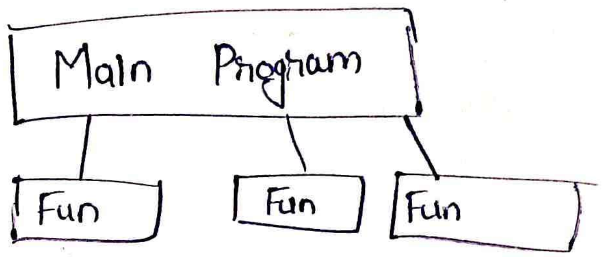
OOPS - CSE - ASHA KHILRANI

$$\begin{tikzcd}[column sep=small] & \text{Programming Language} \arrow[dl, no head] \arrow[dr, no head] & \\ \text{Procedure Oriented programming (POP)} & & \text{Object oriented programming. (OOP).} \end{tikzcd}$$

(1) Procedure Oriented programming (POP)

- In POP layer programs are divided into smaller programs. (procedure)
- In POP a program is written as a ~~separate~~ sequence of procedures (functions)
- Each procedure contains a series of instructions for performing a specific task
- During the program execution each procedure can be called by the other procedure.
- To call a procedure, we have to write procedure name only.
- The major emphasis of these languqge is on the procedure and not on the data
- pop language allow the data to move freely around the system.

- In OOP data is hidden (secure) and cannot be accessed by external functions.
- In OOP, objects communicate with each other through functions.
- In OOP emphasis is on data rather than procedures.
- OOP follows bottom up program design approach.



Lec-3

Difference between POP & OOP

Procedure oriented programming	Object oriented programming
① In POP, program is divided into smaller programs called procedures	① In OOP, program is divided into smaller entities called objects.
② It follow top down program design approach	② It follows bottom up program design approach
③ Importance given to algorithms (procedures) rather than data.	③ Importance given to data rather than algorithms. (functions / procedure).

③ Data Hiding / Information Hiding.

- The information of the data from direct access by the outside function /program is called data Hiding.
- We hide the data for security of data.
- In C++, we achieve data hiding by using private Access specifier.

④ Data Encapsulation.

The wrapping up of data and functions into a single unit (called class) is known as Encapsulation.

- In C++ Encapsulation can be achieved by using class.
- By data encapsulation, data is not accesible to the outside class, only those functions which are wrapped in the class can access it.
- Functions of the class provide the interface between the objects data and outside-objects /functions.

```
      {  x=l;        protected   accessed of base class
         y=t;
      }
   void  sum ()  {
            int  z;
            z = x +y;
           cout << " Result = "<< z;
     }  };
  void  main () {
      clrscr();
      Derived  D;
        D. setxy (5,2);
       D. sum ();
       getch ();
      }
```

MULTIPLE INHERITANCE

In Multiple Inheritance, there is multiple base classes and single derived class.

Example

```
  # include  < iostream.h>
  # include  < conio.h >
    class  Base 1
     { protected:
               int x;
       public
           void  setx (int l)
           {  x =l;  }
           };
```

·Ambiguity in single Inheritance

```
class Base
{ public :
          void display () {
              cout << " Base display "';
          }
   };
   class Derived : public Base {
   public :
        void display ()
         { cout <<" Derived display ";
         } }:

 void main() {
        clscr();
        derived D;
        D. display ()
         getch(); }
```

Method / Function overriding -

Redefinition of base class function in derived class is called as function overriding.

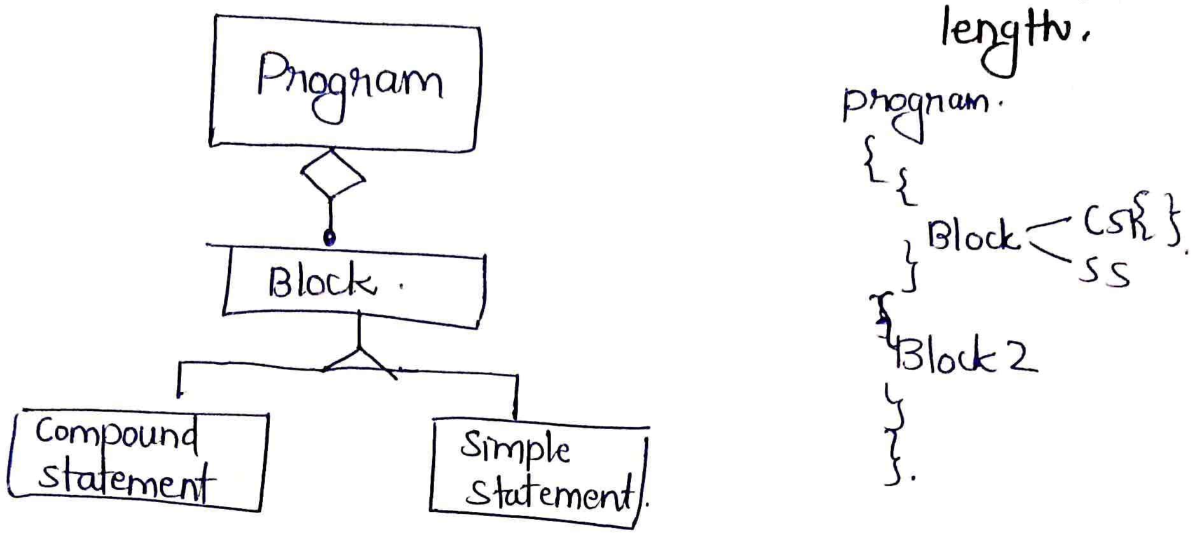
- It means having two or more function with the same name and signature but with different implementation (code).

Dynamic or Runtime polymorphism

It is also known as method overiding in which call to an overridden function is resolved during runtime, not at the compile time.

statements, the recursion terminates with simple statements.

- Here blocks can be nested to arbitary length.



Properties of Aggregation

Following are the properties of Aggregation:-

① Transitivity - If A is a part of B and B is a part of C then A is also part of C.

if $A \longrightarrow B$
if $B \longrightarrow C$
then $A \longrightarrow C$

Ex- if eye is a part of body of face and face is part of body then eye is also part of body.

② Anti Symmetry - In this property, if A is a part of B & B is not part of A then Antisymmetry occurs.

Ex:- If wheel is part of Car but car is not part of wheel.

Similarity -
We cannot create objects of both interface and abstract class, but we can create refrence of both of them.

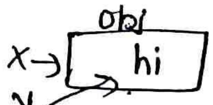
STRINGS.

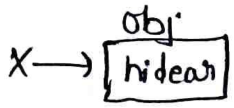
L-41

String is a sequence of characters (alphabets, number, $, %, - etc.) but it is not a primitive or built in type.

- When we create string in java, it actually creates an object ~~aof~~ type string.
- String is immutable object when which means that it cannot be changed once it is created
- whenever we change any string, a new instance is created

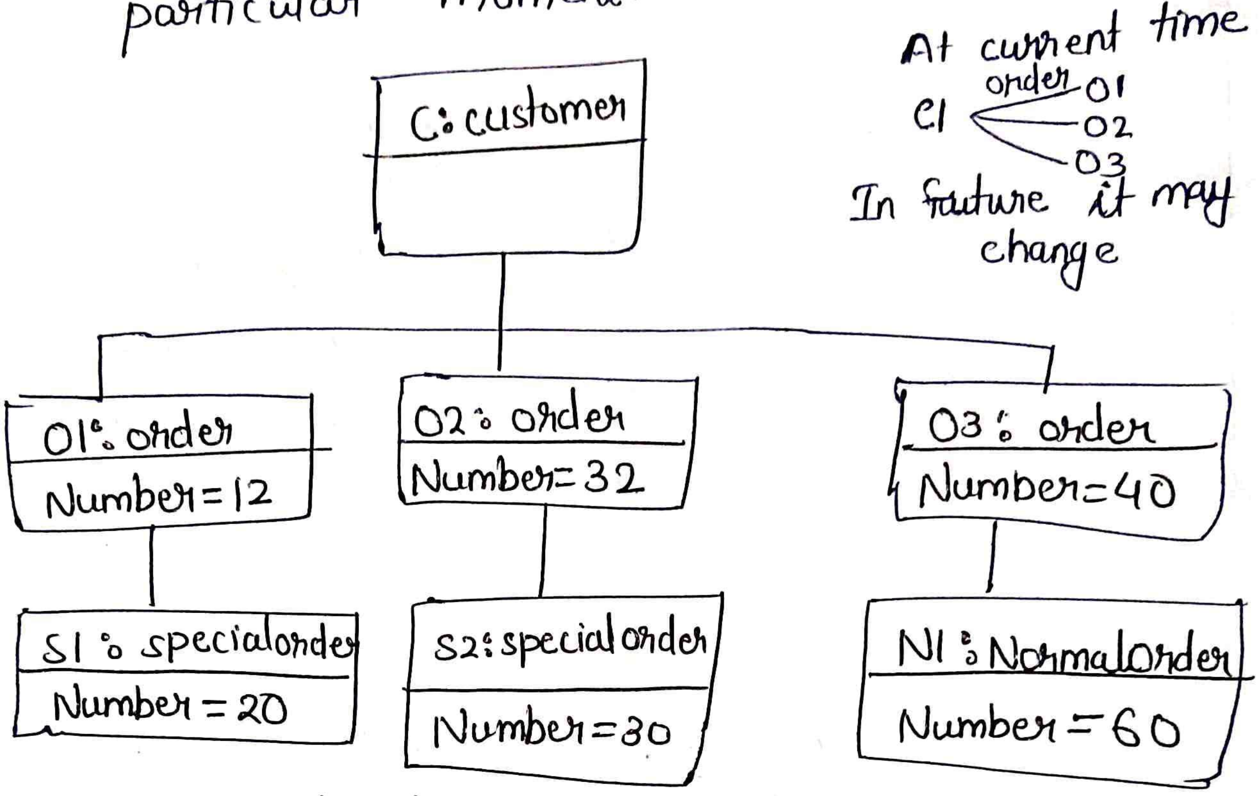
```
String x = "hi";
  string y = "hi";
    x = x. concate ("dear");
```





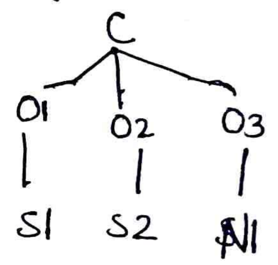
- Java provides the String class to create and manipulate string.
- String is a final class, we cannot extend
- To make the functionality of string class secure it is made as final.

- object diagrams can be imagined as the snapshot of running system at a particular moment.

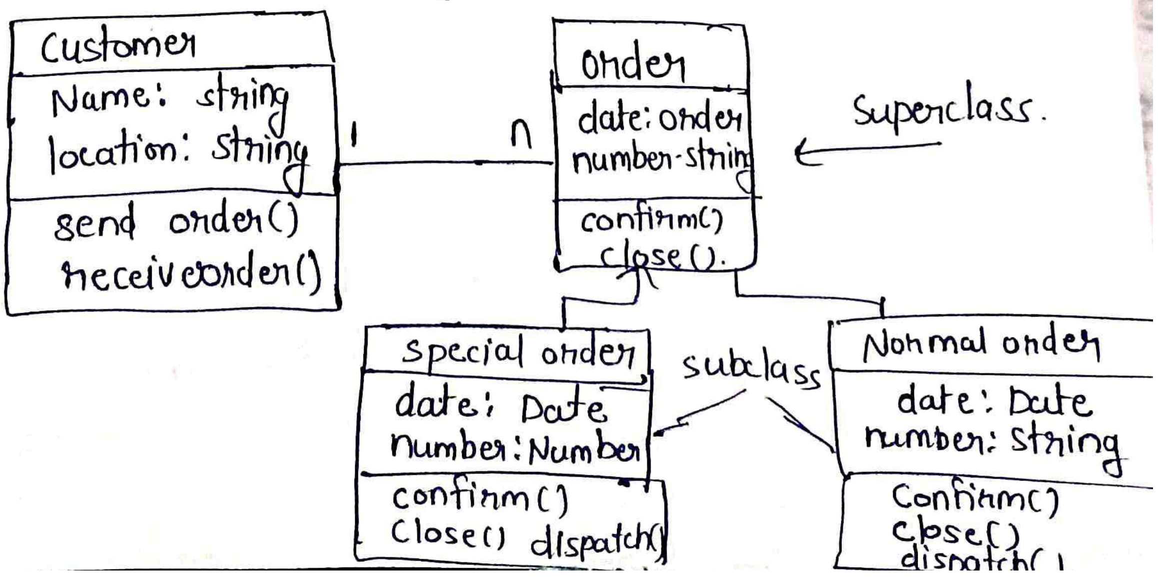


object diagram of order Management sys-

objects
- customer (C)
- order (O1, O2, O3)
- special order (S1, S2)
- Normal order (NO).



class diagram


Customer
Name: string
location: string
send order()
receiveorder()
1
n
Order
date: order
number-string
confirm()
close().
Superclass.
subclass
special order
date: Date
number: Number
confirm()
Close() dispatch()
Normal order
date: Date
number: string
Confirm()
close()
dispatch()