LearnVisualStudio.NET

Visual C# 2005 Express Edition

# Cheat Sheet : C# Language Basics

## Working With Numbers

```
int i = 0;
// convert from a string
int i = int.Parse("1");
// convert froma string and don't throw exceptions
if (int.TryParse("1", out i)) {}
i++; // increment by one
i--; // decrement by one
i += 10; // add 10
i -= 10; // subtract 10
i *= 10; // multiply by 10
i /= 10; // divide by 10
i = checked(i*2) // check for overflow
i = unchecked(i*2) // ignore overflow
```

## Dates and Times

```
DateTime now = DateTime.Now; // date and time
DateTime today = DateTime.Today; // just the date
DateTime tomorrow = today.AddDays(1);
DateTime yesterday = today.AddDays(-1);
TimeSpan time = tomorrow - today;
int days = time.Days;
int hours = time.Hours;
int minutes = time.Minutes;
int seconds = time.Seconds;
int milliseconds = time.Milliseconds;
time += new TimeSpan(days, hours, minutes, seconds,
milliseconds);
```

## Conditional Logic

```
if (i == 0) {}
if (i <= 10) {} else {}
switch (i) {
   case 0:
   case 1:
      break;
   case 2:
      break;
   default:
      break;
}
```

## Compare Operators

```
if (i == 0) // equal
if (i != 0) // not equal
if (i <= 0) // less than or equal
if (i >= 0) // greater than or equal
if (i >  0) // greater than
if (i <  0) // less than
if (o is MyClass) // check type of object
if (o == null) // check if reference is null
```

## Loops

```
for (int i=0; i<10; i++) {}
while (i<10) { i++; }
do { i++; } while (i<10);
foreach (ListItem item in list.Items) {}
```

## Define New Types

```
struct MyValueType {} // defines a value type
class MyClass { // defines a reference type
   // fields
   const string constName = "LearnVisualStudio.NET";
   static string staticName = "LearnVisualStudio.NET";
   readonly string readOnlyName;
   string instName;
   // properties
   public string Name
   { // read/write
      get { return instName; }
      set { instName = value; }
   }
   public string ReadOnlyName { get { return instName; } }
   // methods
   public void Print() { Console.WriteLine(instName); }
   public int Add(int a, int b) { return a + b; }
   public static void PrintStatic() {
   Console.WriteLine(staticName); }
   // constructors
   public MyClass() {} // default constructor
   public MyClass(string name)
   {
      readOnlyName = name;
      instName = name;
   }
}
```

## Inheritance

```
class MyClass : Object {
   // override inherited method
   public override string ToString() {
      // call base class version
      string s = base.ToString();
      return "Hello " + s;
   }
}
```

## Working with XML

```
using System.Xml;
using System.IO;

// Create an XML file
using (XmlWriter xw = XmlWriter.Create(@"c:\names.xml"))
{
   xw.WriteRaw("<names>");
   xw.WriteRaw("<name>Bob</name>");
   xw.WriteRaw("<name>David</name>");
   xw.WriteRaw("</names>");
}

// Load an XML file
XmlDocument doc = new XmlDocument();
doc.Load(@"c:\names.xml");
foreach (XmlNode node in doc.SelectNodes("//name/text()"))
{
   Console.WriteLine(node.Value);
}
```

## Working With Text

```
using System.Text;
using System.IO;

string name = "David";
string hello1 = "Hello " + name;
string hello2 = string.Format("Hello {0}", name);
Console.WriteLine("Hello {0}", name);
StringBuilder sb = new StringBuilder("Hello ");
sb.AppendLine(name);
sb.AppendFormat("Goodbye {0}", name);
Console.WriteLine(sb.ToString());

// Create a text file
using (StreamWriter w = File.CreateText(@"c:\names.txt")) {
   w.WriteLine("Bob");
   w.WriteLine("David");
}

// Read from a text file
using (StreamReader r = File.OpenText(@"c:\names.txt")) {
   while (!r.EndOfStream)   {
      Console.WriteLine(r.ReadLine());
   }
}
foreach (string s in File.ReadAllLines(@"c:\names.txt")) {
   Console.WriteLine(s);
}
```

## Working With a Database

```
using System.Data;
using System.Data.SqlClient;

string cs = @"Data Source=.\SQLEXPRESS;" +
            @"Initial Catalog=NamesDB;" +
            @"Integrated Security=True;";
using (SqlConnection con = new SqlConnection(cs)) {
   con.Open();
   string sql = "INSERT INTO Names(Name) VALUES(@Name)";
   // insert a record
   SqlCommand cmd1 = new SqlCommand(sql, con);
   cmd1.Parameters.Add("@Name", SqlDbType.NVarChar, 100);
   cmd1.Parameters["@Name"].Value = "Bob";
   cmd1.ExecuteNonQuery();
   // insert a second record
   cmd1.Parameters["@Name"].Value = "David";
   cmd1.ExecuteNonQuery();
   // read records
   sql = "SELECT * FROM Names";
   SqlCommand cmd2 = new SqlCommand(sql, con);
   using (SqlDataReader r = cmd2.ExecuteReader()) {
      int iName = r.GetOrdinal("Name");
      while (r.Read()) {
         Console.WriteLine(
           r.IsDBNull(iName)?"Null":r.GetString(iName)
         );
      }
   }
   // read a single value
   sql = "SELECT TOP 1 Name FROM Names";
   SqlCommand cmd3 = new SqlCommand(sql, con);
   Console.WriteLine(cmd3.ExecuteScalar());
}
```