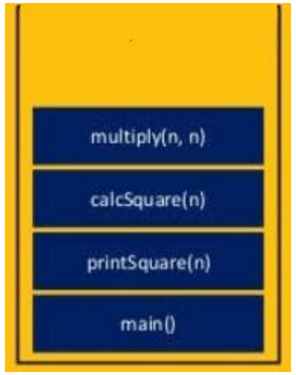
Al Hussein
Technical University
جامعة الحسين التقنية

Data Structures and Algorithms

The Call Stack



Prepared by:

Eng. Malek Al-Louzi

School of Computing and Informatics– Al Hussein Technical University

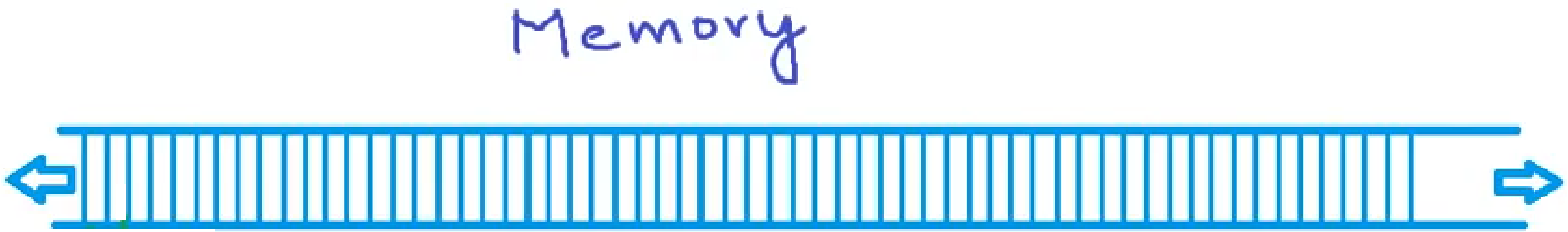
Spring 2021/2022

Outlines

- Introduction
- Local Variables
- The Call Stack
- Notes

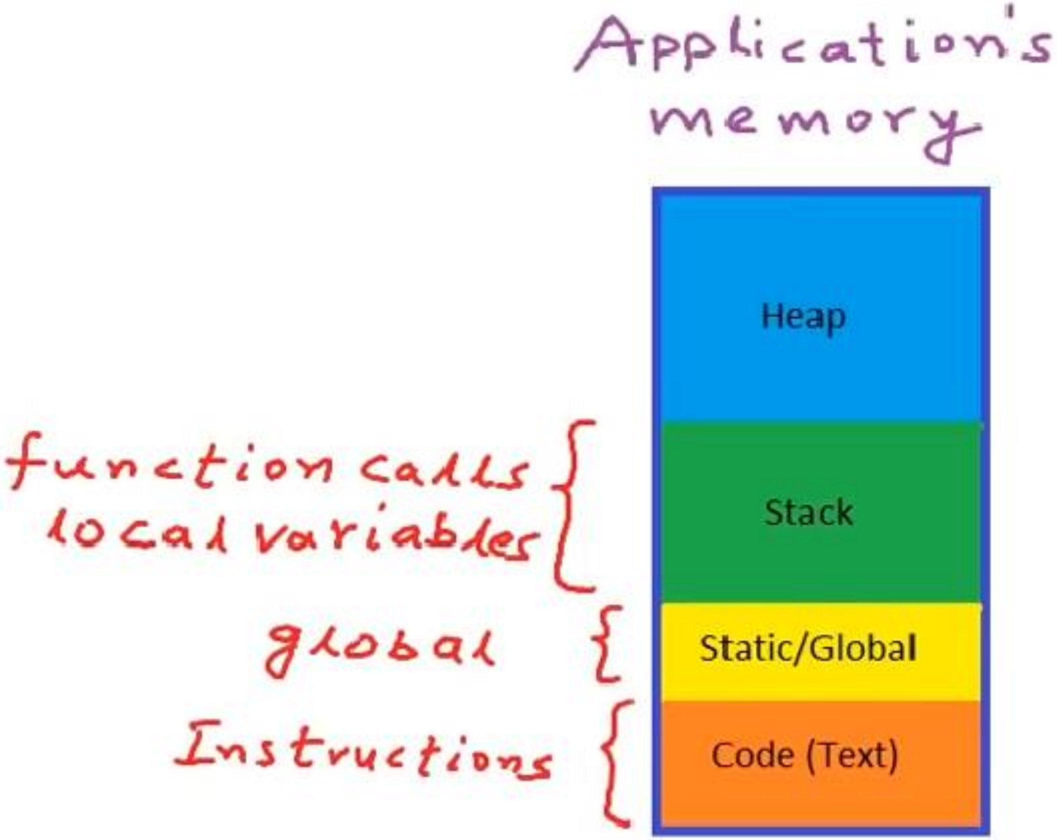
Introduction

- Memory is an important and crucial resource in our machine.
- It always good to know:
  - The architecture of memory.
  - The way Operating System manages memory.
  - The way memory is accessible to us as programmers.



Introduction

- The Memory that is assigned to a program or application in a typical architecture can be divided into Four segments:

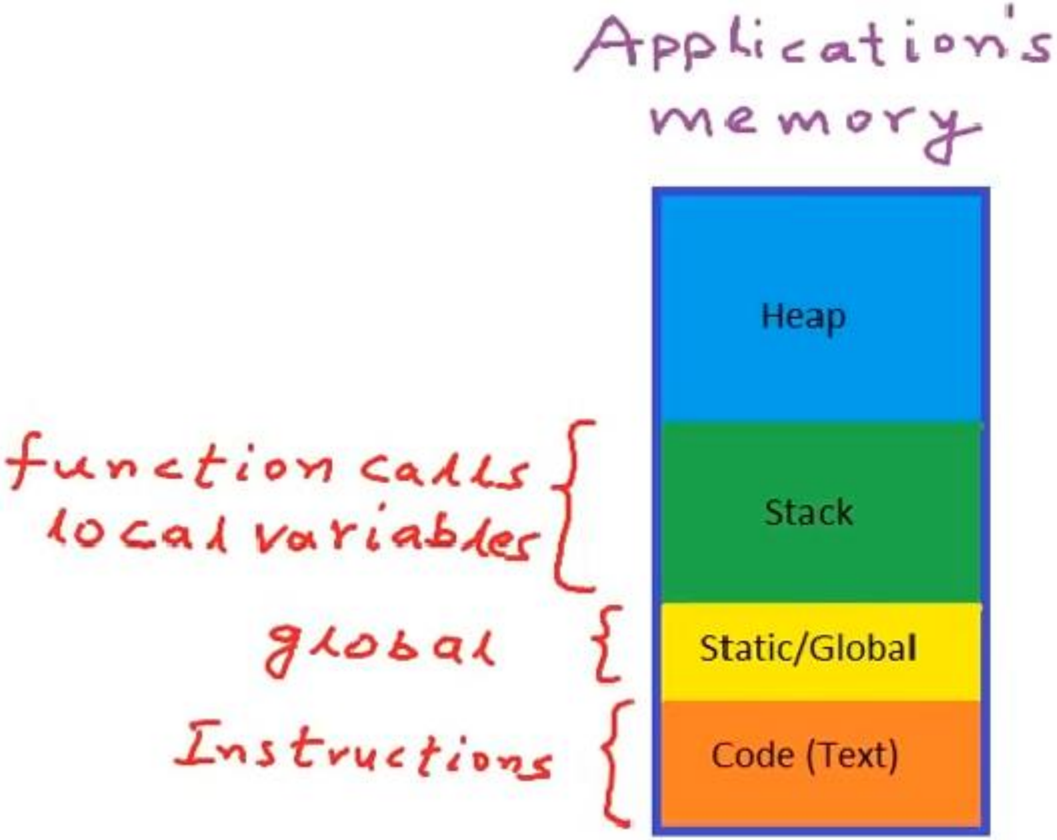


Introduction

- Code (Text):
  - Is assigned to store the instructions that need to be executed.
- Static/ Global:
  - Stores all static and global variables, that have the whole lifetime of an application as long as the application is running.
- Stack:
  - Used to store all the information of function calls and all local variables.
- The amount of memory set aside for the previous three segments does not grow while the application is running.

Local Variables

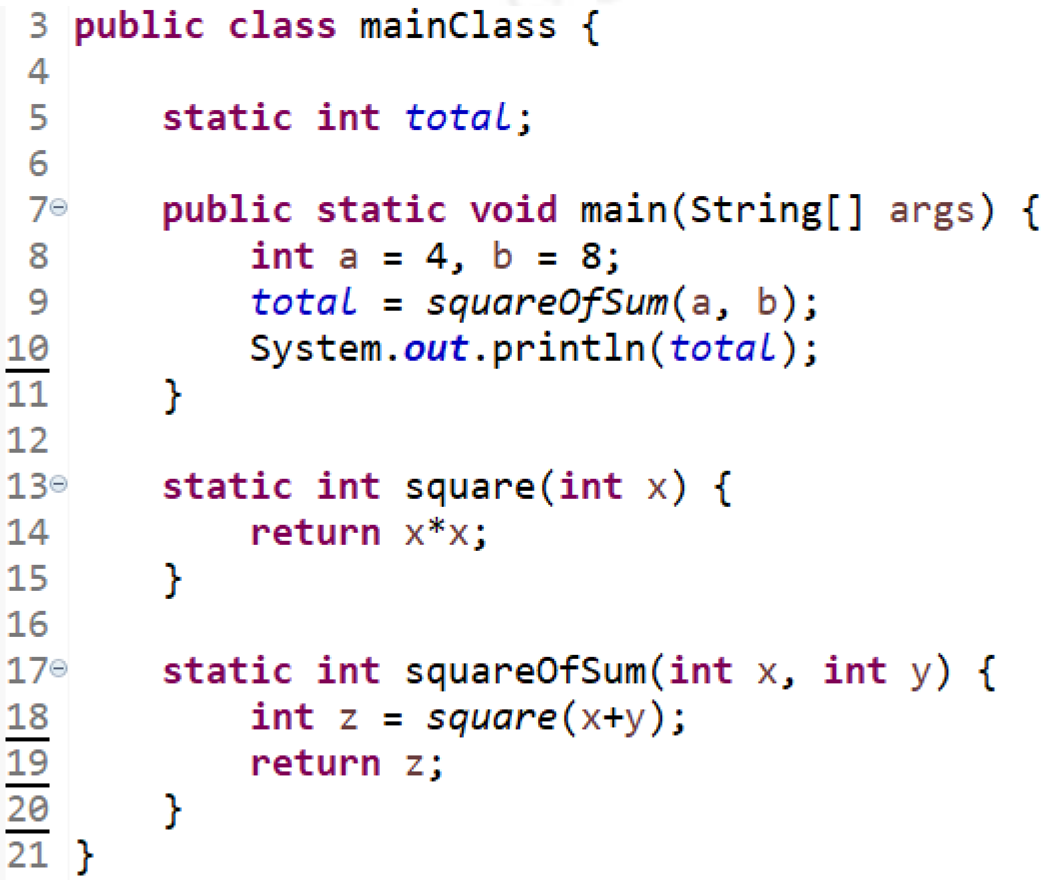
- Local variables are declared inside functions.
- They live only till the time the function is executing.



The Call Stack

- Let us understand how the Stack segment of the memory is used when a program executes.
- We have the following

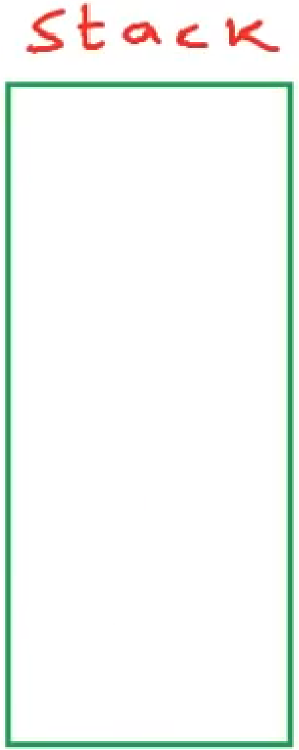
simple Java program.



```
public class mainClass {

    static int total;

    public static void main(String[] args) {
        int a = 4, b = 8;
        total = squareOfSum(a, b);
        System.out.println(total);
    }

    static int square(int x) {
        return x*x;
    }

    static int squareOfSum(int x, int y) {
        int z = square(x+y);
        return z;
    }
}
```

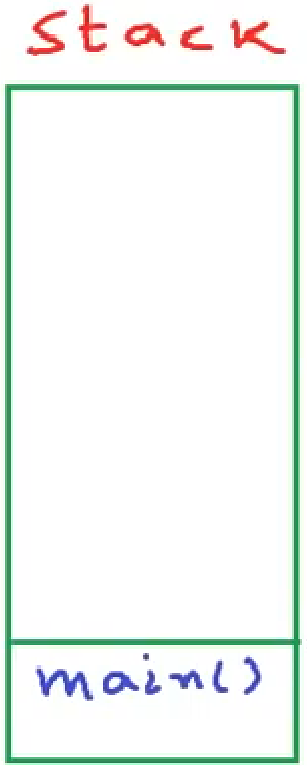
The Call Stack

- We will see what happens in the stack segment when the previous program executes.
- The following green rectangle is the memory reserved as stack segment.



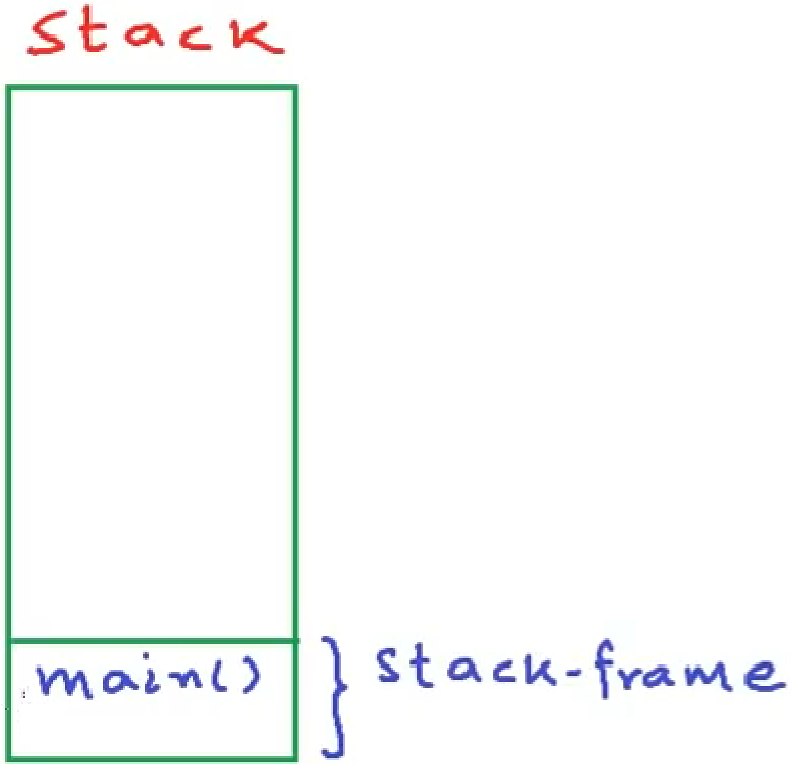
The Call Stack

- When the program starts executing, first the main method is invoked.
- When it is invoked, some amount of memory from the stack is allocated for execution of main method.
- We can say the main method is pushed on the stack.



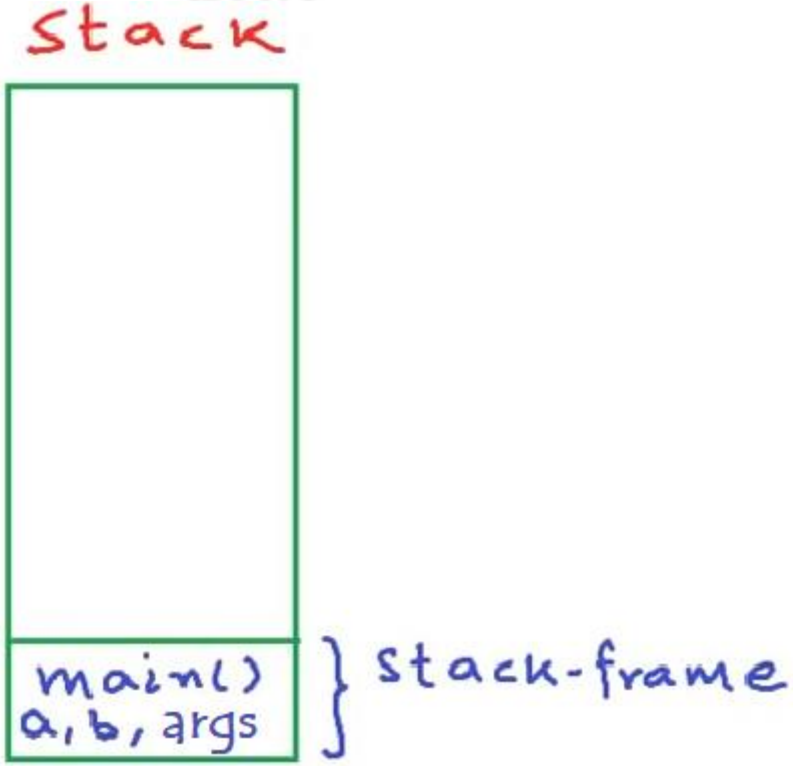
The Call Stack

- The amount of memory allocated in the stack for executing the main can also be called the stack frame.



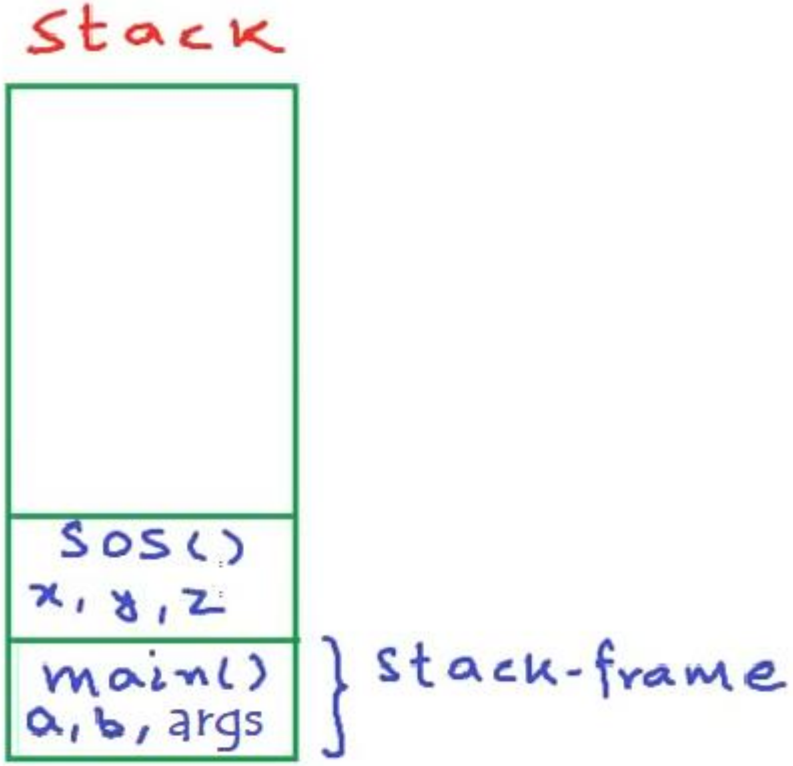
The Call Stack

- All the local variables, arguments and the information where this function should return back to are sored within this stack frame.



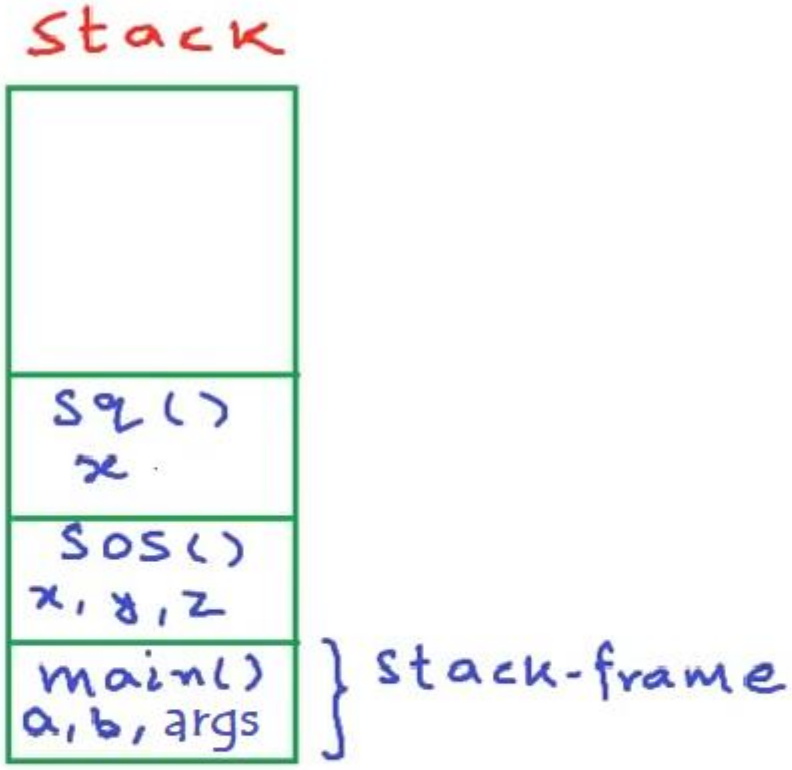
The Call Stack

- The size of the stack frame for a method is calculated when the program is compiling.
- Now, when main calls SquareOfSum method (SOS) for shortcut, then a stack frame is Allocated for it.



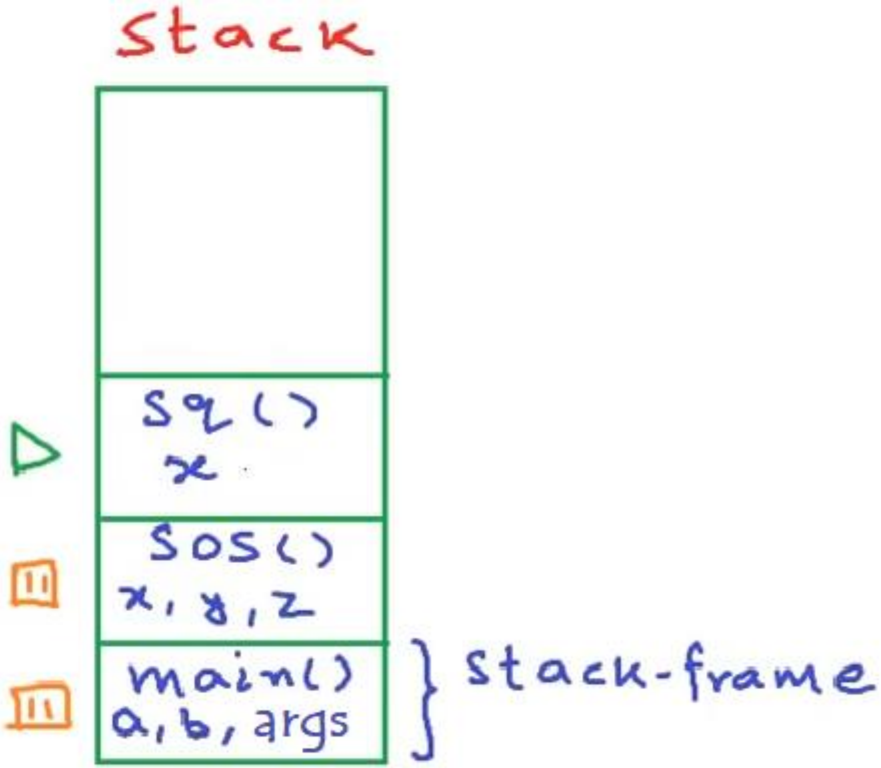
The Call Stack

- Now, SOS calls square (sq) for shortcut, another stack frame for sq is allocated.



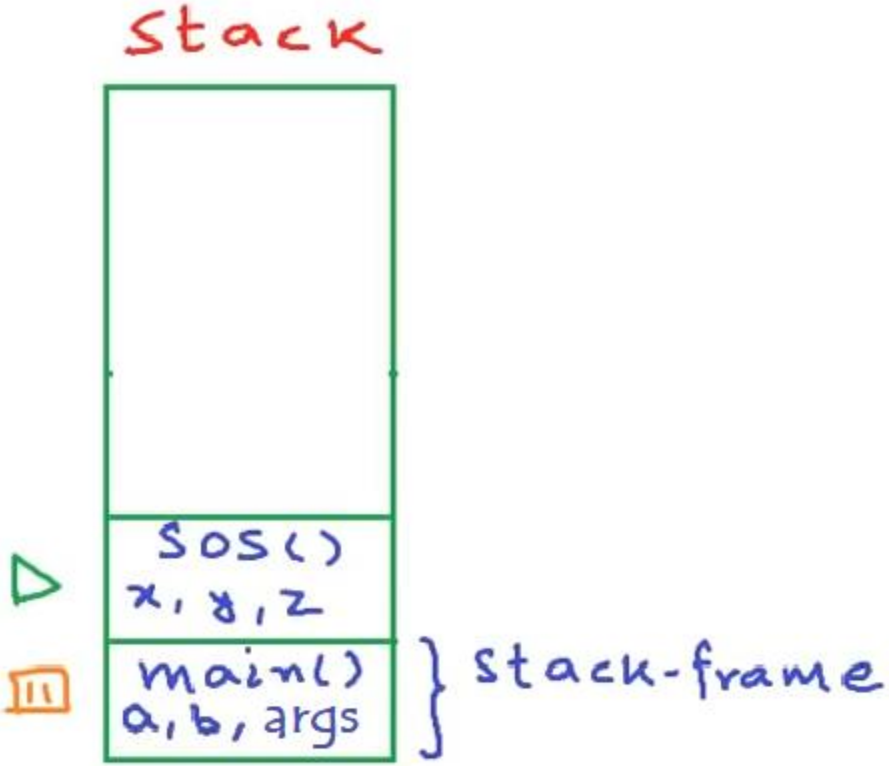
The Call Stack

- At anytime during the execution of the program, the function at the top of the stack is executing.
- The rest are paused, waiting for the function above to finish.



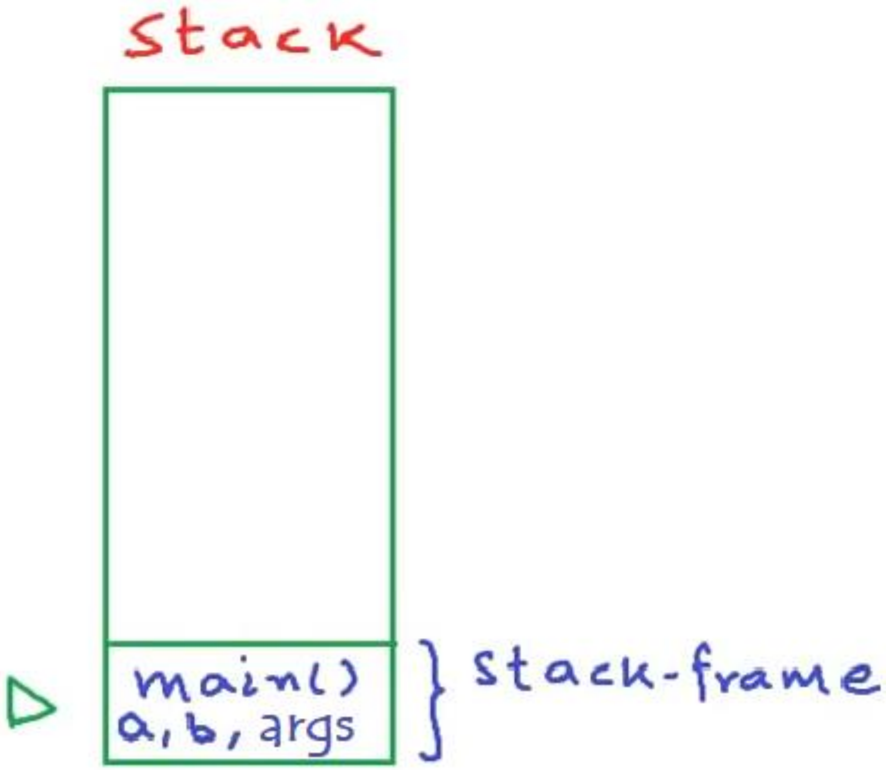
The Call Stack

- When sq finishes, it will be cleared (popped) from the stack memory.
- Now, SOS function will resume execution.



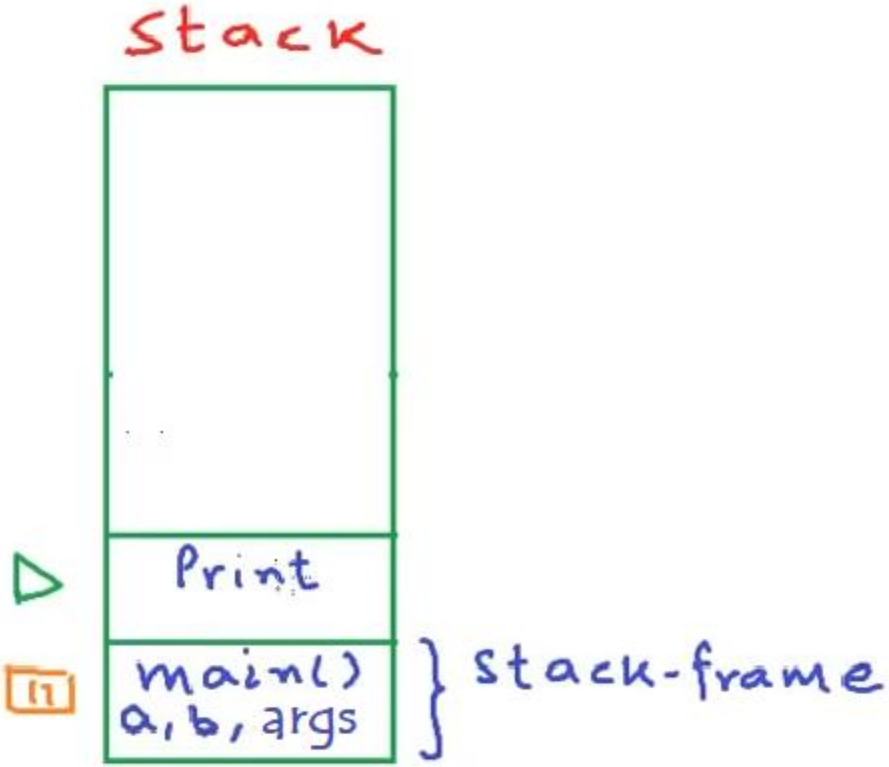
The Call Stack

- When SOS finishes, the main will resume execution.



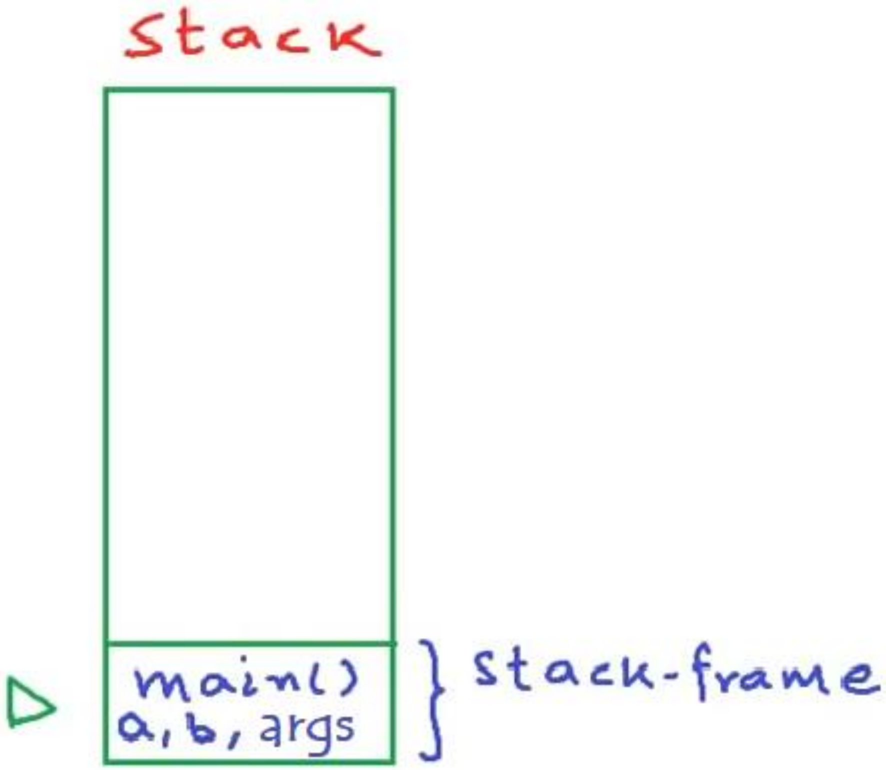
The Call Stack

- Finally, main will call print method, so print will be pushed on the top of the stack.



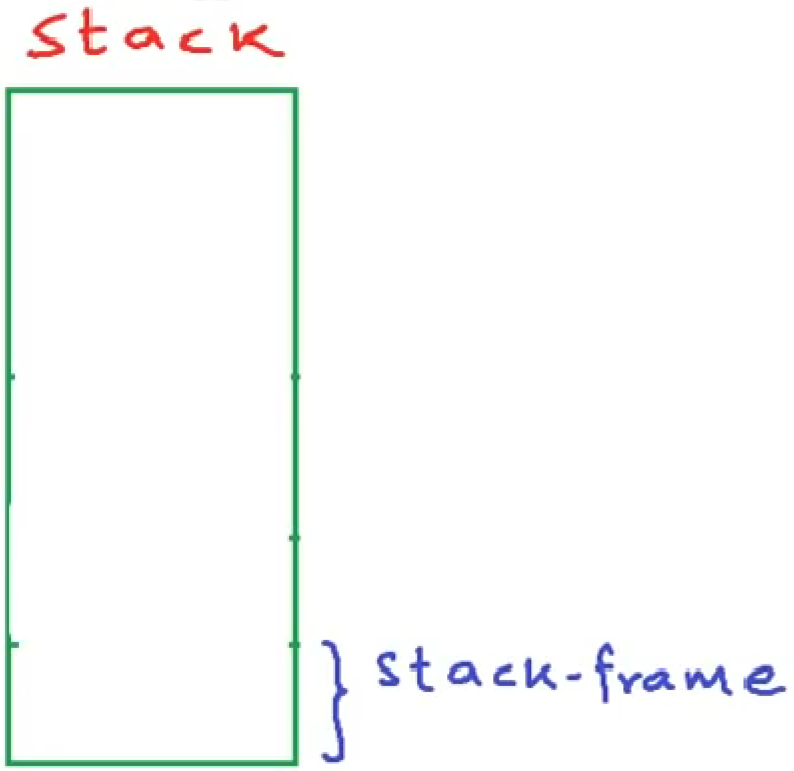
The Call Stack

- Then, print will finish and main will resume.



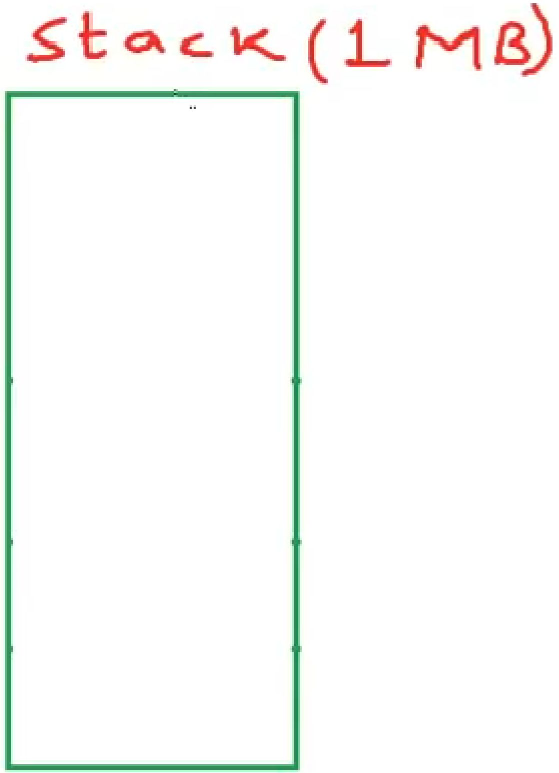
The Call Stack

- Now main will finish.
- When main finishes, the program will also finish.



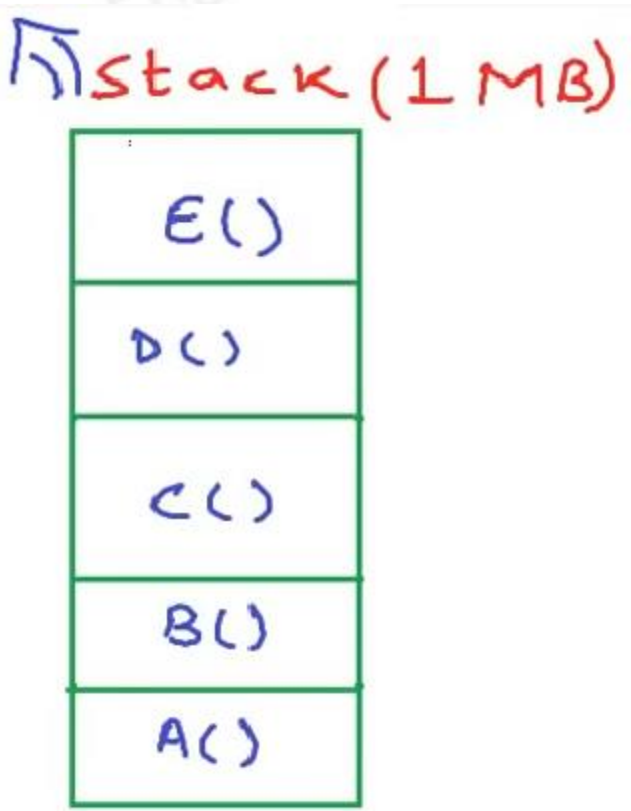
Notes

- When our program starts, the Operating System allocates some amount of memory as stack segment, Let us assume OS allocates 1 MB as stack.



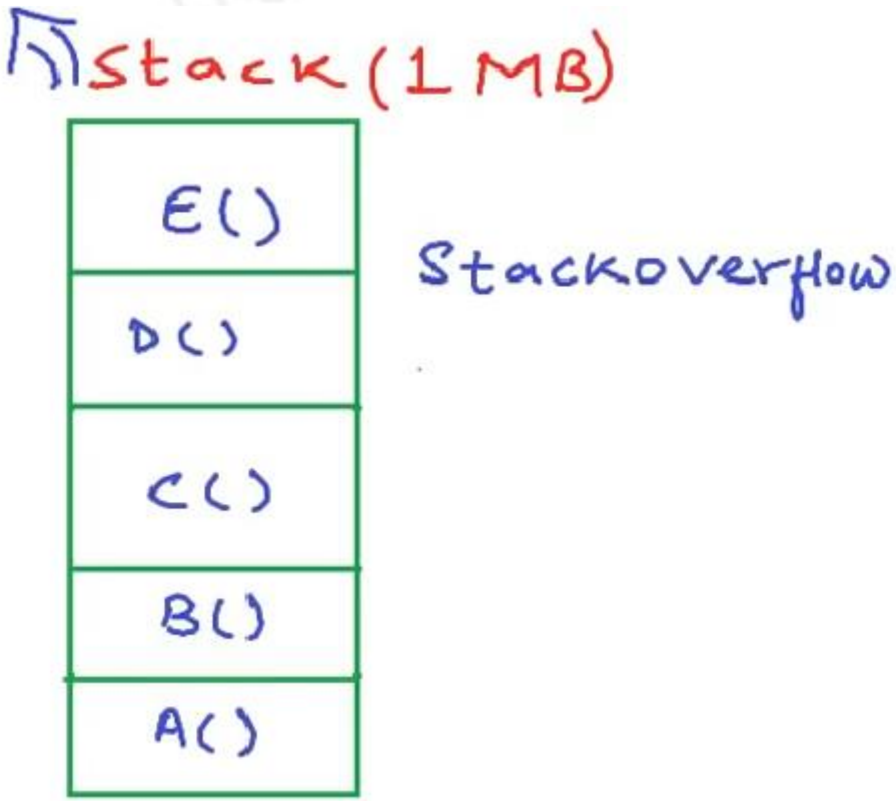
Notes

- The actual allocation of stack frame (the function is pushed in the stack), happens during runtime.
- If the call stack grows beyond the reserved memory for the stack.



Notes

- Then this is called stack overflow, in this case our program will crash.
- For example, if method A calls B, B calls C, we go on calling, and we exhaust the whole space reserved for the stack.



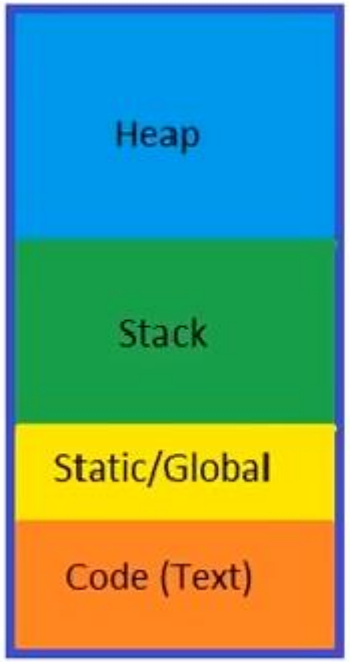
Notes

- As we noticed, There are some limitation of stack segment.
- The memory set aside for stack does not grow during runtime, application cannot request more memory for stack.
- If it is 1 MB, and the allocations of local variables and functions in stack exceeds 1 MB, then the program will crash.
- For this limitation, when allocating complex datatypes, we have the heap segment.

Notes

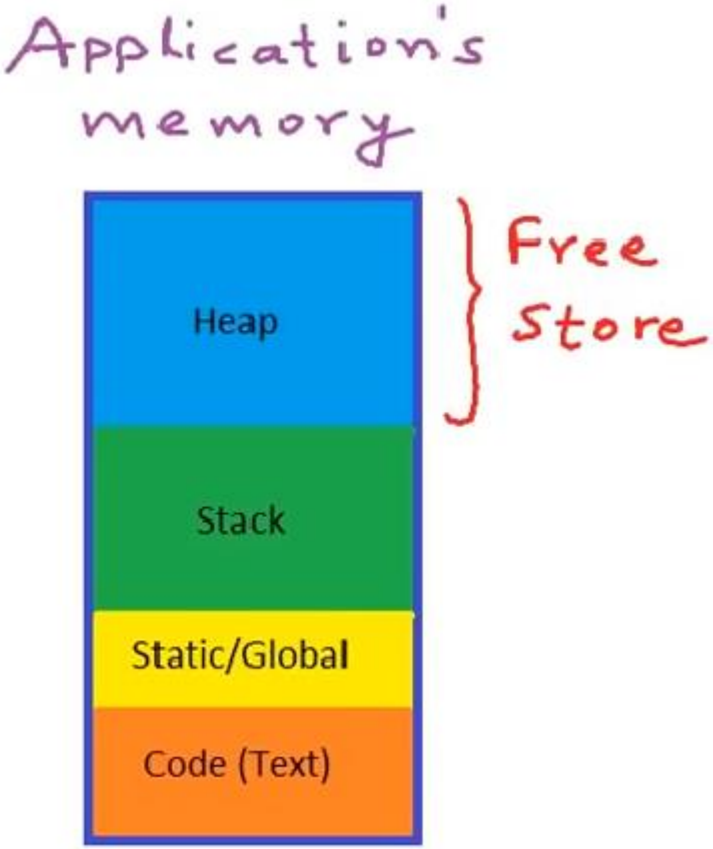
- Unlike stack, application’s heap is not fixed.
- heap size can vary during the lifetime of the application.

Application's memory



Notes

- Heap can grow as long as you don’t run out of memory on the computer itself, sometimes its is called free pool of memory.
- This is a dangerous thing, and that’s why the Garbage Collector is important.



Any Questions???...