28/10/2023

Object Oriented Programming
Class 01

📂 01: Local and global variable

Global variable:
1. *Written outside of the function*
2. *Accessible to all function (Same Copy)*

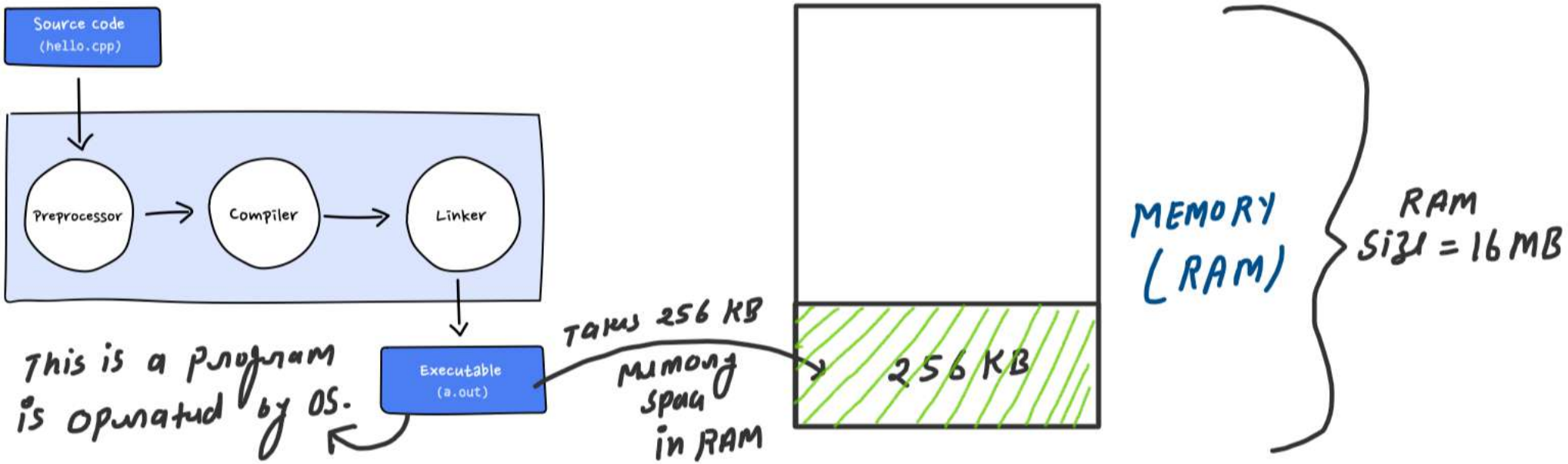
Local variable:
1. *Written inside function*
2. *Accessible inside that function scope only.*
3. *Scoped*

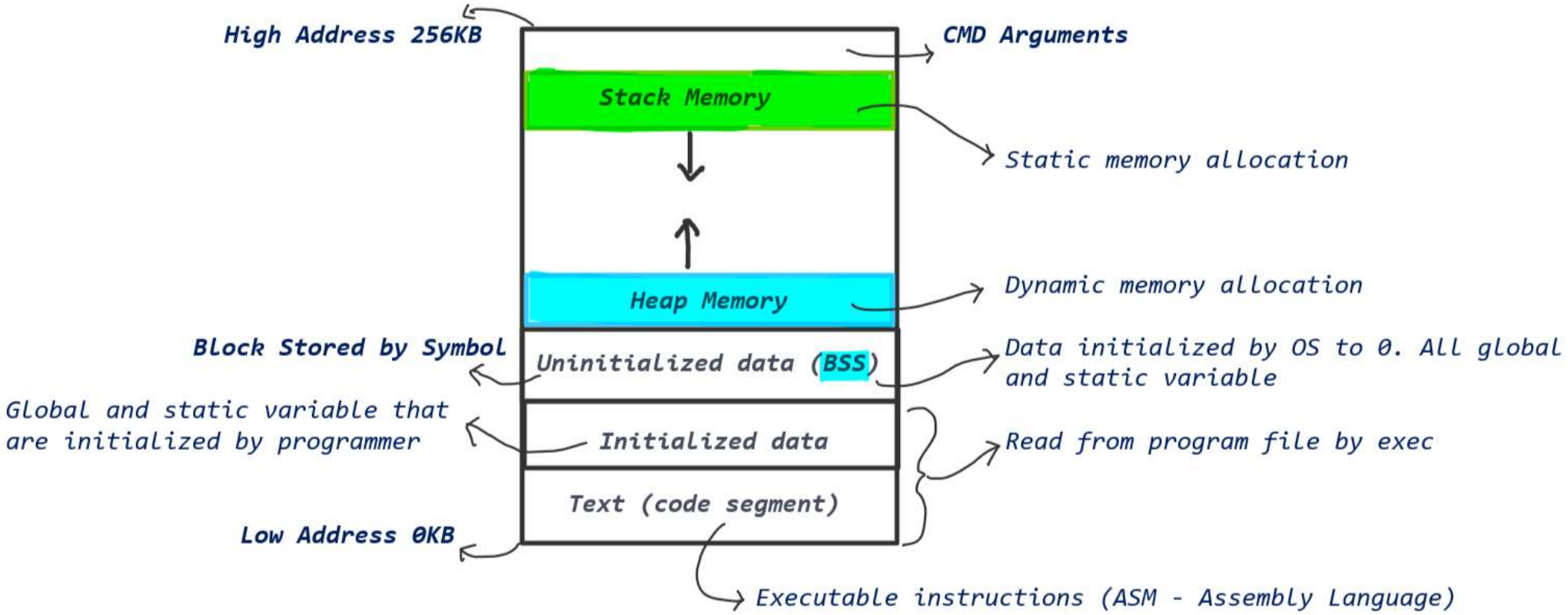
```cpp
#include<iostream>
using namespace std;

int x = 2; // Global variable

void fun(){
    int x = 60;
    cout<< x << endl; // Access local variable to fun()
    ::x = 40; // Udated global variable x by 40
    cout<< ::x << endl; // Access global variable by ::var_name in fun()
}

int main(){

    ::x = 4; // Udated global variable x by 4

    int x = 20; // Local variable to main() function

    cout<< x << endl; // Access local variable to main()
    cout<< ::x << endl; // Access global variable by ::var_name in main()

    {
        int x = 5;
        cout<< x << endl; // Access local variable to scope {}
        cout<< ::x << endl; // Access global variable by ::var_name in scoped{}
    }

    fun();
    return 0;
}
```

Output

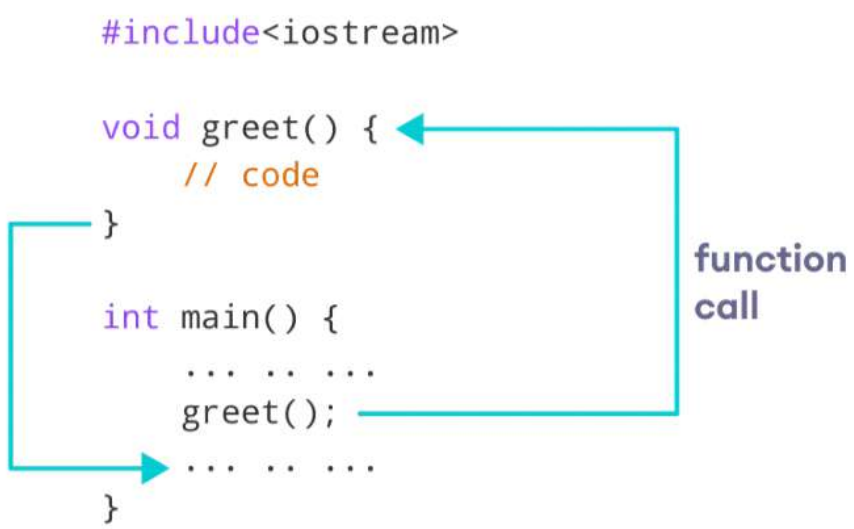
```
20
4
5
4
60
40
```

02: Memory Layout of a program




High Address 256KB
CMD Arguments
Stack Memory
Static memory allocation
Heap Memory
Dynamic memory allocation
Block Stored by Symbol
Uninitialized data (BSS)
Data initialized by OS to 0. All global and static variable
Global and static variable that are initialized by programmer
Initialized data
Read from program file by exec
Text (code segment)
Low Address 0KB
Executable instructions (ASM - Assembly Language)

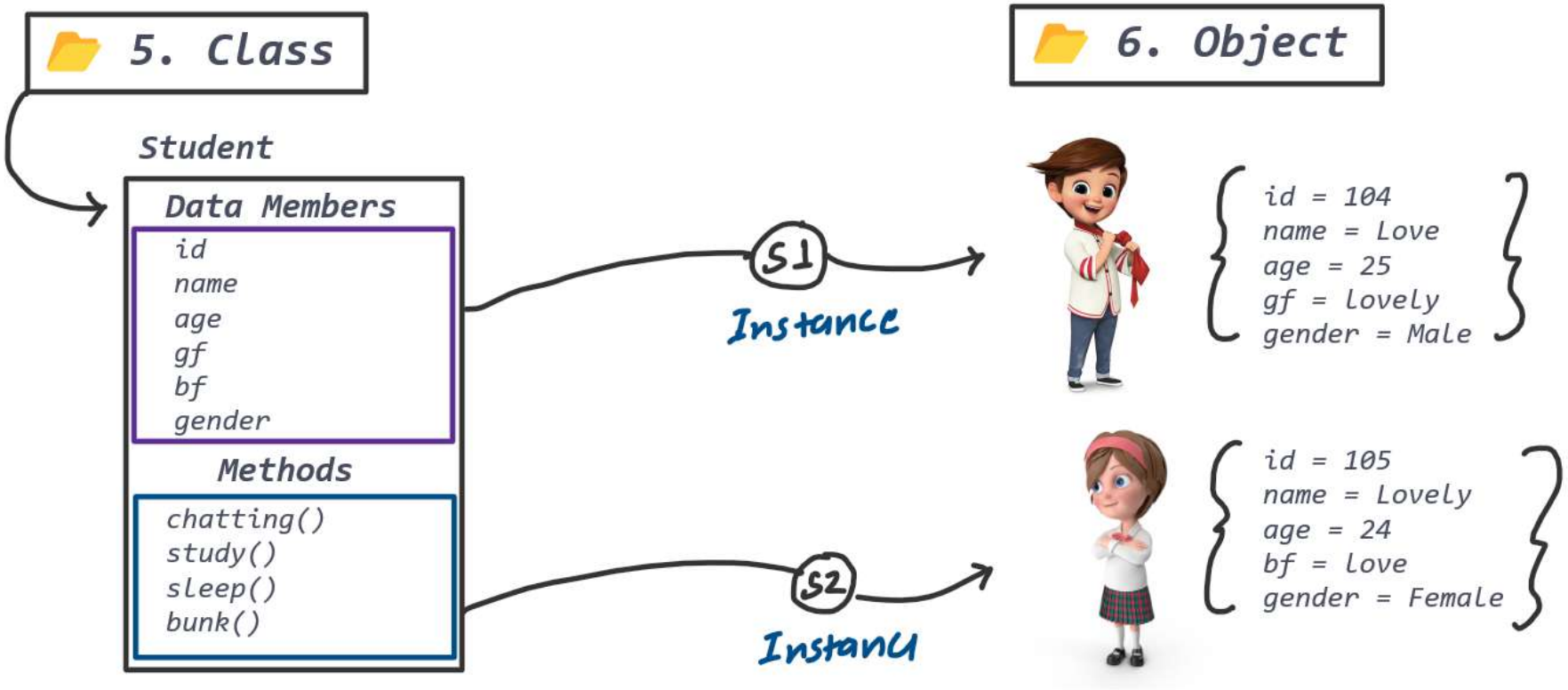
📂 03: Functional programming



jis Code me class AND Object Ki Koi Bat Nahi Ki ja Rahi Hai To iss Code Ko Hum functional Programming Bol sakte Hai.

📂 04: Object Oriented Programming

Object-oriented programming (OOP) *is a computer programming model that organizes software design around data, or objects, rather than functions and logic.*



What is a class:

1. Bundle of properties/states/attributes and behaviors.
2. It is a blueprint and also a user defined data type.

What is an object:

1. This is an instance of a class.
2. An object can be defined as a data field that has unique attributes and behavior.

7. Access modifiers

What is access modifiers:

1. In class, states and behaviors are private by default.
2. They are used to define scope of access.
3. Access modifiers are private, public and protected.

```cpp
#include<iostream>
using namespace std;

class Student
{
private:
    string gf;
    string bf;

    void chatting(){
        cout<< "Chatting" << endl;
    }

public:
    int id;
    string name;
    int age;
    string gender;

    void study(){
        cout<< "Studying" << endl;
    }

    void sleep(){
        cout<< "Sleeping" << endl;
    }

    void bunck(){
        cout<< "Buncking" << endl;
    }
};
```

Privately Info.

Public Info.

```cpp
int main(){
    Student s1;
    s1.id = 104;
    s1.name = "Love";
    s1.age = 25;
    s1.gender = "Male";

    Student s2;
    s2.id = 105;
    s2.name = "Lovely";
    s2.age = 24;
    s2.gender = "Female";

    return 0;
}
```

Public info Accessid in main()

Issi likhne ka koi Short Type hai?

YES, Construction

📂 8. Constructor

What is constructor:

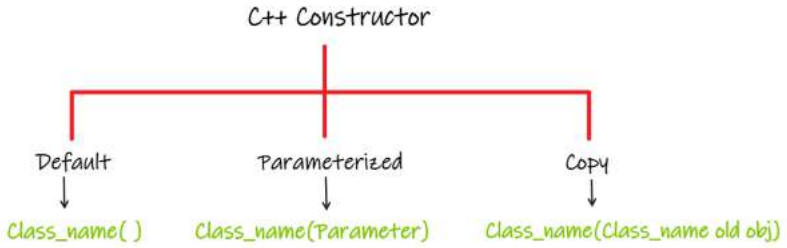
1. *A constructor is a special method (Member Function) that initializes a newly created object.*
2. *The constructor is automatically called when an object is created.*
3. *A constructor has the same name as the class name and has not return data type.*
4. *A constructor will be always public.*

Why use:

1. *It is used to assign the value to class data members.*
2. *It is used to assign default value to class data members.*
3. *It is used when we want to assign sensible value to data members.*

If we do not create any constructor then compiler add default

Two types of constructor: default, parameterized, and copy constructor



```cpp
#include<iostream>
using namespace std;

class Student
{
private:
    string gf;
    string bf;

    void chatting(string name){
        cout<< "Chatting " << name << endl;
    }

public:
    int id;
    string name;
    int age;
    string gender;

    // Default CTOR: assign garbage value
    Student(){
        cout<< "Default ctor called" <<endl;
    }

    // Parameterized CTOR: assign sensible value
    Student(int _id, string _name, int _age, string _gender, string _gf){
        id = _id;
        name = _name;
        age = _age;
        gender = _gender;
        gf = _gf;
        cout<< "Parameterized ctor called for " << name <<endl;
        chatting(name);
    }

    // Parameterized CTOR: assign sensible value
    Student(int _id, string _name, string _gender, string _bf){
        id = _id;
        name = _name;
        gender = _gender;
        bf = _bf;
        cout<< "Parameterized ctor called for " << name <<endl;
        chatting(name);
    }

    void study(){
        cout<< "Studying" << endl;
    }
};
```

Private

Public

Love

Lovly

```cpp
int main(){

    Student s1(104,"Love",25,"Male","Lovely");
    cout<< s1.name << endl;

    Student s2(105,"Lovely","Female","Love");
    cout<< s2.name << endl;

    return 0;
}
```

Output

```
Parameterized ctor called for Love
Chatting Love
Love
Parameterized ctor called for Lovely
Chatting Lovely
Lovely
```

9. Polymorphism (Brief)

Many more form for one things Jaisa EK ladka

GF → BF

SIS → BRO

wife → Hasband ... And so on ....

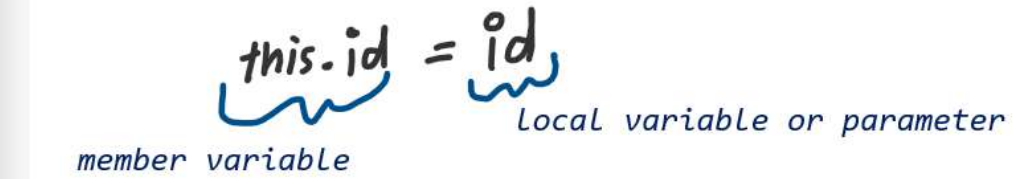
📂 10. this pointer

What is this pointer:
1. *this pointer is a private data member in class*
2. *this pointer points to current object*
3. *this pointer added by compiler privately*

Why use of this pointer:
whenever a local variable or parameter has the same name as a member variable,
the 'this' pointer can be used to explicitly refer to the member variable.

```cpp
#include<iostream>
using namespace std;

class Student
{
private:
    string gf;
    // Student *this;

public:
    int id;
    string name;
    int age;
    string gender;

    // Parameterized CTOR: assign sensible value
    Student(int id, string name, int age, string gender, string gf){
        this->id = id;
        this->name = name;
        this->age = age;
        this->gender = gender;
        this->gf = gf;
        cout<< "Parameterized ctor called for " << name <<endl;
    }
};
```

```cpp

int main(){

    Student s1(104,"Love",25,"Male","Lovely");
    cout<< s1.name << endl;

    return 0;
}
```

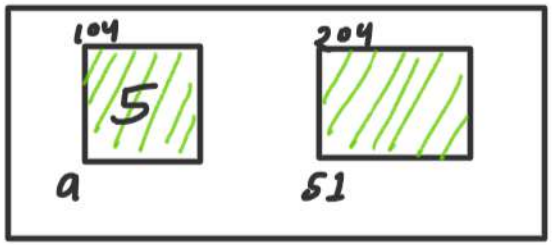


11. Static and dynamic memory allocation

Static Memory Allocation

```
int a = 5;
Student s1;
```

a1 and s1 Kaha allocate hoge?

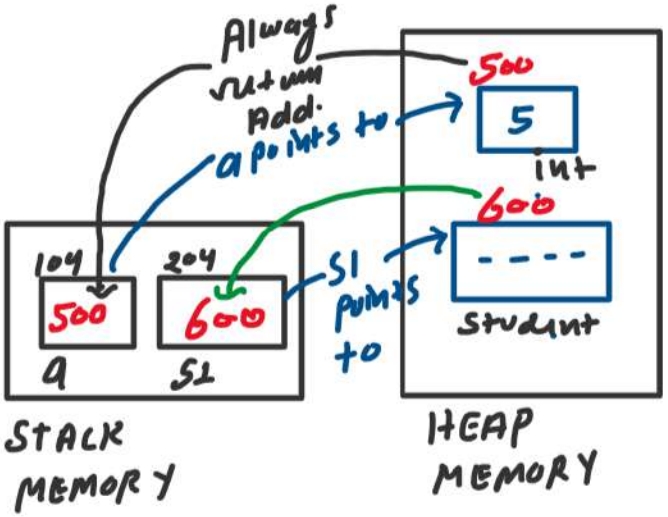


Dynamic Memory Allocation

```
int *a = new int(5);
Student *s1 = new student(...);
```

stack — Heap

a1 and s1 Kaha allocate hoge?



Note

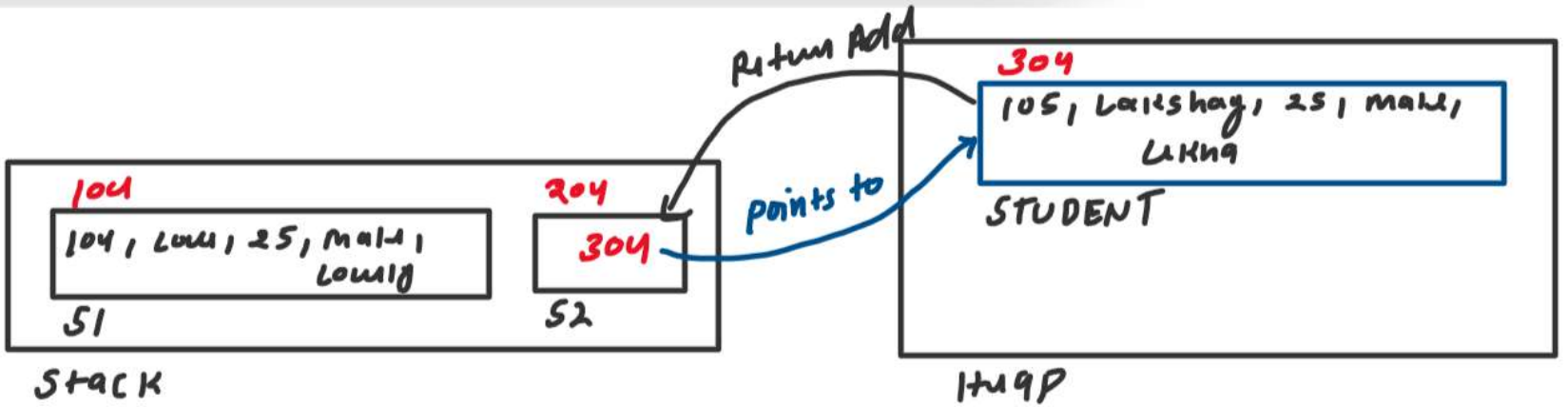
delete a;
delete s1;

```cpp
// 📂 Stack and Heap memory allocation
int main(){

    Student s1(104,"Love",25,"Male","Lovely"); // Stack Memory Allocation
    cout<< s1.name << endl; // Access Member of Class using "." operator in case of Stack

    Student* s2 = new Student(105,"Lakshay",25,"Male","Lekha"); // Heap Memory Allocation
    cout<< (*s2).name <<endl; // Access Member of Class using "(*s2).member" dereference operator in case of Heap
    cout<< s2->name <<endl; // Access Member of Class using "s2->member" this pointer in case of Heap

    return 0;
}
```

Note we can't write syntax to access

*s2-name ✗

(*s2)-name ✓



📂 12. Padding Concept

1. What is the empty class size?
Ans: Size 1 byte (This is smallest addressable size)

Lekin 1 byte kyun hota hai?
Ans: Compiler assigns 1 byte size for an empty class because of isse class ki existence pta chalti hai ki ek empty class bnayi gy hai.
lets assume empty class ka size agar 0 hota to iska mtlb simple hai ki koi bhi class bnayi hi nhi gayi hai.

```cpp
#include<iostream>
using namespace std;

class Student
{
  // Empty Class
};

int main(){

    cout<< sizeof(Student) << endl;
    return 0;
}
```

Output

1

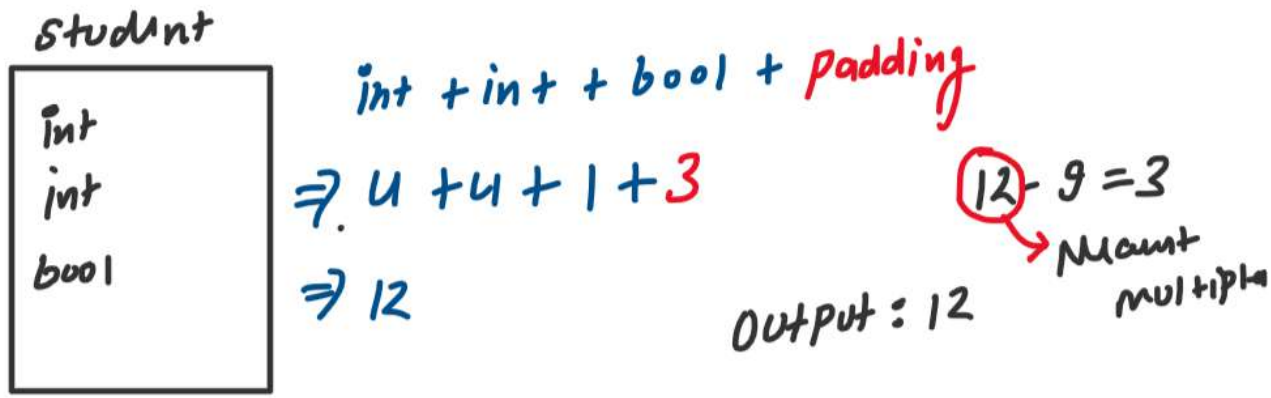
2. What is the non empty class size?

*We can find non empty class size using **padding concept** follows some rules to find the size of non empty class*

***Step 01:** Select biggest data type size*
***Step 02:** Align sum of data types in the boundary of biggest data type size*

Note: When we find the size of non empty class, then sabse pahle bade data type size ke nearest multiple memory par le aao

```cpp
#include<iostream>
using namespace std;

// Non Empty Class
class Student
{
    int a;
    int b;
    bool x;
};

int main(){

    cout<< sizeof(Student) << endl;
    return 0;
}
```



```cpp
#include<iostream>
using namespace std;

// Non Empty Class
class Student
{
    double a;
    double b;
    char x;
};

int main(){

    cout<< sizeof(Student) << endl;
    return 0;
}
```

obj

⇒ 8 + 8 + 1 + padding

⇒ 17 + 7

⇒ 24

Output = 24

24 - 17 = 7

Nearest multiple of 17

```cpp
#include<iostream>
using namespace std;

// Non Empty Class
class Student
{
    double a;
    double b;
    char x;
    bool y;
};

int main(){

    cout<< sizeof(Student) << endl;
    return 0;
}
```

→ 8 + 8 + 1 + 1 + padding

double
double
char
bool

⇒ 18 + (24 − 18)

⇒ 18 + 6

⇒ 24

Output = 24

```cpp
#include<iostream>
using namespace std;

// Non Empty Class
class Student
{
    int a;
    int b;
    bool x;
    string str;
    int c;
};

int main(){

    cout<< sizeof(Student) << endl;
    return 0;
}
```

→ Big

string ka size = 24

int
int
bool
string
int

⇒ 4 + 4 + 1 + 24 + 4 + padding

⇒ 37 + (48 − 37)

⇒ 37 + 11

⇒ 48

Output = 48