TASK BY : AMMAR

MAY 2025

FINALIZER

THE FINALIZER IS A PIECE OF PROGRAM CODE THAT RUNS BEFORE THE OBJECT IS FINALLY CLEARED FROM MEMORY. IT'S LIKE SAYING TO THE OBJECT, "OKAY, YOU'RE LEAVING HERE. IF YOU HAVE ANY LAST THINGS YOU WANT TO DO BEFORE YOU LEAVE, DO THEM NOW." IMAGINE YOU'RE RENTING AN APARTMENT AND THE LANDLORD DECIDES TO DEMOLISH THE BUILDING (SIMILAR TO GC WHEN IT DELETES AN OBJECT). BUT BEFORE DEMOLISHING, HE LETS YOU PACK YOUR THINGS AND LEAVE. THIS IS EXACTLY WHAT THE FINALIZER DOES, GIVING YOU ONE LAST CHANCE TO ORGANIZE YOUR THINGS BEFORE THE DELETION. THIS USUALLY HAPPENS WHEN THE GARBAGE COLLECTOR (WHICH IS RESPONSIBLE FOR CLEARING MEMORY) DECIDES THAT NOTHING IN THE PROGRAM NEEDS THE OBJECT ANYMORE, SO IT DELETES IT AND MAKES ROOM FOR IT. WHEN YOU CREATE OBJECTS IN MEMORY, THERE COMES A TIME WHEN YOU NO LONGER NEED THEM. INSTEAD OF THE MEMORY REMAINING FULL OF UNNECESSARY OBJECTS, SOMETHING CALLED THE GARBAGE COLLECTOR (GC) COMES INTO PLAY, REMOVING THESE OBJECTS FROM MEMORY. THERE ARE TWO TYPES: IMPLICIT COLLECTION (GC): THIS IS THE NORMAL TYPE; THE GC RUNS ON ITS OWN; YOU DON'T HAVE TO DO ANYTHING. EXPLICIT COLLECTION (GC): YOU TELL THE GC TO "RUN NOW."

NESTED CLASS

CLASSES ARE THE HERO BUG THAT BRINGS EVERYTHING TOGETHER: DATA AND BEHAVIOR (DATA = VARIABLES, BEHAVIOR = FUNCTIONS). BUT SOMETIMES YOU ENCOUNTER SITUATIONS THAT REQUIRE MORE THAN A SIMPLE CLASS... SOMETIMES A CLASS WITHIN A CLASS, OR AN OBJECT WITHIN AN OBJECT. THIS IS WHERE OUR STORY BEGINS.IMAGINE YOU HAVE A FACTORY, AND WITHIN THAT FACTORY THERE'S A SMALL DEPARTMENT RESPONSIBLE FOR A SPECIFIC TASK. THIS DEPARTMENT CAN'T OPERATE INDEPENDENTLY; IT MUST BE WITHIN THE FACTORY.THIS IS CALLED A NESTED CLASS — A CLASS WITHIN A CLASS.
IMAGINE THE FACTORY HAS TWO DOORS:
PUBLIC: ANYONE FROM THE STREET CAN ENTER.
INTERNAL: ONLY EMPLOYEES WITHIN THE COMPANY (THE SAME PROJECT) CAN ENTER.

LET'S CONTINUE WITH THE FACTORY EXAMPLE:THERE ARE MACHINES (MEMBERS) INSIDE THE FACTORY. SOME ARE ON PUBLIC DISPLAY (PUBLIC).
SOME ARE IN CLOSED ROOMS (PRIVATE).
PUBLIC: YOU CAN ACCESS IT FROM OUTSIDE THE CLASS.
PRIVATE: YOU CAN ACCESS IT ONLY FROM WITHIN THE CLASS ITSELF.
WHY PUT A CLASS INSIDE A CLASS? LET ME GIVE YOU A SCENARIO: YOU HAVE AN EMAIL PROGRAM IN A CLASS CALLED EMAILCLIENT, INSIDE OF WHICH IS ANOTHER CLASS CALLED SETTINGS.
THESE SETTINGS ARE NOT NEEDED OUTSIDE THE EMAILCLIENT, SO INSTEAD OF LEAVING THEM ALONE, WE NEST THEM.
BENEFIT:
BETTER ORGANIZATION. INNER CLASSES CAN ACCESS MEMBERS OF THE OUTER CLASS. CLUTTER IS REDUCED IF THE SECOND CLASS IS CLOSELY LINKED TO THE FIRST.
LET'S GO BACK TO THE FACTORY AGAIN. THIS FACTORY OWNS MACHINES. IF THE FACTORY SHUTS DOWN, THESE MACHINES ARE SHUT DOWN ALONG WITH IT.
THIS IS CALLED COMPOSITION: THE LARGER CLASS OWNS OBJECTS FROM OTHER CLASSES, AND YOU CONTROL THEIR LIVES. CLASS MEMBERS CHECKLIST SO, THE ENGINEER IN CHARGE OF THE FACTORY CAME TO YOU AND SAID: "I WANT TO MAKE SURE THAT EVERY MEMBER IN THE CLASS IS CONFIGURED CORRECTLY!"
CHECKLIST:
IS THE NAME CLEAR AND DESCRIPTIVE?
DOES THIS MEMBER NEED TO BE PUBLIC OR PRIVATE?
IS THERE AN APPROPRIATE INITIAL VALUE?
DO YOU NEED A PROPERTY INSTEAD OF A DIRECT VARIABLE?
IS THERE DATA SECURITY? THAT MEANS NO ONE CAN CHANGE THE VALUE WITHOUT PERMISSION?

DEBUGGING

DEBUGGING IS THE PROCESS OF DETECTING AND CORRECTING ERRORS IN A PROGRAM. AN ERROR IS ANYTHING THAT CAUSES A MALFUNCTION OR FAILURE IN THE PROGRAM'S OPERATION. IN C#, ERRORS ARE DIVIDED INTO THREE MAIN TYPES:
SYNTAX ERROR (TYPING ERRORS)
RUNTIME ERROR (ERRORS DURING EXECUTION)
LOGICAL ERROR (LOGICAL ERRORS IN REASONING)

SYNTAX ERROR

Syntax Error | Syntax Error Definition:

An error that occurs when code is written in a way that violates the C# syntax.

Example:

Forgetting parentheses { }.

Writing a reserved word incorrectly (such as intt instead of int).

Calling an undefined function.

The program doesn't compile at all if there is a syntax error.

RUNTIME ERROR

Runtime Error | Error During Runtime

Definition:

An error that occurs during program execution, after the code has been correctly written and compiled.

Famous Examples:

Division by Zero

LOGICAL ERROR

Logical Error

Definition: It's an error in the logic of the program. This means the code runs without crashing, but the result is wrong! The code works, but it's doing something you didn't expect. This is the most dangerous type of error because it's easily seen!

Debugger tools help you pause your code and see everything happening in real time.The most important tools are:

Breakpoints: Stop the program at a specific line.

Step Over / Into / Out: Move step by step and view the values.

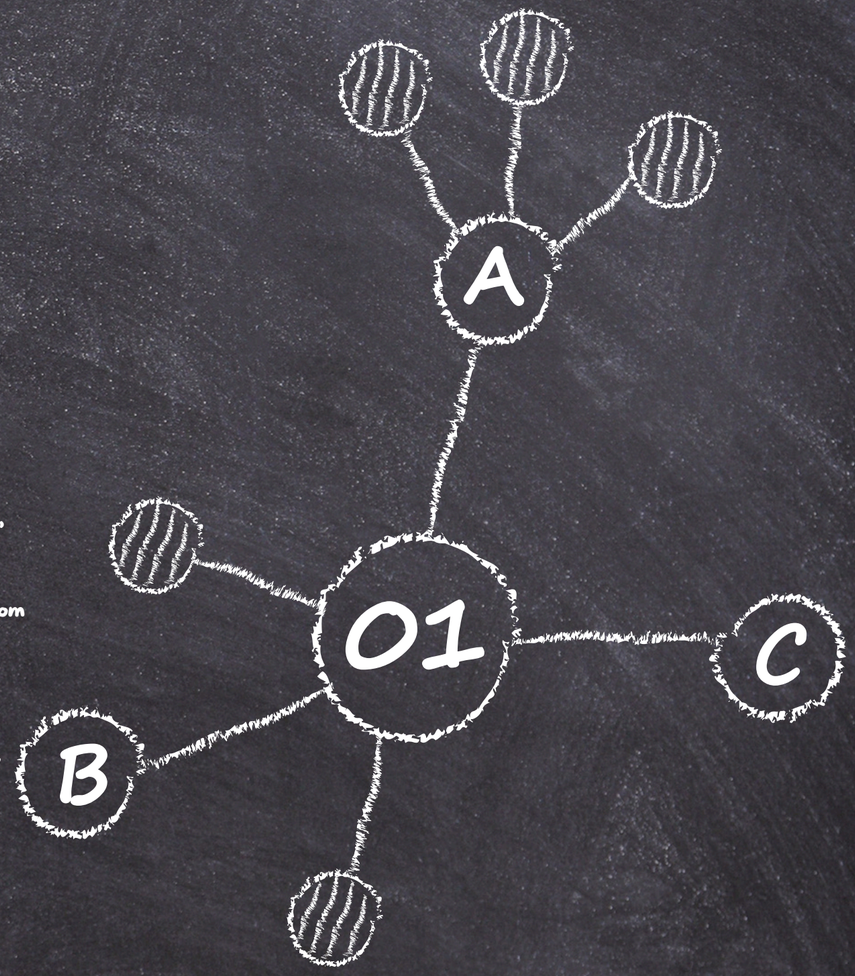
Watch: Track the value of a specific variable at all times.

Immediate Window: Type commands and view their results live.

Tracing: Record the steps of running code using Trace.WriteLine() or Debug.WriteLine().

STRUCT

In C#, we have two basic types of containers that we use to define objects: class = class (flexible and suitable for complex entities) struct = struct (simpler, faster, and suitable for small data)
Definition of struct | Definition of struct: A struct is a value type that stores data directly, not by reference, and is stored in a stack instead of a heap.
Use it when:You want to store simple data sets together. Performance is important (because it's lighter than a class) and it doesn't need to be inherited from another class.
Immutable vs. Mutable | Definition:
Mutable = Values can be changed after creation.
Immutable = Values are constant after creation, like a string.
Tip: It's best to use an immutable struct because mutable structs can cause problems with implicit copying (such as when inserting them into a list or a new variable).
Readonly struct | Read-only struct
Definition:
Readonly struct = A structure whose state cannot be changed after creation.
Why use it?
To ensure immutability, optimize performance, and protect against unintended errors.
DateTime struct | Date and Time Structure
DateTime is the most popular working example of a struct in C#.
Definition:
The DateTime struct is responsible for representing dates and times.
Its attributes:
Immutable and rich in functions: Add, Subtract, Compare, Format, and has constant properties: DateTime.MinValue, DateTime.MaxValue.

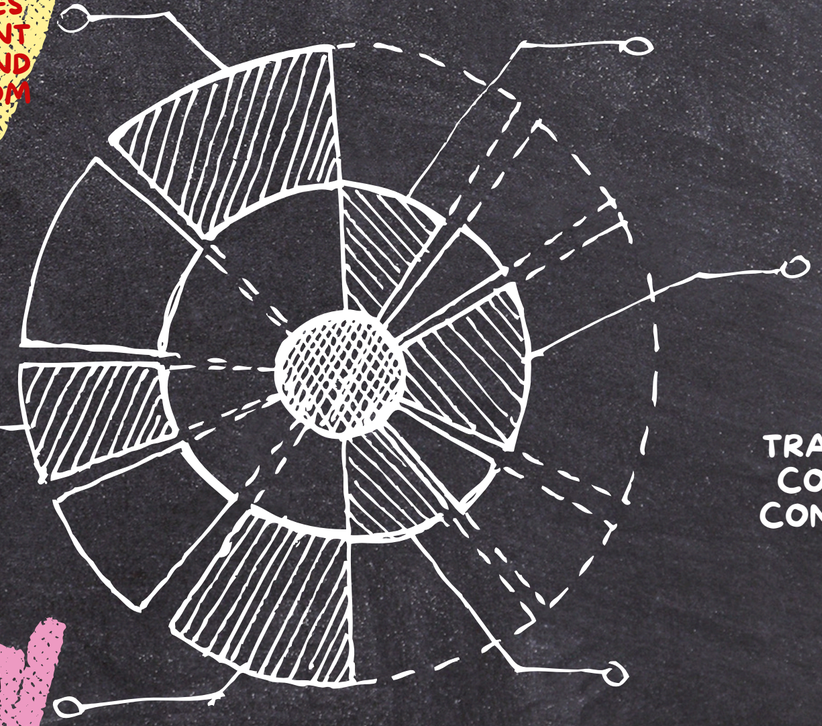


ENUM

AN ENUM IN C# IS A WAY TO DEFINE A SET OF FIXED VALUES (LOGICAL CONSTANTS) THAT HAVE NAMES BUT REPRESENT NUMBERS UNDER THE HOOD. IT HELPS YOU WRITE CLEARER AND EASIER-TO-UNDERSTAND CODE INSTEAD OF USING RANDOM NUMBERS.

DEFINITION | ENUM DEFINITION: ENUM IS SHORT FOR ENUMERATION, AND IS A DATA TYPE THAT ALLOWS YOU TO DEFINE A SET OF PREDEFINED VALUES. IT IS USED WHEN YOU HAVE A FIXED AND KNOWN SET OF VALUES, SUCH AS DAYS OF THE WEEK, COLORS, OR USER PERMISSIONS.

FLAGS ENUM | POINTER ENUM: FLAGS ENUM IS A SPECIAL TYPE OF ENUM THAT ALLOWS YOU TO GROUP MORE THAN ONE VALUE INTO A SINGLE VARIABLE, USING LOGICAL OPERATIONS (BITWISE) AND IS USED IN ACCESS PERMISSIONS FOR MULTIPLE PROPERTIES IN CASES WHERE MORE THAN ONE OPTION MAY BE PRESENT.



```
ENUM TRAFFICLIGHT
    {
    RED,
    YELLOW,
    GREEN
    }

CLASS PROGRAM
    {
STATIC VOID MAIN()
    {
TRAFFICLIGHT SIGNAL = TRAFFICLIGHT.RED;
CONSOLE.WRITELINE(SIGNAL); // يطبع: RED
CONSOLE.WRITELINE((INT)SIGNAL); // 0 :يطبع
    }
    }
```

METRICS

PROVIDE A PLAN FOR TRACKING AND ANALYZING THE KPIS THROUGHOUT THE CAMPAIGN.

WHAT IS A DELEGATE?

A Delegate is an object that represents a reference to a function (like pointers in C++, but safe and controlled in C#). It allows you to pass functions as parameters and change their behavior at runtime.

DELEGATE

A delegate is a data type used to define a method signature. Advantages: Allows flexibility in function implementation. Can be used with events. Can refer to more than one function (multicast delegate).

GENERIC DELEGATE

C# allows you to create a global delegate using Generics.