Documentation for VB.NET: Common Errors, String & Boolean Operations, and Comparison Operators

Common Errors & Exceptions

In VB.NET, errors and exceptions arise when the program encounters unexpected conditions during execution. Two common issues are:

1. Exception:

- Refers to Exceptions. (Press Shift + F5)
- Example 1:

```
Dim y As Integer = 4
y = y / 0
```

- Example 2:

```
Dim x As Integer
x = "Ding"
```

2. NaN (Not a Number):

- This special value represents an undefined or unrepresentable numeric result.
- Example:

```
Dim y As Double = 0
y = y / 0
Console.WriteLine(y)
Console.ReadKey()
```

String & String Operators

Strings in VB.NET are sequences of characters. Below are some key string operations:

1. Concatenation (`&` Operator):

- The `&` operator is used to combine two strings.
- Example:

```vb
Dim firstName As String = "John"
Dim lastName As String = "Doe"
Dim fullName As String = firstName & " " & lastName ' Output: "John Doe"
```

2. Appending (`&=` Operator):

- The `&=` operator appends a string to an existing one.
- Example:

```vb
Dim greeting As String = "Hello"
greeting &= ", World!" ' Output: "Hello, World!"
```

3. Including Double Quotes in a String:

- To include a double quote in a string, use two consecutive double quotes.
- Example:

```vb
Dim quote As String = "She said, ""Hello!"""
Console.WriteLine(quote) ' Output: She said, "Hello!"
```

Boolean & Boolean Operators

Boolean operations deal with true/false values. These are essential for logic and decision-making in programs.

1. Boolean Literal:

- `True` and `False` are the two Boolean literals.
- Example:

```
Dim isActive As Boolean = True
```

2. Boolean Algebra:

- Refers to the use of Boolean logic in decision-making.
- Example:

```
Dim age As Integer
Dim isAdult As Boolean
Console.WriteLine("Please enter your age:")
age = Console.ReadLine()
isAdult = (age >= 18) ' Boolean expression
```

3. Boolean Operators:

- **Not:** Inverts a Boolean value.
- **And:** Returns `True` only if both conditions are `True`.
- **Or:** Returns `True` if at least one condition is `True`.
- **Xor:** Returns `True` if one (and only one) condition is `True`.
- Examples:

```
Dim a As Boolean = True
Dim b As Boolean = False

Console.WriteLine(Not a) ' Output: False
Console.WriteLine(a And b) ' Output: False
Console.WriteLine(a Or b) ' Output: True
Console.WriteLine(a Xor b) ' Output: True
```

Comparison Operators

Comparison operators are used to compare numeric values or expressions and return a Boolean result.

1. Common Comparison Operators:

- **Less than (<)**
- **Less than or equal to (<=)**
- **Greater than (>)**
- **Greater than or equal to (>=)**
- **Equal to (=)**
- **Not equal to (<>)**

2. Examples:

```
Dim x As Integer = 10
Dim y As Integer = 20
Dim c As Boolean
c = x < y
Console.WriteLine(c) ' Output: True
c = x >= y
Console.WriteLine(c) ' Output: False
c = x = y
Console.WriteLine(c) ' Output: False
c = x <> y
Console.WriteLine(c) ' Output: True
```

In comparison expressions, the result will always be True or False depending on the evaluation of the expression.

Conclusion

This section of the documentation provided details on how to handle common VB.NET errors and exceptions, work with strings and Boolean values, and use comparison operators. These operations form the building blocks of logical decision-making in VB.NET applications.