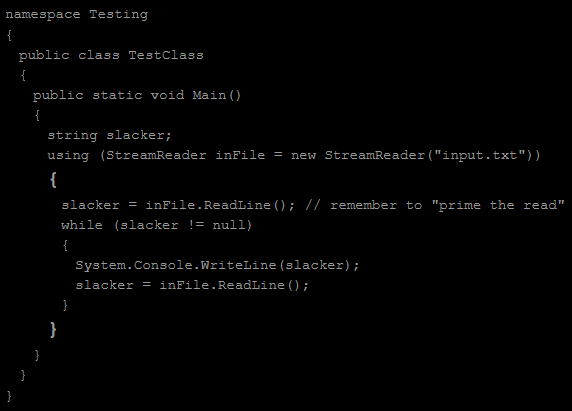
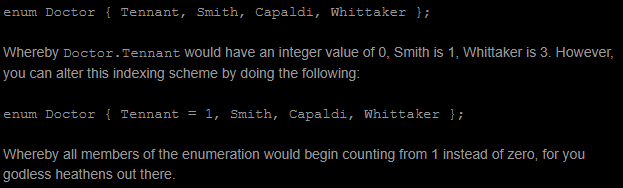
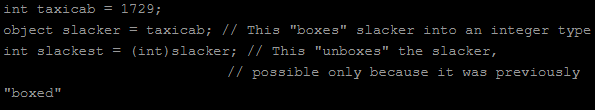
**Your First Program**

* Using Statement
  + Including namespace resources, such as "System.IO"
  + Defining an alias for a namespace or type
  + Simplify the allocation/de-allocation of temporary resources, such as file streams
* Namespace - collection of code associated with a project and development tool

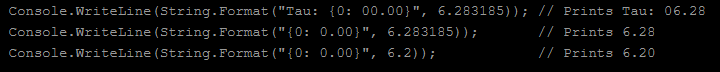
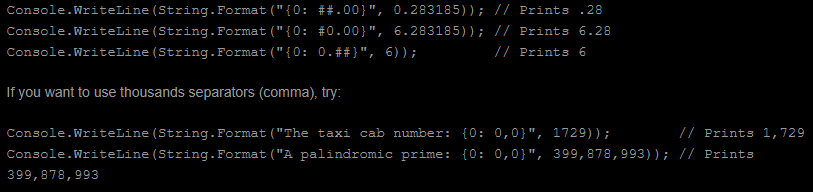
**Input/Output**

* Typecasting - C# has to use an intermediate Convert static class of methods in order to typecast the string-based input as something else. For example, to read a number, you would do:
  + int myInt = Convert.ToInt32(Console.ReadLine());
* Opening an Input File
  + 
  + Using statement for opening an input file doesn't ensure that the file exist or access to read or write from it

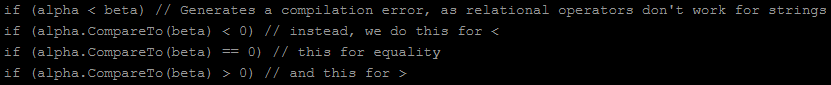
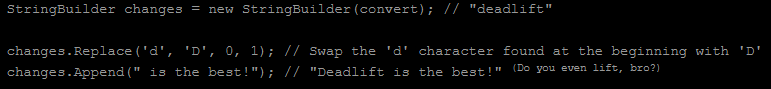
**Data Types and Keywords**

* Enumerated data type and how it works
  + 
* Boxing and Unboxing
  + Boxing - treating a type like an object
  + Unboxing - treating an object like a type
  + Difference between data type casting and boxing/unboxing is the **object** data type
  + 
* Nullable Data Type
  + Can operate the normal way the data type was intended but can take the null value
    - 
  + .HasValue - can see if it has a nonnull value
  + .Value - splits out the nonnull value
  + ?? - alternate assignment
    - Instead of assigning the variable to null, it will assign it to an alternate value
    - 

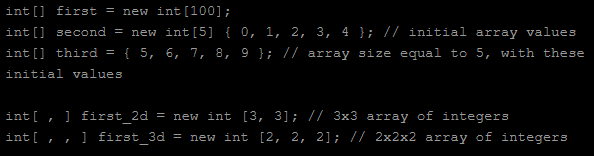
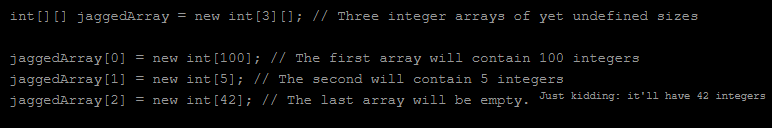
**Formatting for Float/Double Numbers**

* 
  + Output:
    - 
  + -10 is allocating how much space is after the particular variable
  + 0.00 is to format the value to display
    - 0's will unconditionally appear
    - #'s will only appear if there's a digit to fill, otherwise it's blank
* 
* 
* 
  + Percent symbol will automatically go through and multiply the value by 100

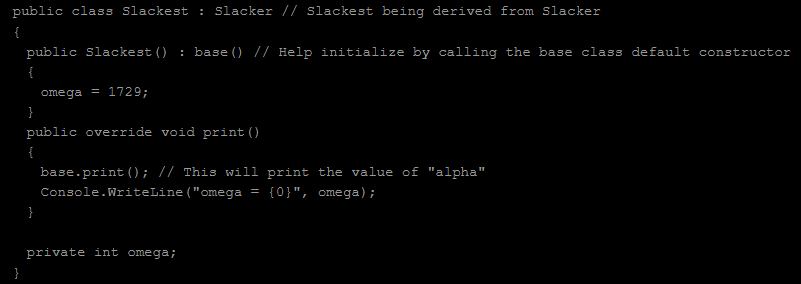
**Strings and StringBuilders**

* Strings are immutable (can't change individual characters in a string when given value. You can just change the whole value completely)
  + String Properties: <https://docs.microsoft.com/en-us/dotnet/api/system.string?view=netcore-3.1>
* In stringbuilders, you can change individual characters
* To compare strings, need .CompareTo
  + 
* 
* StringBuilder needs a .ToString() method to be displayed

**Arrays**

* Conventional
  + Has a fixed size
  + 
* Jagged
  + Array of arrays
  + 
* Two properties - <https://docs.microsoft.com/en-us/dotnet/api/system.array?view=netcore-3.1>
  + Length - size of array
  + Ranks- how many dimensions there are

**Classes**

* Inheritance
  + 
  + Slackest is being inherited from the parent class Slacker
  + Calling parent/superclass method using base keyword

Public Slacker() : base()

{

}

* Override keyword - if want to re-implement base classes' virtual method

Public class Slacker

{

Public virtual void print()

{

}

}

Public class Slackest : Slacker

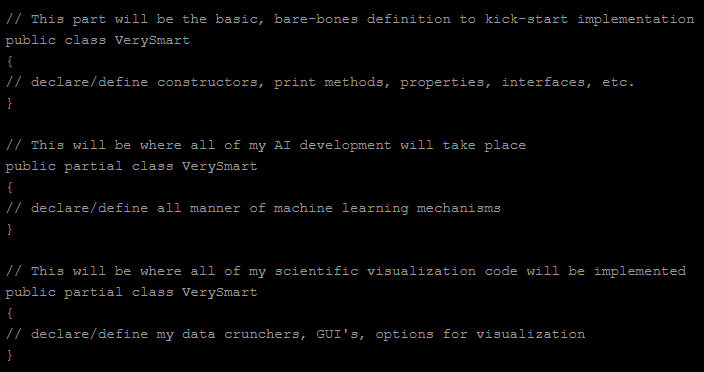
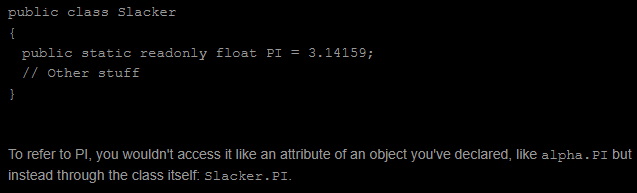
{

Public override void print()

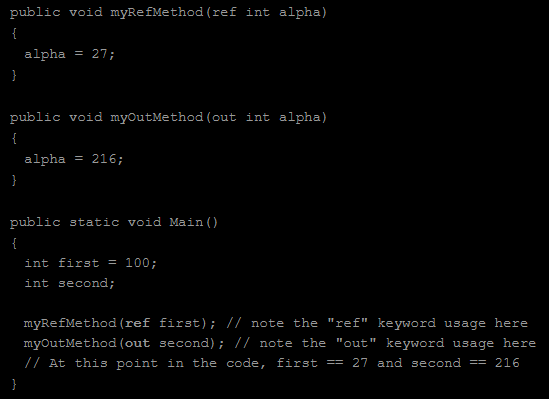
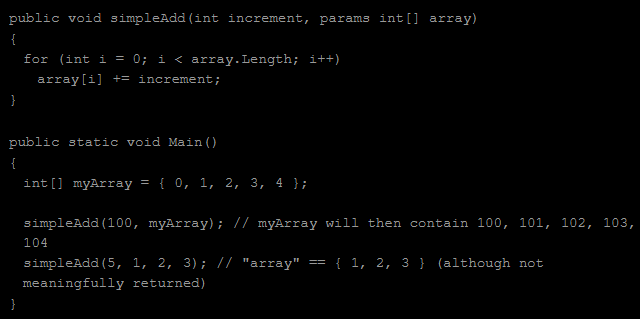
{

}

}

* Partial classes - split up parts of the class among multiple files, developers can work on the same class in different parts of the program
  + 
* Difference between virtual and abstract
  + Virtual methods may be overwritten by other classes
  + Abstract method must have an implementation in all derived classes
  + Virtual is optional, Abstract is mandatory
* Static
  + There will not be an instantiation of it in the program
  + To access it, need to use classname.method.StaticVariable (NEED TO USE CLASSNAME)
  + 

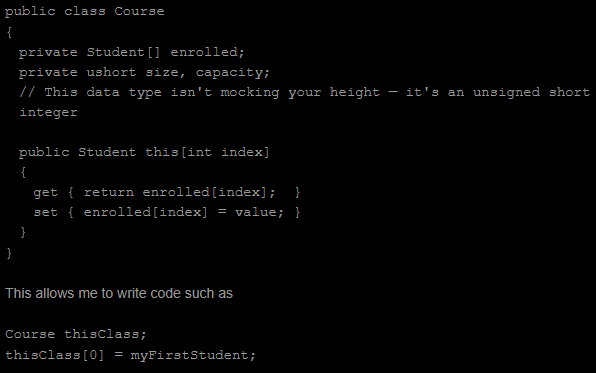
**Methods**

* Ref and out keywords allow variables to be passed by reference
* Ref needs to be initialized
* Out must be uninitialized
* 
* Params
  + Can have 0, 1 or a lot of arguments
  + Needs to be last value in the list of arguments
  + 

**Properties**

* Public interfaces of private attributes
* Naming will be the same except capitalize character
* If set is set to blank, then it is get only
* Get only can also be defined as
  + 

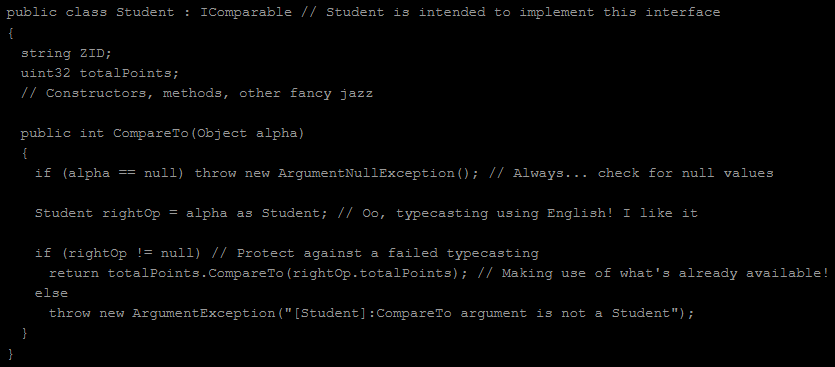
**Indexers**

* Will always be public and have the this keyword
* 

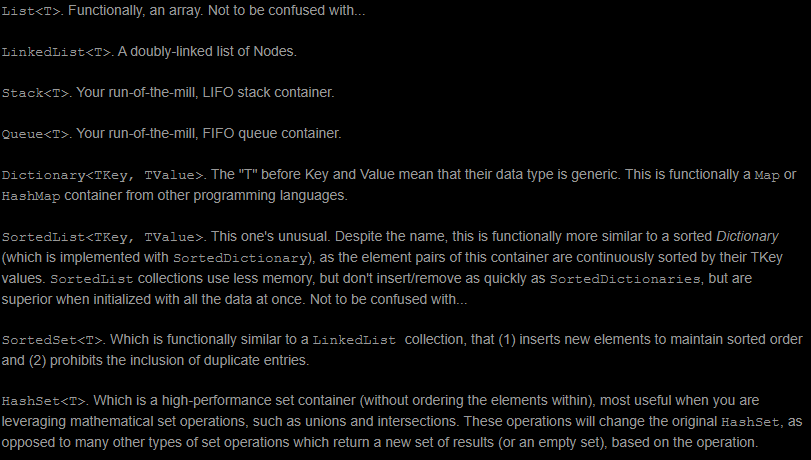
**Interface**

* Don't need to instantiate
* Can be used with Icomparable for comparison using compareTo method
  + Inherit it like from a super class

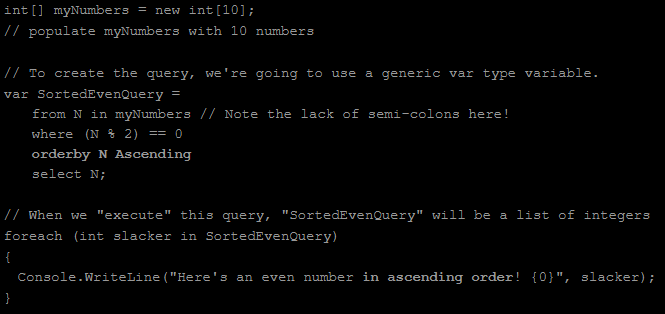
Public class Student : IComparable

* Used to keep stuff in a sorted over
* Can be used with ienumerable for iterating through objects in the class using foreach
* 

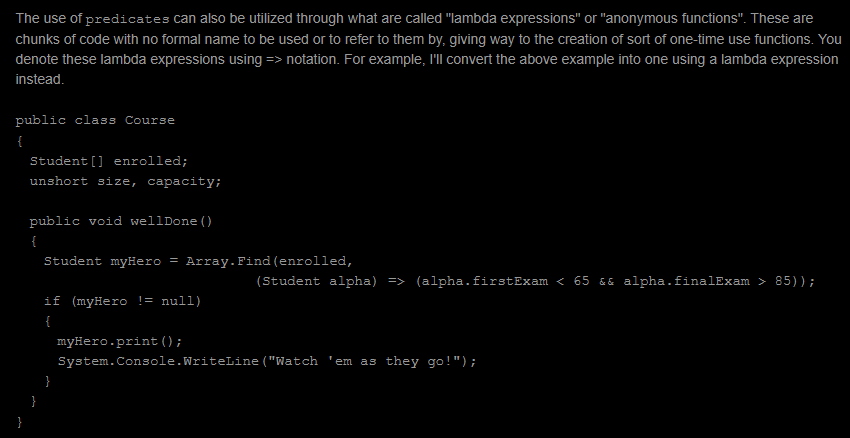
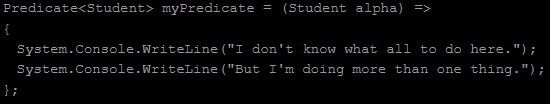
**Collections**

* 
* Collection Properties
  + Capacity - how many elements can be presently sorted without resizing
  + Count - gives how many elements are currently being stored
  + Various, add elements, remove elements, sort, search and clear

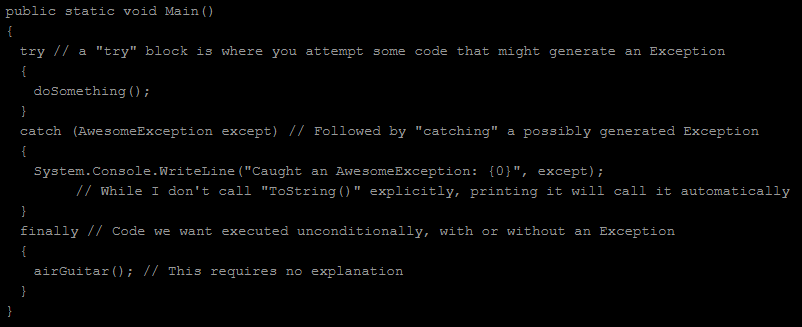
**LINQ**

* Know how to write LINQ queries, Assignments 3 and 4 have them built in
* 

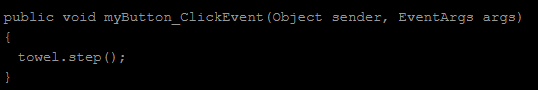
**Predicates** - <http://faculty.cs.niu.edu/~rogness/csci473/gettingStarted/predicates.shtml>

* Return type of a predicate will always be a boolean value
* 
* Using lambda expression or anonymous function as a second argument
  + 
* Expansion of predicate using lambda expression or anonymous function
  + 

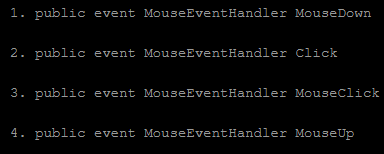
**Exceptions**

* Need at least one catch block with try block, finally block is optional
* 1 - many catch blocks and 0 - 1 finally block
* 

**Events**

* 
* For mouse events, second argument is MouseEventArgs
* For keyboard events, second argument is KeyEventArgs
* For time events, second argument is an ElapsedEventArgs

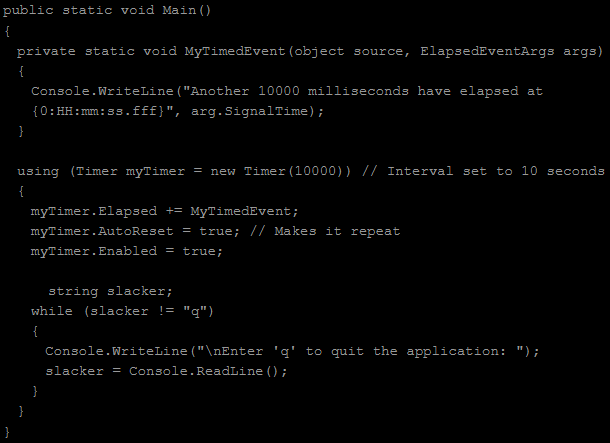
**Mouse Events**

* This is the order of mouse events
* 
* Properties
  + X and Y (X and Y coordinates of where on the screen the event was triggered)
  + Location (a Point structure containing the X and Y coordinate values of where on the screen the event was triggered)
  + Button (indicating what button was pressed, Left, Right, Middle, None, etc.)
  + Clicks (number of times the button was pressed and released
  + Delta (number of notches by which the mouse wheel has been rotated)

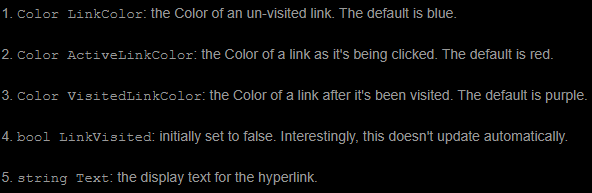
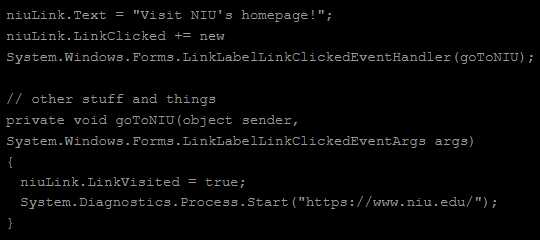
**Keyboard Events**

* KeyCode is an enumerated data type that is used to identify what key was pressed
  + e.KeyCode == Keys.A
* KeyChar is generated by the KeyPressEventArgs and see if it matches KeyCode
* For Alt, Control and Shift keys,
  + e.Alt, e.Control and e.Shift
    - They are boolean variables
  + 

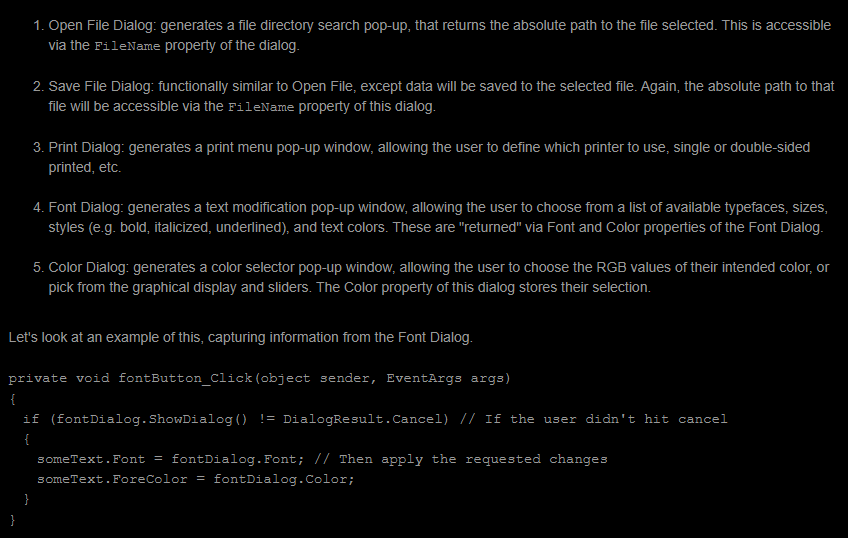
**Timing Events**

* 
* Argument in new Timer is measured in milliseconds
* Elapsed is a delegate being defined and set to MyTimedEvent
* AutoReset allows the timer to repeat in the interval defined
* Enabled is what turns the timer on

**LinkLabels**

* 
* Code to open a webpage in a .Net Project, System.Diagnostics.Process.Start
  + 

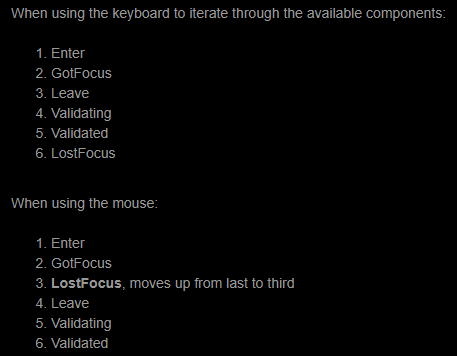
**Dialogs**

* Pop-up windows that present a short bit of information or ask a simple question, before the user dismisses the pop-up window or provides a response
* 
* The statement if (fontDialog.ShowDialog() != DialogResult.Cancel) is for if the user didn't click cancel

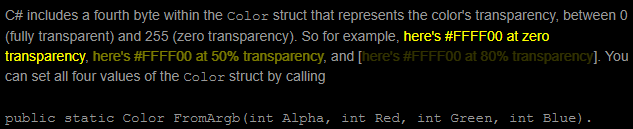
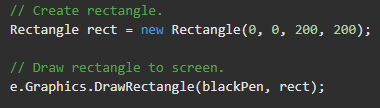
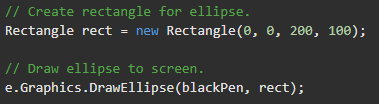
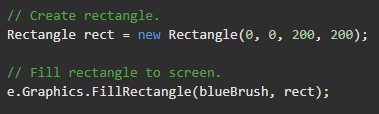
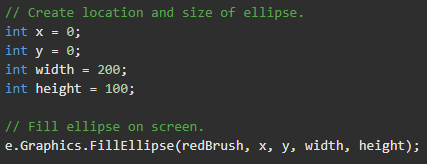
**Tool Tips**

* Used for hiding information on the page that may already be known or used to display clarification

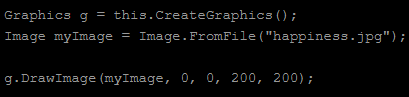
**Focus**

* Used to hide or display a shadow
* Regular EventArgs are used with Focus events
* 

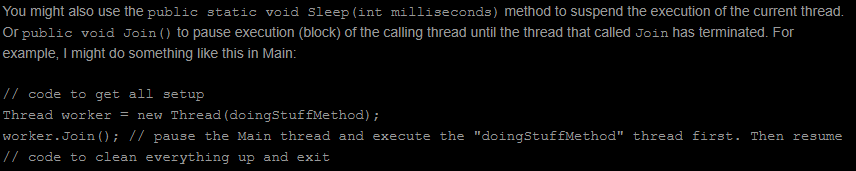
**Drawing Graphics**

* Primary Colors
  + 
* Secondary Colors
  + 
* Alpha represents transparency
  + 
* First parameter that goes into DrawRectangle and DrawEllipse is a pen
  + 
  + DrawRectangle(Pen pen, Rectangle rect);
  + DrawEllipse(Pen pen, Rectangle rect);
* First parameter that goes into FillRectangle and FillEllipse is a brush
  + 
    - Need to have the type of brush when defining a brush like SolidBrush
  + FillRectangle(Brush brush, Rectangle rect);
  + FillEllipse(Brush brush, Rectangle rect);
* DrawRectangle <https://docs.microsoft.com/en-us/dotnet/api/system.drawing.graphics.drawrectangle?view=dotnet-plat-ext-3.1>
  + 
  + 
* DrawEllipse <https://docs.microsoft.com/en-us/dotnet/api/system.drawing.graphics.drawellipse?view=dotnet-plat-ext-3.1>
  + 
  + 
* FillRectangle <https://docs.microsoft.com/en-us/dotnet/api/system.drawing.graphics.fillrectangle?view=dotnet-plat-ext-3.1>
  + 
  + 
* FillEllipse <https://docs.microsoft.com/en-us/dotnet/api/system.drawing.graphics.fillellipse?view=dotnet-plat-ext-3.1>
  + 
  + 

**Image**

* Importing an Image into C#
* 

**Threads**

* 

**ASP.NET: A State of Mind**

* Session State - when writing code that deals with a specific user would write code like this
  + 