

# الفصل الثاني

## ٢

**البيانات والمتغيرات والثوابت والمعرفات والاختفاء**



### أولاً : حرف وأرقام لغة Java

-الحروف الأبجدية Letters وهى الحروف الكبيرة Capital Letters من A إلى Z والحروف الصغيرة small letters من a إلى z

-الأرقام العددية Digits من 0 إلى 9

-الحروف الخاصة Special Characters أو المعاملات أو المؤثرات (الرموز) Symbols or Operators

تستخدم java مجموعة من المعاملات (الرموز) في العمليات الحسابية والمنطقية التي تتم على المتغيرات ويمكن تصنيفها كما يلي:

#### أمعاملات رياضية (حسابية) Arithmetic operators

كما يلي ( مرتبة حسب أولوية التنفيذ) :

الرمز	الاسم	مثال
++	معامل زيادة المتغير بمقدار واحد Increment	Int a; A++;
--	معامل نقصان المتغير بمقدار واحد Decrement	Int a; A--;
*	الضرب Multi plaction	Int w =200; Int z=3*w
/	القسمة Division	Int w =200; Int z=w/2;
%	باقي القسمة Modulus	Int x=15%2///x=1
+	الجمع Addition	Int a=5; Int b=10; Int sum; Sum=a+b;
-	الطرح subtraction	Int a=5; Int b=10; Int w; w=b-a;
=	التساوي Assignment	a = b
++	Increment لإضافة 1 على قيمة المتغير وتستخدم في الحلقات	a++
--	Decrement لإنقاص 1 من قيمة المتغير وتستخدم في الحلقات	a--

### مثال على استخدام المعاملات الحسابية

بفرض أن لدينا متغيرين a ، b وقيمهم 28.5 ، 14.5 ونريد إجراء العمليات الحسابية الأساسية عليهم

1	public static void main(String[] args) {
2	float a=28.5f , b=14.5f;
3	float x,y,z,v,u;
4	x=a+b;
5	y=a-b;
6	z=a*b;
7	v=a/b;
8	u=a%b;
9	System.out.println("a+b="+x+"\n"+"ab="+
	y+"\n"+"a*b=" +
10	z+"\n"+"a/b="+v+"\n"+"a%b="+u);
11	}
	}

```
public static void main(String[] args) {  
    float a=28.5f , b=14.5f;  
    float x,y,z,v,u;  
    x=a+b;  
    y=a-b;  
    z=a*b;  
    v=a/b;  
    u=a%b;  
    System.out.println("a+b="+x+"\n"+"ab="+  
y+"\n"+"a*b="+z+"\n"+"a/b="+v+"\n"+"a%b="+u) ;  
}  
}
```

ويظهر ناتج التنفيذ كما يلي:

### tput - JavaApplication5 (run)

```
run:
a+b=43.0
ab=14.0
a*b=413.25
a/b=1.9655173
a*b=14.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

### ب-معاملات مقارنة أو علائقية Comparison/Relational Operators

تقوم بربط متغير بآخر واختبار العلاقة بينهما

الرمز	الاسم	مثال
=	معامل تخصيص قيمة لمتغير Equal to	Int w=5;
==	يساوى في القيمة	b==A;
!=	لا يساوى Not equal to	a != b;
>	أكبر من Greater than	b>A;
<	أقل من Less than	b<A;
<=	أصغر من أو يساوى Less than or Equal to	b<=a ;
>=	أكبر من أو يساوى Greater than or Equal to	a>=b ;

### ج-المعاملات المنطقية Logical Operator

الرمز	الاسم	المعنى
&&	and	يجب تحقق الشرطين للعودة بالقيمة true
	or	يكتفي تحقق أحد الشرطين للعودة بالقيمة true
!	not	يقوم هذا المعامل بعكس ناتج العلاقة

مثال:

يوجد لدينا المتغيرات  $i$  ،  $f$  ،  $c$  وقيمها  $i=7, f=5.5, c='w'$  والمطلوب إعداد برنامج يوجد نتائج العلاقات التالية:

$(i \geq 6) \&\& (c == 119)$   
 $(i > 6) || (c == 'w')$   
 $(f < 11) \&\& (i > 100)$   
 $(c != 'p') || ((i + f) \leq 10)$

```

1 public static void main(String[] args) {
2     boolean b1,b2,b3,b4;
3     int i=7;
4     float f=5.5f;
5     char c='w';
6     b1=(i>=6)&&(c==119);
7     b2= (i>6)|| (c=='w');
8     b3=(f<11)&&(i>100);
9     b4=(c!='p')||((i+f)<=10);
10    System.out.println("(i>=6)&&(c==119) is "+b1);
11    System.out.println("(i>6)|| (c=='w') is "+b2);
12    System.out.println("(f<11)&&(i>100) is "+b3);
13    System.out.println("(c!='p')||((i+f)<=10) is "+b4);
14    }
15 }
```

```

"/
public static void main(String[] args) {
    boolean b1,b2,b3,b4;
    int i=7;
    float f=5.5f;
    char c='w';
    b1=(i>=6) &&(c==119);
    b2= (i>6) || (c=='w');
    b3=(f<11) &&(i>100);
    b4=(c!='p') || ((i+f)<=10);
    System.out.println("(i>=6) &&(c==119) is "+b1);
    System.out.println("(i>6) || (c=='w') is "+b2);
    System.out.println("(f<11) &&(i>100) is "+b3);
    System.out.println("(c!='p') || ((i+f)<=10) is "+b4);
}

}
```

ويظهر ناتج التنفيذ كما يلي :

#### Output - JavaApplication6 (run)

```
run:
(i>=6)&&(c==119) is true
(i>6)|| (c=='w') is true
(f<11)&&(i>100) is false
(c!='p')||((i+f)<=10) is true
BUILD SUCCESSFUL (total time: 1 second)
```

د-العوامل التي تستخدم لإعطاء قيم للمتغيرات (Assignment Operators)

اسم العامل	رمزه	مثال	الشرح
Basic Assignment	=	a = b	ضع قيمة b في a
Add AND Assignment	+=	a += b	أضف قيمة a على قيمة b و خزن الناتج في a
Subtract AND Assignment	-=	a -= b	أنقص قيمة a من قيمة b و خزن الناتج في a
Multiply AND Assignment	*=	a *= b	أضرب قيمة a بقيمة b و خزن الناتج في a
Divide AND Assignment	/=	a /= b	أقسم قيمة a على قيمة b و خزن الناتج في a
Modulo AND Assignment	%=	a %= b	قسم قيمة a على قيمة b و خزن آخر رقم يبقى من عملية القسمة في a
Left shift AND Assignment	<<=	a <<= 2	أزح آخر اثنين bits و ضعهم في الأول ثم خزن الناتج في a
Right shift AND Assignment	>>=	a >>= 2	أزح أول اثنين bits و ضعهم في الآخر ثم خزن الناتج في a
Bitwise AND Assignment	&=	a &= b	أحسب ناتج جمع الـ bits المشتركة بين a و b و خزن الناتج في a
Bitwise exclusive OR and Assignment	^=	a ^= b	أحسب ناتج جمع الـ bits الغير مشتركة بين a و b و خزن الناتج في a
Bitwise unexclusive OR and Assignment	=	a  = b	أحسب ناتج جمع الـ bits المشتركة و الغير مشتركة بين a و b و خزن الناتج في a

هذا ويوجد بعض الرموز الخاصة تسمى حروف الهروب Escape Character والتي تستخدم للدلالة على أن الحرف التالي لها ذو معنى خاص وهي حروف لها هدف محدد وتبدأ دائما بالشرطة المائلة للخلف \ ويأتي بعدها الحرف أو الرمز أو الاثنين معا ومن أمثلتها ما يلي :

\b	الرجوع للخلف مسافة	\"	كتابة علامة تنصيص مزدوجة
\n	الانتقال لسطر جديد	\'	كتابة علامة تنصيص مفردة
\r	البدء من أول السطر	\?	كتابة علامة استفهام
\t	القفز ٧ مسافات	\\	كتابة شرطة مائلة للخلف
<<	إزاحة إلى اليسار	>>	إزاحة إلى اليمين

مثال

```

1 public static void main(String[] args) {
2     short s=100 ;
3     int i=10000 ;
4     long L=10000000000;
5     float d=105.55f ;
6     char c='G';
7     boolean state=false;
8     System.out.println("s="+s+"\t"+"i="+i+"\n"+"L="+L+"\t"+"d="+
9         d + "\n"+"c="+c+"\t"+"state="+state);
10 }

```

```

/
|     public static void main(String[] args) {
|         short s=100 ;
|         int i=10000 ;
|         long L=10000000000;
|         float d=105.55f ;
|         char c='G';
|         boolean state=false;
|         System.out.println("s="+s+"\t"+"i="+i+"\n"+"L="+L+"\t"+"d="+d+"\n"+"c="+
|         c+"\t"+"state="+state);
|     }
| }

```

ويظهر ناتج التنفيذ كما يلي:

Output - JavaApplication5 (run)

```

run:
s=100          i=10000
L=10000000000    d=105.55
c=G            state=false
BUILD SUCCESSFUL (total time: 2 seconds)

```



مثال :

استخدام المعاملات الحسابية

```
1 public static void main(String[] args) {
2     int a=15;
3     int b=4;
4     int a=15;
5     int b=4;
6     int x,y,z,v,u;
7     float f,c=4.0f;
8     x=a+b;
9     y=a-b;
10    z=a*b;
11    v=a/b;
12    f=a/c;
13    u=a%b;
14    System.out.println("a+b="+x);
15    System.out.println("a-b="+y);
16    System.out.println("a*b="+z);
17    System.out.println("a/b="+v+"\t"+"a/b="+f);
18    System.out.println("a%b="+u);
19 }
20 }
```

```
public static void main(String[] args) {
    int a=15;
    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/b="+f);
    System.out.println("a%b="+u);
}
```

ويظهر ناتج التنفيذ كما يلي:

#### Output - JavaApplication4 (run)

```
run:
a+b=19
a-b=11
a*b=60
a/b=3          a/b=3.75
a%b=3
BUILD SUCCESSFUL (total time: 1 second)
```

#### ثانياً: الكلمات المحجوزة Reserved Words أو الرئيسية Main Words

هي كلمات ذات معنى قياسي سابق التعريف في لغة Java وتستخدم في الغرض المحدد لها فقط ولا يمكن استخدامها كمتغيرات

وهي كلمات ثابتة لها معنى خاص للمترجم ولا يمكن استخدامها لأغراض أخرى في البرنامج

الكلمات المفتاحية لها معنى وغرض خاصين، على عكس المعرفات المخصصة، لا يمكن استخدامها كمعرفات عامة public، هي أيضاً كلمة رئيسية تُستخدم لتمثيل الفئات العامة

الكلمات الأساسية لـ Java لها معنى خاص لمترجم Java يتم استخدامها للإشارة إلى نوع البيانات أو للإشارة إلى بنية البرنامج، جميع الكلمات التالية محجوزة للغة java أي لا يمكن استخدامها كـ Identifiers

abstract	continue	For	New	swtich
assert	default	goto	Package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

### ثالثا: الثوابت Constants

قيم ثابتة يراد الاحتفاظ بها طوال البرنامج ولا تتغير قيمتها أبداً ، وتنقسم الثوابت في لغة الجافا إلى :  
ثوابت عددية Numeric Constants ، ثوابت رمزية Non-numeric Constants

- ١- الثوابت العددية: يمكن تمثيل الثوابت العددية في لغة الجافا كالاتي:  
-الثابت العددي الصحيح Integer : عدد مكون من الأرقام من ( ٠ إلى ٩ ) ولا يحتوي على فاصلة عشرية ، ويمكن أن يتضمن الإشارة ( + أو - ).  
-الثابت العددي الحقيقي Floating constant : عدد مكون من الأرقام من ( ٠ إلى ٩ ) ويحتوي على فاصلة عشرية و يمكن أن يتضمن الإشارة ( + أو - ).

- ٢- الثوابت الرمزية: عبارة عن رموز اللغة وتتكون من الحروف والأرقام وتكون بين علامتي تنصيص أو اقتباس ، ومن الأمثلة على الثوابت الرمزية ما يلي:  
"name" - "Khaled" - "12345" - "30+40 "

وإذا أردنا أن نضع قيمة تظل ثابتة داخل البرنامج في مكان في الذاكرة فإننا نستخدم العبارة final للإعلان أن هذه القيمة ستظل ثابتة طوال تنفيذ البرنامج مثل:

```
final int TABLE_SIZE = 41;  
final float PI = 3.14159;
```

ويمكن تعريف الثابت أيضا على انه قيمة لا تتغير أثناء تنفيذ البرنامج ويتم تعريفه بوضع الكلمة Final قبل نوع الثابت والتي تجعل المترجم يقوم بحجز مكان في الذاكرة لهذا النوع ولا يسمح بتغييره كما يلي :  
Final float pi=3.14;

#### أنواع أخرى للثوابت

- ١-ثوابت صحيحة Integer Constant : هي أعداد ذات قيم صحيحة تتضمن تسلسل من الأرقام سواء سالبة أو موجبة ويجب ألا يحتوي الثابت الصحيح على نقطة عشرية أو أسية
- ٢-ثوابت متحركة النقطة Floating Point : هو عدد للأساس ١٠ يتضمن علامة عشرية أو أس أو الاثنين معا ويرمز للأساس ١٠ بالرمز e أو E ويمكن أن يكون الأس سالب أو موجب ويجب ألا يحتوي على علامة عشرية (مثلا الرقم ٦٠٠٠٠ يكتب 6E4
- ٣-ثوابت حرفية Character Constant : يمكن أن تكون رقم أو حرف أو رمز ومن أمثلتها ( 65 ، ' \n ، ' 8 ' ) ونلاحظ أن الرقم 65 عبارة عن الحرف 'A' حيث أن الثابت الحرفي يمكن أن يكون رقم وهذا الرقم سيتم تحويله إلى الرقم المناظر له في ASCII
- ٤-ثوابت السلسلة String Constant : يتضمن عدد من الرموز والفراغات وتكون محصورة بين علامتي تنصيص مزدوجة " "

### رابعاً : المتغيرات Variables

من أساسيات كتابة البرامج التعرف على المتغيرات وأنواعها وطرق إنشاءها، والمتغير هو وعاء نضع فيه قيمة معينة فيحتفظ بهذه القيمة إلى حين الحاجة إليها لاستخدام المتغير مرة أخرى، القيمة التي يحتفظ بها قد تكون رقم أو حرف أو جملة أو ملف أو صورة أو غيرها من الكائنات التي نعرفها بالحاسب، فالمتغير هو وعاء يحتوي على الكائن أو القيمة التي نرغب بحفظها، ويقع هذا الوعاء بذاكرة الحاسب العشوائية، RAM، وهذا يعني أنه كلما كثر عدد المتغيرات في برنامج ما ازدادت المساحة المحجوزة في ذاكرة الحاسب من قبل هذا البرنامج.

يجب الإعلان عن نوع المتغير قبل استخدامه في البرنامج بالصيغة التالية:

DataType Name = Value؛

### أنواع البيانات Data Type

يقصد بها الأنواع التي ستستخدم في تعريف المتغيرات وتنقسم البيانات إلى: بيانات رقمية صحيحة Integer، بيانات رقمية عشرية Floating integer، بيانات حرفية Character، بيانات منطقية Boolean (False, True)

#### byte Data Type

متغير رقمي يستطيع الاحتفاظ بالأرقام فقط وتكون قيمها أعداد صحيحة موجبة أو سالبة محصورة بين - 128 (حد أدنى) وبين +127 (حد أقصى) وحجمها في الذاكرة 8Bits (1 Byte) وهي أصغر أنواع البيانات الرقمية الصحيحة وغير الصحيحة.  
مثال

1	byte b = 0 ؛	// صحيح
2	byte age = 127؛	// صحيح
3	byte negative = -10 ؛	//صحيح
4	byte theEnd = -128 ؛	// صحيح
5	byte wrong = 128 ؛	خاطئ لأنه تجاوز الحد الأقصى الموجب //
6	byte wrongAlso = -129 ؛	خاطئ لأنه تجاوز الحد الأقصى السالب //

#### حالات استخدام المتغير byte

من الممكن استخدام نوع البيانات byte في الحالات التالية :  
للأرقام الصغيرة التي لا تتعدى حدود احتماله الدنيا والقصى، وهو مدى ليس بكبير لذا يقل استخدام نوع البيانات byte في البرامج .  
-عند قراءة خيارات من المستخدم والتي تتراوح بين ١ إلى ٩ .  
-عند تعريف عمر صغار السن.  
-تسجيل عدد المسافرين في سيارة.  
-قراءة عدد أيام الأسبوع أو الشهر.

### short Data Type

متغير رقمي أيضاً و يعتبر ثاني أصغر أنواع البيانات الرقمية الصحيحة قيمها أعداد صحيحة موجبة أو سالبة محصورة بين -32768 (حد أدنى) ، +32768 (حد أقصى) وحجمها في الذاكرة 16 Bit (ضعف حجم الـ byte و هو 2byte )

1	short s = 0;	// صحيح
2	short cash = 32000;	// صحيح
3	short netativeLimit = -32768;	// صحيح
4	short positiveLimit = 32767;	// صحيح
5	short wrongA = 33000;	// خاطئ لأنه تعدى الحد الأقصى الموجب
6	short wrongB = -32769	// خاطئ لأنه تعدى الحد الأقصى السالب

### حالات استخدام المتغير short

يُعتبر مدى نوع البيانات short معقولاً إلى حد ما ولكنه صغير في عالم البيانات، ولكن من الممكن استخدامه في حالات كثيرة مقارنة بالـ byte مثل:  
- تعريف عدد الأيام المنقضية في السنة.  
- تسجيل مبلغ الرسوم الدراسية لفصل دراسي.  
- حساب حجم شاشة الحاسب بالبيكسل.

### int Data Type

متغير رقمي أيضاً وهو أكثر المتغيرات الرقمية الصحيحة استخداماً في الجافا وتمثل بقيم صحيحة موجبة أو سالبة محصورة بين -2147483648 (حد أدنى) ، +2147483647 (حد أقصى) وحجمها في الذاكرة 32 Bit (4 Byte)

1	int i = 1;	// صحيح
2	int income = 1500777991;	// صحيح
3	int withdraw = -2000987;	// صحيح
4	int positiveLimit = 2147483647;	// صحيح
5	int negativeLimit = -2147483648;	// صحيح
6	int wrongA = 2147483648;	// خاطئ لأنه تعدى الحد الأقصى الموجب
7	int wrongB = -2147483649;	// خاطئ لأنه تعدى الحد الأقصى السالب

### حالات استخدام المتغير int

من أكثر المتغيرات استخداماً لمداها المعقول ولأن أغلب المتغيرات في الحياة يكون لها مدى كبير، فمن الممكن أن يُستخدم في تعريف الحالات التالية :  
- عدد الدقائق التي عاشها الإنسان.  
- الدخل اليومي لشركة ما  
- عداد سيارة.

## long Data Type

أكبر المتغيرات الصحيحة السالبة أو الموجبة في الجافا وتكون أعداد كبيرة جدا ، المدى يتراوح بين: [ ٩,٢٢٣,٣٧٢,٠٣٦,٨٥٤,٧٧٥,٨٠٧ إلى ٢٢٣,٣٧٢,٠٣٦,٨٥٤,٧٧٥,٨٠٨.٩- ] ، إذا تعدى المتغير مدى نوع البيانات long عندها لا بد من التحول إلى نوع البيانات غير الصحيحة ، وحجمه في الذاكرة ٨ بايت في الذاكرة ، أي ضعف نوع البيانات int

1	long i = 1L;	// صحيح
2	long income = 1500777991L;	// صحيح
3	long withdraw = -2000987L;	// صحيح
4	long positiveLimit = 2147483647L;	// صحيح
5	long negativeLimit = -2147483648L;	// صحيح
6	long wrongA = 9999999992147483648L;	خاطئ لأنه تعدى الحد الأقصى الموجب
7	long wrongB = -9999999992147483649L;	خاطئ لأنه تعدى الحد الأقصى السالب

## حالات استخدام المتغير long

عندما تتوقع نمواً مضطرباً في البيانات وقراءة أرقام كبيرة جداً مثل تعريف الحالات التالية:

- متغير المسافة بين الكواكب.
- مصروفات وإيرادات الدول

## float Data Type

متغير رقمي عشري وهو أصغر نوع بيانات رقمي عشري ، وتمثل بأعداد حقيقية بها علامة عشرية وتتحصر بين  $-3.40282347\text{E}+38$  ،  $+1.40239846\text{E}-45$  وحجمه مماثل لـ `int` و هو 4 Byte أي 32 Bit مثلاً ( 2.555 ، 222.357 ، 88.65 ) ولكن حدوده تختلف كثيراً نسبة لاختلاف طريقة التخزين في الذاكرة

1	float f = 999F;	// صحيح
2	float dec = 0.12345679F;	// صحيح
3	float negative = -999999F;	// صحيح
4	float positiveLimit = 2147483647L;	// صحيح
5	float negativeLimit = -2147483648L;	// صحيح
6	float wrongA = 999F;	// خاطئ لأنه تعدى الحد الأقصى الموجب
7	float wrongB = "a";	// خاطئ لأن نوع البيانات ليس رقمي

## حالات استخدام المتغير float

يستخدم نوع البيانات float مع المتغيرات العشرية، وفق الحاجة العامة للبرامج فنوع البيانات float يكفي لأغلب الاستخدامات، من هذه الاستخدامات ما يلي:

- تخزين المبالغ الكسرية
- تخزين النتائج الرياضية
- قياس الأطوال المساحية.

## 09

### حالات استخدام المتغير char

يستخدم نوع البيانات char عند الحاجة الى تعريف حرف واحد فقط مثل: وقت الساعة ص أو م ، تعريف نوع التاريخ هـ أو م ، تعريف حروف لوحات السيارات.

### boolean Data Type

نوع بيانات منطقي، أي يتعامل مع القيمة المنطقية فقط و بما أن المنطق دائماً له قيمتان فقط، صحيح أو خطأ فإنه يأخذ القيمة True أو False ويكون حجمه في الذاكرة 1Bit ، وعند الإعلان عن متغير Boolean وعدم إعطاؤه قيمة تكون القيمة الافتراضية له False

1	boolean b = false;	//صحيح
2	boolean test = true;	//صحيح
3	boolean wronA = no;	//خاطئ
4	boolean wrongB = yes;	//خاطئ

### حالات استخدام المتغير boolean

يستخدم نوع البيانات boolean بكثرة في البرامج لأن البرامج تعتمد على المنطق، وذلك مثل :  
 -وسيلة شهيرة جداً للخروج من الحلقات التكرارية.  
 -معرفة هل أدخل المستخدم قيمة صحيحة أم لا.  
 -الاستعانة بمتغير يحمل القيمة false إلا إذا حدث تغير معين أثناء البرنامج

ويمكن تلخيص ما سبق في الجدول التالي :

النوع	الحجم في الذاكرة	
byte	8 Bits	integer
short	16 Bits	
int	32 Bits	
long	64 Bits	
float	32 Bits	floating
double	64 Bits	
char	16 Bits	character
boolean	1 Bit	boolean



### ملاحظات خاصة بالإعلان عن المتغيرات Variable Declaration

١- يتم الإعلان عن المتغير بكتابة نوع المتغير ثم اسمه مع إمكانية إعطائه قيمة مبدئية

#### Initializing Values

```
int Salary = 750;  
boolean male= true;
```

٢- يجب كتابة جميع أسماء أنواع المتغيرات بحروف صغيرة (double- int-boolean)

٣- للإعلان عن المتغير float أضفنا حرف F للتوضيح أن الرقم يقع في النطاق المحدد للمتغير float ، كما أضفنا الحرف D للتوضيح أن الرقم يقع في النطاق المحدد للمتغير double

٤- للإعلان عن المتغيرات الحرفية Character يجب وضع القيمة بين علامة تنصيص مفردة

٥- للإعلان عن المتغير long يجب كتابته حرف l بعد الرقم كما يلي :

```
long w=1234321l;
```

٦- غالباً نستخدم المتغيرات من النوع boolean مع الجمل الشرطية

٧- المتغيرات من النوع سلسلة strings عبارة عن مجموعة من الأحرف (رموز+أحرف) تستخدم كجزء واحد ويكون تعريفها بوضعها بين علامتي تنصيص مزدوجة مع ضرورة كتابة الحرف الأول ( S ) من كلمة Strings بحجم كبير كما يلي:

```
String w= " Java 2" ;
```

### مثال على المتغيرات Strings

```
1 public static void main(String[] args) {  
2     String s = "*";  
3     for (int i=0;i<=6;i++){  
4         System.out.println(s);  
5         s+="*";  
6     }  
7 }
```

ملحوظة:

S=S+"\*" تعادل S+="\*"

```
public static void main(String[] args) {  
    String s = "*";  
    for (int i=0;i<=6;i++){  
        System.out.println(s);  
        s+="*";  
    }  
}
```

ويظهر ناتج التنفيذ كما يلي:

```
*  
**  
***  
****  
*****  
*****  
*****  
*****  
*****  
BUILD SUCCESSFUL (total time: 0 seconds)
```

٧-تستطيع لغة Java تحويل جميع أنواع المتغيرات في نتائج العمليات المختلفة إلى نوع واحد حيث يمكن تحويل المتغير من نوع بيان إلى نوع بيان آخر وتسمى تلك العملية Data Type Casting وتكون الأولويات عند وجود أنواع المتغيرات في العملية وفقاً للترتيب التالي :

-إذا كان في العملية double تحول المتغيرات الأخرى إلى double والنتيجة تكون من النوع double

-إذا كان في العملية float تحول المتغيرات الأخرى إلى float والنتيجة تكون من النوع float

-إذا كان في العملية long تحول المتغيرات الأخرى إلى long والنتيجة تكون من النوع long

-إذا لم تكن في العملية أي من الأنواع السابقة تحول كل المتغيرات إلى int والنتيجة تكون من النوع int

-وبالنسبة للنوع boolean لا يمكن تحويله إلى نوع آخر أو تحويل نوع آخر إليه

يمكن تحويل الأعداد الصحيحة بأنواعها إلى أعداد عشرية بمعنى تحويل byte ، integer إلى long، Short ، double ، float

مثال

1.5	+	99	=	100.5
double	+	int	=	double
int	→			double

مثال:

تعريف عدة متغيرات من نفس النوع بدون قيم.

1	int a;
2	int b;
3	int c;

يمكن تعريفهم كما يلي:

1	int a, b, c;
---	--------------

مثال:

تعريف عدة متغيرات من نفس النوع وإعطائهم قيم مباشرة عند إنشائهم.

1	int a = 10;
2	int b = 20;
3	int c = 30;

يمكن تعريفهم كما يلي:

1	int a=10, b=20, c=30;
---	-----------------------

مثال:

تعريف عدة متغيرات من نفس النوع و لكن طبيعتهم مختلفة و بدون قيم.

1	int a;
2	int b[ ];
3	int c[ ][ ];

يمكن تعريفهم كما يلي:

int a, b[ ], c[ ][ ];
-----------------------

مثال:

تعريف المصفوفة بعدة طرق كالتالي.

- الطريقة الأولى

int[ ] a;
-----------

الطريقة الثانية

int [ ]a;
-----------

الطريقة الثالثة

int a[ ];
-----------

ملاحظات هامة:

- عند تعريف مصفوفة ومتغير من نفس النوع لا يتم استخدام الطريقة الأولى والسبب انه عند كتابة int[ ] عندئذ يتم اعتبار كل اسم يتم وضعه بعد الفاصلة يمثل مصفوفة وليس متغير عادي وبالتالي فإن ( a ، b ، c ) يمثلون مصفوفات ذات بعد واحد ، ونفس المبدأ ينطبق على المصفوفات الأكثر من بعد واحد.

- Declaration : تعريف متغير بدون إعطائه قيمة

- Assigning : إعطاء قيمة لمتغير تم إنشاؤه سابقاً

- Initialization: تعريف متغير وإعطائه قيمة مباشرة عند إنشائه

أمثله

-عمل Declare لمتغير جديد أي تعريف متغير جديد بدون إعطائه قيمة أولية

```
int a;
```

-عمل Assign لمتغير أي إعطاء قيمة لمتغير كان في الأصل موجوداً

```
a = 10;
```

-عمل Initialize لمتغير جديد أي إنشاء متغير جديد وإعطائه قيمة أولية.

```
double Sum = 0;
```

### خامساً: المعرفات Identifier

هي أسماء تعطى من قبل المستخدم لعناصر البرنامج (المتغيرات-الدوال-المصفوفات) ويتم استخدام المعرفات لتسمية الثوابت والمتغيرات والفئات وكتائنات الفئة

وتتمثل قواعدها فيما يلي:

-يمكن أن يحتوي على حروف أبجدية وأرقام والشرطة التحتية Under Score وعلامة الدولار \$ ولا توجد مسافة بين كل رمز

-يجب أن يبدأ المعرف بحرف أبجدي (صغير أو كبير ويفضل الصغير )

- ألا يتضمن المعرف علامة خاصة عدا الشرطة التحتية

-يجب أن يبدأ اسم الكائن بحرف كبير

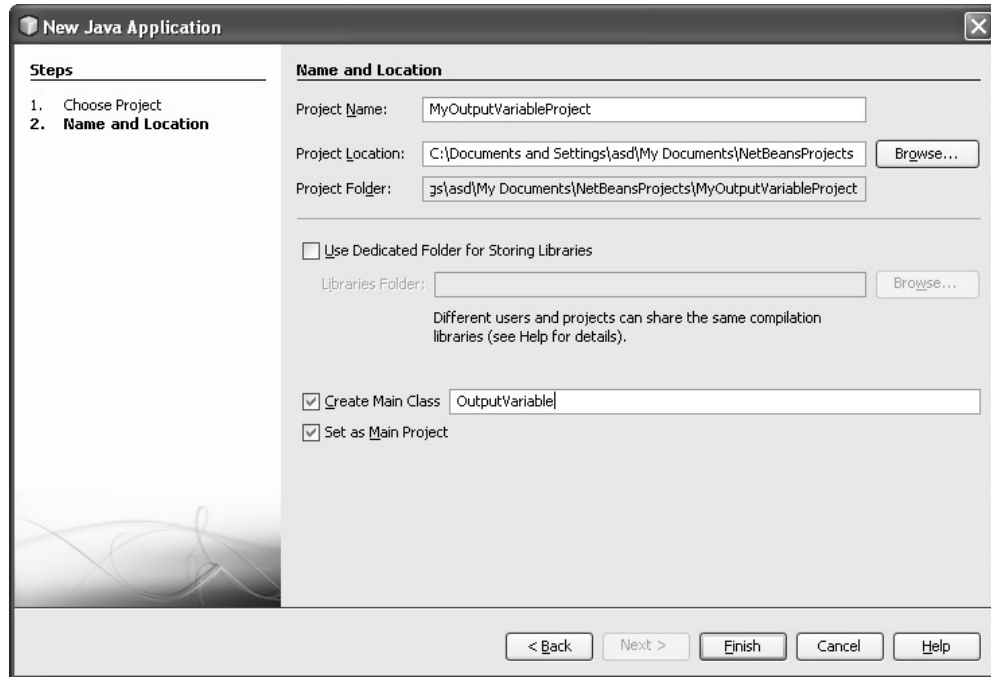
-تنقسم المعرفات إلى فئتين هما: الكلمات الرئيسية، والمعرفات المعرفة من قبل المستخدم، فالكلمات الرئيسية هي معرفات ذات معاني خاصة، على سبيل المثال، يشير الصواب والخطأ إلى الحقيقة المنطقية، والمعرف المعرف من قبل المستخدم هو معرف لكلمة غير محجوزة يتم إنشاؤها بواسطة المستخدم وفقاً لقاعدة تكوين المعرف، على سبيل المثال ، abc هو معرف.

-لغة الجافا حساسة لحالة الأحرف وتطبق مفهوم Case Sensitivity وهذا يعني أنها تميز بين الأحرف الكبيرة و الأحرف الصغيرة على سبيل المثال myjava ، Myjava معرفان مختلفان

مثال

تشغيل برنامج NetBeans IDE ثم فتح قائمة File ونختار منها New

يظهر مربع حوار يتضمن أنواع التطبيقات التي يمكن إنشاؤها، نختار Java (أسفل العنوان Categories) ونختار Java Application (أسفل العنوان Project) ثم نضغط على الزر Next لتظهر نافذة نحدد بداخلها اسم المشروع كما يلي :

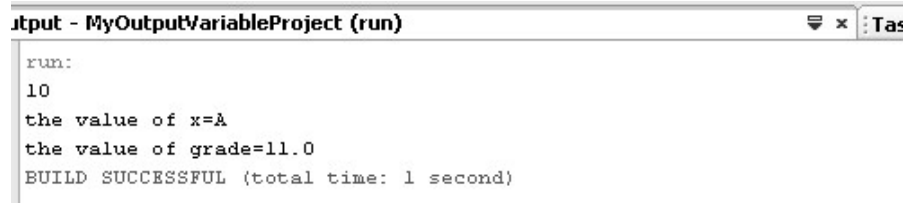


نضغط الزر Finish عندئذ يتم إنشاء التطبيق وتظهر النافذة السريعة التي تتضمن اسم التطبيق ثم تختفي وتظهر نافذة كتابة الكود نكتب بداخلها الكود التالي :

```
public static void main(String[] args) {  
    int value=10;  
    char x;  
    x='A';  
    double grade = 11;  
    System.out.println(value);  
    System.out.println("the value of x=" + x);  
    System.out.println("the value of grade=" + grade );  
}
```

```
public class OutputVariable {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int value=10;
        char x;
        x='A';
        double grade = 11;
        System.out.println(value);
        System.out.println("the value of x=" + x);
        System.out.println("the value of grade=" + grade );
    }
}
```

-تنفيذ البرنامج من خلال فتح قائمة Run ونختار منها Run Main Project ( أو نضغط مباشرة مفتاح F6 ) عندئذ يظهر ناتج التنفيذ بالشكل التالي:



```
Output - MyOutputVariableProject (run)
run:
10
the value of x=A
the value of grade=11.0
BUILD SUCCESSFUL (total time: 1 second)
```

#### سادسا: الجمل Statements

الجملة هي أمر بسيط مكتوب بلغة برمجة يؤدي لحدوث شئ مثل السطر التالي:

```
System.out.println("welcome");
```

#### سابعا: التعبير Expression

التعبير جملة تنتج قيمة من خلال جملة أخرى وتلك القيمة يمكن تخزينها لاستخدامها في البرنامج في وقت لاحق

```
Int x=5;          //Statement
```

```
Int y=10;         //statement
```

```
Int z=x+y;        //Expression
```

### ثامنا: التعليقات أو الملاحظات Comments

عند كتابة كود برمجي يتضمن سطور كثيرة وأردنا العودة إليه بعد فترة لعمل بعض التعديلات فمن الصعب تذكر ذلك الكود وهنا تظهر أهميته إدراج تعليق على الكود يمثل شرح ووصف مختصر عن الكود ولا يقوم المترجم بترجمته حيث تمثل التعليقات شرح للعمل وإضافة معلومات توضيح الغرض من الكود المكتوب وتسهيل التعامل مع الكود بعد فترة- بالإضافة أو الحذف أو التعديل ، ويمكن وضع التعليق في أي مكان بالبرنامج ويمكن أن يخلو البرنامج منها

ويمكن إضافة الملاحظات من خلال إحدى الطرق التالية:  
- لإدراج تعليق على سطر واحد تستخدم علامتان الشرطة المائلة //  
- إذا زاد التعليق عن سطر يجب إدراجه بين علامتين /\*----\*/

### الطريقة الأولى: /\*\* النص \*/

تستخدم لكتابة معلومات التوثيق أو Documentation و لا يقوم المترجم بترجمة الأوامر بين هاتين علامتين.

```
/* هذا تعليق */
* يتألف من
* عدة أسطر
*/
```

### الطريقة الثانية: النص //

تستخدم لكتابة سطر واحد من الملاحظات ولا يقوم المترجم بترجمة الأوامر في هذا السطر إلى أن يصل إلى آخره.  
مثال

```
/**
 *The HelloWorldApp class implements an application that
 *simply displays "Hello World!" to the standard output.
 */class HelloWorldApp{
public static void main(String[] args){
System.out.println("Hello World//";!"!Display the string.
{
{
```

### الطريقة الثالثة: /\*\* النص \*/

لكتابة الملاحظات أيضا و لا يقوم المترجم بترجمة الأوامر بين هاتين علامتين أيضا



## أنواع الأخطاء في الكود البرمجي للجافا Programming Errors

### ١- الخطأ الهجائي Syntax Error

يظهر عند كتابة الأوامر بشكل خطأ أو عند وجود خطأ في قواعد اللغة، فمثلاً عند كتابة جملة For بدون Next أو العكس، وعند كتابة جملة if بدون End if ، وعند كتابة الأمر println بحرف ناقص ، ويظهر هذا الخطأ بشكل سريع ويسهل علاجه ، ويمكن تقليل هذه الأخطاء بالاستعانة بالقائمة المنسدلة التي تظهر عند كتابة الحرف الأول من الأمر المطلوب كتابته أو بقراءة التعليمات التي تظهر عند الضغط على رمز علامة التعجب التي تظهر بالسطر الذي يوجد به الخطأ

وبالتالي فإن الخطأ الهجائي يمثل خطأ في كتابة الكود مثل فتح قوس ونسيان غلقه، كتابة امر خطأ، عدم وضع فاصلة منقوطة في نهاية سطر الكود، وضع علامة تنصيص Double Quotation ولم يغلقها كما يلي:

```
1 public class ShowSyntaxErrors {
2     public static main(String[] args) {
3         System.out.println("Welcome to Java");
4     }
5 }
```

ويتم اكتشاف ذلك الخطأ أثناء عملية الترجمة للكود Detected by compiler حيث انه في حاله وجود خطأ من النوع Syntax Errors وليكن نسيان الفاصلة المنقوطة عندئذ تظهر رسالة خطأ ويقف المؤشر عند أول سطر يتضمن ذلك الخطأ

### ٢- خطأ أثناء التشغيل Runtime error

يظهر هذا الخطأ عند تجريب البرنامج فقط بينما لا يظهر أثناء كتابة البرنامج فمثلاً: عند كتابة أمر فتح قاعدة بيانات غير موجودة أو إعطاء أمر فتح ملف من مشغل أقراص والمشغل غير جاهز عندئذ تظهر الرسالة Runtime error

وهنا الكود مكتوب بشكل صحيح ولا يعترض عليه Compiler ولكن عند تنفيذ الكود تظهر مشكلة غير متوقعة تتسبب في خروج أو قفل البرنامج Causes the program to abort ، مثال لذلك القسمة على صفر infinity Division by zero تكون النتيجة مالاً نهاية infinity

```
public class ShowRuntimeErrors {
    public static void main(String[] args) {
        System.out.println(1 / 0);
    }
}
```

### ٣- الخطأ المنطقي Logical error

لا يعطى هذا النوع أي رسائل خطأ بل قد ينفذ البرنامج بشكل جيد وبدون مشاكل ولكن الخطأ يكون في النتيجة التي يعطيها البرنامج مثل وضع علامة الجمع بدلا من علامة الطرح وبالتالي يتم تنفيذ الكود بدون مشاكل ولكن يقوم بإخراج نتائج تكون غير المتوقعة منه

```
1 public class ShowLogicErrors {
2     public static void main(String[] args) {
3         System.out.println("Celsius 35 is Fahrenheit degree ");
4         System.out.println((9 / 5) * 35 + 32);
5     }
6 }
```

الكود السابق يقوم بتحويل الدرجة من مئوية Celsius الى فهرنهايت Fahrenheit ، وتم كتابة المعادلة التي تقوم بذلك العملية ولكنها مكتوبة بشكل خطأ ، فالحاسب لا يعلم مدى صحة تلك المعادلة ويقوم بتنفيذها كما تم إدخالها ولكنها ستعطي نتائج خاطئة

مثال:

تعريف كلاس بالاسم Person يحتوي على ٤ متغيرات بالإضافة إلى دالة تعرض قيم هذه المتغيرات عندما يتم استدعاءها

```
1 class Person {
2     String name;
3     String sex;
4     String job;
5     int age;
6     void printInfo() {
7         System.out.println("Name: " + name);
8         System.out.println("Sex: " + sex);
9         System.out.println("Job: " + job);
10        System.out.println("Age: " + age);
11    }
12 }
```

### ٤-الاستثناء Exception

الاستثناء في البرمجة يعني وقوع حدث أثناء تنفيذ البرنامج مما يؤدي لتعطيل التسلسل الطبيعي لتعليمات Instruction البرنامج ، فالاستثناء يمثل حدوث خطأ ما وهذا الخطأ ليس في الكود ولكن قد يكون من مصادر أخرى مثل عيب في المكونات المادية للجهاز أو عيب في البرنامج ، وهناك أنواع عديدة من الاستثناءات exception مثل IOException وتعني خطأ في أجهزة الإدخال أو الإخراج والخطأ NumberFormatException (وهي أخطاء التحويل من String إلى رقم ( int , long,..... )

وعند وقوع الاستثناءات ( خلال تنفيذ أحد الدوال Method في الـ Java ) تقوم الدالة بإنشاء هدف Object من نوع الاستثناء Exception Object وتميرير هذا الهدف إلى نظام وقت التنفيذ Runtime System ، وهذا الهدف Exception Object يحتوى على المعلومات الخاصة بالاستثناء (الخطأ) Runtime وهذه المعلومات تشتمل على نوع الاستثناء وحالة البرنامج عند حدوث هذا الخطأ ، عندئذ يكون نظام وقت التنفيذ Runtime System مسئول عن إيجاد جزء الكود المسئول عن معالجة هذا الاستثناء (الخطأ)

وفى لغة الـ Java تسمى عملية " إنشاء هدف الاستثناء Exception Object وتميريره إلى نظام وقت التنفيذ Runtime System بإرسال الاستثناء Throwing an Exception ، وبعد أن تقوم الدالة Method بإرسال استثناء Throw Exception ينطلق نظام وقت التنفيذ Runtime System للبحث عن من يقوم بمعالجة الاستثناء وهو مجموعة من الدوال الموجودة داخل الدالة التي حدث بها الاستثناء ، وهذه الدوال يتم شحنها عند تنفيذ البرنامج إلى منطقة بالذاكرة تسمى Call Stack فيقوم نظام وقت التنفيذ بالبحث تراجعا Backwards في الـ Call Stack بادنا بالدالة التي حدث فيها الاستثناء حتى يجد الدالة التي بها معالج الاستثناء المناسب Exception Handler

ويعرف نظام وقت التنفيذ أن هذا المعالج Exception Handler هو المناسب لو كان نوع الاستثناء الملقى Throw Exception هو من نفس نوع معالج الاستثناء Exception Handler لذلك لو تخيلنا أن الاستثناء عبارة عن كرة تلقى من أسفل إلى أعلى في Call Stack الذي هو مجموعة من الدوال تحتوى على معالجات Handlers ستجد أن تلك الكرة تصعد وتستمر في الصعود حتى تجد الدالة التي بها المعالج المناسب فيقوم بإمسك الاستثناء ومعالجته ، ويطلق على معالج الاستثناء Exception Handler المختار الاسم ملتقط الاستثناء Catch The Exception ، وفى حالة إذا قام نظام وقت التنفيذ بالبحث في جميع الدوال الموجودة في Call Stack ولم يجد معالج الاستثناء Exception Handler المناسب فانه نظام وقت التنفيذ يقوم بإنهاء البرنامج وإزالته من الذاكرة

### المزايا التي يحصل عليها البرنامج باستخدام الاستثناءات

١- فصل كود معالجة الأخطاء عن باقي الكود : حيث كانت عملية اكتشاف الأخطاء وتصنيفها ومعالجتها في البرمجة التقليدية تؤدي إلى تعقيد الكود

٢- نشر الأخطاء ونقلها إلى Call Stack

٣- تجميع وتصنيف كل خطأ بحيث يكون ينتمي كل خطأ إلى مجموعة محددة حيث تنقسم الاستثناءات إلى مجموعات وتختص كل مجموعة بمجموعة من الأخطاء ، ونظرا لأن كل شئ في الـ Java عبارة عن فئات Classes فإن الاستثناءات في الـ Java هو فئات Classes وكل فصيلة تختص بنوع من الاستثناءات وجميع هذه الفئات تشتق من الفصيلة Throwable

## أدوات معالجة الاستثناءات (الأخطاء) في لغة Java

يتم ذلك من خلال الكلمات المحجوزة التالية: `Finally, try, catch, throw, throws` وسوف نتناول هنا التركيب `try, catch` فقط كما يلي:

### -التركيب `try-----catch`

يمثل ذلك التركيب أول أدوات معالجة الاستثناءات أو الأخطاء (`exception`) وهما من أسهل الأدوات، كما تنحصر أوامر القراءة والكتابة بين الكلمتين `try, catch` (تعني جرب القراءة أو الكتابة وإذا لم تنجح فامسك الخطأ الذي تسبب في عدم النجاح في تلك العملية ) وتأخذ الصيغة التالية :

```
1 Try {
2   BufferedReader in = new BufferedReader (new InputStreamReader (System.in));
3   String s = in.readLine(); //اقرأ سطر من لوحة المفاتيح
4 } catch (Exception e) {
5   -----ماذا نريد أن يفعل اذا وجد خطأ في العملية-----
6 }
```

حيث أن ( `e` ) اسم يمكن تغييره يعبر عن خطأ عام أي لم نحدد أي نوع من الأخطاء يجب أن يمسك به البرنامج  
هذا ويمكن طباعة الخطأ على الشاشة كما يلي :

```
1 Try {
2   BufferedReader in = new BufferedReader (new InputStreamReader (System.in));
3   String s = in.readLine(); //اقرأ سطر من لوحة المفاتيح
4 } catch (Exception e) {
5   System.out.println("After Exception"); // يطبع الخطأ
6 }
```

مثال

```
1 Try {
2   الكود الذي يحدث الاستثناء //
3 } catch (Exception Object Declaration)
4 {
5 }
```

مثال:

```

1 class Excp
2 {
3
4 }
5 public class Main {
6     /**
7      * @ param args the command line arguments
8      */
9     public static void main(String[] args(
10 {
11     int d=0;
12     int a=60/d;
13     System.out.println("After Exception");
14 }
15 }

```

وعند تنفيذ ذلك الكود نلاحظ أن المترجم قام بترجمته بنجاح ولكن أظهر الرسالة التالية :

```

Output - JavaApplication12 (run)
run:
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at javaapplication12.Main.main(Main.java:23)
Java Result: 1
BUILD SUCCESSFUL (total time: 1 second)

```

تخبرنا تلك الرسالة بما يلي:

-وجود خطأ أثناء وقت التنفيذ ولاكتشاف الخطأ نقرأ الرسالة فنجد أنه حدث استثناء Exception وهذه هي أول كلمة في الرسالة لكن أين حدث الاستثناء؟ ، نجد أن الرسالة تنص على أنه في Thread main أي أنه حدث خطأ عند تنفيذ الدالة الرئيسية main ونجد أن هذا الاستثناء من النوع Arithmetic وقد نتج عن محاولة القسمة على صفر كما تدلنا الرسالة السابقة على اسم الفصيلة التي حدث بها الاستثناء كما هو واضح بين الأقواس

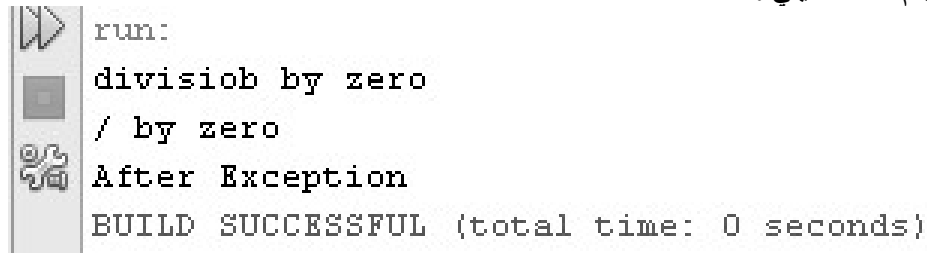
نستنتج مما سبق أن الدالة main وجدت بها استثناء Exception وقد ألقت الاستثناء إلى نظام وقت التنفيذ Runtime system وقام نظام وقت التنفيذ بالبحث عن يعالج هذا الاستثناء فلم يجد فالقي الاستثناء ثم خرج من النظام، كما نجد أنه لم يصل إلى السطر ١٣ لأنه توقف عند الاستثناء الحادث ولمعالجة هذا الاستثناء نكتب البرنامج كما يلي:

```

1 class Excp
2 {
3 }
4 public class Main {
5     /**
6      * @ param args the command line arguments
7      */
8     public static void main(String[] args)
9     {
10         Try{
11             int d=0;
12             int a ;
13             a=60/d ;
14         } catch(ArithmeticException e)
15         {
16             System.out.println("divisiob by zero");
17             System.out.println(e.getMessage( ));
18         }
19         System.out.println("After Exception");
20     }
21 }

```

ونلاحظ عند تنفيذ ذلك الكود أن المترجم قام بترجمته بنجاح بدون ظهور رسائل خطأ كما قام بتنفيذ السطر رقم ٢٠ كما يلي :



```

run:
divisioib by zero
/ by zero
After Exception
BUILD SUCCESSFUL (total time: 0 seconds)

```

ويرجع السبب في ذلك إلى أن الدالة أخبرت المترجم بأن يحاول تنفيذ الكود في الجزء Try Block وإذا حدث استثناء (وهو ما حدث) يتم التقاطه من خلال الجزء Catch Block وتم اظهر الرسالة التي تفيد حدوث القسمة على صفر وأخيرا ظهور نتيجة السطر ٢٠ لان الاستثناء الحادث قد تم التقاطه ومعالجته.

### إرشادات لمعالجة الأخطاء في الجافا

بعض الأمثلة التي تتضمن أخطاء مقصودة للتعرف على طبيعة الخطأ وبالتالي كيفية معالجته

مثال:

عمل `Declare` لمتغير جديد ثم عرض قيمته وهو فارغ.

```
1 int a;  
2 System.out.println(a);
```

لاحظ الصورة التالية لتفهم كيف يقوم برنامج `Netbeans` بإبلاغك بالخطأ قبل أن تقوم بتشغيل البرنامج.

الشيء الموضوع تحته خط رمادي يعني أنه وجد شيء غير مستخدم، أي لا حاجة له. مرر الماوس فوقه و سيخبرك بالسبب كما في الصورة التالية

```
variable a might not have been initialized  
----  
(Alt-Enter shows hints)
```

int a;  
  
System.out.println(a);

هذه الإشارة تعني أنه وجد خطأ في هذا السطر. مرر الماوس فوق اللمبة و سيخبرك ما الخطأ كما في الصورة التالية

```
Variable a is not used
```

الشيء الموضوع تحته خط أحمر يعني أنه هو سبب الخطأ. و هو يظهر نفس رسالة اللمبة عند تمرير الماوس فوقه

```
Variable a is not used
```

إذاً المشكلة الوحيدة هنا أننا حاولنا عرض محتوى متغير فارغ

ويخبرنا البرنامج أن المشكلة سببها المتغير `a` لأننا لم نفعل له `initialize` ووضع تحته خط أحمر للتنبيه لأنه إذا بقي هذا الخط ظاهراً وقمنا بتشغيل البرنامج فسيظهر لنا خطأ و يخبرنا في أي سطر وجده و ما الذي سببه.

## الفصل الثاني: البيانات والمتغيرات والثوابت والمعرفات والاختفاء

```
Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - variable a might not have been initialized
    at harmash.Harmash.main(Harmash.java:10)
Java Result: 1
```

هنا نخبرنا في أي سطر وجد الخطأ

هنا نخبرنا نوع الخطأ

ولإصلاح هذا الخطأ يجب إعطاء المتغير `a` قيمة قبل عرض قيمته ، الجملة `Variable is not used` تعني أننا حتى الآن لم نستخدم هذا المتغير أي يمكننا الاستغناء عنه.

مثال:

أحياناً يستخدم المبرمج مباشرة اسم متغير دون أن يكون قد قام بتعريف هذا المتغير أي لم يفعل له `Declare` ، هنا سنقوم بعرض قيمة متغير غير موجود إطلاقاً.

```
System.out.println(a);
```

في هذا الحالة سيظهر لك هذه الرسالة إذا وضعت الماوس فوق اللمبة.

```
4
5
6
7
8
cannot find symbol
symbol: variable a
location: class Harmash
----
(Alt-Enter shows hints)
System.out.println(a);
```

المشكلة عدم تعريف المتغير `a` قبل وضعه في الدالة.