Class - objects

- Class = Data + Functions
- objects → Instance of a class. ('Instantiation' Process of creating an object)

  Ex: Rectangle R1;
- Data Hiding: Data is concealed within the class. (private, public)
- Data Hiding | Data Security. (Both are different in nature)
- 'Member Function', 'Method'
- Default (usually): Data — Private
  Functions — public
- Member function → can be accessed only by the object of that ~~cls~~ class.

Example Constructor

R1.Setdata(10, 20)
↳ Class Member Access operator.

CONSTRUCTORS :→

```
class integer
{
    int i, j;
    void getdata();
    void setdata();
    integer()
    {
    }
    integer(int j)
    {
        i = j;
    }
    void displaydata();
};
main()
{
    integer i1(100), i2, i3;
    i1.displaydata();
    i2.setdata(200);
    i2.displaydata();
    i3.getdata();
    i3.displaydata();
    return 0;
}
```

▷ Constructor : - Special member function.
- Setup values of the data members while defining the object.
- Executed automatically. (When the object is created.)
- Same name as the class.
- No return type.

▷ Overloaded constructors :

```
integer ()          integer (int j)
{                   {
                      i = j
}                   }
```

▷ - If no constructor → implicit constructor
- Once we declare an one argument constructor it is necessary to define the implicit constructor.

DESTRUCTOR :

- Called automatically when an object is destroyed.
- ~ (tilde) Name ()
- - class example
  - int data ; (private)
  - constructor ; destructor ; (public)
  - example ()
    {
    inside constructor.
    }
  - ~example ()
    {
    inside destructor ;
    }
- main ()
  {
  example e ;
  return 0 ;
  }

Do not have a return value.

- When the control goes outside main () object 'e' gets

Constructor :→

1. Default Constructor.
2. Parameterized Constructor.
3. Copy Constructor,

Example of the copy constructor.

```
#include <iostream>
using namespace std;
class A
{
  private: len, breadth;
  public:
      A(int l, int b)
      {
        len = l;
        breadth = b;
      }
      A(A &O)
      {
        len = O.len;
        breadth = O.breadth
      }
      double calculate_Area()
      {
        return len * breadth;
      }
};
```

```
int main()
{
    A a1(10, 20.5);
    a1.Calculate_Area();
    A a2(a1); or A a2 = a1;
    a2.Calculate_Area();
}
```

– Complex No. using the concept of classes:

Ex:

```
class complex
{
private:
    float real, img;
public:
    complex()
    {
    }
    complex(float r, float i)
    {
    real = r;
    img = i;
    }
    void display_data()
    {
    cout << "Real =" << real << endl;
    cout << "Imaginary =" << img << endl;
```

```
void add_complex(Complex c1, Complex c2)
    {
    real = c1.real + c2.real;
    img = c1.img + c2.img;
    }
Complex mul_complex(Complex c2)
    {
    Complex t;
    t.real = real * c2.real - img * c2.img;
    t.img = real * c2.img + c2.real * img;
    return t;
    }
};
main()
  {
  Complex c1(2.0, 2.0), c2(1.4, -2.5), c3, c4;
  c3.add_complex(c1, c2);
  c3.display_data();
  c4 = c3.c1.mul_complex();
  c4.display_data();
  return 0;
  }
```

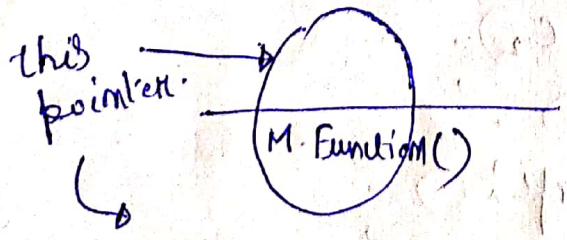
Operator Overloading:

```
Complex operator + (Complex C)
{
    complex t;
    t.real = real + t.real;
    t.img = img + t.img;
    return t;
}

main()
{
    complex C1(2.0, 3.9), C2(1.4, -2.5), C3, C4;
    C3 = C1 + C2;   ⟶ Internally C3 = C1.operator+(C2);
    C3.display_data();
}
```

↳ This is basically a concept of the **this** pointer.

this pointer :—



↳ Points to the object itself.

Example:

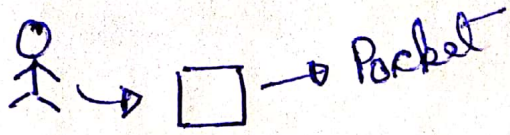
```
class A
{
  private:
    int i;
  public:
    void Set_data(int ii)
    {
      i = ii;
      cout << endl << "My objects Address = "
           << this;
      this -> i = ii;
    }
    void Show_data()
    {
      cout << i;
      cout << endl << "My objects Address = " << this
      cout << this -> i << endl;
    }
};
Main()
{
  A a;
  a.Set_data(10);
  a.Show_data();
  return 0;
}
```

Friend Function:

```
//using namespace std;
class Sum
  {
    int a, b;
  public:
    void get_number(void);
    friend int add(void);
  };

void Sum :: get_number(void)
  {
    cout << "Enter the values of a, b" << en
    cin >> a >> b;
  }

int add(void)
  {
    Sum S;
    int temp;
    S.get_number();
    temp = S.a + S.b;
    return temp;
  }
int main()
  {
    int result;
    result = add();
    cout << "Sum = " << result << endl;
    return 0;
  }
```



Example Friend Function:

```
class Box
{
    double width;
public:
    friend void printWidth(Box b);
    void SetWidth(double wid);
};

// Member function definition

void Box :: SetWidth(double wid)
{
    width = wid;
}

void print(Box b)
{
    cout << "Width of the box=" <<
                width << endl;
}

int main()
{
    Box b;
    b.SetWidth(10.0);
    printWidth(b);
}
```

this - pointer:

Example:

```
class Example
{
private:
    int i;
public:
    void Setdata(int ii)
    {
        i = ii;
        cout << endl << "My object's address is"
             << this;
        this -> i = ii;
    }

    Void Showdata()
    {
        cout << i;
        cout << endl << "My object's address
             is" << this << endl;
        cout << this -> i << endl;
    }
};

int main()
{
    Example e1;
    e1.Setdata(10);
    e1.Showdata(); return 0;
}
```