Interview

**Framework vs Library**

**Definition: framework is a platform.** may also include code libraries, a compiler, and other programs used in the software development process.

Library – Made up of useful reusable code (Classes, Functions, Procedures <Function w/ no return>, Scripts, Configuration)

Framework is a collection of library?

Supporting Structure?

Both of them are dependent

Framework and Library differ in rules where in framework is more strict or fixed

Reusability

Frameworks are patterns

**ASP.NET**

Server side Web development application framework in .NET framework

Dynamic web sites – client-server relationship that occurs. Actions performed by user such as request are sent to the server.

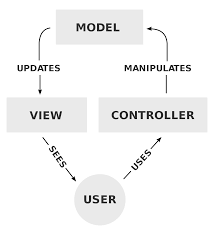
**MVC asp.net**

Implements the Model View Controller pattern

Easy integration with JS

**model** directly manages the **application's** data, logic and rules.

**View** is a user interface.

Controller are responsible for controlling the flow of the application execution.

What is OOP?

Object-oriented programming is a programming paradigm based on the concept of "objects", which can contain data, in the form of fields, and code, in the form of procedures.

Advantages:

It makes programming much easier by making it easier to understand because of the way it uses real life objects as tools in programming.

Object – instance of a class

Class – blueprint or template of an object.

Pillars of OOP:

Abstraction – Hides irrelevant data but reveals only the important ones. Uses abstract methods to hide data.

An abstract method is a method that is declared, but contains no implementation.

Encapsulation – Hides irrelevant data. HIDES COMPLEXITY AND PROVIDE SECURITY to the data. Uses access modifiers (properties). (restricts/ permits)

Inheritance – inheriting states and behavior of a different class (parent class / base class to subclass/ child class)

Polymorphism - occurrence of something in different forms. Mostly used in method overloading as well as method overriding. (e.g. polymorphic array, abstract methods-can be override that’s why it’s morphing).

Create array of classes then make an instance of their derived class. Making the derived class have multiple forms of its child classes.

Arithmetic Operators

Assignment Operators

Comparison Operators

Logical Operators

**Bootcamp Deck 1**

Visual C#

* Runs on Dot Net framework

**.NET framework**

A development platform created by Microsoft for windows.

Main components of .NET framework:

CLR(Common language runtime) –a virtual machine like JVM for java. Manages the execution of code.. it uses JIT to convert **Intermediate Language** to **Machine Language**

PL – IL – CLR – ML

.NET framework class library – collection of classes, interfaces, value types

**Common Type System (CTS)**

Identifies types supported in CLR

CTS provides support to the five categories of type:

* Classes
* Structures
* Enumerations
* Interfaces
* Delegates

**Types**

Value types or reference type. Both are data types

Cannot override value types unlike referenced type

Reference usage:

Student Stud = new student() // cannot affect/override student class since it uses new to declare a copy

Stud1 = student //can override

Classes(reference type) and Structs(Value type)

Basic constructs of common types in .Net framework

Each is a data structure that encapsulates a set of data and behaviors that belong together as a logical unit

Type members:

* Fields
* Properties
* Methods
* Constructors
* Events
* Nested Types

Built in Types

Built-in object, strings, numeric types (int, floating, double, etc.)

Custom Types/ User-defined types

* Classes
* Structs
* Interface
* Enums

.NET framework library itself is a collection of custom types

Variables

C# is a strongly-typed language(strict restrictions on intermixing values with different data types).

Nullable Types

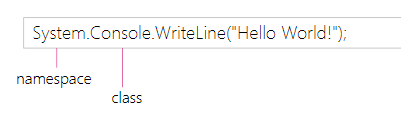
C# allows you to assign null values to value type variables

**Access Modifiers – accessibility level**

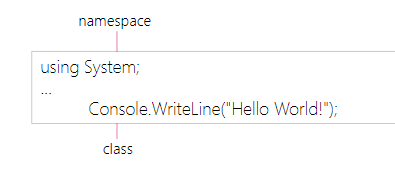
* Public
* Private – access only in same class
* Protected – same assembly and derived class has access
* Internal – same assembly only
* Protected Internal – same assembly and derived class

**Namespaces**

Containers of classes



The using keyword can be used so that the image above or the complete name isn’t necessary.



**Struct**

Suitable for representing values

**Members of a class or struct:**

* Fields – a variable of any type that is declared directly in a class or struct. Can be read or set directly. Use fields on variables with private or protected accessibility.
* Properties – Get and Set. Associated with Fields
* Constants – uses the keyword “const”. Immutable values meaning do not change for the life of the program
* Methods – actions to be performed in a class or struct
* Events - Triggers
* Operators
* Indexers
* Constructors – initializes the class
* Destructors – Destructs instances of a class. Used by garbage collector to free up space. Called when program exits.
* Nested Types – Class defined within another class.

**Sample Code:**

Class SampleClass

{

public static string SampleVariable = “This is a static member of SampleClass”.

private int \_mySample; // Field

public int sampleProperty {get; set;} // Properties. Contains get and set. Can be coded lengthily

public const myConstant = 100; //Constant. Maintains value. Static

public SampleClass() //Constructor. Same name with class

{

//initialize

}

Class NestedClass

{

}

}

int myVariable = SampleClass.myConstant; // accessed the constant

SampleClass.NestedClass newObject = new SampleClass.NestedClass(); // Calling a class with complete name without using the “using” keyword

Console.WriteLine(SampleClass.SampleVariable); //To access a static member

**Methods:**

**Parameters and Arguments(**values passed in a method with parameter/s**)**

**Method Overloading**

Using the same method names in a class but have different parameters

**Exception Handling**

* Try, Catch, Throw
* The try statement defines a code block to run (to try). The catch statement defines a code block to handle any error. The finally statement defines a code block to run regardless of the result. **The throw statement defines a custom error**.

**Collections**

Groups of related objects

Is a class

System.Collections.Generic //Library called to be able to use Lists

**Arrays**

int[] myArray = new int[5];

int[] anotherArray = {1,2,3,4,5,6,7};

myArray[0] = 1;

**Enumeration Type**

Uses the keyword “enum”

enum Days {Sunday, Monday, Tuesday};

Days today = Days.Monday;

int dayNumber = (int)today;

**conditional statements:**

**Selection Statement**

If , elseif, else

**Switch statements**

**Iteration Statements**

do, while, for, foreach

**Jump Statements**

break, continue

**Bootcamp Deck 2**

**Object Oriented Programming**

Programming Paradigm based on real-world objects

Pillars

Interfaces

Class modifiers: Abstracts and Sealed

Type conversions: Implicit and Explicit

Partial Class and Partial Methods

**Bootcamp Deck 3**

**ADO.Net**

A platform. Programming using entity models. Accesses and manipulates data

ADO.Net Entity Framework

Provides access to relational, XML, and application data

Two components of ADO.NET:

.NET framework Data Provider – used for connecting database, executing commands, and retrieving results

* Connection
* Command
* DataReader
* DataAdapter

Dataset – explicitly designed for data access. Representation of data

**ADO.NET LINQ**(Language-Integrated Query)

Extends powerful query capabilities to C# language

Select, Where, From, OrderBy

**ADO.NET Entity Framework**

Object-relational mapper

Goal is to decrease the amount of code

Data annotations provided by ADO.NET Entity Framework:

* Key
* Required
* Range

**Unit Testing Framework**

* TestClass
* TestMethod
* TestElements
* ADO.NET refresh

The common way to access a database from inside an ASP page is to:

1. Create an ADO connection to a database
2. Open the database connection
3. Create an ADO recordset
4. Open the recordset
5. Extract the data you need from the recordset
6. Close the recordset
7. Close the connection