Ken's C# Notes

Data Types

C# is case sensitive
C# is strongly typed
C# does NOT default numeric values to 0 or empty strings to null

Converting between types

aDouble = **Convert.ToDouble**(anInt);

Type		Prefix	to convert to
bool	true or false	bln	= Convert.ToBoolean(expression)
byte	0 to 255 (unsigned)	byt	= Convert.ToByte(expression)
char	single Unicode char	chr	= Convert.ToChar(expression)
DateTime	0:00:00 (midnight)		= Convert.ToDateTime(expression)
decimal	large integer, use for currency values	dec	= Convert.ToDecimal(expression)
double	large float	dbl	= Convert.ToDouble(expression)
int	large/long signed int	int	= Convert.To**Int16**(expression) int.Parse(aString)
long	larger/longer signed int	lng	= Convert.To**Int32**(expression)
sbyte	-127 through +128		= Convert.To**Int64**(expression)
short	int (-32,768 to +32,767)	sho	= Convert.ToSByte(expression)
float		flt	= Convert.To**Single**(expression)
uint			= Convert.To**UInt32**(expression)
ulong			= Convert.To**UInt64**(expression)
ushort	unsigned short (0 to 65,535)		= Convert.To**UInt16**(expression)
string		str	= Convert.ToString(expression)
Object		obj	
	global	g_	
	private to class	m_	
	form	frm	
	button	btn	
	label	lbl	
	database	cn	
	DateTime	dte	

Constants

const *datatype name = value*
const double c_pi = 3.14159265;

Arrays

(zero is the first element)

```
string[] aStringArray;
aStringArray = new string[10];

int[,] intMeasurements;
intMeasurements = new int[3,2];
```

Strings

Backslashes are escape chars. To treat as a literal prefix the string with @

```
string aFile = @"c:\temp";
```

Chars get single quotes

```
char aChar = 'a';
```

```
aString.Length;

aSubString = aString.Substring(0,5);  // returns the first 5 characters
i = aString.IndexOf("brown");  // -1 means not found
anotherString = aString.Trim();  // removes leadingand trailing spaces
anotherString = aString.TrimStart();
anotherString = aString.TrimEnd();
anotherString = aString.Remove(10,5);  // remove 5 chars starting at index 10
anotherString = aString.Replace("findThis", "replace with this"); // first or all?
```

DateTime

```
DateTime dteNow = DateTime.Now;      // date time
DateTime dteNow = DateTime.Today;   // date

DateTime dteBirthday = new DateTime(1969,7,22);  //    7/22/1969 12:00:00 AM

anInt = dteBirthday.Month;  // returns 7
anInt = dteBirthday.Day;
anInt = dteBirthday.DayOfWeek;  // Tuesday

DateTime aDte = dteBirthday.ToLongDateString();  //   Tuesday, July 22, 1969
```

DateTime aDte = dteBirthday.ToShortDateString(); // 7/22/1969
DateTime aDte = dteBirthday.ToLongTimeString(); // 12:00:00 AM
DateTime aDte = dteBirthday.ToShortTimeString(); // 12:00 AM

DateTime anotherDateTime = dteBirthday.AddDays(6);
DateTime anotherDateTime = dteBirthday.AddHours(anInt);
DateTime anotherDateTime = dteBirthday.AddMilliseconds(
DateTime anotherDateTime = dteBirthday.AddMinutes(
DateTime anotherDateTime = dteBirthday.AddMonths(
DateTime anotherDateTime = dteBirthday.AddSeconds(
DateTime anotherDateTime = dteBirthday.AddYears(

Lists

First index = 0
Adding items to a list
aList.Items.Add("another item");
Removing items from a list
aList.Items.Remove("another item"); // removes only the first occurrence
aList.Items.RemoveAt(0); // removes first item in list, throws error if empty
Clearing a List
aList.Items.Clear();

ListBox

anItem = aListBox.SelectedItem;
anIndex = aListBox.SelectedIndex; // -1 = nothing selected

aListBox.Items.Add("first item");

Hash (Associative Array)

```
private Hashtable mDictionary_hash = new Hashtable();

mDictionary_hash.Add(pieces[0], pieces[1]); // add key,value

aValue = mDictionary_hash[aKey].ToString();

mDictionary_hash ["some key"] = "Here I've changed the value.";

foreach (string aKey1 in mDictionary_hash.Keys) {
    mDictionary_misses.Add(aKey1,0);
}

// getting the Hash last entry (awkward!)
ArrayList theKeys = new ArrayList(mDictionary_hash.Keys);
aKey = theKeys[max].ToString();
aValue = mDictionary_hash[aKeyValue.aKey].ToString();
```

Operators

= =
! =
! not
^ logical XOR
?: conditional

Functions (Methods)

```
private void aMethod() {                                  // no parms, no returned value
private int aMethod(string strText, int aCount) {  // parms, returns int
   return someValue;
```

Returning multiple values

```
struct key_value_pair {
  public string aKey;
  public string aValue;
}

private key_value_pair aMethod() {
  key_value_pair aKeyValue;
  aKeyValue.aKey = "Freed";
  aKeyValue.aValue = "Ken";
  return aKeyValue;
```

Decisions, Looping

Just like C. If else is:

```
if (expression) {

} else if (another expression) {

} else {

}
```

```
// trailing while = until
do {

} while (expression);
```

```
// leading while
while (expression) {

}
```

Iterating through a collection

```
        for (int I = 0; I < this.Controls.Count; i++ ) {
          …
}
```

Try-Catch

```
try {


} catch (System.FormatException) {

} finally {
  // code when the try OR catch section completes
}
```

Exceptions

System.FormatException
System.OutOfMemoryException

Dialog Boxes

MessageBox.Show(*message text, caption, button, icon)*;
MessageBox.Show("hello world"); // .Net framework class

Common MessageBoxButtons. Enumerators
AbortRetryIgnore, OK, OKCancel, YesNoCancel, YesNo, RetryCancel

Common MessageBoxIcon. Enumerators
Exclamation, Information, None, Question, Stop, Warning

DialogResult. Enumerators
Abort, Cancel, Ignore, No, None, OK, Retry, Yes

if (MessageBox.Show("Would you like to do X?",
"MessageBox sample",
MessageButtons.YesNo,
MessageBoxIcon.Question) = = DialogResult.Yes) {

Custom Dialog Boxes

are just modal forms, that return a DialogResult
Done by using a button's "DialogResult" property
If DialogResult property is set on a button, pressing closes the form and returns the result to the <frm>.ShowDialog(); caller.

File IO

```
/* read theFile and populate a global array */
string fullFileName = gDICTIONARY_DIR + "\\" + theFile;

if (System.IO.File.Exists(fullFileName)) {

    System.IO.StreamReader objFile = new System.IO.StreamReader(
                    new FileStream(
           fullFileName,FileMode.Open),Encoding.UTF8
                                    );
    // line by line read
    aFileLine = objFile.ReadLine();  // returns null after last line
```

or

```
    // read the whole thing
    string strContents;
    strContents = objFile.ReadToEnd();

    // clean up
    objFile.Close();
    objFile.Dispose();
```

Good link: http://www.fincher.org/tips/Languages/csharp.shtml

ADO.NET

for accessing databases

```
using System.Data.OleDb;

OleDbConnection cnADONetConnection = new OleDbConnection();
cnADONetConnection.ConnectionString =
   @"Provider=Microsoft.Jet.OLEDB.4.0;
   Data Source=C:\temp\contacts.mdb";
cnADONetConnection.Open();

// to populate a database, you need a "DataAdapter"
OleDbDataAdapter cnADONetAdapter =
      new OleDbDataAdapter([cmd text],[connection]);

OleDbDataAdapter cnADONetAdapter =
      new OleDbDataAdapter("Select * From Contacts", cnADONetConnection);

…  todo, lookup and denote the rest of this, when needed


cnADONetConnection.Close();
cnADONetConnection.Dispose();
```

Classes, Objects

Interface consists of: Properties, Methods, Events
Project -> Add Class

static members belong to the class as a whole
(instance members beloing to the instance, and are created with "new")

Visual Basic: differentiates between class methods and public methods
Visual C#: All methods must exist in the context of a class.
Globally available methods can be achieved by defining *static* methods in the class
Static methods are always available, even if there's no instances of the class
You cannot access a static method through an instance (exception error)
Methods/Functions
scope datatype functionName (parameters) {
private void OpenPicture() { // doesn't return a value
private int OpenPicture(string thePicture, string aUser) { // returns int
return thePicture.Length;
}

Static Methods
public static int aMethod (string someText) {

// late (runtime) binding (uses plymorphism, virtual objects)
object anObject; // pate binding = declared as some arbitrary "object"
anObject = new aClass; // late binding = var set later to ref a specific class

// early binding
aClass anObject; // a specific class
if (…) {
anObject = new aClass();

anObject = null; // should be released when out of scope, but this makes sure

anObject.Dispose(); // make a Dispose method to be called by the garbage collector

Scoping

```
namespace Sams1
{
    // class level data goes here
    string aString;
    aString = "Ken Freed";
```

```
string anotherString = "Joe Boggs";

public partial class Form1 : Form
{
    private void OpenPicture()
{
```

Misc

Use the Solution Explorer to get to a form's code without adding an event handler
F5 or Debug->Start Debug *runs the project*
<file open dialog box object name>.FileName // to get the file selected

Debug Output

Console.WriteLine ("Debug: the value is:" + Convert.ToString(anInt));
Debug.WriteLine ("Debug: the value is:" + Convert.ToString(anInt));
System.Diagnostics.Debug.WriteLine("Results " + lngResults);
? txtInput.Text // uses immediate window

Forms

Snapping to grid:
Tools->Options->Windows Forms Designer

this.Hide(); // hides a form
this.Close(); // to terminate a form or program

<object>.Dispose();

Showing a form:
aFormObject.Show(); // non modal form of the window
aFormObject.ShowDialog(); // modal form of the window
or
aFormObject.Visible = true;

To use one form from another, you need
using <namespace>
at the top; e.g.:

```
using Sams1_PictureViewer;
```

TabControl
In Containers
TabPages
C# names each page TabPageN

ToolStrip
Is for menu item icons/pictures, usually located below the menu bar