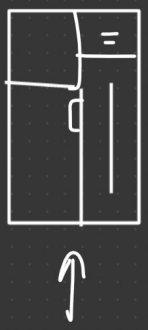
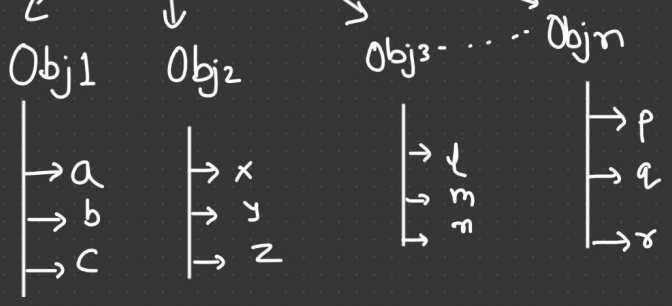
Objects and classes



9555031137

Real entity

Blue print of Object



Class

Data member → set of attributes (height, weight, length, breadth, colour...)

+

member function → set of operation (addition(), running(), eating().....)

```
class Student
{

   Char name[20];
   int age;
   int rollno;

};
```

By default Access Modifier is private

```
struct Student
{
   char name[20];
   int age ;
   int rollno;
};
```

Structure of C.

User defined Data Type

```
struct Student s;
```

s.age
s.rollno

```
int x;
```

```
class Addition
{
 public :
   int x;
   int y;
   int add()
   {
     return x+y;
   }
};
```

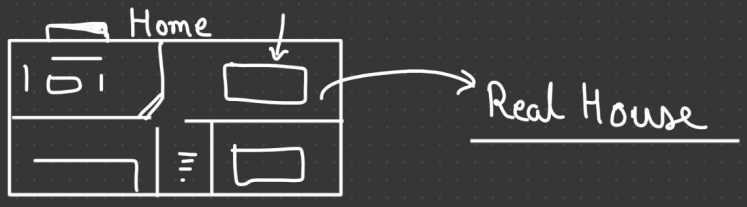
Data member + Member function ⇓ Class

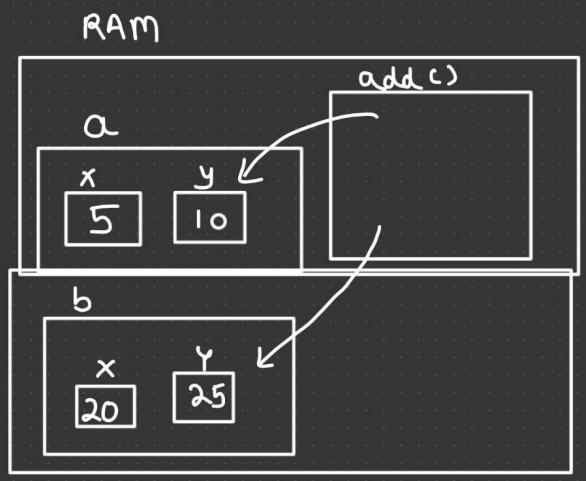
C Lang:-

```
int add(int x, int y)
 {
    return x+y;
 }
int main()
{
  int z = add(2, 3);
  printf("%d", z);
}
```

& No memory has been allocated at the time of Class Creation.

```
int main()
{
  Addition a, b;
  a.x = 5;
  a.y = 10;
  b.x = 20;
  b.y = 25;
  int z = a.add();
  int p = b.add();
  cout<<"Addition is"<<z<<endl;
  Cout<<"Addition is"<<p;
  return 0;
}
```

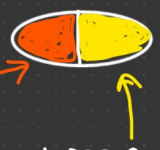




printf("Addition is %d", z);

OOPs features :-

- ✓ ✱ Abstraction ⟶ Data hiding
- ✓ ✱ Encapsulation ⟶ class = Data member + member function



- ✓ ✱ Polymorphism ⟶ Poly = many (EK naam, Anek Kaam)

<< , >>

- ✓ ✱ Inheritance
  - ✱ Reusability of code ⟶ with the help of inheritance.

```
Class  Addition
{
  public :
        int x,y;
        int add();
        int sub();
};
```

Class = 10 x 50 = 500 line.

= 20 x 100 = 2000 line

↓

20 - 30 line

```
int  Addition :: add()
{
  return x+y;
}

int Addition::Sub()
{
  return x-y;
}
```

```
Class A
{
    int add();

};
int A::add()
{
}
```

Constructors

```
int x;      → declare (Garbage value)
x = 5;      → Assign
```

```
int x = 5;  → Initialization
```

```
Addition a;  → declare
a.x = 5;
a.y = 10;    } → Assign
```

* we can initialize our object using Constructors in C++.
* It a special function whose name is same as the name of our class.
* It does not have any return type.

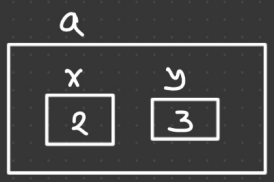
```
Class Addition
{
 public :
        int x, y;

        Addition (int a, int b)
        {
            x = a;
            y = b;
        }

    int add()
    {
      return x+y;
    }

};
```

```
int main()
{
   Addition a = Addition(2,3);
   int z = a.add;
   Cout<<z;
   return 0;
}
```



- Access Modifier of the constructor must be public.
- If we want that no one can create object of my class then I need to make my constructor private.
- we can make more than one constructors in our class.
- If we do not make any constructor in our class then Compiler itself creates a default constructor for our class.
- If you create any constructor then compiler will not create any default constructor. The responsibility of creating default constructor is passing to you now.

we can call the constructor implicitly & explicitly:-

```
Addition a = Addition(2,3);  ⟶ Explicit call
Addition a(2,3);             ⟶ Implicit call.
```