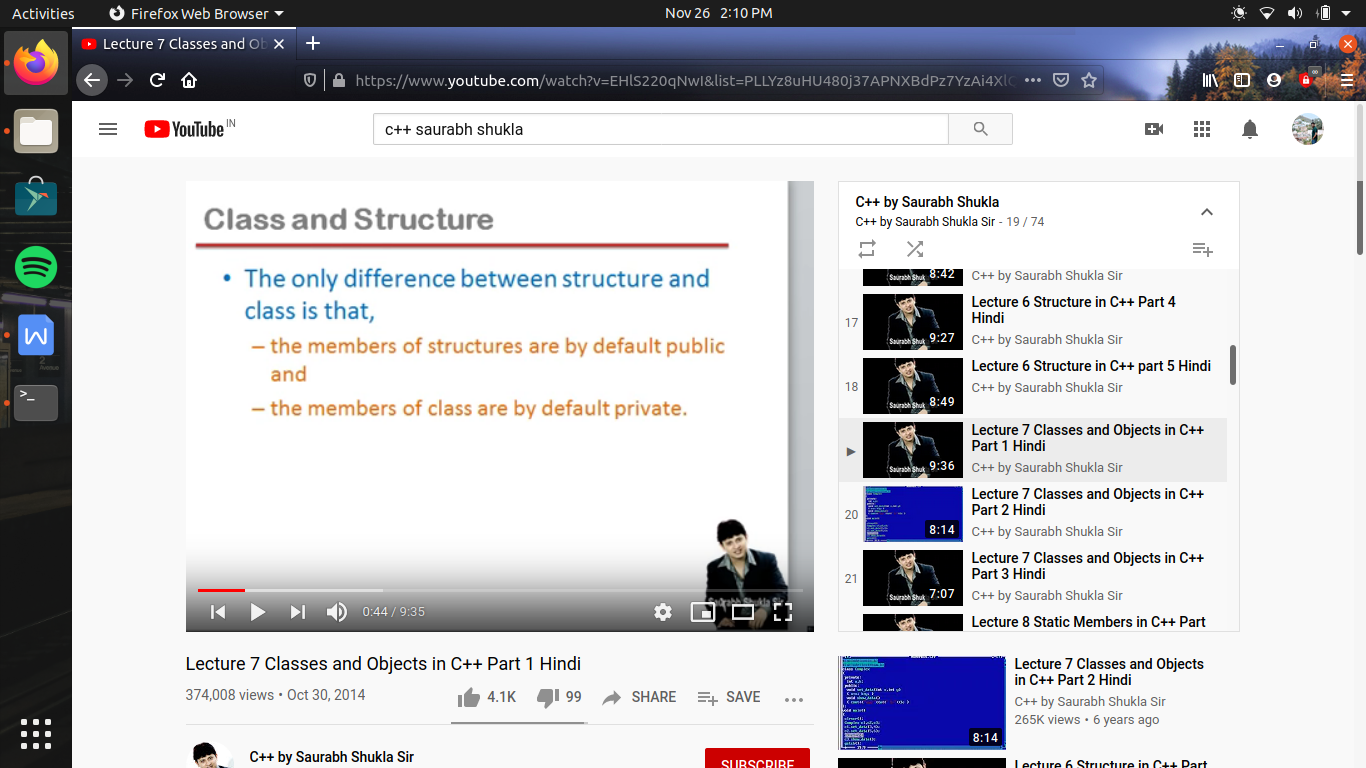
classes and objects

difference b/w structure and class

Structure ke member by default public mane jate hai balaki class ke andar member by default member private mane jate hai

this is the only difference b/w class and structure

baki sari bate or sare concept hum class ke leye padege wo structure ke leye bhi utane hi applicable hoge



/\* this program to understand the difference b/w structure and class \*/

// jaise hum janate hai complex no. ,i.e real part and imaginary part ka combination hota hai complex no.

//toh ek complex no. ko store karane ke leye hame 2 no. store karane padege , or es ka mtlb 2 variables chahiye

//toh hum aisi class banayege jo complex value ko represent karega jis ka ek object 2 value contain karega

//so pahale sturucture banayege fir class ka concept us se compare karate hue deekhege

#include<iostream>

using namespace std;

/\*

struct complex

{

private :

int a,b;

public:

void set\_data (int x,int y)

{a=x;b=y;}

void show\_data ()

{

cout<<"\n a ="<<a<<"\n b ="<<b<<endl;

}

};

int main()

{

complex c1; //here c1 called as a variable

c1.set\_data(3,4);

c1.show\_data();

}

\*/

class complex

{

private : //here we don't need to write private , by default private hi hoga , but for understanding hum likh dete hai

int a,b;

public:

void set\_data (int ,int ); //decleration of function which define outside the body of class

void show\_data ()

{

cout<<"\n a ="<<a<<"\n b ="<<b<<endl;

}

};

void complex:: set\_data (int x,int y) // here we use class membership label complex::

{a=x;b=y;}

int main()

{

complex c1; // ab yaha c1 variable na bol kr ese object bolege , so as like variable , object also consume memory

c1.set\_data(3,4);

c1.show\_data();

}

//if agar hum class ke bahar koe member define kiya hai toh use bhi us class ka member mana ja sakata hai toh us ki bhi kuch rules hai

//pahale funtion ko class body ke andar declare karana hoga

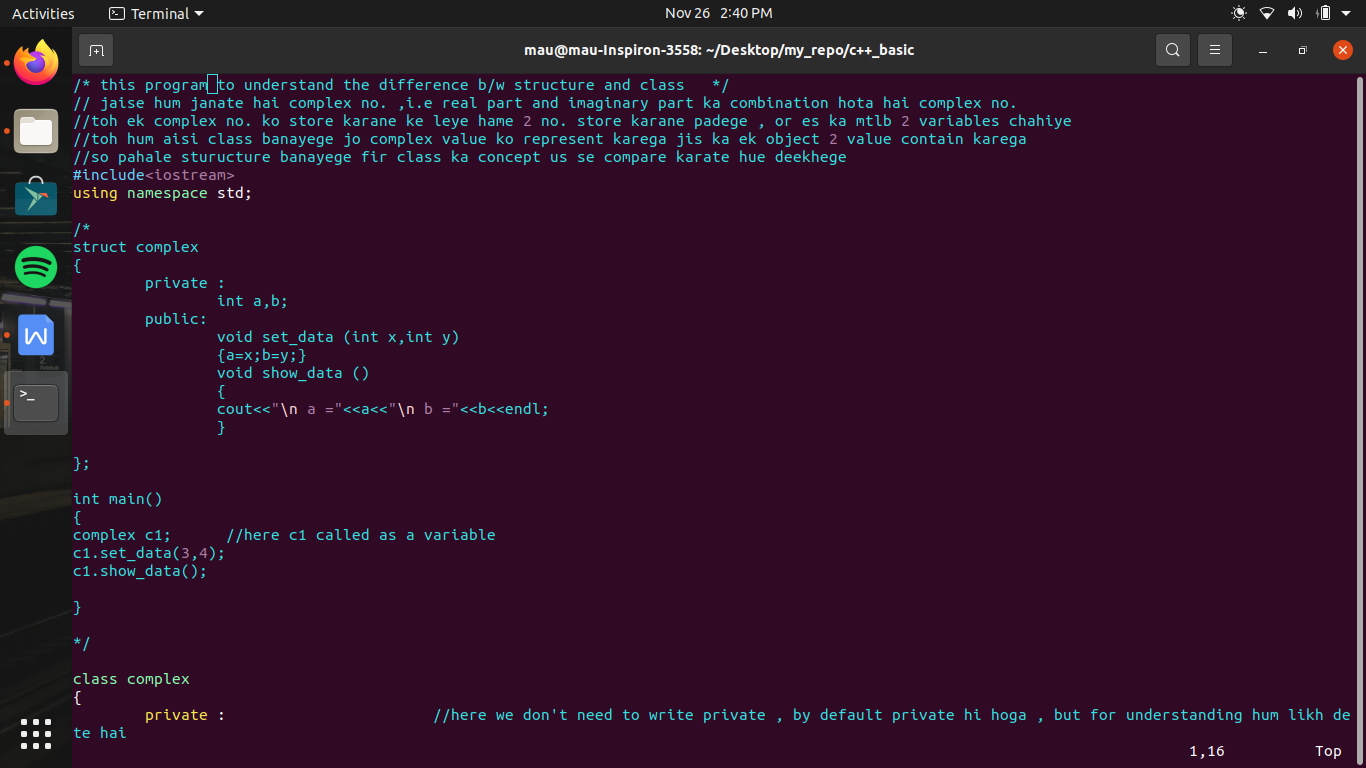
//or jab bahar define kar rahe hai toh function ke name or us ke data type ke beech membership label jarur lagaya jayega

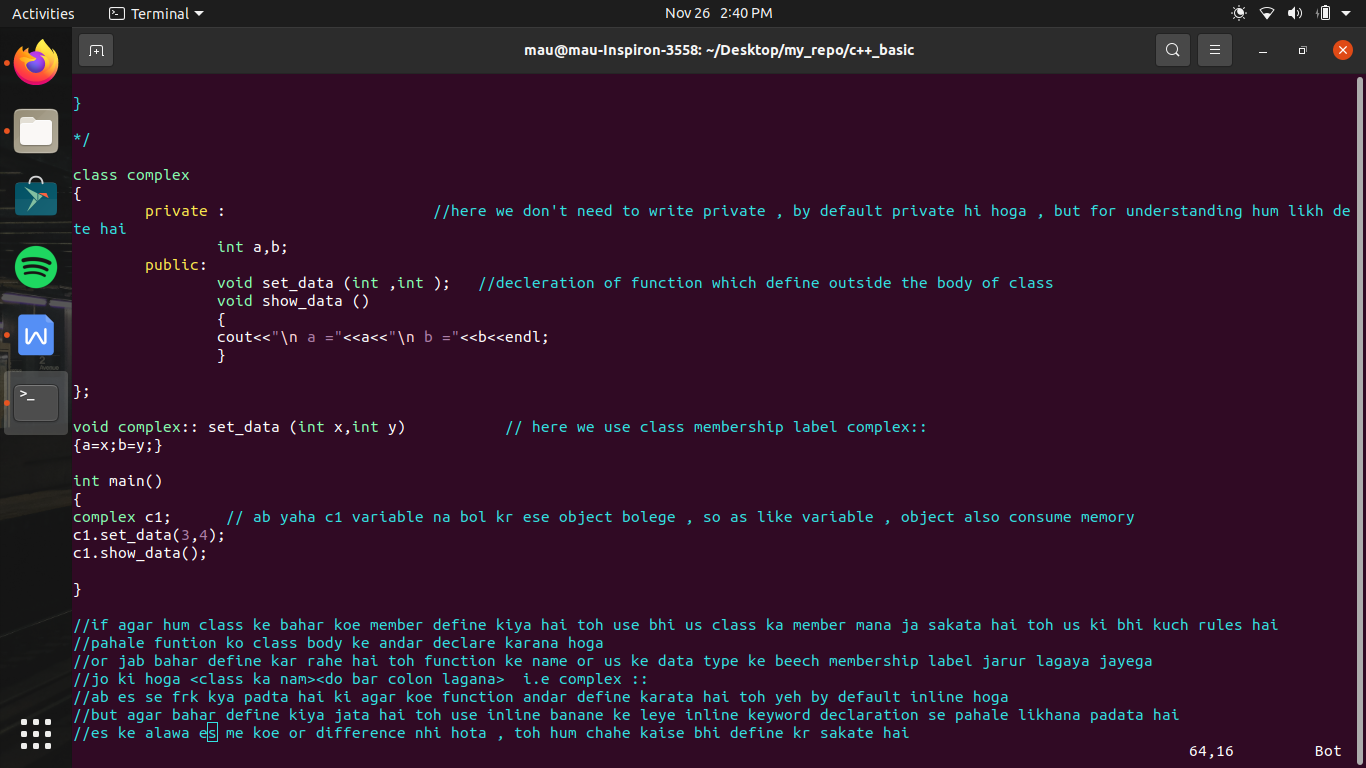
//jo ki hoga <class ka nam><do bar colon lagana> i.e complex ::

//ab es se frk kya padta hai ki agar koe function andar define karata hai toh yeh by default inline hoga

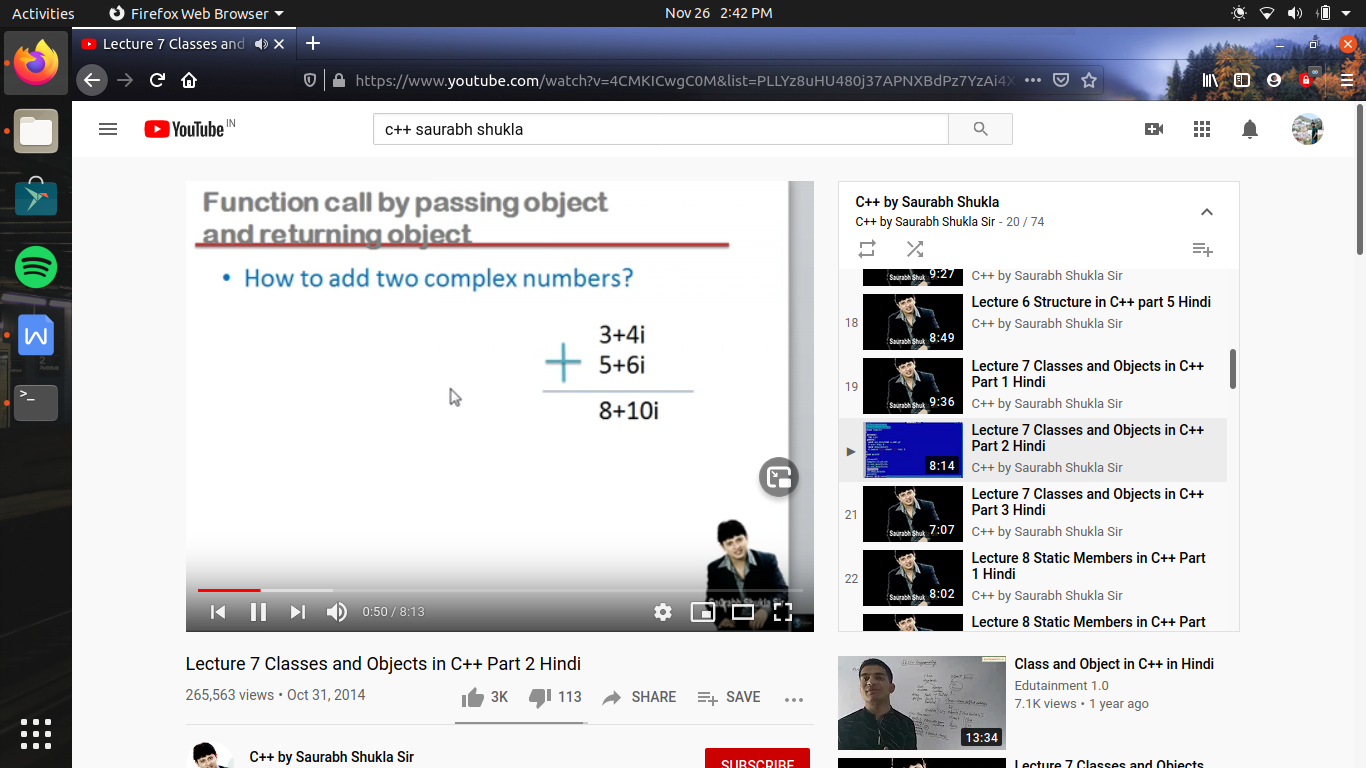
//but agar bahar define kiya jata hai toh use inline banane ke leye inline keyword declaration se pahale likhana padata hai

//es ke alawa es me koe or difference nhi hota , toh hum chahe kaise bhi define kr sakate hai





function call by passing object and returning object



/\* function call by passing object and returning object \*/

//here we are adding two complex numbers

#include<iostream>

using namespace std;

class complex

{

private :

int a,b;

public:

void set\_data (int ,int );

void show\_data ()

{

cout<<"\n a ="<<a<<"\n b ="<<b<<endl;

}

complex add(complex c2)

{

complex added\_value;

added\_value.a=a+c2.a; //as here c1 call add function so c1 function's variable access directly and as we pass c2 so by dot operator we access c2 data

added\_value.b=b+c2.b;

return (added\_value);

}

};

void complex:: set\_data (int x,int y)

{a=x;b=y;}

int main()

{

complex c1,c2,c3;

c1.set\_data(3,4);

c2.set\_data(5,6);

// c3=c1+c2; //agar hum es tarah se 2 complex no. ko add karane ki kosis kr rahe hai toh yeh galat hai

//ab yeh galat kyu hai ese samajahate hai

//c1 and c2 dono ka type hai complex , + operator operation perform karega addition ka lekin

//compiler ko har operator ka meaning already pata hota hai i.e har operator ki coding operator ko malum hai

//operants ko dhek ke compiler decide karata hai ki konsi wali conding es me bind honi chahiye

//ab jab compiler deekhega ki operator ke dono operants int type ke hai toh wo + ki kon si coding chalayega use pata hai , or agar float ke case me hai toh bhi use pata hai ki kon si conding chalani hai

//lekin operator ke dono operants primitive type k nhi hote i.e int , char , float , double

//but agar non primitive type ke hote hai jaise ki jab bhi hum koe naya data type banate hai

//toh jab bhi compiler deekhega non primitive opearants toh compileer un ko operate karane wali definition nhi janata hai ese leye wo eror dega

//yani c1 or c2 ke beech + likhana error dega kyu ki compiler ese resolve nhi kr pa raha hai

//yani compiler ko nhi ata ki c1 or c2 ko kaise add karana hai

//but age hum ek chapter padege operator overloading us me padege class ke andar kaise es ki definition likhi jaye taki yeh line workout ho

//toh fil hal hum es ka ek dusara tarika use karate hai hum es ka ek function banate hai

//or es function ko class me member function ke roop me banana padega ab aisa kyu

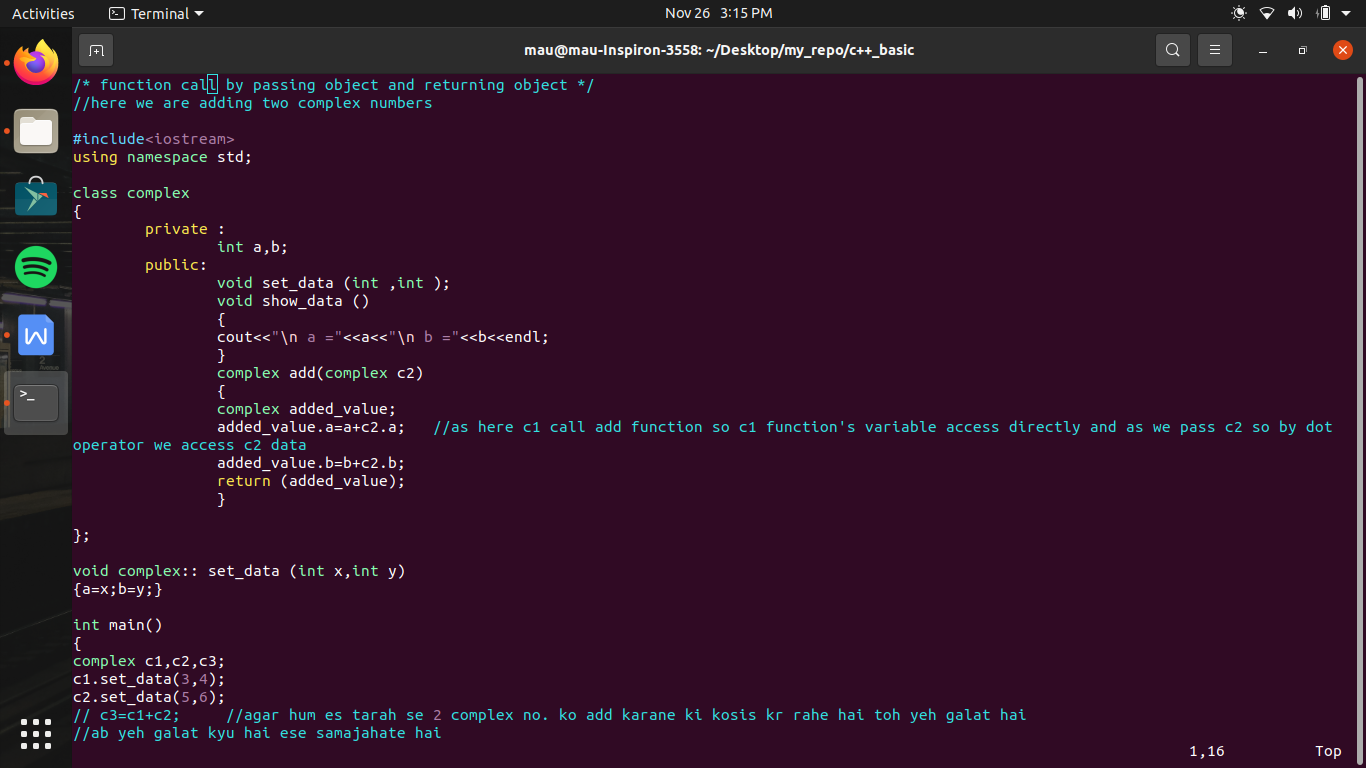
//kyu ki agar hum member funtion ki strength ho soche toh member function ko yeh adhikar hota hai ki wo class ki kisi bhi chij ko use kar sakate hai

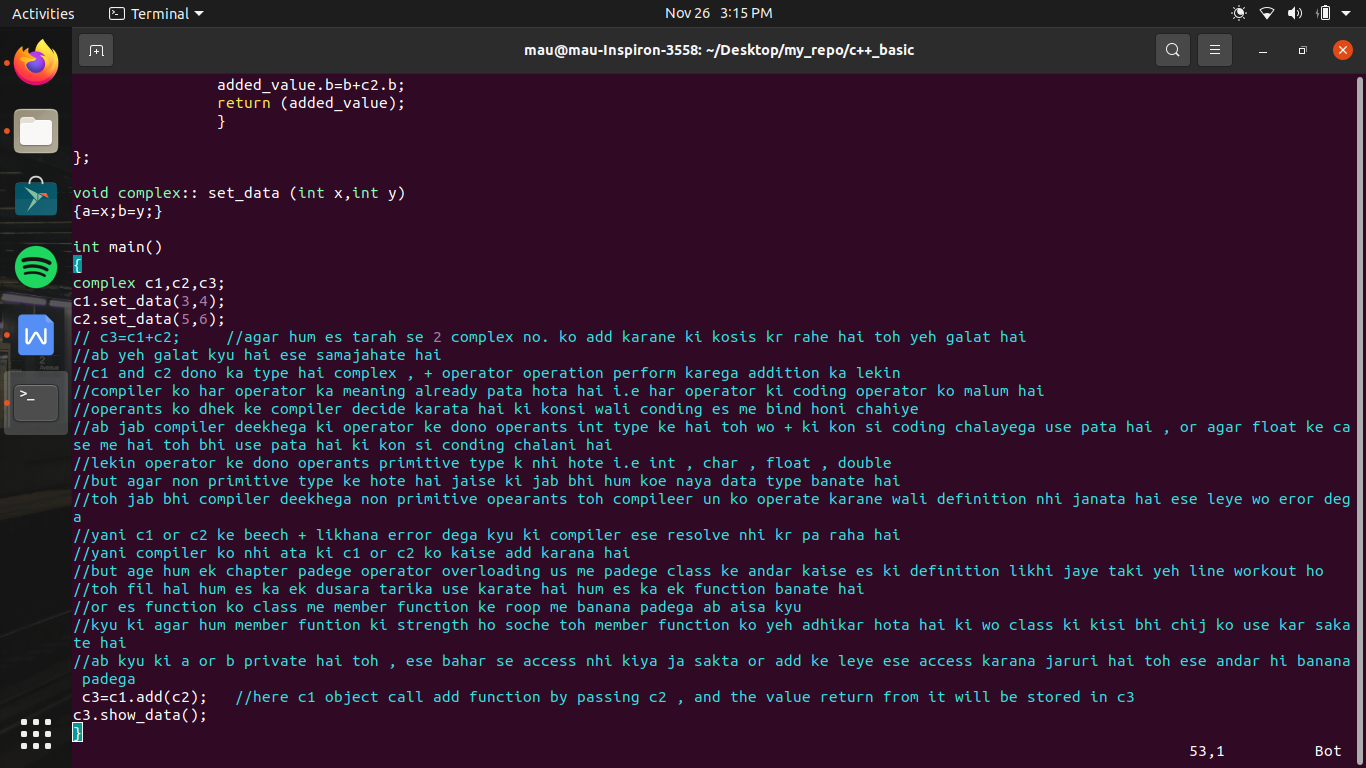
//ab kyu ki a or b private hai toh , ese bahar se access nhi kiya ja sakta or add ke leye ese access karana jaruri hai toh ese andar hi banana padega

c3=c1.add(c2); //here c1 object call add function by passing c2 , and the value return from it will be stored in c3

c3.show\_data();

}





technical jargons:-

instance -> means example

class ke andar jo variable banate hai use bolatehai instance member variable

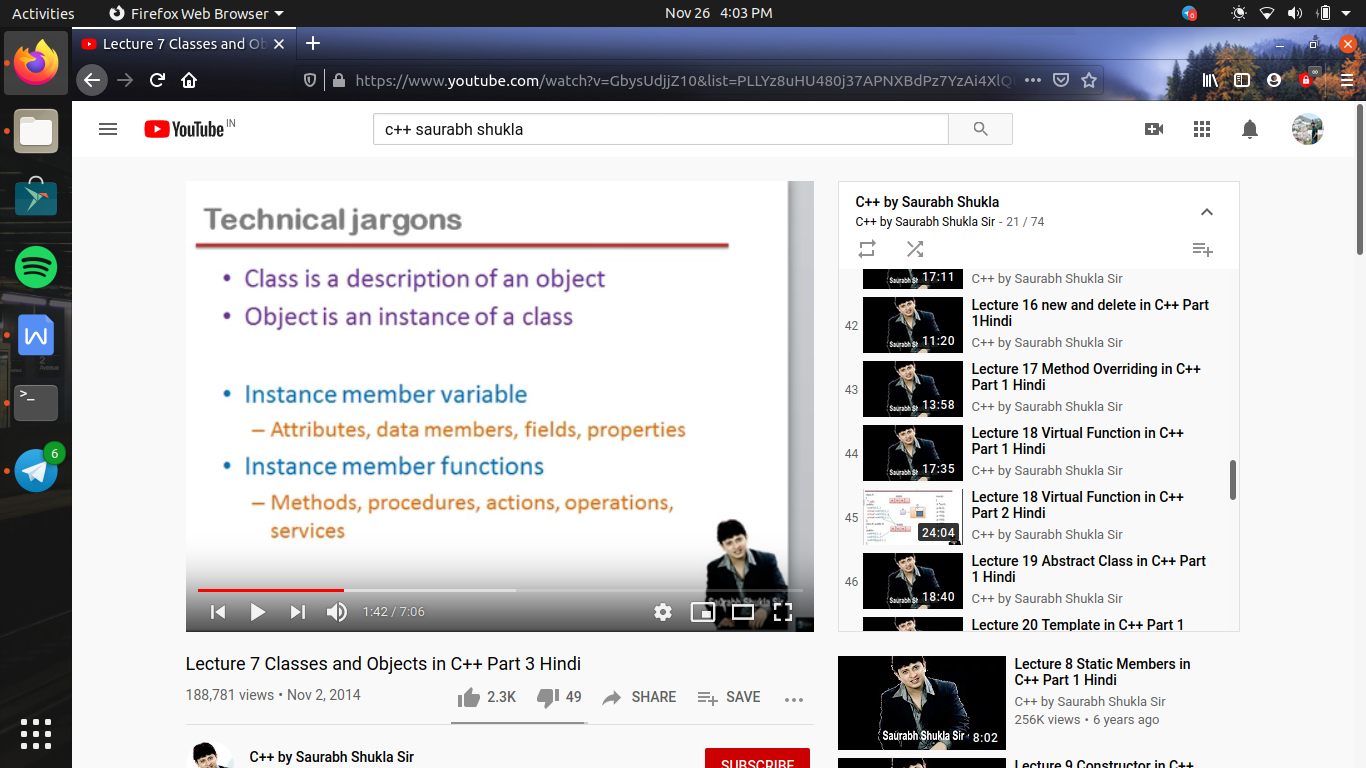
or class ek andar jo function banate hai use kahate hai instance member function

because hi tabhi hai jab object banaye jate hai or yeh bina object ke access bhi nhi ho sakate

kisi bhi objectv ki state ka mtlb us member ki value ka collection , or ab agar member ke variable ki ek bhi value me change kr du toh hum kahege ki object ki state change ho gayi hai

ab OOPs yeh kahata hai ki object ki state tabhi change honi chahiye jab hum object ke kisi function ko call kare use se pahale change nhi honi chahiye i.e why we made variable as private and function as public

object ke jitane bhi member function hai us ke collection ko behaviour ho object kaha jata hai yani object kay kya kr skata hai , kaise kaise perform kr sakata hai function



instance member variable ko kayi namo se jana jata hai

jaise - attribute , data member , fields, properties

and

instance member functions are called as -

methods, procedures, action, operations, services