**Call Stack in C# 🚀**

**What is a Call Stack?**

The **Call Stack** in C# is a **last-in, first-out (LIFO) data structure** that keeps track of method calls during program execution. It helps manage:

* **Method execution flow**
* **Local variables**
* **Return addresses** (where execution should resume after a method finishes)

**How the Call Stack Works**

1. When a method is called, a **stack frame** is **pushed** onto the stack.
2. The stack frame contains:
   * **Method parameters**
   * **Local variables**
   * **Return address**
3. When the method completes, the stack frame is **popped** off, and execution returns to the caller.

**Example of Call Stack in Action**

csharp

CopyEdit

using System;

class Program

{

static void MethodA()

{

Console.WriteLine("Inside MethodA");

MethodB(); // Call MethodB

Int a=7;

Console.WriteLine(a);

}

static void MethodB()

{

Console.WriteLine("Inside MethodB");

MethodC(); // Call MethodC

}

static void MethodC()

{

Console.WriteLine("Inside MethodC");

}

static void Main()

{

Console.WriteLine("Starting Program");

MethodA(); // Call MethodA

Console.WriteLine("Program Finished");

}

}

**Call Stack Execution Flow**

1. Main() is the first method to execute.
2. Main() calls MethodA(), which is **pushed** onto the stack.
3. MethodA() calls MethodB(), which is **pushed** onto the stack.
4. MethodB() calls MethodC(), which is **pushed** onto the stack.
5. MethodC() finishes execution and is **popped** off the stack.
6. Execution returns to MethodB(), which then completes and is **popped**.
7. Execution returns to MethodA(), which then completes and is **popped**.
8. Finally, execution returns to Main(), and the program ends.

**Call Stack at Each Step**

| **Step** | **Stack (Top → Bottom)** |
| --- | --- |
| Start | Main() |
| MethodA() called | MethodA() → Main() |
| MethodB() called | MethodB() → MethodA() → Main() |
| MethodC() called | MethodC() → MethodB() → MethodA() → Main() |
| MethodC() returns | MethodB() → MethodA() → Main() |
| MethodB() returns | MethodA() → Main() |
| MethodA() returns | Main() |
| Main() finishes | *(Stack is empty)* |

**Call Stack Overflow (StackOverflowException)**

If there are too many function calls without returning, the stack overflows, causing a **StackOverflowException**.

🔴 **Example of Stack Overflow:**

csharp

CopyEdit

void RecursiveMethod()

{

RecursiveMethod(); // Infinite recursion

}

💡 Since there is **no exit condition**, the stack keeps growing until memory runs out.

✅ **Fix: Use Base Condition**

csharp

CopyEdit

void RecursiveMethod(int count)

{

Count=count-1;

if (count == 0) return;

RecursiveMethod(count - 1);

}

Void Main(string args[]){

Int count=5;//Global

RecursiveMethod(count);

Consol.WriteLine(count);//5

}

**Key Takeaways**

✅ **LIFO (Last In, First Out)** structure.  
✅ Stores **method calls, local variables, return addresses**.  
✅ **Automatically managed** (stack frames are created & removed).  
✅ **StackOverflowException** happens when too many calls are made without returning.  
✅ Used for **method execution tracking** and debugging.