**----GOAL OF MODULE----**

To teach attendees of C# language and .net framework core concepts in gradual manner, while introducing computer science concepts that can be applied in other programming languages

It should be minimal and lean on content only with the most important parts left

Ultimate goal is that after the module a person, who attended it, could read small-medium sized project code, edit it or extend it. Also, would be able to create simple solutions from scratch with readable code design.

**----BEFORE START----**

-Do we accept everyone?

-Minimal requirements for attendant PC setup

-Github account (all material and assignments will be put into github)

Tooling:

1. Visual studio (community);

Mid-Soft introduction (10-15 min):

1. Usual day @ work;
2. Scare ‘em;

#Manifesto:

1. Less depth (time constraint).

Lecture order:

1. Take turns @ presenting their own homework @ the start of lecture;
2. Every lecture has dedicated homework assignment;

(I plan that visual studio intro will come by default during curriculum, other stuff taught up front can be over complicated to people who does not know much about programming and would not stick as information without proper context).

Things like configuration and entry point are not of a core importance.

**----TRAINING----**

**-#1 Getting familiar with syntax (ALLA->GSAG)**

                -Variables

                -Loops

                -Conditional statements

                -Simple data structures (Array, List)

#Homework 1 - find integer in a list and return it's index (Console app)

#Homework 2 - sort an integer list (Console app)

**-#2 Object Oriented Programming (GSAG->KKAJ)**

                -Classes

                -Structs

                -Access modifiers (By default, but can be made explicit)

                -Objects

                -Fields

                -Properties

                -Methods

#Homework 3 - Implement linked list by given contract (Console app) (No generics, string value)

#Homework 4 - Implement abstract reader and concrete file, console readers for data input

**-#3 Advanced Object Oriented Programming (KKAJ->GSAG)**

                -Interfaces

                -Generics

                -Equality (reference, value)

                -Variable references and out parameters

                -Optional arguments, default values

                -Delegates (Func, Action) Yes, this part includes both, again – less specific = less complicated

                -Events At first I skipped this concept but rethought and agree that it is mandatory, so let’s include

#Homework 5 - LinkedList should implement IList (Homework 3)

#Homework 6 - LinkedList should become generic (Homework 5)

#Homework 7 - Extensible methods in Nodes like ToString virtual (Homework 6)

**-#4 Data structures (GSAG -> ALLA)**

                -Key value collection / dictionary

                -Hash Table

                -Queue

                -Stack I chose not to have both (queue and stack) just because it is simply LIFO vs FIFO so maybe not worth spending time on both?

                -Tree

                -Graph

-IEnumerable (Good suggestion, just moved lower as it is not a mandatory concept to grasp before getting into collections)

                -Linq to collections

#Homework 8 - Implement a graph data structure

#Homework 9 - Implement DFS and BFS methods for graph structure

**-#5 Resource management (ALLA->VVAS)**

                -Memory space = heap / stack

                -Un-/managed memory Managed memory will be covered in first topic, while unmanaged might be left to future curriculums? I do not think average C# developer does stuff with unmanaged as BAU activities

                -Object life time (initialization, generations?, garbage collection)

                -Casting

                -IDisposable

                -Static vs Instance Probably worth mentioning

#Homework 10 - Abstract reader (Homework 4) should implement IDisposable as abstract method and concretes should override it

**-#6 Multihreading and Asynchronous programming (VVAS->KKAJ)**

                -Multithreading

                -Asynchronous programming

                -Concurrency abstractions

                -Concurrent Collections I do not think this is a good place for such topic, plus I do not think it is necessary to building coding foundation

#Homework 11 - Make LinkedList threadsafe (Homework 7)

#Homework 12 -Make abstract reader async, implement that in concrete readers (Homework 10)

**-#7 Code design (KKAJ->ALL)**

                -~~SRP~~ S.O.L.I.D. I chose only SRP not to overwhelm people with full S.O.L.I.D., SRP is one of the best simple patters that forces to think about code design

                -DRY

                -KISS

                -Naming conventions and comments

                -Design patterns? I do not think this should be part of the curriculum as it is advanced topic and will not be digested easily

**----Reading List----**

C# 7.0 in a nutshell

Clean Code

Paskaitos - 7 +2.

GITHUB:

1. <https://github.com/DuckAssociation>
2. <https://github.com/DuckAssociation/Homework>
3. https://github.com/DuckAssociation/CSharpCourse