Day 40 11/04/2025

OOPS..

Object oriented programming System (or) structure

class & Object & method

public class calc { → class (is like a blueprint)

public int Add (int a, int b) {

return a+b;

}

}

called by object

method.

→ objectname. → (building something based on the blueprint)

c = new calc(); → creating a object

call

int sum = c.Add(1,3)

call the method

isDigit( ) -> Check it the given value is digit or not.

continue; -> it continue to next loop.

return; -> it will stop the whole code.

break; -> it will stop the single loop

method -> (Access modifier)

private <return type> <methodname> (parameter)

↳ within a class

eg: (int a, int b)

↓

datatype

{

// statement

// return

}.

public -> outside the class.

if object is not create => the instance memory is Not alloted

```
class Name {
public String a = "Hi";
}
```

It will calculate all the variable inside the class to allocated memory

name | memory → For each object created

* no memory is allocated for a because no object is created.

```
Name n = new Name();
```

* now memory is allocated for a

we can create static memory to get value from the class without creating object.

→ if we change static value it will change for every thing

```
public static int y = 0;
```

Constructor (default constructor)

every class have constructor so if we need to modify a the default Constructor we need use same name as the class.

```
public class car
{ public String color ; -> field
  default constructor
  public car () {
      color = black;
  } }
```

if we did not give any value after creating the object of car it will take color as black which is in default constructor

Parameterised constructor;

```
Public class myclass
{
    Public int x;        -> Fields
    Public String name;

Public My class (int x, String name)
    {
        this.x = x;  -> name from object
        this.name = name;
    }
}
```

myclass obj = new myclass (10, "Hi")
argument

can create an object of myclass and initialize its fields by passing arguments to the constructor.

Readonly :- Once the value is set cannot modifies it

```
Public readonly int x;
```

C# -> have automatic garbage collection

if memory is going to be more than allocated memory it will dispose the least used object

Property

public int x { get; set; }

↓

property