com

المتغيرات Variables:

- المتغيرات هى مكان نستطيع تخزين قيمة فيه أثناء عمل البرنامج ، وهذا المتغير نستطيع تغيير قيمته فى أي وقت.
- ونستطيع تخيل المتغير أنه عبارة عن وعاء لتخزين قيمة ما ، ونستطيع تغيير هذه القيمة وقتما نريد. والقيمة المخزونة فى هذا المتغير تتوقف على نوع المتغير ، فتعال معى نتعرف على أنواع البيانات.

أنواع البيانات Data Types:

تنقسم أنواع البيانات فى لغة Java إلى فئتين رئيستين:

أ- بيانات مبنية ومعرفة فى أصل اللغة Primitive Data Types.

ب- بيانات يقوم المبرمج ببنائها User Defined Data Types.

البيانات المعرفة فى أصل اللغة Primitive Data Types:

وكما قلنا فهى البيانات التى تأتى مبنية فى أصل اللغة ، وهى تنقسم إلى أربعة أنواع:

1- البيانات الرقمية الصحيحة Integer Literals مثل (1 ,155 ,10 ,5- ,1-).

2- البيانات الرقمية ذات الفصلة العشرية Floating-Point Literals مثل (0.55, 1.33).

3- البيانات الحرفية Character Literals مثل 'A', 'B', 'X'.

4- البيانات المنطقية Boolean Literals مثل (false, true).

ولكل نوع من الأنواع السابقة نطاق ومساحة يحتلها فى الذاكرة كما فى الجدول التالى حيث يتضح فيه نوع البيانات Data Type والمساحة التى يحتلها فى الذاكرة وأيضاً النطاق الرقمى الذى يستطيع كل نوع من هذه الأنواع استيعابه.

Range النطاق		المساحة فى الذاكرة	النوع Data Type
إلى	من		
البيانات الرقمية الصحيحة Integer Literals			
$2^7 - 1$	-2^7	8 bits	byte
$2^{15} - 1$	-2^{15}	16 bits	Short
$2^{31} - 1$	-2^{31}	32 bits	Int
$2^{63} - 1$	-2^{63}	64 bits	Long
البيانات الرقمية ذات الفصلة العشرية Floating-Point Literals			
3.4×10^{38}	-3.4×10^{-38}	32 bits	float
1.7×10^{308}	-1.8×10^{-324}	64 bits	Double
البيانات الحرفية Character Literals			
$2^{16} - 1$	0	16 bits	char
البيانات المنطقية Boolean Literals			
false	true	1 bit	boolean

النطاق والمساحة التى يحتلها أى متغير مذكور فى الجدول السابق لا تختلف باختلاف نظام التشغيل Operating System كما فى اللغات الأخرى

الإعلان عن متغير Variable Declaration:

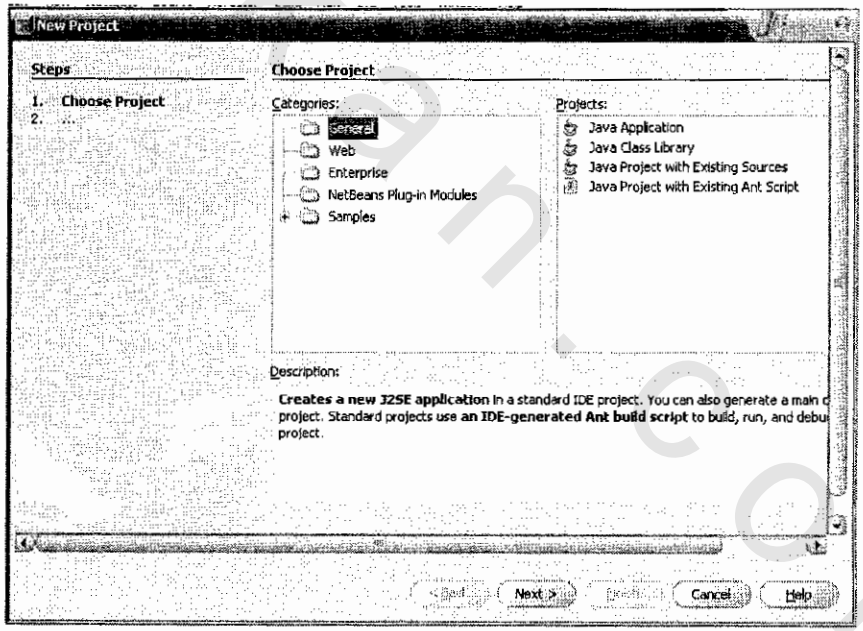
- يتم الإعلان عن المتغير بكتابة نوع بيانات المتغير ثم اسم المتغير مع إمكانية إعطائه قيم مبدئية Initial Values.
- ولتسمية متغير ما فإن لك حرية تسمية المتغير على ألا يبدأ الاسم برقم ولا يكون بين حروفه مسافة أو قوس أو الحروف الخاصة مثل (@و#و...) ، وفيما يلى أمثلة للإعلان عن المتغيرات.

```
int salary = 500;
boolean male = false;
float age=35.6F;
char c='k';
```

- وكما ترى فى الأمثلة السابقة ، أنه لكى نقوم بالإعلان عن متغير مثل salary ، فإننا نسبق اسم المتغير بالنوع الذى ينتمي إليه – وهو النوع int- ثم يليه اسم المتغير salary ، ويليه قيمة المتغير - وهى 500- وذلك عن طريق المؤثر "=" ، والمتغير salary يحتل مساحة فى الذاكرة قدرها 32 بت لأنه من نوع int كما هو واضح فى الجدول السابق.
- ويظهر ذلك فى برنامج متكامل كما يلي فى الخطوات التالية.

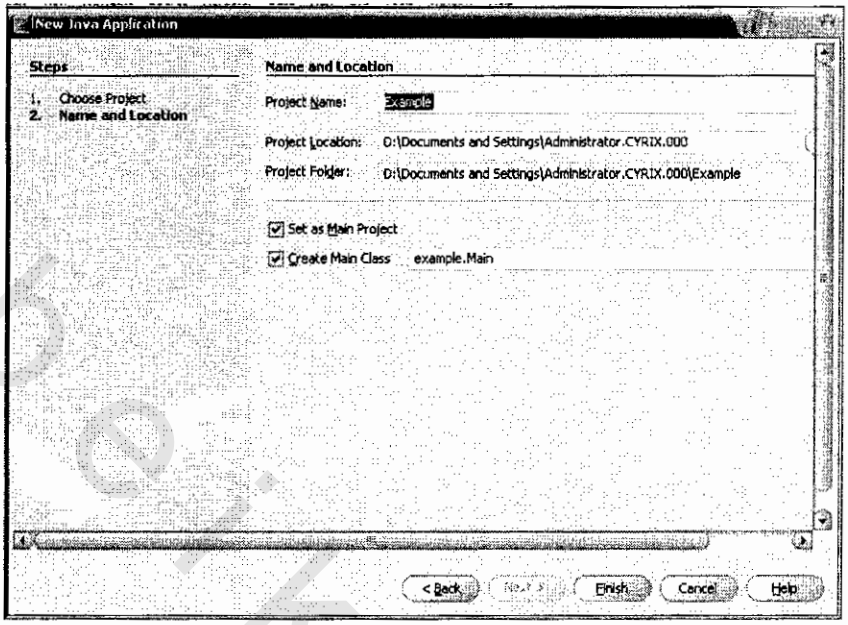
مثال (1): المتغيرات Variables:

- قم بتشغيل البرنامج Net Beans وقم بإعداد تطبيق جديد كما فى الشكل (2-1).



(الشكل 2-1)

- فى هذا الشكل يتم تحديد نوع التطبيق وهو النوع الأول Java Application.
- اضغط الزر Next ليتم الانتقال إلى الشاشة كما فى الشكل (2-2).



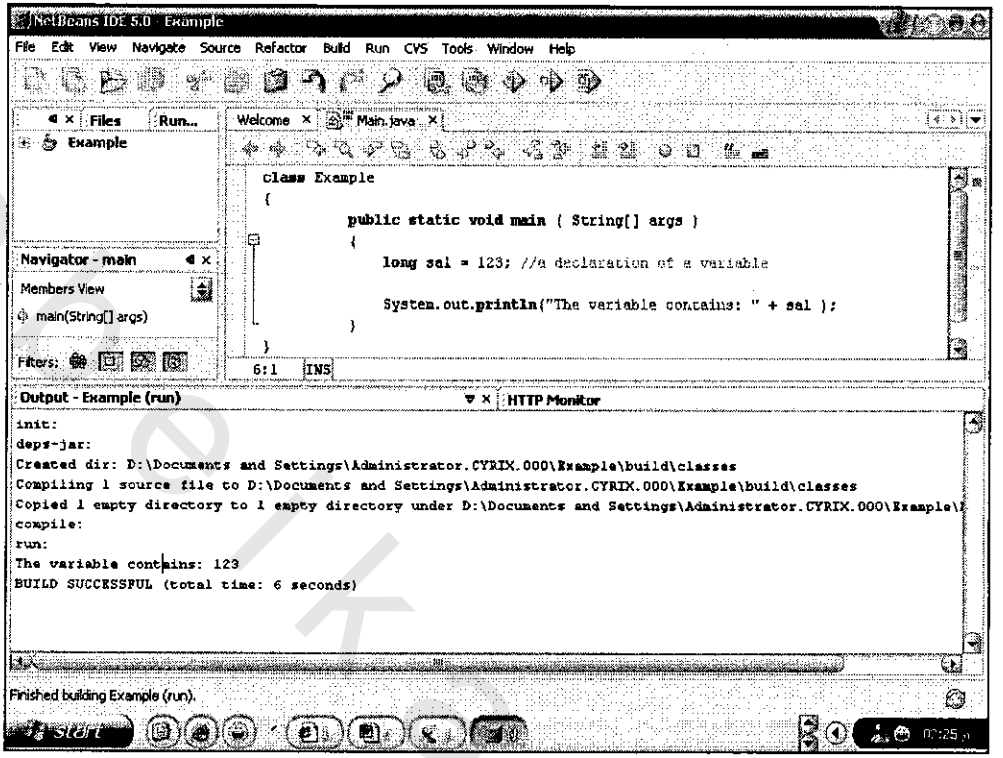
(الشكل 2-2)

- اضغط الزر Finish لتحصل على شاشة سطور الأوامر.
- قم بالتعديل في سطور البرنامج كما في السطور التالية:

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

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

- قم بتنفيذ البرنامج بالضغط على الزر F6 لتحصل على نتيجة التنفيذ أسفل الشاشة الرئيسية للبرنامج كما فى الشكل (2-3).



(الشكل 2-3)

كما تجدر الإشارة إلى أنه توجد عدة طرق للإعلان عن المتغير كما فى الصيغ التالية:

- dataType variableName;
- dataType variableNameOne, variableNameTwo;
- dataType variableName = initialValue;
- dataType variableNameOne = initialValueOne, variableNameTwo = initialValueTwo;

الجمل Statements:

- الجملة Statement هى أمر بسيط مكتوب بلغة برمجة (فى حالتنا مكتوب بلغة Java) وتؤدى كتابة الجملة Statement إلي حدوث شئ في البرنامج.
- وكمثال للجمل Statements:

```
int x = 10;
System.out.println ("Welcome to Java");
emp1.salary = 500;
```

- وهناك بعض الجمل Statements يؤدي تنفيذها إلي إنتاج قيمة ما ، فمثلاً عندما تجمع رقمين فى البرنامج ، فهذا يؤدي إلي حساب ناتج الجمع.
- هذا النوع من الجمل تسمى تعبير Expression.

التعبيرات Expressions:

- التعبير Expression هو أي جملة تتكون من أوامر اللغة وتؤدي إلي حساب قيمة ما فيما عدا جمل مساواة متغير بقيمة معينة ، ففي هذه الحالة لا يتم حساب قيمة من جملة المساواة.
- وهذه القيمة المحسوبة من التعبير Expression نستطيع تخزينها لاستخدامها فى البرنامج فى وقت لاحق.
- وكمثال للتعبير Expression:

```
int x =5;       // Statement
int y =10;      // Statement
int z =x+y;     // Expression
```

التعامل مع الأنواع المختلفة للبيانات:

والآن لنوضح كيفية التعامل مع الأنواع المختلفة من البيانات.

البيانات الرقمية الصحيحة Integer Literals:

- تعلمنا في النقاط السابقة أنه توجد أربع أنواع للبيانات الرقمية الصحيحة Integer Literals وهم byte و short و int و long.
- ويتوقف اختيارك لنوع البيانات Data Type من هذه الأربعة أنواع على نطاق القيمة التى تمثلها ، وكلما كانت القيمة صغيرة كلما كان من الأفضل اختيار النوع ذو النطاق الأقل كلما أمكن وذلك توفيراً لمساحة الذاكرة.

```
byte x = -120;
short y=1500;
int x=101111;
long = 9.2 x 1010;
```

البيانات الرقمية ذات الفصلة العشرية Floating-Point Literals:

- بينما تستطيع البيانات الرقمية الصحيحة Integer Literals تمثيل القيم الصحيحة فقط ، فإن البيانات ذات الفصلة العشرية Floating-Point Literals تستطيع تمثيل القيمة التى تحتوى على قيم صحيحة وكسور مثل 123.456 - ,39.99.
- ولغة Java تشتمل على نوعين من بيانات الفصلة العشرية هما float & double ، واستخدام كل نوع يتوقف على الدقة التى تريدها ، فكيف ذلك؟
- إن استخدام البيانات الرقمية ذات الفصلة العشرية Floating-Point Literals يزداد تعقيداً فى عمليات القسمة والضرب ، فمثلاً لو أردنا إجراء عملية القسمة التالية $\frac{10}{3}$ ، فستجد أن النتيجة هي 3.333 ، ولكن فى الواقع – إذا تحرينا الدقة– فستجد عمليات القسمة مستمرة والجزء العشرى فى الرقم يزداد ويستمر إلى ما لا نهاية.
- إذاً نتيجة قسمة $\frac{10}{3}$ قد تكون 3.3333 أو 3.33333333 أو 3.33333333333333 أو دقة أكثر فتكون 3.33333... ، إذاً فكلما ازداد الرقم 3 فى الجزء العشرى كلما زادت دقة النتيجة ، والدقة المطلقة هى أن نضيف العدد 3 إلى ما لا نهاية فى الجزء العشرى.
- وللتعامل مع الرقم السابق ، يجب أن نقرر قبل استخدام أى نوع من بيانات الفصلة العشرية Floating-Point Literals كم رقم نريده فى الجزء العشرى ، فإذا كان الرقم الذى نريده يقع فى نطاق float فنختار float ، أما إذا كان أكبر من ذلك فنختار double.
- تذكر أن قيم نطاق النوع float هى $3.402823x10^{38}$ – إلي 3.402823×10^{38}.
- والقيم التى تقع بين هاتين القيمتين تسمى أحادية الدقة Single-Precision ، ولكى أوضح أن الرقم يقع فى هذا النطاق ، فإن الإعلان عن المتغير من هذه الفئة يكون كالتالى.

```
float x = 105.1095F;
```

- وذلك بأن نضيف حرف F إلى نهاية الرقم ، كما يمكن تمثيل السطر السابق بالطريقة التالية أيضاً.

```
float x= 1.051095e2F;
```

- ومعنى e2 أى هي ضرب الرقم في 10^2.
- أما القيم التى تقع فى النطاق $\approx -1.8 \times 10^{308}$ to 1.8×10^{308} ، فتسمى بالدقة المضاعفة Double-Precision.
- ولكى أعلن عن متغير من النوع double ، فإن الكود يكون كالتالى

```
double num=3.14D;
```

- وكما قلنا فى استخدام المتغيرات من النوع int ، أنه كلما كانت القيم التى يمثلها المتغير تقع فى نطاق أقل ، فالأفضل استخدام نوع البيانات الواقع فى النطاق الأقل وليس الأعلى توفيراً لمساحة الذاكرة.

البيانات الحرفية Character Literals:

- عادة تحتاج فى برامجك إلى طريقة لتمثيل الحروف characters وليس فقط الأرقام ، والحرفيات Characters هى أى رمز يستخدم فى الكتابة ، ومن أشهرها الحروف الأبجدية B,A Capital and Small ، كما تعد الأرقام 9-0 أيضاً من الحروف ، وأيضاً علامات التعجب والمفاتيح الخاصة الموجودة فى لوحة المفاتيح Keyboard يمكن أيضاً اعتبارها حروف.
- ولكى نقوم بالإعلان عن متغير من نوع حرف char ، فإن الكود يكون كالتالى.

```
char x = 'a';
```

- فى السطر السابق قمنا بالإعلان عن متغير من نوع char ، وخصصنا له القيمة a مع ملاحظة أنه يجب وضع القيمة بين فصلة أحادية Single Quote.
- وهناك بعض الحروف الخاصة التى يتم تمثيلها بشكل مختلف عن الحروف العادية مثل تمثيل زر الجدولة tab أو زر الإدخال Enter ، والجدول التالى يبين لنا كيفية تمثيل

بعض من هذه الحروف الخاصة.

الحرف Character	التمثيل
Back Slash	'\\'
Back Space	b'\'
Carriage return	'\r
double quote	' \" '
Form Feed	' \f '
Line Feed	'\n'
Single quote	' \' '
Tab	'\t'

مثال:

```
char b= '\b';
```

البيانات المنطقية Boolean Literals:

- فى أحيان كثيرة نحتاج لتحديد قيمة من اثنين إما نعم أو لا ، وفى هذه الحالة نستطيع استخدام النوع boolean.
- وكمثال

```
boolean  Fileopen = true;
```

وعموماً تستخدم البيانات من نوع boolean مع الجمل الشرطية التى ستتحدث عنها لاحقاً.

جميع أسماء أنواع المتغيرات تكتب بحروف صغيرة small كالتالي

int, float, long, double, short, byte, char, boolean

الثوابت Constants:

- الثابت Constant هو قيمة لا تتغير أبداً أثناء عمل البرنامج ، وذلك على العكس من المتغير الذى نستطيع تغيير قيمته باستمرار أثناء عمل البرنامج ، ولكى نقوم بتعريف ثابت ما ، فإننا نستخدم الكلمة final ، ويكون التعريف كالآتى:

```
final double pi=3.14;
```

- فكلمة final تجعل المترجم Compiler يقوم بحجز مكان خاص فى الذاكرة لهذا النوع ولا يسمح بتغيير قيمته.

التعليقات Comments:

- التعليق هو نص لا يلتفت إليه المترجم أثناء عملية الترجمة Compilation ، وتعد كتابة التعليقات من المهام الهامة جداً لأننا فى بعض الأحيان ننسى ونريد أن نتذكر كيفية بناء وعمل جزء معين من الكود وعلاقته بباقى الكود ، أو عندما يريد أخرون الاطلاع على الكود ويريدون أن يفهموا أجزاء البرنامج.
- وهناك نوعان من التعليق هما:

1- تعليق السطر الواحد:

لكتابة سطر واحد من التعليق ، لابد أن يسبق هذا السطر علامتان من الشرطة المائلة // كالآتى.

```
//this is single line comment
```

2- تعليق مكون من سطور متعددة:

لكتابة تعليق مكون من سطور متعددة ، نبدأ الجزء الذي نريده كتعليق بالعلامة /* وننتهى بالعلامة */ كالآتى:

```
/* This is a Multiple Line Comment
hi my students */
```

بيانات من نوع الفصائل Class Types:

- بالإضافة إلى أنواع البيانات الثمانية السابقة ، فيوجد أنواع من البيانات مبنية فى لغة Java تسمي بالفصائل classes ، ويمكننا التعامل معها على أنها أنواع من المتغيرات وفى نفس الوقت يمكننا التعامل معها على أنها فصائل classes مثل الفصائل String و Color.
- فمثلاً يمكننا التعامل مع الفصيلة String كأنها نوع من أنواع البيانات كالتالى:

```
String name = "Ahmed";
```

- ويمكننا التعامل معها على أنها فصيلة class حيث يمكننا أن نقوم بإنشاء هدف object - كما سنتعلم فى الفصول القادمة - كالتالى:

```
String str=new String();
str.trim();
```

- فى هذا المثال قمنا بإنشاء هدف object ثم قمنا باستدعاء دالة Method من هذه الفصيلة class ، وهذا النوع من البيانات ستتعرف عليه بالتدريج في الفصول القادمة.

بيانات يقوم المستخدم ببنائها User Defined Data Types:

- كما سنتعرف فى فصل البرمجة بالأهداف OOP، فإنه يمكننا إنشاء فصيلة class ، وفي هذه الحالة تعتبر هذه الفصيلة class نوع من أنواع البيانات التى ينشئها المبرمج.

المؤثرات Operators:

- المؤثر Operator هو رمز خاص يستخدم فى العمليات الحسابية والمنطقية التى تتم على المتغيرات.

أنواع المؤثرات Operators Types:

تتوافر في لغة Java العديد من أنواع المؤثرات Operators هي:

1. المساواة Assignment.
2. الحسابي Arithmetic.
3. النصي String.
4. المنطقي Logical.
5. المقارنة (العلائقي) Comparison (Relational).
6. الشرطي Conditional.
7. الأحادي Unary.
8. المؤثرات على مستوى البت Bit-wise Operators.

الجدول التالي يوضح أنواع المؤثرات Operators المختلفة ووظيفتها.

المؤثر Operator	النوع	الوظيفة	مثال
=	المساواة Assignment	مساواة قيمة الطرف الأيسر بالطرف الأيمن	a = 5
+=	المساواة Assignment	جمع الرقمين ووضع الناتج في الطرف الأيسر	a += 10
=-	المساواة Assignment	طرح الرقمين ووضع الناتج في الطرف الأيسر	a -= 10
*=	المساواة Assignment	ضرب الرقمين ووضع الناتج في الطرف الأيسر	a *= 10
/=	المساواة Assignment	قسمة الرقمين ووضع الناتج في الطرف الأيسر	a /= 10
%=	المساواة Assignment	قسمة الرقمين ووضع باقي القسمة في الطرف الأيسر	a %= 10
+	الحسابي Arithmetic	جمع الرقمين	a + 10
-	الحسابي Arithmetic	طرح الرقمين	a - 10
*	الحسابي Arithmetic	ضرب الرقمين	a * 10
/	الحسابي Arithmetic	قسمة الرقمين	a / 10
%	الحسابي Arithmetic	إيجاد باقي قسمة الرقمين	a % 10

المؤثر Operator	النوع	الوظيفة	مثال
&&	Logical المنطقي	تنتج القيمة true إذا تحقق كلا الشرطين	a == 5 && b == 10
||	Logical المنطقي	تنتج القيمة true إذا تحقق أحد الشرطين	a == 5 || b == 10
!	Logical المنطقي	تقوم بعكس القيمة true إلي false والقيمة false إلي true	!a
==	Logical المنطقي	تنتج القيمة true إذا تساوي كلا الطرفين	a == 5
!=	Comparison المقارنة	تنتج القيمة true إذا لم يتساوي كلا الطرفين	a != 5
>	Comparison المقارنة	تنتج القيمة true إذا كان الطرف الأيسر أكبر من الطرف الأيمن	a > 5
<	Comparison المقارنة	تنتج القيمة true إذا كان الطرف الأيسر أصغر من الطرف الأيمن	a < 5
>=	Comparison المقارنة	تنتج القيمة true إذا كان الطرف الأيسر أكبر من أو يساوي الطرف الأيمن	a >= 5

المؤثر Operator	النوع	الوظيفة	مثال
<=	المقارنة Comparison	تنتج القيمة true إذا كان الطرف الأيسر أصغر من أو يساوي الطرف الأيمن	a <= 5
+	النصي String	تقوم بعمل وصل للنصوص Concatenation	"mostafa" + "maged"
(condition) ? val1 : val2	الشرطي Conditional	تنتج قيمتين مختلفتين بناء علي الشرط الموجود	y = (x > 5) ? 3 : 4
++	الأحادي Unary	تقوم بزيادة قيمة المتغير بواحد	a++ ++a
-	الأحادي Unary	تقوم بعكس إشارة المتغير	-a
--	الأحادي Unary	تقوم بإنقاص قيمة المتغير بواحد	a-- --a

المؤثرات الحسابية Arithmetic Operators:

المؤثرات الحسابية هى (*,+,-,/,%,++,--)

أمثلة على المؤثرات الحسابية:

المؤثر (+): وهو يقوم بعملية الجمع.

```
int a = 10;
int b =20;
int Sum;
Sum = a + b; // Sum =30
```

المؤثر (-): وهو يقوم بعملية الطرح

```
int salary = 500;
int deduction = 85;
int netsalary;
netasalary = salary – deduction;
```

المؤثر (*): وهو يقوم بعملية الضرب

```
int Salary = 500;
int Bonus = 2* Salary;
```

المؤثر (/): وهو يقوم بعملية القسمة

```
int Salary = 500;
int halfsal =500/2;
```

المؤثر (%): وهو باقى القسمة Modulo

```
int x = 10% 3 ;                    // x = 1
```

المؤثر (++):

وهو يعمل على زيادة قيمة المتغير بواحد ، فمثلاً لكى نقوم بزيادة المتغير Count بقيمة 1 كنا نقوم بعمل الآتى:

```
int Count;
Count = Count +1;
```

ونستطيع فعل ذلك بطريقة مختصرة عن طريق (++)

```
Count ++;      // doing the Same as Count = Count+1
```

وتنطبق هذه الطريقة أيضاً على المؤثر (- -) الذى يعمل على نقصان قيمة المتغير بواحد.

أمثلة مختلفة على اختصارات المؤثرات Operators:

يقوم هذا المثال بإلاعلان عن متغير من نوع int ثم يقوم بإضافة 5 علي قيمته الأصلية:

```
int x=10;
x=x+5;
```

يمكننا فعل ذلك بطريقة مختصرة كالآتي:

```
x+=5;
```

وينطبق ذلك على بقية المؤثرات Operators:

```
x=x-5;
x-=5;
x=x*5;
x*=5;
x=x/3;
x/=3;
```

أسبقية التعامل مع المؤثرات Operator Precedence:

- والآن بعد أن تعرفنا على المؤثرات الحسابية تعال معى نتعرف على أسبقية معالجة المؤثرات من خلال هذا المثال.

```
int answer;
answer = 3+5*3;
```

- إذا قمت بحساب النتيجة لتكون 24 فأنت مخطئ.
- ولكي نفهم السبب ، فلا بد أن نسأل أنفسنا ما هو ترتيب تنفيذ العمليات السابقة ، هل الجمع أم الضرب؟ الحل بسيط ويخضع للجدول التالى الذى يرتب تنفيذ معالجة العمليات حيث يكون أول صف صاحب أعلي أسبقية في التنفيذ وهكذا بالترتيب.

Operator المؤثر	Description الوصف
( )	الأقواس
+ +	الزيادة بواحد Increment
– –	النقصان بواحد Decrement
*	الضرب Multiplication
/	القسمة Division
%	باقي القسمة Modulus
+	الجمع Addition
–	الطرح Subtraction
=	المساواة Assignment

- وبذلك تكون نتيجة تنفيذ المثال السابق هى 18 لأن عملية الضرب لها أسبقية فى التنفيذ عن الجمع ، أي أن عمليات الضرب تتم معالجتها أولاً.
- وإذا أردنا أن نجعل المترجم Compiler يقوم بعملية الجمع أولاً ، فإننا نقوم بعمل الآتي:

```
answer = (3+5) *3
```

- وهذا لأن الأقواس لها أسبقية عن الضرب.

التحوير Casting:

- إن عملية التحوير Casting هى عملية جعل المترجم Compiler يقوم بالتعامل مع نوع من البيانات على أنه نوع أخر ، وإليك هذا المثال لإظهار الحاجة إلى التحوير Casting.

```
int Result=0;
Result= 10-5-1.5f;
```

- فكما ترى فى المثال ، فإنه عندنا متغير اسمه Result من نوع int ونحاول القيام بعملية حسابية وتخصيص قيمتها لهذا المتغير ولكن هناك مشكلة!
- المشكلة هنا أن ناتج العمليات الحسابية هو (3.5) ، وكما ترى فإن هذه القيمة من نوع float وأنت تحاول تخصيصها لمتغير من نوع int ، ولو حاولت ذلك سوف يقوم المترجم Compiler بإصدار رسالة خطأ. إذن ما الحل؟
- الحل هو إجراء عملية تحوير Casting للقيمة Result أى أن نقوم بتحويل ناتج العملية الحسابية إلى نوع int وذلك كالأتى:

```
Result = (int) (10-5 – 1.5f);     //Result =3;
```

- فى هذا المثال نحدد للمترجم Compiler أن يقوم بالعمليات الحسابية ثم يقوم بمعاملة الناتج على أنه من نوع int ، وبذلك لن يقوم المترجم Compiler بإصدار أى رسالة خطأ، ولكن فى هذه الحالة سنفقد الجزء العشرى من الناتج ليصبح الناتج 3 بدلاً من 3.5.
- مثال أخر:

```
byte R2;
R2=20*20;
```

- يقوم المترجم Compiler بإصدار خطأ لأن الناتج أكبر من نطاق المتغير.
- الحل أن نقوم بكتابة الآتي:

```
R2= (byte) (20*20);  // R2=-112
```

- مثال أخر:

```
int x = 3/2
```

- هنا لا يقوم المترجم Compiler بإصدار رسالة خطأ وستكون قيمة الناتج 1 فلماذا؟
- هذا لأن الرقمين 2 و 3 هى قيم من نوع int ، والمترجم Compiler يقوم بإسقاط أى قيمة غير صحيحة من الناتج.
- ولكن ما الحل إذا أردنا الإجابة بشكل دقيق؟
- قد يكون الحل الذي تقترحه عزيزي القارئ أن تعدل نوع المتغير x كالآتى:

```
float x=3/2;
```

- ولكن إذا قمت بفعل ذلك سيظل الناتج 1 لأن المترجم Compiler يقوم بالعملية الحسابية على أساس أن 2, 3 هما من نوع int والحل هو كالأتى:

```
float x = 3f/2f  ;    //x=1.5;
```

قاعدة عامة فى عمليات التحوير Casting:

- نستطيع القول بأن المتغير ذا النطاق الأكبر يستطيع احتواء المتغير ذي النطاق الأقل بدون إجراء عمليات تحوير Casting.

```
byte x = 50;          // implicit Casting
short Z=x;
```

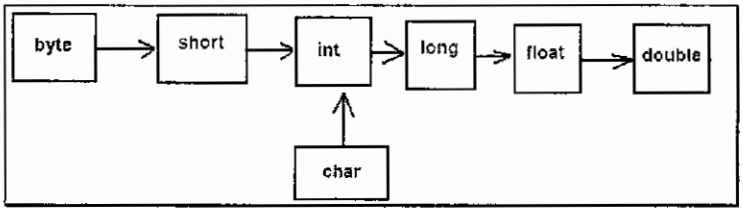
- فى هذا المثال لا نحتاج إلى عملية تحوير Casting لأن المتغير Z من نوع short وهو أكبر فى النطاق من المتغير x الذى ينتمى إلى النوع byte.

```
int i = 100;
short x = i;
```

- هنا يقوم المترجم Compiler بإصدار رسالة خطأ لأن نطاق المتغير i أكبر من نطاق المتغير x ، إذن فلا بد من إجراء عملية تحوير Casting.

```
short x = (short) i;     // Explicit Casting
```

والرسم التالي يوضح لنا بشكل أكبر هذه العملية.



هذا المخطط يوضح لنا أننا نستطيع تخصيص متغير من نوع byte إلى متغير من نوع short أو float بدون عملية تحوير Casting.

ونستطيع تمثيل المخطط السابق فى المثال الآتى:

```
byte a=20;
short b;
int c;
long d;
float e;
double f;
b=a;
c=b;
d=c;
E=d;
F=e;
```

فى هذه الحالات لا توجد مشكلة مع المترجم Compiler.

المؤثرات العلائقية Operators Relational :

وهذه المؤثرات Operators تقوم باختبار العلاقة بين متغيرين ، وهذه المؤثرات Operators هى:

المساواة (= =) وأقل من (<) وأكبر من (>) وأقل من أو يساوى (<=) وأكبر من أو يساوى (>=) ولا يساوى (!=).

المؤثران = و == والفرق بينهما:

المؤثر (=):

هذا المؤثر يقوم بتخصيص القيمة التى فى يمين العلاقة ووضعها فى المتغير الموجود فى يسار العلاقة ، فمثلاً:

```
int x = 5;
```

معنى هذا التعبير أن يتم وضع القيمة 5 فى المتغير x.

المؤثر (= =):

هذا المؤثر يقوم باختبار ما إذا كانت القيمة الموجودة فى يسار العلاقة تساوى الموجودة فى الناحية اليمنى أما لا ، فمثلاً:

```
if ( x == 5)
```

ستكون نتيجة هذه الجملة أما true أو false ، فإذا كانت النتيجة true نقوم بكتابة ما نود عمله فى هذه الحالة ، أما لو كانت النتيجة false فنقوم بكتابة ما نود عمله فى هذه الحالة أيضاً.

وبالمثل بالنسبة لباقى المؤثرات العلائقية Relational Operators فجميعها تقوم بإرجاع القيمة true أو false بناء على الشرط الذى تكتبه ، وعموماً فإن هذه المؤثرات Operators تستخدم مع الجمل الشرطية التى سنتعلمها فى الفصول القادمة.

المؤثرات المنطقية Logical Operators:

- المؤثرات المنطقية تمكننا من إجراء عمليات المقارنة بين أكثر من تعبير Expression.
- المؤثرات المنطقية هى (&& و || و !و ^) ، فتعال معى نفهم كل من هذه المؤثرات Operators.

المؤثر AND (&&):

- هذا المؤثر Operator يتطلب أن تكون جميع التعبيرات Expressions لها القيمة true لكى يكون ناتج المقارنة الكلي هو true كما يتضح فى المثال التالى.

```
int x=10;
int y=5;
int s=2;
int t=3;
if (x>y && t>s)
then do something
```

- هذا الشرط سوف يتحقق لأن كلاً من التعبيرين صحيح لأن x>y وفى نفس الوقت s<t.
- أما لو كان أحد الحدين غير صحيح false فإن الشرط بأكمله يصبح غير صحيح false كما فى المثال التالى:

```
(4+3 = = 9) && (3+3 = = 6)
```

- فعلى الرغم من أن الحد الأيمن صحيح true ، فإننا نجد أن الحد الأيسر غير صحيح false ، وبالتالى يكون ناتج الشرط false.

المؤثر OR (||):

- هذا المؤثر Operator يتطلب فقط أن تكون قيمة أحد الحدين true لكى يكون ناتج الشرط true ، فإذا كان كلا الحدين true فيكون ناتج الشرط أيضاً true ، أما إذا كان كلا الحدين false فيكون ناتج الشرط أيضاً false ، فمثلاً:

```
(3+6 = = 5) || (4+4 == 8)
```

- هذه العلاقة تكون نتيجتها true لأن أحد الطرفين يرجع true لأن 8 = 4+4.

المؤثر OR EXCLUSIVE (^):

- هذا المؤثر يستخدم لتحديد ما إذا كان التعبيرين Expressions مختلفين أم لا ، فإذا كان كلا الطرفين false أو true ، فإن ناتج العلاقة يكون false بسبب تساوي قيمة التعبيرين Expressions ، أما لو كانت قيمة أحد التعبيرين Expressions مختلفة عن الأخر ، فإن ناتج العلاقة يكون true بسبب عدم تساوي قيمة التعبيرين Expressions.
- وكمثال ، فإن التعبير Expression التالي تكون نتيجته true:

```
(5+7 ==12) ^ (4+3==8)
```

- لأن أحد الطرفين فقط هو الصحيح وهو (5+7 ==12).
- ولكن التعبير Expression التالي تكون نتيجته false:

```
(5+7 ==12) ^ (4+3==7)
```

- لأن كلا الحدين true ، كما أن العلاقة التالية تنتج false لأن كلا الحدين false:

```
(5+7==13) ^ (4+3==8)
```

المؤثر Not (!):

- يقوم هذا المؤثر Operator بعكس ناتج العلاقة ، فمثلاً (4+3==5) تكون نتيجتها false ولكن مع استخدام المؤثر ! كالتالي:

```
!(4+3==5)
```

- تصبح هذه العلاقة true لأن المؤثر (!) قد قام بعكس ناتج العلاقة.

أسبقية تنفيذ المؤثرات المنطقية Logical Operators والعلائقية Relational Operators:

- لقد تعرفنا على ما أسبقية معالجة العمليات فى المؤثرات الحسابية Arithmetic Operators ، والآن لنتعرف على أسبقية المعالجة مع المؤثرات المنطقية Logical Operators والعلائقية Relational Operators حيث يكون أول صف صاحب أعلي أسبقية في التنفيذ وهكذا بالترتيب.

المؤثر Operator	الوصف
!	Not
=> =<	Relational
== =	Equality
&&	Logical And
||	Logical OR

1. تعال معى لنرى أسبقية معالجة المؤثرات Operators من خلال المثال التالى:

```
(3<5) && (2==1) || (7==7)
```

2. والآن هل هذه العلاقة true أم false؟
3. الحل هو أن هذه العلاقة true ، ولكن لماذا!؟
4. والإجابة true لأن المؤثر && له أسبقية فى التنفيذ عن المؤثر || ، ولكى تحل هذا المثال بشكل صحيح ، تعال نضع أنفسنا مكان المترجم Compiler لنرى كيف سيقوم بالتنفيذ.

5. فى البداية سيقوم المترجم Compiler بتنفيذ العلاقة التى بها المؤثر && ويكون الشكل كالتالى:

```
((3<5) && (2==1)) || (7==7)
```

- ولأن ثلاثة أقل من 5 ، فالناتج سيكون true ولأن 2 لا تساوى 1 فالناتج سيكون false.

```
((true) && (false)) || (7==7)
```

- وبعد ذلك يجئ دور المؤثر && الذى يقوم بالمقارنة بين القيمتين true و false ، فتكون النتيجة false ويصبح شكل العلاقة كالتالى:

```
false||(7==7)
```

- ولأن العلاقة 7==7 تنتج true ، فيكون شكل العلاقة كالتالي:

```
False || True
```

- وبالطبع لأن أحد طرفى العلاقة هو true والمؤثر هو || ، فتصبح العلاقة كلها true.

المؤثرات على مستوى البت Bit-Wise Operator

- يستخدم هذا المؤثر Operator لتعديل المتغير على مستوى البت Bit.
- ويوجد لدينا مؤثرين Operators من هذا النوع وهما:
  - إزاحة لليمين Shift Right(<<).
  - إزاحة لليسار (>>) Left Shift.
- ولكى نفهم عمل هذا المؤثر Operator ، فإليك المثال التالى:

```
int x = 8;
int y,z;
y=x>>1;
x=8;
z= x<<2;
```

- والسؤال الآن ما هى قيمة y , z؟
- الإجابة هى أن y=4, z=32.
- ولكن لماذا !؟

- الحل يكمن ببساطة فى أننا لابد أولاً من تمثيل العدد 8 بالنظام الثنائى Binary فيكون كالتالى $8_{10} = 1000_2$

0	0	0	0	1	0	0	0

- والآن فى حالة الإزاحة لليمين بمقدار 1 يكون الشكل كالتالى $4_{10} = 100_2$

0	0	0	0	0	1	0	0

- فيكون y=4.
- أما فى حالة x=8 فإن الإزاحة ناحية اليسار بمقدار 2 تجعل الشكل كالتالى

$32_{10} = 100000_2$

0	0	1	0	0	**0**	0	0

- فيكون الناتج 32 لأن العدد 100000 فى النظام الثنائى Binary يساوى 32 فى النظام العشرى Decimal.
- والآن بعد أن تعرفنا على أنواع المتغيرات والمؤثرات المختلفة تعال معى نطبق ما تعلمناه من خلال المثال التالى.

مثال (2): المتغيرات Variables:

- فى هذا البرنامج سنقوم بحساب عمر شخص ما بالأيام والساعات والثوانى ، سنقوم بحساب راتبه ، واختبار ما إذا كان رجلاً أم سيدة.
- الآن حاول فتح المثال من الإسطوانة المرفقة ونفذ المثال أو افتح محرر النصوص Text Editor الذى تكتب فيه الكود واكتب المثال كالآتى.

```
1.  //Example Chapter 2 Example 1
2.  public class Chpt2Exmpl
3.  {
4.  public static void main(String arg[])
5.  {
6.  //Declaration Section Goes here
7.  short age_Inyears=80;
8.  int age_Indays;
9.  int age_Inhours;
```

```
10.  long age_InSeconds;
11.  float salary,bonus,netSalary;
12.  String name="Ahmed";
13.  boolean sex=true;
14.  //print Person Name
15.  System.out.println ("\n Welcome Mr. "+ name);
16.  //Calculate the Age in Days
17.  age_Indays=age_Inyears*365;
18.  //print out the Data
19.  System.out.println ("\n Mr. "+name+"\n Your Age
     Calculated in Days = "+age_Indays);
20.  //Calculate the Age in Hours
21.  age_Inhours=age_Indays*24;
22.  //print out the Data
23.  System.out.println ("\n Mr "+name+ "\n Your Age
     Calculated in Hours = "+age_Inhours);
24.  //Calculate the Age in Seconds
25.  age_InSeconds=(long)age_Inhours*3600;
26.  //print out the Data
27.  System.out.println("\n Mr "+name+ "\n Your Age
     Calculated in seconds = "+age_InSeconds);
28.  salary=2870.78f;
29.  //Calculate the Bonus
30.  bonus=salary*.1f;
31.  //Calculate the netSalary
32.  netSalary=salary+bonus;
33.  //print out the Data
34.  System.out.println ("\n Mr. "+name+ "\n Your net Salary =
     "+netSalary+ " LE");
35.  //check the Gender
36.  if(sex==true)
37.  //print out the Data
38.  System.out.println ("\n Mr. "+name+ "\n Your Gender is
     male");
39.  else
```

```
40.  //print out the Data
41.  System.out.println ("\n Mr. "+name+ "\n Your Gender is
     Female");
42.  }
43.  }
```

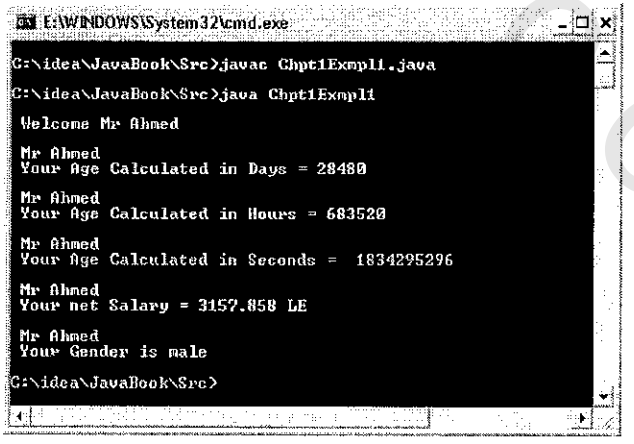
شرح السطور:

- كما ذكرنا من قبل ، فإن كل شئ فى لغة Java مبنى على الفصائل Classes ولذلك نجد أنه فى السطر رقم 2 نقوم بإخبار المترجم Compiler أننا نريد أن ننشئ فصيلة Class واسمها هو Chpt2Exmpl.
- فى السطر رقم 3 نجد قوس بداية الفصيلة class ({).
- فى السطر رقم 4 نكتب الدالة الرئيسية ()main وشكلها كالآتى:

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

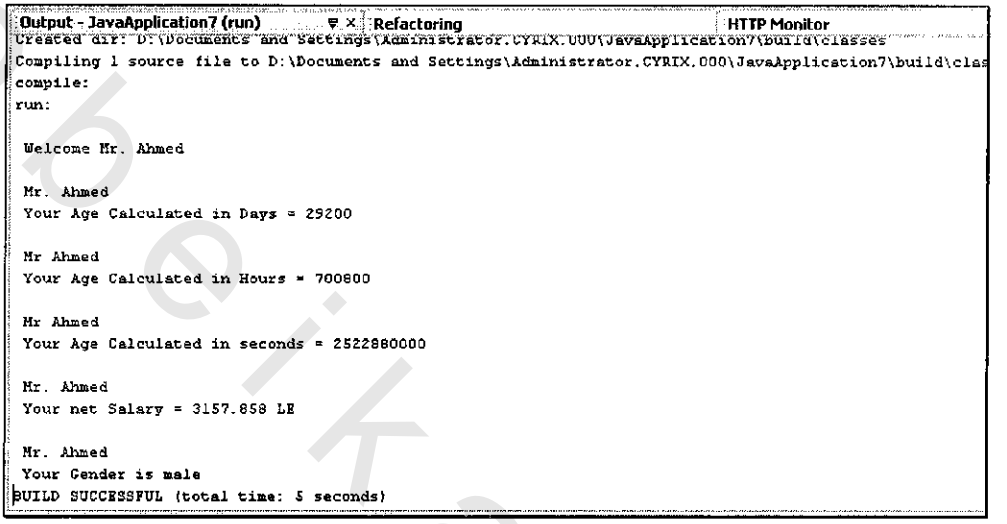
- فى السطر رقم 5 نجد قوس بداية الدالة الرئيسية ()main.
- فى السطور من 6 إلى 13 قمنا بتعريف مجموعة مختلفة من المتغيرات منها ما قد قمنا بإعطائه قيمة أولية Initial Value مثل المتغيرات Sex , name, age_inyears.
- فى السطر رقم 15 قمنا بطباعة رسالة ترحيب تحمل اسم الشخص ، ولطباعة أى نص على الشاشة ، فإننا نستخدم الدالة ()System.out.println ثم نضع بين الأقواس ما نريد طباعته من بيانات.
- وحيث أننا نريد طباعة نص ، فلا بد من وضع هذا النص بين علامتى تنصيص Quotations كالتالي " " ، ثم نريد أن نطبع اسم الشخص ، ولأن اسم الشخص عبارة عن متغير ، فذلك لا يتم وضعه بين العلامتين " " ، وكما ترى أيضاً فقد وضعنا المؤثر + الذي يخبر المترجم Compiler أننا نريد إضافة قيمة متغير ما إلى النص الأول ، وهكذا إذا أردنا أن نضيف نصاً أو اسم متغير أخر ، فإننا نضع علامة + أخرى وهكذا دواليك.
- فى السطر 17 رقم قمنا بحساب عمر الشخص بالأيام وذلك عن طريق ضرب قيمة المتغير age_inyears -الذى أعطيناه قيمة مبدئية 80- فى 365.
- فى السطر رقم 19 طبعنا اسم الشخص مع عدد الأيام ممثلة فى المتغير age_indays.

- فى السطر رقم 21 نقوم بحساب عمر الشخص بالساعات وذلك عن طريق ضرب المتغير age_indays -الذى احتفظ بالقيمة السابقة عن العملية الحسابية- في 24.
- فى السطر رقم 23 قمنا بطباعة عمر الشخص بالساعات.
- فى السطر رقم 25 قمنا بحساب عمر الشخص بالثوانى.
- فى السطر رقم 27 قمنا بطباعة عمر الشخص بالثوانى.
- فى السطر رقم 28 قمنا بتخصيص قيمة للمتغير Salary وهو من نوع float.
- فى السطر رقم 30 قمنا بحساب المكافآت التى يحصل عليها.
- فى السطر رقم 32 يتم حساب صافى المرتب.
- فى السطر رقم 34 يتم طباعة المرتب.
- فى السطور من 35 إلى 41 نختبر ما إذا كان الشخص ذكراً أم أنثى عن طريق المتغير Sex الذى من نوع boolean.
- فى السطر رقم 42 نجد قوس نهاية الدالة الرئيسية main().
- فى السطر 43 نجد قوس نهاية الفصيلة class.
- والآن قم بحفظ الملف باسم Chpt1Exmpl1.java ثم قم بعملية الترجمة Compilation عن طريق الأمر javac Chpt1Exmpl1.java ، إن لم يكن هناك أخطاء ، فقم بكتابة الأمر java Chpt1Exmpl1 لتنفيذ البرنامج (كما يمكنك فتح البرنامج من الإسطوانة المرفقة) وترى نتيجة تنفيذ البرنامج كما هو واضح فى الشكل (2-4).



(الشكل 2-4)

- يمكنك كتابة البرنامج داخل برنامج NetBeans ونفذه لتحصل على نتيجة التنفيذ كما هو واضح فى الشكل (2-5).



(الشكل 2-5)

ملخص الفصل:

- تعلمنا في هذا الفصل ثمانية أنواع رئيسية Primitive Data Types فى لغة Java وهى byte, char, short, int, long, float, double, Boolean وتعرفنا علي مساحة ونطاق كل نوع.
- كما تعرفنا على الجمل Statements والتعبيرات Expressions والفرق بينهما.
- كما تعرفنا على كيفية إنشاء المتغيرات وإعطائها قيم مبدئية Initial Values.
- كما تعرفنا علي الثوابت Constants.
- كما تعرفنا على المؤثرات Operators وأنواعها وأسبقية كل منها فى التنفيذ.
- في الفصل القادم سوف نتعرف - بإذن الله - علي كيفية كتابة أوامر جمل التحكم Control Statements ، فتابع معنا الفصل القادم.