OOPS (CN)

What is OOPS?

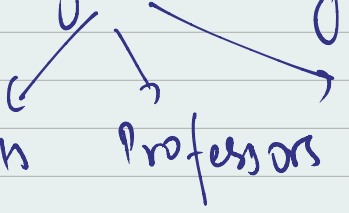
↳ a programming style which mainly revolves around objects.

What are objects?

↳ simple entity (real-world entity)

→ we want that we involve these real world entities as much as close to our code.

eg. College university



} visualization is easy !! while designing the software.

eg. Student → object

- ↳ name
- ↳ roll no.
- ↳ age
- ↳ address
- ↳ contact no.

} properties of the object.

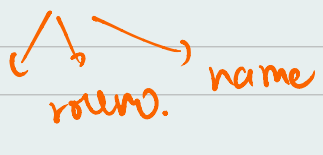
Function / Methods :-

↳ tasks which our object can perform.

eg. Student

- ↳ changeAddress
- ↳ setRollNumber

→ let say i want to create 20 students



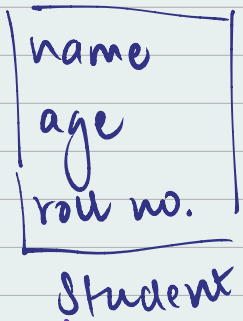
For every student, i have to explicitly tell that each student has these 3 properties

S1 → age, roll no., name

S2 → age, roll no., name

But i don't want this!!! (very lengthy and cumbersome)

→ I will just make a template (CLASS)



इसे सिर्फ इस template की copy बनानी है, हर बार अलग से properties define करने की जरूरत नहीं है।

(OBJECTS)

S1: name, age, roll no.

S2: name, age, roll no.

S3: name, age, roll no.

How can we create a class in code?

```
class Student {
    int rollNo;
    int age;
};
```

Student → name of the class

; → important

```
class Product {
    int weight;
    char name[100];
};
```

How do we create objects?

normal integer variable →

```
int a;
```

a → variable

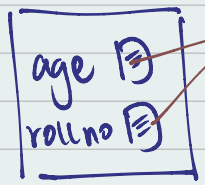
यूज़र डिफाइन variable बनाना है student type का

→ classes are also called USER DEFINED DATATYPES.

Student s1; ← object

↓

as soon as this line will be written, ⇒

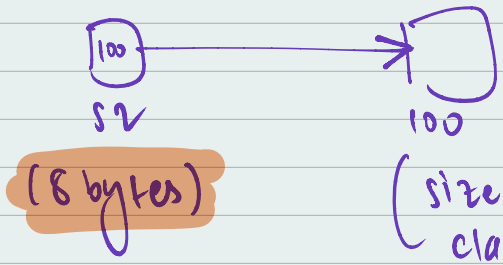


random garbage value

s1

→ Dynamic creation of object

Student *s2 = new Student;



{ normal जैसा variables को handle करते हैं वैसा ही है }

eg. #include<iostream>

class Student {

int age;
int rollno;

};

int main() {

// Create objects statically
Student s1;
Student s2;

// create objects dynamically
Student *s3 = new Student;

s1.age = 23; } → this is how you access the properties of your class

}

How to access the properties of dynamically created objects?

```
int *a = new int;
```


```
*a=5;
```
(इससे value allocate करते है)

```
Student *S6 = new Student;
```


```
S6.age = 27;
```
(is this enough?) NO!!

```
*S6.age = 27;
```
(पहले dereferं करेगा, उसके बाद access करेगा)

dereference

↓

could be written as,

```
S6->age = 27;
```

Access Modifiers

1. public → class के बाहर कहीं भी use कर सकते है।
2. private → properties and methods उस class के अंदर ही visible है बस।
3. protected

→ By default, class की properties और methods PRIVATE होती है।

Eg.

```
#include<iostream>

class Student{
  public:
    int age;
```

```
int rollno;
};
int main() {
    // Create objects statically
    Student S1;
    Student S2;

    // create objects dynamically
    Student *S3 = new Student;

    S1.age = 23;
}
```

Getters and Setters

eg.

```
#include<iostream>
class Student {
  public:
    int rollno;
  private:
    int age;
  public:
    void setAge(int _age) { age = _age; }
    void display() { print(age, rollno.); }
};

int main() {
    // Create objects statically
    Student S1;
    Student S2;

    // create objects dynamically
    Student *S3 = new Student;

    S1.age = 23;
```

this is called a setter!! (similarly a getter can be used)

i can access the private member within the class.

we will not be able to access the age because it's a private member

s1.setAge(23); → but you can access the private properties like this!!

}

Q: if we are able to access and change the value of private variables, then what is the use of private access modifier?

कुछ फायदा ही नहीं हुआ क्योंकि public func^n के माध्यम से हम access तो कर ही पा रहे हैं।

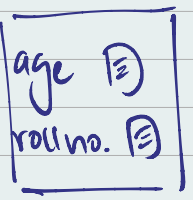
→ suppose i want कि age को कोई negative न set कर दे, so i can add a check in the setter func^n.

eg.
```
public:
    void setAge(int a){
        if(a<0) return;
        age=a;
    }
```
} i can add multiple constraints like this while making a variable private

Constructors

→ When we write,

Student s1;



what happens internally?

s1.Student();

this will be called!!

A func^n with exact same name as of our class will be called automatically.

Q: what is the role of this func^n?

initializing our data members with garbage values.

→ This special func^n called CONSTRUCTOR

↳ same name as class
↳ no return type

default constructor will have no input arguments as well.

→ default constructor will be called automatically when we create our object. (for. every object)

default constructor

```
Student(){

}
```

→ doesn't matter if you create your class statically OR dynamically.

eg.
```
class Student()
{
    int age;
    int rollNo;

    Student(){
        cout << "constructor called!!";
    }
}

Student S1;
Student *S2 = new Student;
```

O/p:-
constructor called!!
constructor called!!

→ As soon as you created your own constructor, your inbuilt (default constructor) constructor will be destroyed.

→ we can make constructors with parameters as well.

eg. ①.
```
Student(int r)
{
    rollNo = r;
}
```

→ parameterized constructors

②.
```
Student(int a, int r)
{
    age = a;
    rollNo = r;
}

Student S1(10);
Student S2(20,10);
```

→ कैसे interpret होगा?

S1.Student(10);

→ For one object, only 1 constructor will be called.

Dynamically:-

Student *S3 = new Student (10,20);

② constructor will be called!!

→ कौन सा constructor चलेगा, it depends upon which what params are you passing while creating the object.

this keyword

what will happen if we type

eg. class Student {
public:
int age, rollNo.;
Student (int rollNo)
{
rollNo = rollNo;
}
};

this line will assign this rollNo to this rollNo.

error नहीं आयेगा!!

Student S1(23);

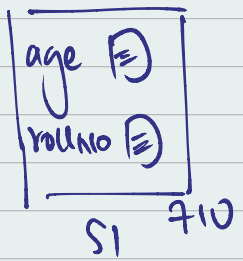
→ That's why we use this keyword.

it holds the address of current object.

eg. void display ()
{
cout << this;
}

Student S1;

this will print the address of the object



O/p: 710

So, we can write as,

eg.
```
class Student {
public:
    int age, rollNo.;
    Student (int rollNo)
    {
        this -> rollNo = rollNo;
    }
};
Student s1(23);
```

argument वाला rollNo.

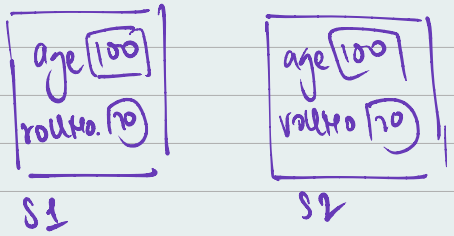
s1. का rollNo (वही object जिसने call किया है)

Copy constructor

→ Suppose i want to copy the contents of Student s1 to Student s2.

One of the approaches could be,

```
s2.age = s1.age
s2.rollNo = s1.rollNo
```



we don't need to do this, we can do

interpreted as

```
Student s2(s1);
s2.Student(s1);
```

but इसके लिए हमारा कोई constructor बनाया है नहीं जो object को as a parameter accept करे

→ We get another inbuilt constructor called **Default Copy Constructor**

Copy constructor with Dynamic Objects

eg. `Student *s3 = new Student (16, 20);`

s3 (710) → age (16), rollNo (20) at 710

16 → age, 20 → rollNo

s3 is an address and not the object

i want to use copy constructor,

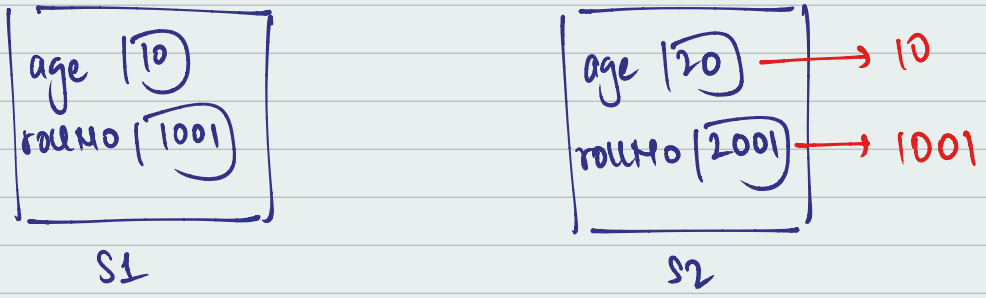
`Student su(s3);` → is this correct? NO!!

we have to write,

Student S4(*S3);

Copy assignment operator (=)

eg. Student S1(10, 1001);
Student S2(20, 2001);



→ Now, i want that S2 में S1 की values आ जाए।

→ we can do;

S2 = S1;

interpreted as

S2.age = S1.age;
S2.rollNo = S1.rollNo;

→ copy constructor is used at the time of object creation but copy assignment operator could be used at any time. (after construction of object)

eg. Student *S3 = new Student;

*S3 = S1; → for dynamically created objects.

Destructors

↳ same name as class
↳ no return type
↳ no input arguements

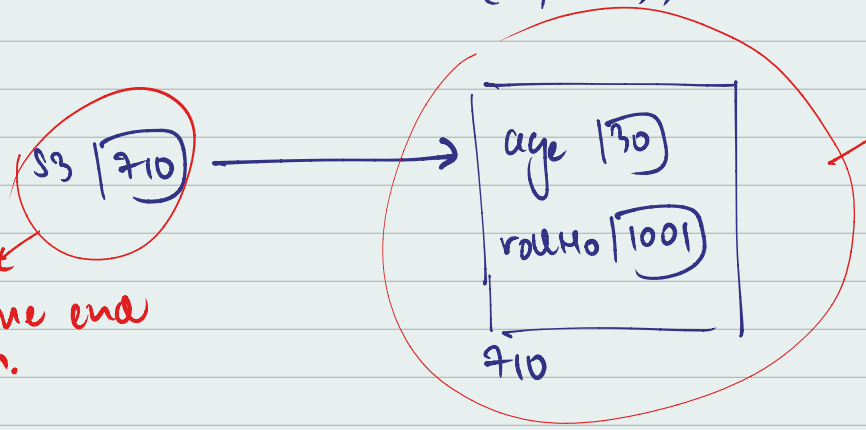
eg. ~Student() { } → will be called automatically at the time when the object is destroyed.

→ You can make only 1 destructor. (cannot make multiple destructor).

Note:-

for dynamically created objects, you have to manually delete the space that have been allocated to the object.

```
Student *S3 = new Student(30,1001);
```



This will be destroyed at the end of the program.

But this space will be in memory.

→ So, we have to manually delete it by using the keyword "delete".

```
delete S3;
```

if you do not delete, that could cause memory leaks.

Q. What will happen when we write,

```
Student S5 = S4;
```

This is same as,

```
Student S5;
S5 = S4;
```

} interpreted as → `Student S5(S4);`

for optimization, why not call copy constructor itself?

→ copy constructor will be called!!

PRACTICE

Q. Define a class Fraction with its members as numerator and denominator. It will have function to display the fraction, add the fraction, multiply and also the fraction should be in its simplest form.

Q: Also practice the complex class function as well.