Today's Lecture
- C# Primer (Continued)
  - STL Equivalents
  - Properties
  - Delegates
- Example Code
  - Student List
  - Teaser
    - Balance Board / WPF

Reminders
- Blog Post (Week)
- C# Programming

Interfaces are not just software

Interfaces - Remote

Small Group Exercise
- Think of non-GUI interfaces
- Best remote?
- Worst remote?

Split into groups of 2-4 students

Fun Link – Brief History Programming

1970 - Guy Steele and Gerald Sussman create Scheme. Their work leads to a series of “Lambda the Ultimate” papers culminating in ‘Lambda the Ultimate Kitchen Utensil.’ This paper became the basis for a long running, but ultimately unsuccessful run of late night infomercials. Lambdas are relegated to relative obscurity until Java makes them popular by not having them.
### History – Part 2

1983 - Bjarne Stroustrup bolts everything he's ever heard of onto C to create C++. The resulting language is so complex that programs must be sent to the future to be compiled by the Skynet artificial intelligence. Build times suffer; Skynet's motives for performing the service remain unclear but spokespeople from the future say 'there is nothing to be concerned about, baby;' in an Austrian accented monotone. There is some speculation that Skynet is nothing more than a pretentious buffer overrun.

### History – Part 3

1996 - James Gosling invents Java. Java is a relatively verbose, garbage collected, class based, statically typed, single dispatch, object oriented language with single implementation inheritance and multiple interface inheritance. Sun loudly heralds Java's novelty.

2001 - Anders Hejlsberg invents C#. C# is a relatively verbose, garbage collected, class based, statically typed, single dispatch, object oriented language with single implementation inheritance and multiple interface inheritance. Microsoft loudly heralds C#'s novelty.

### Simple student list

- Revisit the classic list of stuff
  - Student
    - First name
    - Last name
    - GPA
    - Major
  - Make a list of them and display it
- Homework 2
  - List and manipulate files

### Add a new class....

- Pick up where we left off
- Name != Class

### Class Design – C++

```cpp
class Student
{
private:
    String m_sFirstName;
    // ...

public:
    Student ()
    { ... }

    void getFirstName ();
}
```

### Class Design – C#

- Several changes
  - No header file
  - Per item ACL
    - Designate each item / function
    - Default is private
  - Similar modes
    - public, protected, private
  - Can also make "global"
    - `static` – do not need to instantiate
Big Change - Properties

- Normally do get / set functions
  - `getFirstName()`
  - `setFirstName()`

```java
class Student {
    String m_sFirstname;
    public String FirstName {
        get { return m_sFirstName; }
        set { m_sFirstName = value; }
    }
}
```

Let's see that again....

```java
public Type PropertyName {
    get { // Stuff to get }
    set { // Stuff to set }
}
```

- Return type must match property type
  - `get` value is the same as the property type

Using Properties

```java
Student theStudent;
theStudent = new Student();

// Old School
theStudent.setFirstName("Aaron");
Console.WriteLine(theStudent.getFirstName());

// New school
theStudent.FirstName = "Aaron";
Console.WriteLine(theStudent.FirstName);
```

- Everything is a "pointer" (except primitives*)

Automatic Properties

```java
public String FirstName {
    get; 
    set; 
}
```

- New feature in .NET 3.0
- Automatically creates the variable underneath

Let's see that again....

```java
public Type PropertyName {
    get { // Stuff to get }
    set { // Stuff to set }
}
```

Where do we use this?

```java
Instance.PropertyName
```

- Real power is when we tie it to controls
- `Student.FirstName` - `TextBox.DataBindings`

- Any changes in the control automatically update our variable
  - WHOOOT!
Populate the class

☐ See example code on wiki

Swapping over to Visual Studio

Override a function

public override string ToString()
{
    string temp;
    temp = firstName + " " + lastName + " -> GPA: " + GPA + " -> Major: " + Major;
    return temp;
}

override -> virtual
Place in child

toString exists in every class
Part of base object

Code Example

Student theStudent;
theStudent = new Student();
theStudent.firstName = "John";
theStudent.lastName = "Doe";
theStudent.GPA = float 3.67;
theStudent.major = "Eng";

Instance.PropertyName

Instance: theStudent

Properties: FirstName, LastName, GPA, Major

Collections

☐ What is similar to STL Vector?

- List - Collection

List<Type> varName;
varname = new List<Type>();

List<Student> listStudents;
listStudents = new List<Student>();

Using the List type

theList = new List<Student>();

student theStudent;
theStudent = new Student();
theStudent.firstName = "John";
theStudent.lastName = "Doe";
theStudent.GPA = float 3.67;
theStudent.major = "Eng";

theList.Add(theStudent);

student theStudent2;
theStudent2 = new Student();
theStudent2.firstName = "Brady";
theStudent2.lastName = "Quinn";
theStudent2.GPA = float 3.70;
theStudent2.major = "Eng";

theList.Add(theStudent2);

Dictionary<string, int> myDict = new Dictionary<string, int>();
myDict["key1"] = 1;
myDict["key2"] = 2;

Example output

If we want to have it iterate inside the list,
We have to make our own holder class that inherits or has an internal List class
Neat operator - foreach

```
foreach (Type VarName in CollectionName)
{
    // Do whatever
}
```

```
foreach (Student theSt in m_StudentList)
{
    Console.WriteLine(theSt);
}
```

Writing out the student list

- Sketch out on the board

See example code on Wiki

Code commentary

```
/// -> Triple slash
/* Similar to doxygen / javadoc */
/* Visual Studio will help auto-populate */
```

```
public String Major
{
    get { return m_Major; } 
    set ( m_Major = value; )
}
```

Preview - WPF

- Data binding in action

How does it work?

```
<ListBox Grid.RowSpan="4" Margin="0,1,1,1" Name="listBox1" ItemsSource="{Binding Source={StaticResource StudentList}}" />
<TextBox Grid.Column="1" Margin="10,16,12,0" Name="textBox1" Text="{Binding ElementName=listBox1, Path=SelectedItem.FirstName}" />
<TextBox Grid.Column="1" Grid.Row="1" Margin="10,16,12,0" Name="textBox2" Text="{Binding ElementName=listBox1, Path=SelectedItem.LastName}" />
<TextBox Grid.Column="1" Grid.Row="2" Margin="10,15,9,1" Name="textBox3" Text="{Binding ElementName=listBox1, Path=SelectedItem.GPA}" Height="39" VerticalAlignment="Bottom" />
<Button Grid.Row="4" Margin="13,12,18,10" Name="button1" Click="button1_Click">Load Info</Button>
```

Yikes!

WPF = WTF?

Key Part:

```
Text="{Binding ElementName=listBox1, Path=SelectedItem.FirstName}"
```

Other control

Property

Property
Questions?

- Homework 2
- Week 2 - Blog