Contents

[Day 2 .Net & App Architecture 2](#_Toc71023684)

[.Net 2](#_Toc71023685)

[.Net SDK vs Runtime 2](#_Toc71023686)

[Visual Representation of .Net5 (Review recording for imaging) 2](#_Toc71023687)

[.Net Standard 2](#_Toc71023688)

[Common Language Runtime (CLR) 3](#_Toc71023689)

[Another representation of the compilation process as described 3](#_Toc71023690)

[Managed Code 3](#_Toc71023691)

[Unmanaged Code 3](#_Toc71023692)

[Garbage Collector 4](#_Toc71023693)

[Discussions 4](#_Toc71023694)

[App Architecture: Organizing your code 5](#_Toc71023695)

[Good Developing Technique 5](#_Toc71023696)

[Separation of Concerns 5](#_Toc71023697)

[Classes 5](#_Toc71023698)

[Namespace 5](#_Toc71023699)

[Projects 6](#_Toc71023700)

[Solution 6](#_Toc71023701)

[Discussion 6](#_Toc71023702)

Revature

# Day 2 .Net & App Architecture

<Spaced out and forgot to start notes>

## .Net

-Cross platform/ Platform specific

>Write code once and implement in different Environments

## .Net SDK vs Runtime

-Software Development Kit (SDK) -**Read and Write**

>Includes everything you need to both build and run application

-Runtime **-Read only**

>Includes just the resources you need to run your application.

>Runtime is also included in the SDK

## Visual Representation of .Net5 (Review recording for imaging)

-.Net5

>Runtime Components

>Compilers

>Languages

-Tools

>Visual Studio I.E VS (Code/Mac/IDE)

>CLI

## .Net Standard

-Set of APIs that are implemented by .Nets Base Library

## Common Language Runtime (CLR)

-Services

>Memory Management

)Automatic garbage collector

* Deletes variables and storages after it is removed from scope

>JIT compilation *(C# isn’t much different than Java)*

) Assist in Cross Platforming (Visual explanation in Recording)

) IL = Interpreter Language (your .exe files)

) Machine Code = Standard binaries ( 1’s and 0’s)

) C# uses CLR , Java uses JVM

>Exception Handling

) Handles Syntax and Logical errors at compile and runtime

### Another representation of the compilation process as described

.cs file =compiles > .Net Compiler = translate > Intermediate Language = passes > CLR = passes to >

JIT Compiler = compiles > native code (Machine Code I.E Binary)

## Managed Code

-Any code managed by the CLR

>C# is an example

## Unmanaged Code

-Code that is developed outside .Net

>C is an example

-Can still be executed, Wrappers are required

## Garbage Collector

-Maintains your Heap memory

>Checks for objects in the managed heap that are no longer in use, and frees this memory for use.

>The Collector creates generations to track objects in the heap on how often that object is used or is referenced. The longer the generation the more it is referenced.

-Memory Outside of the CLR

>These resources must be handled manually using IDisposable Interface dispose() method

)Database connections is an example of something that must be handled with dispose()

)Classes that require IDisposable, should already inherit this Interface in their own code.

/// End of PP

## Discussions

1. What is an SDK?

SDK handles both the build and runtime of an application.

1. What’s the role of the CLR?

Memory Management IE Garbage collector

Runtime environment offered by .Net, compiles the IL to Machine Code using JIT.

Exception Handling during runtime

Allows for easily integrating between other languages

Helps with scalable and Multithreaded applications

1. What’s the role of the BCL?

Contains common runtime libraries used during C# Development.

AKA Base class library.

1. What is managed code?

Code that is covered under the .NET manager (CLR)

Managed code is anything compliant with the .NET CLR

1. What is unmanaged code?

Code that is not covered or compliant with the .NET CLR

1. What’s the garbage collection process?

Way to make memory management easier. Collect and end processes that are no longer in use

1. **Is managed code and unmanaged code, vocabulary specific to the CLR? (Question is not required knowledge, but knowledge is always good =D! )**

No, … and yes. While the specific verbiage is specific to CLR, the concept is used across other developments like the JVM. Ask other developers, and see what they call it. (Scala, Kotlin are other examples but they must be translated first)

# App Architecture: Organizing your code

*“There’s a place for everything, and everything in its place.”*

**Classes; How do you define those classes? How do you group those class?**

## Good Developing Technique

-Modularizing your code

>Separates your logic into separate categories.

## Separation of Concerns

-Group your data based on logic for whether they belong together.

-Avoid spaghetti code! Shoot for a Lasagna code.

>don’t have your code to be tangled, but to be layered.

-This is to help make your code more readable, extendable, Scalable, maintainable code.

>If everything has their place, and nothing is tangled together, clients can read your code more clearly.

>Makes it easier to explain your code as well!

## Classes

-Blueprints to the actual objects in your project.

-Building blocks of program.

-used to encapsulate logic and data together.

-abstract classes to handle higher end logic.

-EVERYTHING MUST EXISIT IN A CLASS.

-Do not create a superclass that handles everything in your code.

>This will help with debugging your code. Easier to find where the problem is by what the problem is.

## Namespace

-Logical grouping by theme of functionality

-to use classes from other namespaces, you must use the ‘*using’* statement.

>Difference between ‘*using*’ statement, keyword, and block.

-Similar to Java’s ‘*package*’

-the physical grouping of types are called ‘*assembly*’.

## Projects

-Contains all files that are to be compiled into .exe, library, or website

-Much like classes, don’t have a super project.

-Also contains compiles settings, as well as any dependencies (other classes you implement for your code to work) that your code might need.

-Contains an XML document that lists all different; Frameworks, Output type, References, etc.

>The XML and requirements will differ based on the nature of the project

-file extensions help define what the nature of the Project is (.csprojec = C# Project and vb.net = .net project.)

## Solution

-Not an “answer”

-a container for related projects. IE your final packaging of your application.

/// End of PP

## Discussion

Q: One main method per project or per Solution?

A: Per Solution

Q: A Solution can contain several projects?

1. Solution made of many Projects, Projects are made up of several namespaces, and namespaces are made up of many classes.

Q: Do you have any links for .Net Architecture?

1. (Found in resource package of PP)

Q: Can there be multiple namespaces to a file?

1. You can have multiple namespaces in a project. Like in java, certain classes belongs to certain packages. A type is part of a namespace. A namespace is part of a project.

Q: what’s the difference between namespace and assembly? What are we referencing with physical grouping?

1. When something becomes a physical file. Namespaces are virtual logical, Assembly is the folder location.

Venkata: Solution = Apartment Complex, Project = Apartment, Namespace= Specific apartment housing, Class= Stuff that exists within the apartment.

* What does separation of concerns mean?

Organizing your classes in ways that makes sense. Modularizing your code.

Advantages?:

Makes it easier to debug your code.

Makes it easier to explain your code to others.

Avoids naming complexes

IE: Readability and writability <<< VERY IMPORTANT

* What are classes?

Blueprint of the objects.

Define a data structure, Objects hold the actual data. Creating a class, you are creating a data instance. Different way of describing it.

* What is the difference between Assembly and namespaces?

<Missed while editing the file>

* What is the difference between projects and namespaces?

Projects are made up of many namespaces.

* How many projects can be included in a solution?

One or more

Multiple Projects belong to one Solution

[[1]](#endnote-1)

1. You can find Recordings on [Revature Connect](https://connect.revature.net/). You can also find the Notes at [↑](#endnote-ref-1)