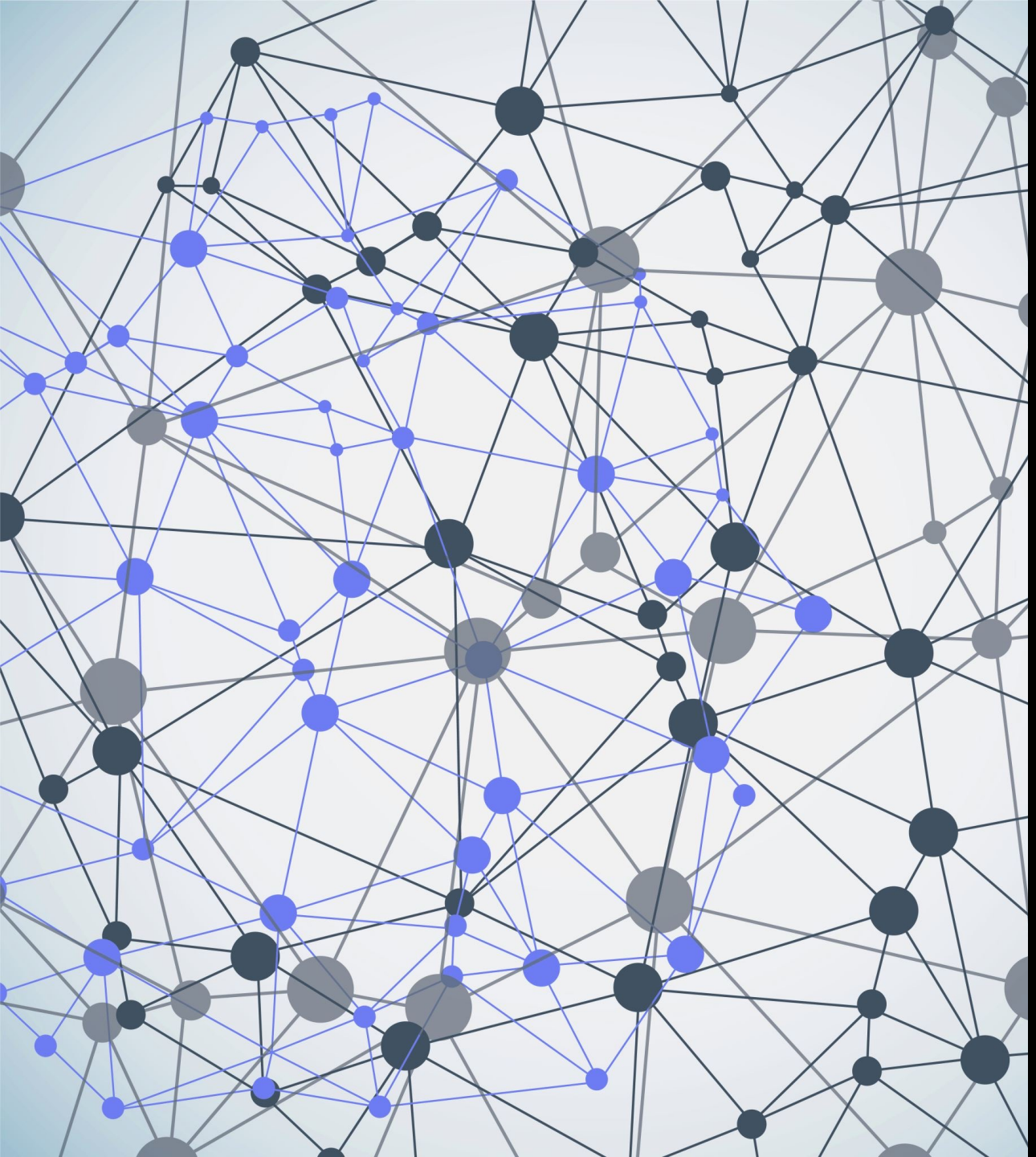


98-361

SOFTWARE DEVELOPMENT FUNDAMENTALS (USING VB.NET)

1. Understand computer storage and data types

Stack	Heap
Number types	
Boolean	
Char	
Pointer to class	Classes
Pointer to string	Strings
Pointer to object	Objects

1a. how a computer stores programs and the instructions in computer memory

- The stack is a relatively small area for storing data, with high-speed access.
- Compared to the stack, the heap:
  - Does not have any real limits on memory size, so it can contain larger amounts of data,
  - Is not as efficient as the stack, and
  - Is slower compared to the stack.
- Variables such as strings will have:
  - Small pointers (or handles) on the stack, pointing to
  - Where they are on the heap.

1b. memory stacks and heaps

- Stacks are a Last-In First-Out (LIFO) storage structure.
  - Stack memory is used for local variables, but not strings.
- Structures are *value types* and uses stack allocation.
  - number types, bool and char as also value types.
  - Stack memory is cleared when it is no longer necessary, e.g. when a method ends.
- Classes are *reference types* and uses heap allocation.
  - class, interface, delegate, record, dynamic, object, string are also reference types.
  - Heaps are cleared using garbage collection, which happens automatically at some point.
- When you pass a reference of something (say to a function), it is allocated on a heap.
- When a class is instantiated, it goes on the heap, and a pointer to it goes on the stack.

1c. memory size requirements for the various data storage types, numeric data and textual data

- C# is strongly typed. Every variable will have a type.
- Declare variables like

    bool myvar = false; // one equals sign is used for assignment.

    var myvar = false; // computer chooses variable type.

    const int myvar = 87; // constants are values which do not change.

    private float myvar = 87; // this variable is "private" (as opposed to "public") to where it has been defined.

- Order of calculations:
  - Brackets first
  - i++ means that i will increase at the end of the statement (postfix increment operator). See also: i--
  - ++i means that i will increase right now (prefix increment operator). See also --i.
  - Other symbols are + - % (for remainder)
  - And then + and -.

1c. memory size requirements for the various data storage types, numeric data and textual data

Name	.NET name	Range	Precision	Size	Letter
Sbyte	System.SByte	-128 to 127 (3 digits)	Whole number	1 byte	
Byte	System.Byte	0 to 255	Whole number	1 byte	
Short	System.Int16	-32,768 to 32,767 (5 digits)	Whole number	2 bytes	S
Ushort	System.UInt16	0 to 65,535	Whole number	2 bytes	
Integer	System.Int32	-2.147 billion to 2.147 billion (9 zeros)	Whole number	4 bytes	I or %
UInteger	System.UInt32	0 to 4.294 billion	Whole number	4 bytes	
Long	System.Int64	-9 quintillion to +9 quintillion (18 zeros)	Whole number	8 bytes	L or &
Ulong	System.UInt64	0 to +18 quintillion	Whole number	8 bytes	
Single	System.Single	+/- (38 zeros)	About 6-9 digits	4 bytes	F or !
Double	System.Double	+/- (308 zeros)	About 15-17 digits	8 bytes	R or #
Decimal	System.Decimal	+/- (28 zeros)	28-29 digits	16 bytes	D or @

The .NET name is expressed in bits (a zero or 1). There are 8 bits in a byte (zero to 255).
"U" means unsigned – no negative values allowed
Letters can be used with numbers, e.g. 1000000L, for a data type

LISB or BaSIL

1c. memory size requirements for the various data storage types, numeric data and textual data

Name	Description	Size	Letter
Boolean	true or false (defaults to false)	1-4 bytes	
String	text	variable	"
Char	1 Unicode character	2 bytes	" and c
Date	Contains a date and time (even if midnight)		#

2. Understand computer decision structures

2b. If decision structures

- If takes the structure

  if (condition) {

  then-statement;

  }

- The condition must result in true or false.
- The {}s are optional if you only have one statement as opposed to a block.

2c. multiple decision structures, such as If…Else and switch/Select Case

- if-else takes the structure

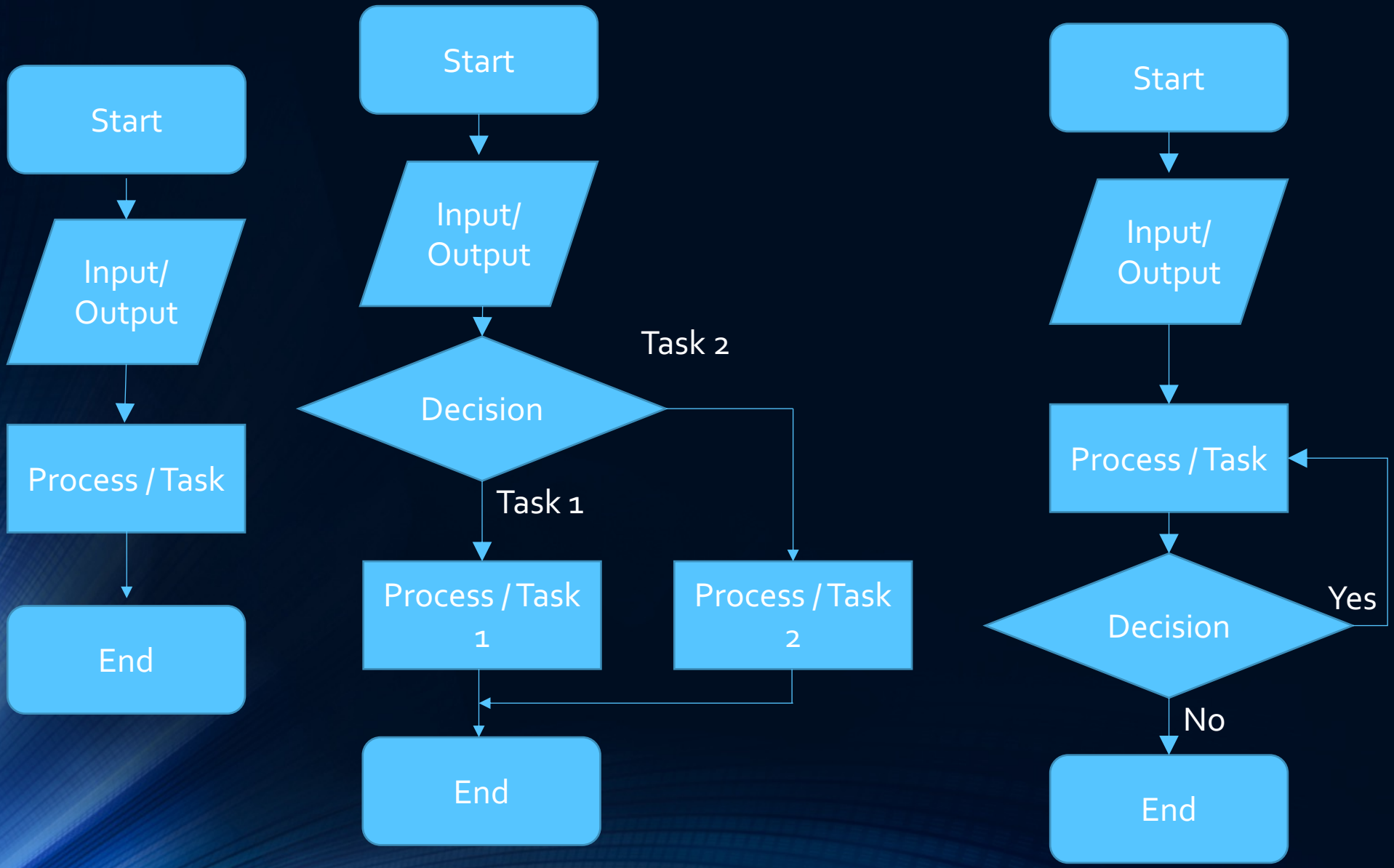
```
if (condition) {
    then-statement;}
else {
    else-statement;}
}
```

- switch allows you to compare against many things:

```
switch (condition)
{
    case first_answer [when second_condition]:
        statements;
        break;
    case second_answer: //The next section will run on either second_answer or third_answer
    case third_answer:
        statements;
        break;
    default:
        statements;
        break;
}
```

- default is optional and is run if nothing else matches.
  - If default is not used, then if nothing matches, nothing is run.

Reading flowcharts



2e. decision tables

Question	Response 1	Response 2	Response 3	Response 4
Is it dry?	Yes	Yes	No	No
Am I OK to get wet?	Yes	No	Yes	No
Final Result	Go out	Go out	Go out	Stay inside

	Response 1	Response 2	Response 3	Response 4
Am I well?	Yes	Yes	No	No
Do I want to go outside?	Yes	No	Yes	No
Final Result	Go outside.	Stay indoors	Stay indoors	Stay indoors

2e. decision tables

Question	Response 1	Response 2	Response 3	Response 4
Is it dry?	True	True	False	False
Am I OK to get wet?	True	False	True	False
OR Result	True	True	True	False

	Response 1	Response 2	Response 3	Response 4
Am I well?	True	True	False	False
Do I want to go outside?	True	False	True	False
AND Result	True	False	False	False

2f. evaluating expressions

- Conditions:
  - == the same as (equality).
  - != not the same as (inequality).
  - < is less than
  - > is more than
  - <= is less than or equals to (only one = sign)
  - >= is more than or equals to
- Multiple conditions can be used:
  - && means that both conditions must be true for the overall expression to be true – otherwise, it is false.
  - || means that only one condition must be true for the overall to be true – otherwise, it is false.
  - Make sure you use && and ||. & and | are used for bit-wise conditions.
- The order of precedence is:
  - Mathematical symbols.
  - Conditions (not equality or inequality)
  - Equality
  - && and ||
- Consider:
  - 4 + 5 > 8
  - 4 > 3 && 2 > 1

3. Identify the appropriate method for handling repetition

3a. For loops

- For loops are for running a certain number of times:

```
for (int i=1; i<=10; i++) // initializer; condition; iterator
{ // how important is it to have <= instead of < ?
// write code here
}
```

- The initializer occurs before the loop starts, and only occurs once.
- As soon as the condition evaluates as true, the loop terminates.
  - This could mean that the loop runs zero times.
- The iterator occurs at the end of the loop.
- An infinite loop would start with: *for (; ;)*

3b. While loops

- while loops executes statements until the "while" expression becomes false.
  - It could be used instead of a for loop – but it is slightly more complicated.
- The "while" expression is evaluated at the start of the loop. It does not use a "do".

```
while (expression)
{
    // statements here;
};
```

- You can leave a do…while loop using

```
break;
```

- You can go to the bottom and re-evaluate the while expression by using

```
continue;
```

3c. Do…While loops

- do … while loops executes statements until the "while" expression becomes false.
  - It is always run once.
  - It could be used instead of a for loop – but it is slightly more complicated, and a for loop might run zero times.
- The "while" expression is evaluated at the end of the loop.

```
do
{
    // statements here;
} while (expression);
```

- You can leave a do…while loop using

```
break;
```

- You can go to the bottom and re-evaluate the while expression by using

```
continue;
```

3d. recursion

- StackOverflowException occurs when there are too many nested method calls. The limit is not fixed.

```
public class Example
{
  private const int MAX_RECURSIVE_CALLS = 1000;
  static int ctr = 0;
  public static void Main()
  {
    Example ex = new Example();
    ex.Execute();
    Console.WriteLine("\nThe call counter: {0}", ctr);
  }
  private void Execute()
  {
    ctr++;
Console.WriteLine("Call number {0} to the Execute method", ctr);
if (ctr <= MAX_RECURSIVE_CALLS)
      Execute();
Console.WriteLine("Call number {0} to the Execute method", ctr);
    ctr--;
  }
}
```

4. Understand error handling -structured exception handling

- try-catch-finally
  - If something happens in the try section, the code then goes through to the catch.
  - Regardless, it then goes through to the finally section (unless a StackOverflowException is encountered).

```
try
{ // code
}
catch (Exception e)
{ // code
}
finally
{ // code
}
```

  - StackOverflowException occurs when there are too many nested method calls.
  - FileNotFoundException is when you try to open a file and it is not there.
  - DivideByZeroException is when you try to divide by zero.

5. Understand the fundamentals of classes

5a. properties, methods, events, and constructors

- Methods are blocks of code.
  - They generally return values – and the variable type is in the definition.
  - You return the value using the return, which also ends the method (lines afterwards will not be run).

    return myval;

- The program will start at the Main method.
  - You must have only one Main method.
  - It is static – it can be called even when there no instance of the class has been created.
- Methods can return values back to the code which started it.
  - Methods which do not include the word void.

5a. properties, methods, events, and constructors

- When you create a new object – MyClass, this will also run the method MyClass.MyClass when initialising the object.
  - This is called a Constructor, and the process of initialising it is also called instantiation.
  - A Constructor which takes no arguments is called a parameterless or default constructor. This is run when no arguments are provided to "new".

```
public MyClass()
{
    this.Title = "No title";
}
```

  - A Constructor will always have a parameterless constructor (unless the class is static), so it can create a new object.
  - A Constructor may be overloaded, with different number or different types of parameters in each constructor.
- You may see "this" – "this" refers to this class.

```
public MyClass(string varTitle)
{
    this.Title = varTitle;
}
```

5a. properties, methods, events, and constructors

- Properties are exposed class members.
- Just like variables, they can be used to assign values and return values.
- They differ from variables because they have code inside it.
- They could be read-write, read-only or write-only, depending on whether they have a "get" or "set" accessor.

```
private int myVar;

public int Property

{

    get

    {

            return myVar;

    }

    set

    {

            myVar = value;

    }

}
```

- This can be shortened, using auto-implemented properties, to:

```
public int Property

    {get; set;}
```

- This also removes the parameterized constructor.
- You can also create objects more easily:

```
var item = new SaleItem{ Name = "Shoes",
Price = 19.95m };
```

5a. properties, methods, events, and constructors

- Events can trigger code.
- In one module:

```
public event EventHandler Changed;
...
Changed (this, EventArgs.Empty);
```

- In another module, you need to subscribe to this public event:

```
Rectangle r = new Rectangle():
r.Changed += new EventHander(r_Changed);
r.Length = 10; // this triggers the Event, which triggers the r_Changed.
static void r_Changed(object sender, EventArgs e)
{
Rectangle r = (Rectangle)sender;
Console.WriteLine("Value Changed: Length = {0}, r.Length);
}
```

- .

Events

Module: Program
Sub: Main

Class: Computer
Sub/Function: Typing

SoundOfTyping = Item.Typing()

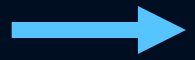
```
Function Typing() As String
'Do something about the typing
Console.WriteLine("I am typing")
```

Events

Module: Program
Sub: Main

Class: Computer
Sub/Function: Typing

```
SoundOfTyping = Item.Typing()
```



```
Function Typing() As String
'Raise an event about Typing
```

```
'An event has happened.
'Do something about the typing.
```

Events

Class: Program
Method: Main

Class: Computer
Sub/Function: Typing

Other classes or modules

SoundOfTyping = Item.Typing()

Function Typing() As String
'Raise an event about Typing

'An event has happened.
'Do something about the typing.

'An event has happened.
'Do something about the typing.

5b. how to create a class

- A class defines a type of object or data type.
  - It is not the object itself, but objects can be created from it.
  - You can create a new class for each object.
  - Classes can also be referred to as “types”.
- You create it using:

```
class Class_Name
{
    // your code comes here.
}
```

- They are contained in namespaces.
  - Namespaces can be used to organize large code projects.
  - References to namespaces can also be brought into code using the keyword using

```
using System;
```

5c. how to use classes in code

- You can refer to a class in the same namespace by calling it:

  myClass

- If it is in another namespace, you need to use the following syntax (unless you have previously used "Using") to create a fully qualified class name:

  myNameSpace.myClass

- If you wish to create an object (an instance of the class), you can use the new keyword:

  myClass myVar = new MyClass()

6-8. Terminology

- Abstraction – un-exposing methods and properties, so they are not called on directly by consumers of the type or class.
- Encapsulation – treating a group of properties, methods etc. as a single unit.
- Inheritance – creating a new (derived) class based on an existing (base) class. Any methods used in the base class are inherited by the derived class.
  - The relationship between base and derived is like a parent and child.
- Polymorphism – having multiple classes that can be converted between them.
  - The methods used in the multiple classes can be named the same but use different code.
  - The methods used in the derived classes can override the base class.

6. Understand inheritance: inheriting the functionality of a base class into a derived class

- You can create a base class, and then a derived class which has characteristics of that class.
  - The base class contains members which can be inherited.
  - The derived class inherits these members.
  - The derived class can be called a "is a" of the base class.
    - For example, a laptop "is a" computer.
  - A derived class can inherit from only one base class. This is called "single inheritance"
- Base class:

  class Polygon
- Derived class:

  class Rectangle: Polygon

6. Understand inheritance: inheriting the functionality of a base class into a derived class

- Abstract classes show that it is incomplete or has something missing.
  - You need to use the "abstract" modifier.
- Virtual members show that it can be overridden.
  - Virtual members (methods, properties, events and indexers) are complete – there isn't anything missing.
  - Members are non-virtual (sealed) by default.
  - You cannot use virtual with static, abstract, private or override modifiers.
- Abstract classes can be shared or modified by derived classes:

```
abstract class BaseClass
{
    abstract public class MyMethod();
}
class DerivedClass: BaseClass
{
    public override double MyMethod() // and then the method;
}
```

- The "Sealed override" modifier would show where something cannot be overridden by another derived class.
  - Others would not be able to customise your class.
  - But others would also not be able to make their derived class not work correctly.

7. Understand polymorphism: extending the functionality in a class after inheriting from a base class, overriding methods in the derived class

- If you access a derived class as a base class, the overrides will still be called.

```
DerivedClass B = new DerivedClass();
B.MyMethod();  // Calls the new method.

BaseClass A = (BaseClass)B;
A. MyMethod();  // Also calls the new method.
```

- If you want to hide a base class member, then use new. If you use "new" instead of "override", then A.MyMethod() will call the old method.
- A derived class can call the base method by using the keyword base

```
base.MyMethod();
```

- In this way, the derived class can concentrate on the extra derived behavior.

7. Understand polymorphism: extending the functionality in a class after inheriting from a base class, overriding methods in the derived class

If…	Then…
Method in derived class does not have "new" or "override" keywords.	Compiler will issue a warning. Method will act as if "new" was used.
Derived Method is preceded by "new" keyword	Method is independent of base class method.
If it is cast to the Base method	Method will call the base method.
Derived Method is preceded by "override" keyword and Base Method is defined as "virtual".	Objects of the derived class will call this override method.
If it is cast to the Base method	Method will call the derived method.

8. Understand encapsulation

- Encapsulation is a group of:
  - Properties,
  - Methods,

  being treated as a single unit.
- All of the necessary code is there.
- If anything needs to be changed in the encapsulated unit, it only needs to be changed there, and not in other code.

8a. creating classes that hide their implementation details while still allowing access to the required functionality through the interface

- Interfaces contains definitions for a group of related functionalities that a non-abstract or struct class must implement.
- It is similar to an abstract base class with only abstract members.
- An interface cannot be run directly.
- A class/struct can implement multiple interfaces.
- You can define an interface by using the interface keyword:

```
interface IsSame<T>
{
    Bool IsSame(T obj);
}
```

- By convention, interface names begin with a capital I.
- Any class/struct that implements this interface must contain a definition for an Equals method that matches the input and output the interface specifies, such as

```
Public bool IsSame(Car car)
{
    Return (this.Make) == (car.Make)
}
```

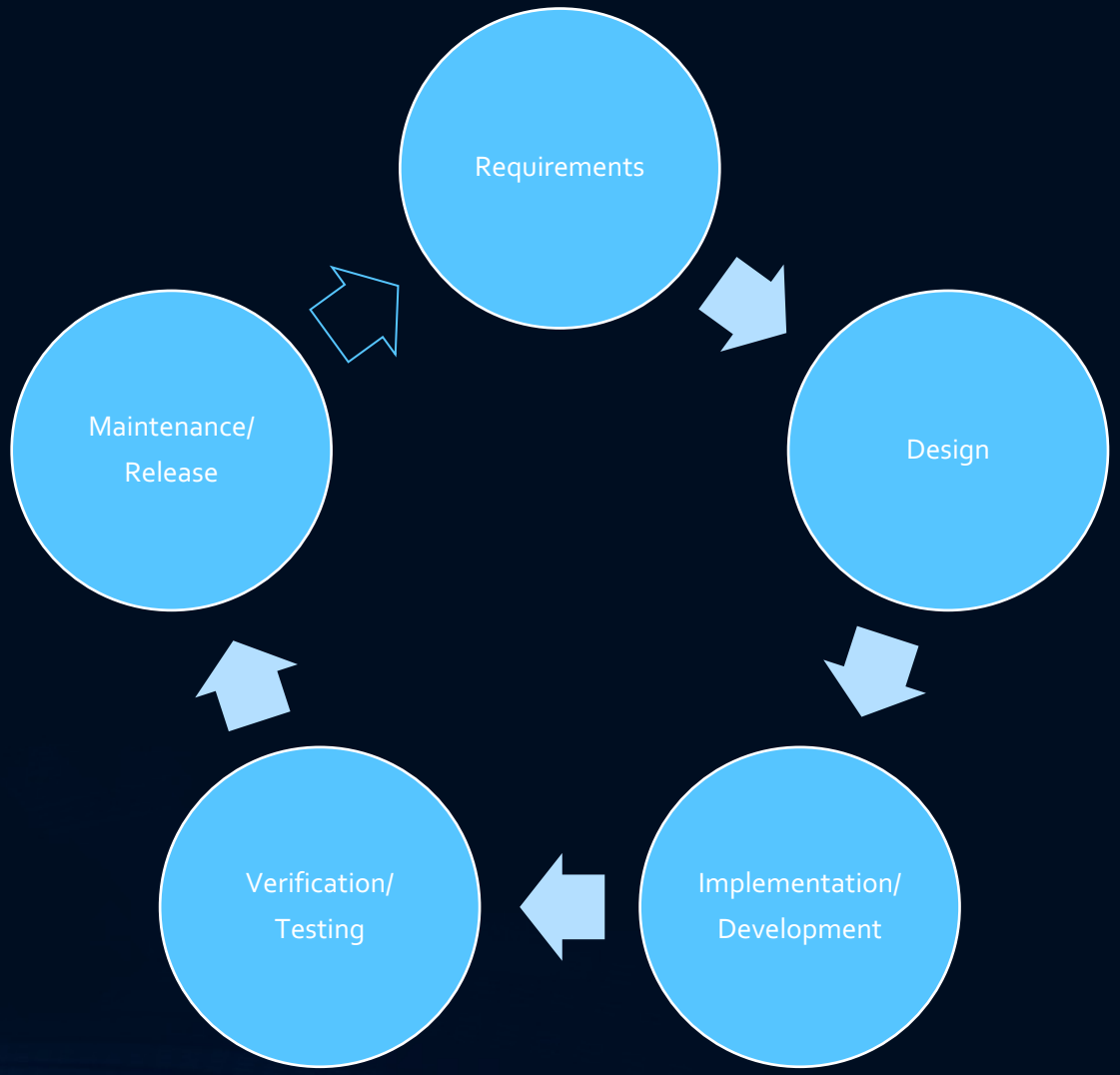
8b. access modifiers

- The type or member can be accessed by:
- public:
  - any other code in the same assembly or
  - another assembly that references it.
- private:
  - code in the same class or struct.
- protected:
  - code in the same class, or
  - in a class that is derived from that class.
- internal:
  - code in the same assembly only.
- protected internal:
  - code in the assembly in which it's declared, or
  - from within a derived class in another assembly.
- private protected:
  - only within its declaring assembly,
  - by code in the same class or
  - in a type that is derived from that class.

9. Understand application life cycle management

9a. phases of application life cycle management

- In a waterfall methodology, problems detected in earlier phases are more fixable/cheaper to fix than in later phases.
- "Agile" computing would do this over a 1-4 week cycle, to get a deliverable.
- In an Agile scrum (a process for managing software development), this iteration would be called a "sprint".



9a. phases of application life cycle management

- Requirements
  - Collecting requirements for stakeholders
    - Some of these may conflict.
  - Determining needs to satisfy a vision.
  - Done by Business Analyst, or Computer Systems Analyst.
- Design
  - Planning the software solution.
    - This may involve creating a model or prototype.
    - It could involve the outline of design classes or flowcharts.
  - Done by (Solutions) Architect or engineer.
- Implementation
  - Building the software.
    - Classes, methods, interfaces.
    - Publishing the software
  - Done by a Computer Programmer or Developer.
- Verification / testing
  - Finding software bugs.
    - Interaction between objects and components.
    - Does it meet the requirements?
  - Testing input/output.
  - Done by Quality Assurance or Tester.
- Release / maintenance
  - Getting feedback by stakeholders.
  - Recommendations for improvement / fixing problems.
  - More than just support – that can also be done by looking at documentations or telephone calls.
  - Done by Technical support with end users.

9b. software testing

- Black-box testing treats the software as a sealed box
  - You have no knowledge of how the software does its work.
  - You examine functionality by knowing what it is supposed to do.
  - It requires the tester to have a "test case", which has input(s) and what the output should be like.
  - No programming knowledge is required.
  - Due to the lack of knowledge as to how the software works, the tester may be testing the same process multiple times, or may not test part of it at all.
- White-box testing looks at the internal workings of the program:
  - You can test individual units, how the units are integrated, or the overall system.
    - Unit or integration testing may not detect missing requirements.
  - Because you have knowledge of the internal workings, you can ensure you go down every code path or branch.
    - However, if you only test a branch once with one input, you may not notice it works differently with a different input.

11. Understand algorithms and data structures

11a. arrays

- An array is a group of variables with a similar type (and purpose).
  - The [] indicates that it is an array.
    - int[] items = {10, 11, 12, 13, 14];
  - It is zero-based
    - items[0] == 10;
    - items[5] is an error.
- Arrays can be single-dimensional, or multi-dimensional.
  - int[] is a single-dimensional array.
  - int[,] has two dimensions.
- int[] items = new int[5] has 5 members, not 6, with indexes from 0 to 4.
- Array values are stored on the heap, with a pointer stored on the stack.

11a. stacks

- A stack is a Last-In First-Off (LIFO) collection.

```
using System;
using System.Collections;
public class SamplesStack  {

  public static void Main()  {

    // Creates and initializes a new Stack.
    Stack myStack = new Stack();
    myStack.Push("Hello");
    myStack.Push("World");
    myStack.Push("!");

    // Displays the properties and values of the Stack.
    Console.WriteLine( "myStack" );
    Console.WriteLine( "\tCount:    {0}", myStack.Count );
    Console.Write( "\tValues:" );
    PrintValues( myStack );
  }

  public static void PrintValues( IEnumerable myCollection )  {
    foreach ( Object obj in myCollection )
      Console.Write( "    {0}", obj );
    Console.WriteLine();
  }
}

/*This code produces the following output.
myStack
    Count:    3
    Values:    !    World    Hello
*/
```

- Methods includes:
  - Clear() – Removes all objects from the Stack.
  - Contains(Object) – is an element in the Stack?
  - Peek() – gets the top item without removing it.
  - Pop() – gets the top item and removes it.
  - Push(Object) – puts an item on top of the Stak.

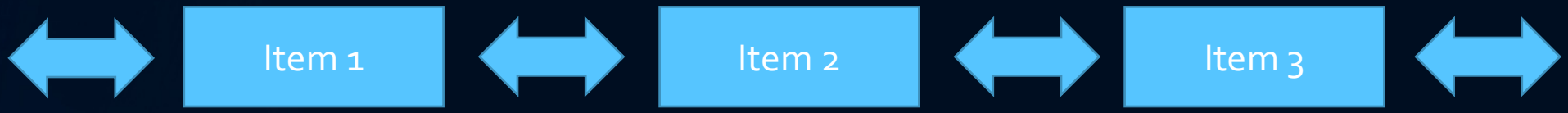
11a. queues

- A queue is a First-In First-Off (FIFO) collection.

```
using System;
using System.Collections;
public class SamplesStack  {

  public static void Main()  {

    // Creates and initializes a new Stack.
    Stack myStack = new Queue();
    myStack.Push("Hello");
    myStack.Push("World");
    myStack.Push("!");

    // Displays the properties and values of the Stack.
    Console.WriteLine( "myStack" );
    Console.WriteLine( "\tCount:    {0}", myStack.Count );
    Console.Write( "\tValues:" );
    PrintValues( myStack );
  }

  public static void PrintValues( IEnumerable myCollection )  {
    foreach ( Object obj in myCollection )
      Console.Write( "    {0}", obj );
    Console.WriteLine();
  }
}

/*This code produces the following output.
myStack
    Count:    3
    Values:    Hello World !
*/
```

- Methods includes:
  - Clear() – Removes all objects from the Queue.
  - Contains(Object) – is an element in the Queue?
  - Peek() – gets the top item without removing it.
  - Dequeue() – gets the top item and removes it.
  - Enqueue(Object) – puts an item on bottom of the Queue.
- Queues are implemented as a circular array.

11a. linked lists



- Lists,
  - https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.linkedlist-1
- Methods
  - AddAfter(location, varString)
    - location could be from a Find
  - AddBefore(location, varString)
  - AddFirst(varString);
  - AddLast(varString);
  - Clear();
  - Contains(varString);
  - Find(varString);
  - FindLast (varString);
  - Remove(location);
  - RemoveFirst();
  - RemoveLast();

11a. sorting algorithms

- Start with http://rosettacode.org/wiki/Category:Sorting_Algorithms
- Bubble sort
  - Compares two adjacent items to see if they should be sorted.
  - http://rosettacode.org/wiki/Sorting_algorithms/Bubble_sort#C.23
- Quick sort
  - Separates data into two partitions – one below a certain level, and one above.
  - Then sub-divides partitions again – and so on, until all sub-partitions have just one item.
  - http://rosettacode.org/wiki/Sorting_algorithms/Quicksort#C.23

11b. performance implications of various data structures
11c. choosing the right data structure

- Arrays
  - Stores elements of the same type
  - Size based on initial declaration
  - They are identified by the array index number
  - Have to go through all items if you want to search
- Stacks/Queues
  - Data structure – elements can be of different types
  - Items are moved onto top (stacks) or bottom (queues) only.
  - Items are removed from top only.
  - Can search for items using Contains()
- Linked list
  - Data structure
  - Items are connected (or doubly connected) to each other.
  - Items can be inserted into the top, bottom or middle.
  - Can search for items using Contains()

12. Understand web page development

12a. HTML

- Hypertext Markup Language – HTML

```
<!DOCTYPE html>
<html>
  <head>
     <title>The Title of the page</title>
  </head>

  <body>
    <h1>A heading</h1>
    <p>A paragraph.</p>
    <p>A paragraph with a <a href="http://www.microsoft.com">Link</a>.</p>
    <div></div>
  </body>
</html>
```

- href is an attribute. <a> is an anchor tag. The "/" closes the tag – it is required.
- Hypertext Transfer Protocol (HTTP) transmits documents such as HTML.
- Hypertext Transfer Protocol Secure (HTTPS) is for secure communication.

12b. Cascading Style Sheets (CSS)

- Stores formatting. If the formatting needs to be updated for a website, only one file needs to be changed.
- It can also be used for responsive design, allowing your content to look good on any screen.
- In the head, you need either:

```
<link rel="stylesheet" href="styles.css">
```

- or

```
<link rel="stylesheet"
href="https://www.microsoft.com/styles.css">
```

- The CSS file could say:

```
body {
  background-color: powderblue;
}
h1 {
  color: blue;
  font-family: arial;
  font-size: 150%;
border: 2px solid blue;
}
p {
  color: red;
}
```

- There are 140 standard color names.
- #Is there a new menu named Styles in VS?#

12c. JavaScript

```
<!DOCTYPE html>
<html>
<head>
<script>
function IWantWhite() {
  varselect = document.getElementById("ColorSelection");

varselect.options[varselect.selectedIndex].text = "White";
}
</script>
</head>
<body>
<form>
Select your favorite color:
<select id="ColorSelection">
 <option>Blue</option>
 <option>Red</option>
 <option>Pink</option>
 <option>Black</option>
</select>
<input type="button" onclick="IWantWhite()" value="I prefer white">
</form>
<script>
favoritecolor = prompt("Please enter your favorite color");
message = "Your favorite color is " + favoritecolor + ".";
</script>
<noscript>No JavaScript! </noscript>
</body>
</html>
```

- <noscript> can be used to show a message for users who do not use JavaScript.

13. Understand Microsoft ASP.NET web application development

13a. page life cycle,

Stage	Description
Page request	The page request occurs before the page life cycle begins. Does it need to be parsed and compiled (therefore beginning the life of a page), or can a cached version be sent?
Start	In the start stage, page properties are set. Is the request is a postback or a new request and sets the IsPostBack property? The page also sets the UICulture property.
Initialization	Controls on the page are available and each control's UniqueID property is set. A master page and themes are also applied to the page if applicable.
Load	During load, if the current request is a postback, control properties are loaded with information recovered from view state and control state.
Postback event handling	If the request is a postback, control event handlers are called. After that, the Validate method of all validator controls is called, which sets the IsValid property of individual validator controls and of the page.
Rendering	Before rendering, view state is saved for the page and all controls. The Render method for each control is called.
Unload	The page has now been fully rendered, sent to the client, and is ready to be discarded. At this point, page properties are unloaded and cleanup is performed.

13b. event model,

Stage and event	Relevant event
Page request	
Start	PreInit
Initialization	Init, InitComplete
Load	PreLoad, Load
Postback event handling	Such as Button's Click events.
	LoadComplete, PreRender, PreRenderComplete, SaveStateComplete
Rendering	
Unload	Unload

ASP.NET event names are Page_ plus the relevant event.

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs) Handles Me.Load
    Response.Write("Event: " & System.Reflection.MethodBase.GetCurrentMethod().Name & ".<br/>")
```

13c. state management,

- The state (previous information) is either controlled by the client or server.
- If the client:
  - Cookies
    - Text files containing data

      Response.Cookies("destination").Value = "CA"

      Response.Cookies("destination").Expires = DateTime.Now.AddDays(1)
  - View State. The ViewState property provides a dictionary object.
    - ViewState["color"] = "red";
  - Control state. ViewState can be turned off by developers at a page level. Control state cannot be – but it is more complicated.
  - Query Strings
    - Information appended to the end of a URL
    - http://www.microsoft.com?message=hi
  - Hidden Fields
    - When a page is sent to the server, hidden field content can also be sent.

      <asp:hiddenfield id="ExampleHiddenField"

      value="Example Value"

      runat="server"/>
- If the server:
  - Application state – storage accessible from all pages:

    Application["WelcomeMessage"] = "Welcome to the Contoso site.";
  - Session state - storage accessible from all pages, but only for a single session/user

    Session["FirstName"] = FirstNameTextBox.Text;

13d.client-side versus server-side programming

- Client-side means doing it on your computer.
  - Does not require network traffic.
  - For simple tasks, quicker response.
  - Complex tasks might overwhelm the computer.
  - JavaScript code is executed client-side – you could use it for checking data that has just been entered, to see if it is in the expected format.
- Server-side means doing it on the Server.
  - More internet traffic.
  - Can access more resources (such as databases).
  - If complex, might be better on a more powerful computer.
  - If you are having something execute on the Web server, then you will need somewhere to have:

    runat="server"
  - Server is the only acceptable value – there is no client attribute.

14. Understand web hosting

14a. creating virtual directories and websites,

- To add a website, right-hand click on the Default Web Site and select "Add Website…".
- Virtual Directory map (provide an alias) to places on a computer or network.
  - The alternative is physical directories – areas within (say) C:\inetpub\wwwroot .
- They can be used to separate out areas of a website, or to host files such as graphics.
- Virtual directories are not stored within C:\inetpub\wwwroot – they can be elsewhere on the network.
- On the IIS, right-hand click on the Default Web Site and select "Add Virtual Directory".

14b. deploying web applications,

- Uploading files to a website can be done using:
  - FTP – File Transfer Protocol is like Windows Explorer, but for the Internet.
    - Use can enter ftp:// in Windows Explorer
    - See FileZilla as an example of a programs which has a pane for Local and a pane for Internet.
- From Visual Studio, you need to click Publish using the Publish Web wizard.
  - In Publish web application select the Publish method.
  - In the Connection tab:
    - Choose "Web Deploy Package" if you want to give it to someone else to deploy.
    - Choose "Web Deploy" to deploy it directly.
    - Enter things like location/URL and Site/application (if you are creating a package, this can be overridden when deployed).
  - In the Settings tab, select the Configuration:
    - Use the Debug configuration when you are deploying to a test environment and want to debug. However, this has no effect when the web site project is compiled dynamically.
    - Otherwise, use the Release configuration.
  - Packages can be installed:
    - Using IIS Manager,
    - Using the .deploy.cmd file, or
    - Using Web Deploy commands from the command line or PowerShell.
  - See https://docs.microsoft.com/en-us/visualstudio/deployment/deploying-applications-services-and-components?view=vs-2019
  - If you are publishing ASP.NET applications to IIS, make sure you have the ASP.NET feature installed in IIS – otherwise it won't run!

14c.understanding the role of Internet Information Services

- IIS allows you to turn a computer into a web host.
  - It can also host ASP.NET, PHP applications, and WordPress sites.
  - It does not need to be configured on a computer that publishes it, or where the files are located, unless that computer also hosts it.
- To activate it, use the Windows search bar to Turn Windows features on or off.
  - Or run Control Panel - Programs and Features – Turn Windows features on or off (left-hand side).
- Once activated, look for IIS in the Windows search bar.
- Go to the Default Web Site, right-hand click it, and Browse.
  - It opens up "localhost".
- Add HTML files into C:\inetpub\wwwroot, especially the root page index.html

15. Understand web services

15a.web services that will be consumed by client applications,

- Web services allows for your functionality to be exposed on the Internet.
- Go to File – New – Project – Visual C# Projects – ASP.Net Core Web Application – Web Application.
- Then Add New Item – Web Service (ASMX)
- You can change the default WebService1 name and Service1.asmx to something different.
- Your code should now start with a reference to the WebService – this is required for you to work with it.
- Add functionality (as many as you want), starting with [WebMethod] if you want it to be exposed:

```
[WebMethod]
public int GiveMeAOne
{
return 1;
}
```

- Click on Build – Build.
- Go to the Web service page (replace [NameOfWebService] with your Web Service's name:
  - http://localhost/[NameOfWebService]/[NameOfWebService].asmx
- Remember this address – you will need it in the next step.

15b. accessing web services from a client application,

- In Visual Studio, under Project types, create another ASP.Net Web Application - Web Forms.
- Go to Project – Add Service Reference – Advanced… - Add Web Reference.
  - In older versions of Visual Studio, it was under Project – Add Web Reference.
- Enter the URL of the Web Service.
- Select "Add Reference".
- Expand the Web References section of Solution Explorer. Note the Namespace.
- Add the following code in Main:
  - localhost.[FileOfService] [NameOfService] = new localhost.[FileOfService]();
- Then write some code using the methods you have created:
  - Console.Write([NameOfService].GiveMeAOne);

15c. SOAP

- SOAP is Simple Object Access Protocol.
- It allows for exchange of data between client and server in XML using HTTP
  - HTTP is easy to transfer between systems.

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
  <HelloWorld xmlns="http://tempuri.org/" />
 </soap:Body>
</soap:Envelope>
```

15d. Web Service Definition Language (WSDL)

- Web Services Description Language WSDL is another XML-based description language.
- However, this is used for describing functionality:
  - Service – system functions.
  - Port/Endpoint – address or connection point.
  - Binding – Interface and SOAP binding style.
  - PortType/Interface – Web services, operations, and messages used.
  - Message – Information needed to perform an operation.
  - Types – Describes the data.

```
<service name="MyWebService">
<port name="MyWebServiceSoap" binding="tns:MyWebServiceSoap">
<soap:address location="http://localhost:1236/MyWebService.asmx"/>
</port>
<port name="MyWebServiceSoap12" binding="tns:MyWebServiceSoap12">
<soap12:address location="http://localhost:1236/MyWebService.asmx"/>
</port>
</service>
```

16a. UI design guideline categories,

Topic	Description
Controls	User Interface that allow users to interact with your App.
Commands	Actions that users can take, such as in toolbars, menus and ribbons.
Text	Any text the end user can see.
Messages	Any message users see, such as errors, warnings and confirmations.
Interaction	Ways in which the end user can interact, such as keyboard, mouse, touch (with gestures).
Windows	The container for controls, menus etc.
Visuals	Elements other than controls, such as layout, font, color, icons etc.
Windows Environment	Elements such as desktop, taskbar, file system.

16b. characteristics and capabilities of Store Apps

Topic	Description
Music, Pictures, Videos	Access to a user's Music, Pictures and Videos, including background media player
Removable Storage	USB memory sticks and external hard drives
Internet and Public Networks	Access
Home and work networks	Access
Appointments, Contacts	Access
Phone Calls, VoIP calling	Access and the ability to make calls
User Account Information	Access to user's name and pictures

16b. characteristics and capabilities of Store Apps

- Windows Forms have a Graphical User Interface with windows. It does not need Internet access.
- They also react to events, either internally called or through user interaction with controls.
- When a Windows Forms application starts, Form Events as raised as follows:
  - Form.Load
  - Form.Activated
  - Form.Shown
- When it closes, the following Form Events are raised:
  - Form.Closing and Form.FormClosing
  - Form.Closed and Form.FormClosed
  - Form.Deactivate
  - Application.ApplicationExit

16b. characteristics and capabilities of Store Apps

- Multiple Document Interface (MDI) applications are where multiple child windows run in a parent window.
  - In the main Windows Form, set IsMdiContainer to True.
  - In Solution Explorer, right-hand click the Project, then Add – New Item and select Windows Form.
  - In the main Windows Form, add a new Menu item, and enter similar to:

```
protected void MDIChildNew_Click(object sender, System.EventArgs e){
  Form2 newMDIChild = new Form2();
  // Set the Parent Form of the Child window.
  newMDIChild.MdiParent = this;
  // Display the new form.
  newMDIChild.Show();
}
```

- The opposite of a Multiple Document Interface (MDI) is a Single-document Interface.

16c. identify gestures

Command	Number of fingers	Gesture
Select an item	1	Tap on the touchpad.
Scroll	2	Place on the touchpad and slide horizontally or vertically.
Zoom in or out	2	Place on the touchpad and pinch in or stretch out.
Show more commands (similar to right-clicking)	2	Tap the touchpad, or press in the lower-right corner.
See all open windows	3	Place on the touchpad and swipe them away from you.
Show the desktop	3	Place on the touchpad and swipe them towards yourself.
Switch between open windows	3	Place on the touchpad and swipe right or left.
Open Cortana	3	Tap three fingers on the touchpad
Open action center	4	Tap four fingers on the touchpad.
Switch virtual desktops	4	Place four fingers on the touchpad and swipe right or left.

17. Understand console-based applications - characteristics and capabilities of console-based applications

- Console-based applications run from the command line. There is no GUI, but it can accept user input.
- Command-line parameters can also be included in the command line.
- Go to Create a new Project – Console App (.NET Core).
  - If you cannot see the Console App, then click on "Install more tools and features", and then ".NET Core Cross-platform development" workload – Modify.
- Enter a project name in "Configure your new project".
- Enter your code in the Main module.
- Use:
  - *Console.WriteLine(varString);* to show a string, and press the Return key.
  - *Console.Write(varString);* to show a string, and not press the Return key.
  - *Console.ReadLine();* to ask a question and get the answer.

18. Understand Windows Services - characteristics and capabilities of Windows Services

- Windows Services have no user interface – they run in the background.
- To create a service, go to New – Project – Windows Service (.NET Framework).
- Enter a name for the service.
- Add code into the constructor:
  - Use eventLog1.WriteEntry(varString); to write to the Event Log.
  - Use OnStart for what to do when it starts.
- Add a timer to allow it to poll:
  - using System.Timers;
  - Timer timer = new Timer();
  - timer.Interval = 60000; // 60 seconds
  - timer.Elapsed += new ElapsedEventHandler(this.OnTimer);
  - timer.Start();
- And then add programming into the OnTimer method.
- Should this Windows Service start when the computer starts? Change the StartType for the serviceInstaller1 to:
  - Service_Stopped.
  - Service_Paused.
  - Service_Running. # or is it Manual, Automatic, Stopped?#
- To manually install it once it has been compiled, so it shows as a service in the Service Control Manager (SCM), use:
  - The Developer Command Prompt for Visual Studio,
  - Navigate to the file's directory.
  - installutil nameoffile.
- Services can be restarted in the SCM.
  - Go to Computer Management – Services.
  - or run Set-Service in PowerShell.

    Set-Service -Name BITS -StartupType Automatic

19. Understand relational database management systems

19a. characteristics and capabilities of database products

- RDBMS – Relational Database Management System

19b. database design

- One-to-one, one-to-many relationships.
- What is a data warehouse (OLAP) compared to an OLTP?
- Define:
  - Primary Key – uniquely identifies a row in a table. Each value is unique. Only 1 in a table.
  - Foreign Key – links to a primary key, and enforces the relationship.
  - Unique Constraint – each value is unique. There can be multiple unique constraints in a table.
  - Check Constraint – Allows for certain values only to be included in a column (e.g. not negatives).
  - Default Constraint – If no value is included when the row is inserted, this value will be instead.
    - Used for row numbering, or for when the data was inserted, for example.
  - Index – like an index in a book, allows you to find specific rows more easily.
    - Best used with the WHERE clause.

20. Understand database query methods

20a. Structured query language (SQL)

- USE database_name – goes to a different database
  - Often followed by GO
- SELECT
- UPDATE

20b. creating and accessing stored procedures,

- Stored Procedures are pieces of code when are stored in the database.
- Once they are first run, they are compiled, and so subsequent runs can be done quicker.

CREATE PROC name_of_procedure AS

//code

EXEC name_of_procedure

20d. selecting data

- The following will return all rows:

```
SELECT *
FROM name_of_table
WHERE condition
[ORDER by name_of_column ASC/DESC]
[;] // optional in some variants
```

- Condition can include: WHERE name_of_condition LIKE 'ABC%'
- SQL comes in different variants. It may use <> instead of !=
- Use JOIN to combine with another table.

21. Understand database connection methods

- DSN (data source name) - http://support.microsoft.com/kb/305599
  - Stores and centralizes connection information for a data source. You can view DSNs by opening the Data Sources (ODBC) Control Panel applet. It allows you to configure an application to reference the DSN without having to know the connection string to the actual data source.
- Not to be confused with DNS (Domain Name Systems), which translates names to IP addresses.