



COMPSCI 280 S1 2011 Enterprise Software Development

Programming Fundamentals
Operators & Expressions



Agenda & Reading

- ▶ **Agenda:**
 - ▶ Operators
 - ▶ Arithmetic
 - ▶ Increment/Decrement
 - ▶ Logical
 - ▶ +
 - ▶ Shifting
 - ▶ Assignment
 - ▶ Comparison
 - ▶ The Math Methods
 - ▶ Operators Precedence
- ▶ **Recommended Reading:**
 - ▶ Operators (C# Programming Guide)
 - ▶ <http://msdn2.microsoft.com/en-us/library/ms173145.aspx>
 - ▶ C# for Java Programmers
 - ▶ Chapter 3
 - ▶ Microsoft Visual C# 2008 Step by Step
 - ▶ Chapter 2
- ▶ **Hands-On Lab:**
 - ▶ Lecture05Lab



Operators & Expressions

- ▶ **Operators**
 - ▶ Operators are special symbols used for mathematical functions, some types of assignment statements, and logical comparisons.
 - ▶ Types of Operators
 - ▶ Arithmetic
 - ▶ Assignment
 - ▶ Comparison
 - ▶ Concatenation
 - ▶ Logical
- ▶ **Expressions**
 - ▶ An expression is a statement that can convey a value
 - ▶ An expression can be any combination of variables, literals, and operators.
 - ▶ They also can be method calls, because methods can send back a value to the object or class that called the method.



Arithmetic Operators

▶ Arithmetic

Operator	Meaning	Example
+	Addition	3 + 4
-	Subtraction	5 - 7
*	Multiplication	5 * 5
/	Division	14 / 7
%	Modulus - Remainder of division	20 % 7

23 / 5 returns 4
23.0 / 5 returns 4.6

7 % 4 returns 3
4 % 7 returns 4

▶ Increment & Decrement

- ▶ Increment/decrement its operand by 1

```
double x = 1.5;
Console.WriteLine(++x);
x = 1.5;
Console.WriteLine(x++);
```

2.5

1.5

Operator	Meaning		Meaning
++	Increment	Prefix	result is the value of the operand <u>after</u> it has been incremented
		Postfix	result is the value of the operand <u>before</u> it has been incremented
--	Decrement	Prefix	result is the value of the operand <u>after</u> it has been decremented
		Postfix	result is the value of the operand <u>before</u> it has been decremented



Logical

▶ Logical Operators

p	q	!p	p & q	p q	p ^ q
F	F	T	F	F	F
F	T	T	F	T	T
T	F	F	F	T	T
T	T	F	T	T	F

Operator	Type	Meaning
!	unary	negates its operand
&	binary	logical conjunction on two Boolean expressions (logical AND)
	binary	logical disjunction or inclusion on two Boolean expressions (logical-OR)
^	binary	logical exclusion on two Boolean expressions (exclusive-or)

▶ Short Circuit Logical Operators

- ▶ Only evaluates its second operand if necessary
- ▶ && : If p evaluates to False, then the second expression is not evaluated
- ▶ ||: If p evaluates to True, then the second expression is not evaluated

p	q	p && q	p q
F		Returns False, Q is not evaluated	F
F			T
T		F	Returns True Q is not evaluated
T		T	

```
int i=0;
bool a = 23 < 14 && 4 > 4 / i;
bool b = 23 > 14 || 4 > 4 / i;
```

Run-time error:
Divide by zero

```
int i=0;
bool a = 23 < 14 && 4 > 4 / i;
bool b = 23 > 14 || 4 > 4 / i;
```



Operators (con't)

▶ The + Operator

- ▶ For numeric types, + computes the sum of its two operands.
- ▶ When one or both operands are of type string, + concatenates the string representations of the operands

```
Console.WriteLine("5" + "5");
Console.WriteLine(5 + "5");
```

▶ Shifting Operators

Operator	Name	Meaning
<<	left-shift operator	its first operand left by the number of bits specified by its second operand
>>	right-shift operator	shifts its first operand right by the number of bits specified by its second operand

```
Console.WriteLine("0x{0:x}", 8 << 1);
Console.WriteLine(32 >> 3);
```

1000 (left shift) ->
10000 = 0x10 (16)

100000 (right shift)
-> 100 = 4



Assignment Operators

▶ Assignment operators

- ▶ A shorthand notation for performing an operation on and assigning a new value to a variable

Operator	Meaning	Example	Meaning
+=	Addition before assignment	a += b	a = a + b
-=	Subtraction before assignment	a -= b	a = a - b
*=	Multiplication before assignment	a *= b	a = a * b
/=	Division before assignment	a /= b	a = a / b
%=	Modulus before assignment	a %= b	a = a % b
&=	And assignment operator	p &= q	p = p & q
=	Or assignment operator	p = q	p = p q
<<=	left-shift assignment operator	a <<= b	a = a << b
>>=	right-shift assignment operator	a >>= b	a = a >> b
??	The ?? Operator Returns the left-hand operand if it is not null, or else it returns the right operand.	x = a ?? b	If a is null, x = b; otherwise x = a

```
object y = x ?? "IT IS NULL";
Console.WriteLine(y);
```



Comparison Operators

▶ Comparison operators

- ▶ Comparison operators compare two expressions and return a Boolean value that represents the relationship of their values.

Operator	Name	Example
<	less than	x < 10
<=	less than or equal to	x <= 10
>	greater than	y > 10
>=	greater than or equal to	y >= 10
==	equality	x == y
!=	inequality	x != y
?:	conditional operator returns one of two values depending on the value of a Boolean expression	i == 0 ? "ZERO" : "NOT ZERO"

▶ Note:

- ▶ For value types, == returns true if the values of its operands are equal
- ▶ For reference types, == returns true if its two operands refer to the same object.



Note:

```
MyPoint myp1 = new MyPoint(1, 2);
MyPoint myp2 = new MyPoint(1, 2);
Console.WriteLine(myp1 == myp2);
```

false

```
class MyPoint
{
    public int x;
    public int y;
    ...
}
```

- ▶ The equality operator, ==
 - ▶ For reference types, == returns true if its two operands refer to the same object.
 - ▶ For the string type, == compares the values of the strings

```
string s1 = "Hello";
string s2 = "Hello";
Console.WriteLine(s1 == s2);
```

true

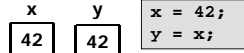
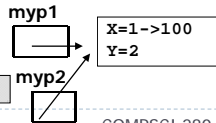
```
Point p1 = new Point(10, 20);
Point p2 = new Point(10, 20);
Console.WriteLine(p1 == p2);
```

Note: Point is a value type; return true

- ▶ The assignment operator, =
 - ▶ Value Types
 - ▶ Assignment to a variable of a value type creates a copy of the value being assigned.
 - ▶ Reference Types
 - ▶ Assignment to a variable of a reference type creates a copy of the reference rather than a copy of the referenced value.
 - ▶ So Changing the contents of the destination object would affect the contents of the source object

```
myp1 = myp2;
myp2.x = 100;
```

```
Console.WriteLine(myp1.x);
```



The Math Methods

- ▶ The following table offers a partial list of the math methods in System.Math class.

Method	Purpose
Abs(n)	Returns the absolute value of n
Atan(n)	Returns the arctangent, in radians, of n
Cos(n)	Returns the cosine of the angle n. The angle n is expressed in radians
Exp(n)	Returns the constant e raised to the power of n
Sin(n)	Returns the sine of the angle n
Sqrt(n)	Returns the square root of n
Tan(n)	Returns the tangent of the angle n
Pow(x,y)	Returns the specified number x raised to the specified power y
Round(n)	Rounds a value to the nearest integer or specified number of decimal places. If n is halfway between two integers, one of which is even and the other odd, then the even number is returned.
Round(n, mode)	Rounds a double-precision floating-point value to the nearest integer. A parameter specifies how to round the value if it is midway between two other numbers. Mode: AwayFromZero, ToEven



Example:

- ▶ Math.Round:

- ▶ To nearest integer

```
Console.WriteLine("Round(2.4)=" & Math.Round(2.4))
Console.WriteLine("Round(2.9)=" & Math.Round(2.9))
Console.WriteLine("Round(2.5)=" & Math.Round(2.5))
Console.WriteLine("Round(3.5)=" & Math.Round(3.5))
```

2
3
2
4

- ▶ AwayFromZero

```
Console.WriteLine(Math.Round(2.5, MidpointRounding.AwayFromZero))
Console.WriteLine(Math.Round(3.5, MidpointRounding.AwayFromZero))
Console.WriteLine(Math.Round(4.5, MidpointRounding.AwayFromZero))
```

3
4
5

- ▶ ToEven

```
Console.WriteLine(Math.Round(2.5, MidpointRounding.ToEven))
Console.WriteLine(Math.Round(3.5, MidpointRounding.ToEven))
Console.WriteLine(Math.Round(4.5, MidpointRounding.ToEven))
```

2
4
4

- ▶ Note:

- ▶ (int) 3.5 returns 3
- ▶ Convert.ToInt32(3.5) returns 4



Operator Precedence

- ▶ When several operations occur in an expression, each part is evaluated and resolved in a predetermined order called operator precedence.

x++, x--
Unary: +, -, !, ~, ++x, --x
*, /, %
+, -
<<, >>
<, <=, >, >=, is, as
==, !=
&, ^,
&&, , ?:
~, +=, -=, *=, /=, %=, &=, =, ^=, <<=, >>=

Note: almost the same as in Java, but:

- logical and bitwise operators of the same sort have equal precedence in Java (e.g., & and &&)
- in Java, & has precedence over ^ which has precedence over |

- ▶ Note:

- ▶ When operators of equal precedence appear together in an expression, the compiler evaluates each operation as it encounters it from left to right.
- ▶ Use parentheses to force precedence
- ▶ Do not clutter expressions with parentheses when the precedence is correct and obvious, i.e., don't do a (3 * 4) + 5 when a 3 * 4 + 5 will do – but don't rely on every reader of your code to know the table above by heart