C# for Java Developers

Microsoft® Visual Studio® 2010

Keywords

Java	C#	Java	C#
abstract	abstract	native	extern
assert	Debug.Assert (*method*)	new	new
break	break	null	null
case	case	package	namespace
catch	catch	private	private
class	class	protected	internal
const	const	public	public
continue	continue	return	return
default	default	static	static
do	do	strictfp	n/a
else	else	super	base
enum	enum	switch	switch
extends	:	synchronized	lock
false	false	this	this
final	sealed	throw	throw
finally	finally	throws	*n/a*
for	for / foreach	transient	[Nonserialized] (*attribute*)
goto	goto	true	true
if	if	try	try
implements	:	… (*varargs*)	params
import	using	void	void
instanceof	is	volatile	volatile
interface	interface	while	while

Note: The `const` and `goto` keywords in Java have no function.
The C# `const` and `goto` keywords are operational.

C# keywords not found in Java, or with different behavior

Keyword	Description	Keyword	Description
base	Provides access to members in a parent class	operator	Declares an overloaded operator
checked	Enables arithmetic overflow checking	out	Declares an output parameter (method call) Declares a covariant type parameter(generic interface)
delegate	Declares a type-safe reference to a method	override	Declares a method that overrides a method in a base class
event	Declares an event	protected	Declares a member that is accessible only within the class and descendant classes (different semantics from Java protected keyword)
explicit	Declares a narrowing conversion operator	readonly	Declares a field to be read-only
implicit	Declares a widening conversion operator	ref	Declares a reference to a value type
in	Declares a contravariant type parameter for a generic interface	struct	Defines a new value type
new	Declares a method that hides a method in a base class (method modifier)	virtual	Declares a member that can be overridden

Operators

Java	C#	Description
x.y	x.y	Member access "dot" operator
f(x)	f(x)	Method invocation operator
a[x]	a[x]	Array element access operator
++, --	++, --	Increment and decrement operators (pre and post-fix)
new	new	Object instantiation operator
instanceof	is	Type verification operator
(T)x	(T)x	Explicit cast operator
+, -	+, -	Addition and subtraction operators (binary). Positive and negative operators (unary)
+	+	String concatenation operator
!	!	Logical negation operator
&&, ||	&&, ||	Conditional AND and OR operators (short-circuited evaluation)
&, |, ^	*n/a*	Conditional AND, OR, and XOR operators (full evaluation of operands)
~	~	Bitwise complement operator
&, |, ^	&, |, ^	Bitwise AND, OR, and XOR operators
<<, >>	*n/a*	Signed left-shift and right-shift operators
>>>	>>	Unsigned right-shift operator
*, /, %	*, /, %	Multiply, divide, and modulus operators
==, !=	==, !=	Is-equal-to and is-not-equal-to operators
<, >, <=, >=	<, >, <=, >=	Relational less-than, greater-than, less-than-or-equal-to, and greater-than-or-equal-to operators
x?y:z	x?y:z	Conditional operator
=	=	Assignment operator

C# operators not available in Java:

Operator	Description	Operator	Description
<<	Unsigned left-shift operator	default(T)	Returns the default value for a type (generics)
<<=	Unsigned left-shift compound assignment operator	sizeof(T)	Returns the size of a type
??	Null-coalescing operator	stackalloc	Allocates a block of memory on the stack
=>	Lambda expression operator	typeof(e)	Returns the type of an expression
as	Safe casting operator	unchecked	Disables arithmetic overflow checking
checked	Enables arithmetic overflow checking	unsafe	Enables unsafe code

C# identifiers that should not be used as names for types, methods, or variables:

dynamic	join	set
from	let	value
get	orderby	var
group	partial	where
into	select	yield

Common Datatypes

Java	C#	Java Range	C# Range
boolean	bool	true of false	true/false
byte	sbyte	-128 to 127	-128 to 127
char	char	0 to 2^{16} - 1	0 to 2^{16} - 1
double	double	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$
float	float	$\pm 1.5 \times 10^{45}$ to $\pm 3.4 \times 10^{38}$	$\pm 1.5 \times 10^{45}$ to $\pm 3.4 \times 10^{38}$
int	int	2^{31} to 2^{31}-1	-2^{31} to 2^{31}-1
long	long	-2^{63} to 2^{63}-1	-2^{63} to 2^{63}-1
short	short	-2^{15} to 2^{15}-1	-2^{15} to 2^{15}-1
String	string	*n/a*	*n/a*
Object	object	*n/a*	*n/a*
Date	DateTime	1st Jan 1970 to *implementation dependent value*	1st Jan 0001 to 31st Dec 9999

Datatype examples

Java	C#
`int i = 1;`	`int i = 1;`
`byte b = 1;`	`byte b = 1;`
`double d = 1.1;`	`double d = 1.1;`
`long l = 1;`	`long l = 1;`
`short s = 1;`	`short s = 1;`
`boolean found = true;`	`bool found = true;`
`char c = 'z';`	`char c = 'z';`
`String title = "Hello World";`	`string title ="Hello World";`

Array examples

Java	C#
`int[] data1;`	`int[] data1;`
`int data2[];`	*n/a*

C# datatypes not provided as part of Java

Type	Range	Size(bits)
decimal	28-29 significant figures	128
byte	0 to 255	8
uint	2^{32}-1	32
ulong	0 to 2^{64} -1	64
ushort	0 to 2^{16}-1	16

Java wrapper classes mapped to C# structs

Java Wrapper	C# Struct
Boolean	Boolean
Byte	Byte
Character	Char
n/a	Decimal
Double	Double
Float	Float
Integer	Int32
Long	Int64
Short	Int16
Void	*n/a*

Development Areas

Microsoft® .NET

Windows®

Windows Phone®

Windows Azure™

Microsoft® Silverlight®

C# Features Not Available in Java 6

- LINQ
- Delegates
- Events
- Properties
- Indexers
- Lambda Expressions
- Partial Classes
- Dynamic Runtime Language
- Structures
- Operator Overloading
- Partial Methods
- Optional Parameters

C# for Java Developers

Microsoft® Visual Studio® 2010

Program and Class Example

C#

```
// Bring System namespace into scope
// - contains the Console class
using System;

namespace Example
{
    // Program does not have to be in Program.cs
    // Main method does not have to be in a public class
    class Program
    {
        // Main method has a capital "M"

        static void Main(string[] args)
        {
            Car myCar = new Car();
            myCar.Drive();
        }
    }

    // Abstract base class
    abstract class Vehicle
    {
        public virtual void Drive()
        {
            // Default implementation
        }
    }

    // Car inherits from Vehicle
    class Car : Vehicle
    {
        // Override default implementation
        public override void Drive()
        {
            Console.WriteLine("Car running");
        }
    }

}
```

Java

```
package Example;

// Optional import
import java.lang.*;

// Program class must be in a file called Program.java
// The main method must be defined in a public class

public class Program {

    public static void main(String[] args){

        Car myCar = new Car(); myCar.Drive();
        }
}

// Abstract base class
abstract class Vehicle{

    public void Drive() {
        // Default implementation
    }
}

// Car inherits from Vehicle
class Car extends Vehicle {

    // Override default implementation
    public void Drive() {

        System.out.println("Car running");
    }
}
```

C# Struct and Enum Example

```
enum Month
{
    January, February, March, April, May, June,
    July, August, September, October, November, December
}

struct Date
{
    private int year;
    private Month month;
    private int day;

    public Date(int ccyy, Month mm, int dd)
    {
        this.year = ccyy - 1900;
        this.month = mm;
        this.day = dd - 1;
    }

    public override string ToString()
    {
        string data = String.Format("{0} {1} {2}",
          this.month,this.day + 1 ,this.year + 1900);
        return data;
    }

    public void AdvanceMonth()
    {
        this.month++;
        if (this.month == Month.December + 1)
        {
            this.month = Month.January;
            this.year++;
        }
    }
}
```

Interface Example

C#

```
interface IDateCalculator
{
    double DaysBetween(DateTime from,DateTime to);
    double HoursBetween(DateTime from,DateTime to);
}

class DateCalculator : IDateCalculator
{
    public double DaysBetween(DateTime start,DateTime end)
    {
        double diffInDays =
            end.Subtract(start).TotalDays;
        return diffInDays;
    }

    public double HoursBetween(DateTime start,DateTime end)
    {
        double diffInHours =
            end.Subtract(start).TotalHours;
        return diffInHours;
    }
}
```

Java

```
interface IDateCalculator {
    double DaysBetween(Date from,Date to);
    double HoursBetween(Date from,Date to);
}

class DateCalculator implements IDateCalculator
{
    public double DaysBetween(Date start,Date end) {
        long startTimeMs = start.getTime();
        long endTimeMs = end.getTime();
        double diffInMs = endTimeMs - startTimeMs;
        double diffInDays =
            diffInMs / (24 * 60 * 60 * 1000);
        return diffInDays;
    }

    public double HoursBetween(Date start,Date end) {
        long startTimeMs = start.getTime();
        long endTimeMs = end.getTime();
        long diffInMs = endTimeMs - startTimeMs;
        double diffInHours =
            diffInMs / (60 * 60 * 1000);
        return diffInHours;
    }
}
```

C# Switch Example

```
string month;

switch (month) // string variables supported
{
    case "Jan":
        monthNum = 1;
        break; // doesn't allow fall through
    case "Feb":
        monthNum = 2;
        break;
    default:
        Console.WriteLine("Invalid Value");
        break;
}
```

C# Delegate/Lambda Example

```
class Program
{
    delegate int Calculation(int a, int b);

    static void Main(string[] args)
    {
        int x = 2;
        int y = 3;
        Calculation add = (a, b) => { return a + b; };

        int answer = add(x, y);

        Console.WriteLine(answer); // output: 5
    }
}
```

Properties Example

C#

```
class Circle
{
    private double radiusCM;
    public double RadiusMeters
    {
        get { return radiusCM / 100; }
        set { radiusCM = value * 100; }
    }
}

class Program
{
    static void Main(string[] args)
    {
        Circle myCircle = new Circle();

        myCircle.RadiusMeters = 50;
        double radius = myCircle.RadiusMeters;
    }
}
```

Java

```
public class Circle {

    Properties property;
    double radiusCM;

    public Circle()
    {
        property = new Properties();
    }
    void setRadiusMeters(int rad)
    {
        radiusCM = rad /(double)100;
        property.setProperty("RadiusMeters",
            Double.toString(radiusCM*100));
    }
}

public class Program {

    public static void main(String[]args){
        Circle circ = new Circle();

        circ.setRadiusMeters(50);
        double radius =
              new Double((String)
                  circ.property.get("RadiusMeters"));
    }
}
```

C# LINQ Example

```
using System;
using System.Linq;

class Program
{
    static void Main(string[] args)
    {
        string[] names = { "John", "Jim", "Bert",
        "Harry"};

        var matchingNames = from n in names
        where n.StartsWith("J") select n;

        foreach (string match in matchingNames)
        Console.WriteLine(match);
    }
}
```