Road to Java

From VB.NET

Written by:
Josip Medved <jmedved@jmedved.com>

Thanks to:
DIR <www.dir.hr>

0.01 2004-12-06

Contents

Foreword

This bunch of letters is made out of my personal need. For long I have been VB and then VB.NET programmer with limited experience in C-like languages. But suddenly I got myself in situation that requires Java. I took some books and started learning but none of them was satisfactory. They all either treated me as an idiot or they treated me as an expert and all I wanted is someone to tell me the difference (between VB.NET 2003 and Java not between idiot and expert).

This script is intended for VB.NET non-beginners trying to understand Java. Some things will be described in details but for some there is need for basic programming knowledge. I will not waste my keyboard on some basic stuff like what are data types or what is compiler.

There is hidden agenda also - to improve my English so don't get mad when you stumble on weird looking words and tenses - just contact me and we'll see what to do about that.

P.S. Since this script got to it's existence because of my own itch, there is no scope defined. I will write for as long I feel like it and it will cover all things I think it should. If you have itch too, you may contribute some text or just correct me.

Elementary, dear Data

For kickoff here is table of elementary VB.NET data types and their equivalent in Java.

VB.NET			***Java***		
Boolean	False or True	2 b	boolean	False or True	-
Byte	0 to 255	1 b	byte	-128 to 128	1 b
Char	0 to 65,535	2 b	char	Unicode 0 to Unicode 2^{16}-1	2 b
Date	0001-01-01 to 9999-12-31	8 b	Date class	-	-
Decimal	depends	16 b	BigDecimal class	depends	-
Double	IEEE754	8 b	double	IEEE754	8 b
Integer	-2^{31} to 2^{31}-1	4 b	int	-2^{31} to 2^{31}-1	4 b
Long	-2^{63} to 2^{63}-1	8 b	long	-2^{63} to 2^{63}-1	8 b
Object class	-	-	-	-	-
Short	-2^{15} to 2^{15}-1	2 b	short	-2^{15} to 2^{15}-1	2 b
Single	IEEE754	4 b	float	IEEE754	4 b
String class	approx. 2G Unicode characters	-	String class	approx. 2G Unicode characters	-
does not exist			void	-	-

Biggest difference lays in way variables are declared:
Instead `Dim x As Integer` we use `int x;`. This is most difficult thing to get adjusted to (along with semicolon on end of each line). Also note that Java types are case sensitive (as whole language is) and `Int x;` is not same as `int x;`. Forgetting this can lead to notorious bugs.

Those who didn't work in any C-like languages may notice void data type. This is usually used as indication for function that returns nothing - we have Sub for that.

Initialization

All variables in class body default to same values as in VB.NET (0 for integers, false for booleans and so on). But variables defined in function defaults to random values as in C/C++. Big difference here is compiler insisting on assigning default value. This can be done similar to VB.NET:

```
int a = 23;
int b = 0x17;
int c = 027;
```

Here we can see three ways of defining default values. In a case we assign simple integer value to variable. In second example we can see that same value can be defined in hexadecimal notation (&H17 in VB.NET). Third case shows octal value. You can define octal numbers by using 0 as first number. In real life there is no or very little need for it since octal numbers are very rare.

Star days

In .NET there is single System.DateTime date type for dealing with all problems. This is not case in Java. There is java.util.Date class but most of functionality we are used to have is deprecated and not to be used. To create date variable that will hold our date proper way is to use java.util.Calendar class:

```
java.util.Calendar calendarVar =
                           java.util.Calendar.getInstance();
calendarVar.set(2004,12,6,0,0,0);
java.util.Date dateVar = calendarVar.getTime();
```

As you may see this is way too complicated for normal use (and don't forget to put these 0's - current time is used if they are not there). Reason is internationalism concept that wasn't there in Java so they patched it. There is danger lurking for VB programmers in way of getting month from date also:

```
int monthIndex = cal.get(java.util.Calendar.MONTH);
```

It seems not too complicated but big surprise is that monthIndex is 0 for January, 1 for February and so on. You read it right. Even months start from zero. This is too much for me.

There is no swap

From the old VB days we had ByVal and ByRef. They had been misused through history but sometimes you just had need for them. For example, swap routines could be made really easy with them. In Java there is no swap. Language it self has capability to transfer something ByRef but it is forbidden for programmer to do so. Only proper way to return parameter is function. No more than one parameter is to be returned. Remember this.

Basic

Any comment?

To make comments in VB.NET we used apostrophe character (') or Rem keyword. In Java there are more options. First is to use /* and */ pair. All in between these characters is considered to be remark. This is useful for making multi line comments. We can also use // to comment remainder of row. This acts exactly as ' in VB.NET. Last style is used to produce JavaDoc comments. It begins with /** and ends with */. Things in between are considered to be part of documentation.

Operate me

Standard arithmetic operators are:

Operator	***VB.NET***	***Java***
Addition	`c = a + b`	`c = a + b;`
Substraction	`c = a - b`	`c = a - b;`
Multiplication	`c = a * b`	`c = a * b;`
Division	`c = a / b`	`c = a / b;`
Integer division	`c = a \ b`	`c = a / b;` make sure that both variables are integers
Division remainder	`c = a Mod b`	`c = a % b;`
Power	`c = a ^ b`	-

If we don't need another variable and we need to keep things short:

Operator	***VB.NET***	***Java***
Addition	`c += a`	`c += a;`
Substraction	`c -= a`	`c -= a;`
Multiplication	`c *= a`	`c *= a;`
Division	`c /= a`	`c /= a;`
Integer division	`c \= a`	`c /= a;` make sure that both variables are integers
Division remainder	-	`c %= a;`
Power	-	-

Incrementing or decrementing by one is special story:

Operator	***VB.NET***	***Java***

Incrementing	`c += 1`	`c++;`
		`++c;`

Substraction	`c -= 1`	`c--;`
		`--c;`

Note that these are not all that Java or VB.NET can offer but they are most often used.

Logic is logic

Relational operators create boolean result:

Operator	***VB.NET***	***Java***
Equivalent	`a = b`	`a == b`
Not equivalent	`a <> b`	`a != b`
Less than	`a < b`	`a < b`
Less than or equal	`a <= b`	`a <= b`
Greater than	`a > b`	`a > b`
Greater than or equal	`a >= b`	`a >= b`

Since all things are almost same, only problems can arise by forgetting correct equivalence operators syntax.

Boss me around

What If

If is most basic instruction and one way to control program flow:

Form	VB.NET	Java
One liner	`If a = b Then x`	`if (a==b) x();`

Simple

VB.NET

```
If a = b Then
   x
End If
```

Java

```
if (a==b) {
   x();
}
```

With else

VB.NET

```
If a = b Then
   x
Else
   y
End If
```

Java

```
if (a==b) {
   x();
} else {
   y();
}
```

Worst case scenario

VB.NET

```
If a = b Then
   x
ElseIf c = d Then
   y
Else
   z
End If
```

Java

```
if (a==b) {
   x();
} else if (c==d) {
   y();
} else {
   z();
}
```

You must notice () around operators. There is also possibility to leave out {} when only one statement is used. Note that this is not recommended practice.

Flip the switch

Sometimes you need to do branching based on one variable. For that we have Select Case in VB.NET and switch in Java.

Form	VB.NET	Java

Basic

VB.NET

```
Select Case a
   Case 1: x
   Case 2: y
   Case Else: z
End Select
```

Java

```
switch (a) {
   case 1: x(); break;
   case 2: y(); break;
   default: z(); break;
}
```

Form	VB.NET	Java
Not so basic		-

VB.NET:

```
Select Case a
   Case 1, 2, 3: x
   Case 4 To 10: y
   Case Else: z
End Select
```

Java: -

Here we have few hidden dangers. One of them is break; statement after each branch. Without that statement, all cases would evaluate. While this seems pretty stupid to me, it is standard in C-like languages. Also note that there is no equivalent for second form of behavior. If you have such needs, you are stucked with if.

What for

Form	VB.NET	Java
Simple		
Double barrel	-	

Simple, VB.NET:

```
For i As Integer = 0 To 9
   x(i)
Next
```

Simple, Java:

```
for(int i=0; i<=9; i++) {
   x(i);
}
```

Double barrel, VB.NET: -

Double barrel, Java:

```
for(int i=0, int j=0; i<=9; i++,j++) {
   x(i,j);
}
```

While simple form is supported in both languages. Complicated form where we increment two variables at time is supported only in Java. This is rarely needed but it is nice to have. Also note that separator for if is semicolon (;), not comma (,). If you accidentally use comma, you may find your self in trouble.

While

Form	VB.NET	Java
Basic		

Basic, VB.NET:

```
While a = b
   x
End While
```

Basic, Java:

```
while (a==b) {
   x();
}
```

As you can see, thing are not that different. Notice that this is same as Do While ... Loop statement in VB.NET but this form is used since it is closer in words.

Do

Form	***VB.NET***	***Java***
Basic		
Opposite		-

Basic

```
Do
   x
Loop While a = b
```

```
do {
   x();
} while (a==b);
```

Opposite

```
Do
   x
Loop Until a <> b
```

-

Notice that there is no until form in Java but that can be countered with clever use of not operator.

Something to end with

We got used to using Exit For, Exit While, Exit Do and similar statements. In Java you've got single break statement to get you out of path. Also there is countinue statement which will continue with next value:

```
for (int i=0; i<10; i++) {
   if (i%2==0) continue;
   System.out.println(i);
}
```

Result of that will be:

```
1
3
5
7
9
```

This is one of rare things that I really miss in VB.NET.

Let's go

Standard starting point of all programs is main function in selectable class:

```
public class Hello {
   public static void main(String[] args) {
      System.out.println("Hello world!");
   }
}
```

That same code in VB.NET would be:

```
Public Class Hello
   Public Shared Sub Main()
      System.Console.WriteLine("Hello world!")
   End Sub
End Class
```

After we get used to using } as single replacement for whole bunch of our End constructs and new way of declaring sub's only thing left is learning new namespaces.

Recommended reading

Bruce ECKEL: Thinking in Java

Definitely one of best books ever. Can be find at www.bruceeckel.com. It is put here because it is The Book for Java.

Ian F. DARVIN: Java Cookbook

Can be used as fast lookup for Java patterns.

David FLANAGAN: Java Examples in a Nutshell

Lots of examples, most of them are only basics but it is useful as reference.

Steve HOLZNER: Eclipse

Pretty good book on Eclipse IDE.

Robert SIMMONS, Jr: Hardcore Java

Some useful implementation details.