OOPS (Object Oriented Programming System)

C++ consists of all the programming concept that is real life concept →

i) Class
ii) Object
iii) Polymorphism
  a) Function Overloading
  b) Function Overriding
  c) Operator Overloading (Java doesn't support it)
iv) Inheritance
v) Constructor
vi) Destructor
vii) Scope Resolution Operator.
viii) Inline function.
ix) Interface (C++ doesn't support)
x) Applet
xi) Superclass
xii) Access Specifier.
  a) Private
  b) Default (C++ doesn't Support).
  c) Protected
  d) Public

i) Class: It is a user-defined datatype, which consists of function & variable. Class concept is used to organize the program. C program is unstructured and insecure programming. To secure the program in a structured way C++ concept has been introduced. Variable & function within a class is known as member variable & member function respectively. That means variable & function together forms data member of a class.

Syntax:

```
class classname
{
  public variablelist;
  public functionlist;
  }
  };
int main()
{
  class obj;

}
```

ii) **Object:** It is a runtime instance of a class, which is used to hold the data member of a class i.e. variable & function.

Through the object creation of the class we can call the function & variable of a class.

Syntax:

```
A ob;
↑  ↑
Class  Object
name   Name.
```

iii) **Polymorphism:** Poly = "many" & morphism = "forms". Function having same name with different forms is known as polymorphism. Polymorphism are of 3 types →.

i) **Function Overloading:** In case of function overloading only one class is acquired and within the class minimum 2 function must be present. Here function name, return type and access specifier must be same and parameter or arguement passing may or may not be same.

If the no. of parameter is same then datatype must be different and if the no. of parameter is different then datatype can be same.

Syntax: (Type 1).

Same datatype, different no. of parameter

```
class A
{ public void calc (int x)
  {
  }
  public void calc (int x, int y)
  {
  }
}
```

Syntax: (Type 2).
Same no. of parameter, Different Datatype.

```
class A
{ public void calc(int x).
  {
  }
  public void calc(double x)
  {
  }
}
```

ii) **Function Overriding**:. In case of function overriding minimum two class is required and within both the class minimum one function must be present. All the function name, return-type, access specifier and parameter passing must be same. ~~Child~~

Child class will prioritise its own function rather than parent class function i.e. child class function's output will be displayed and parent class function's output will be hidden.

Syntax:

```
class A                          ]
{ public void calc(int x)        ]  Parent class.
  {                              ]
  }                              ]
}
class B : public A.              ]
{ public void calc (int y)       ]  Child class.
  {                              ]
  }                              ]
}
```

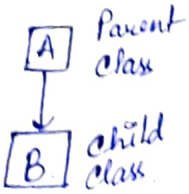
iv) **Inheritance :** It is one of the most important concept in C++. minimum two classes are required. i) Parent class Base class or Super class and another is ii) child class or Derived class or Sub class.

The class who inherits the features of another class is known as child class and the class which is inherited by another class is known as Parent class.

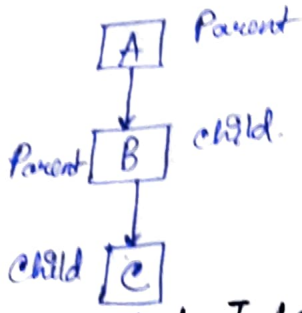
Types :

i) Single level Inheritance
ii) Multi-Level Inheritance
iii) Multiple Inheritance
iv) Hierarchical Inheritance
v) Hybrid Inheritance

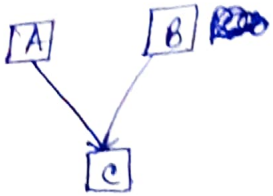
Single Level Inheritance.



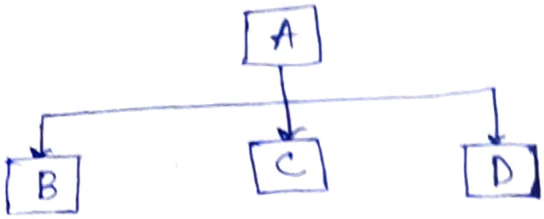
Multi Level Inheritance.



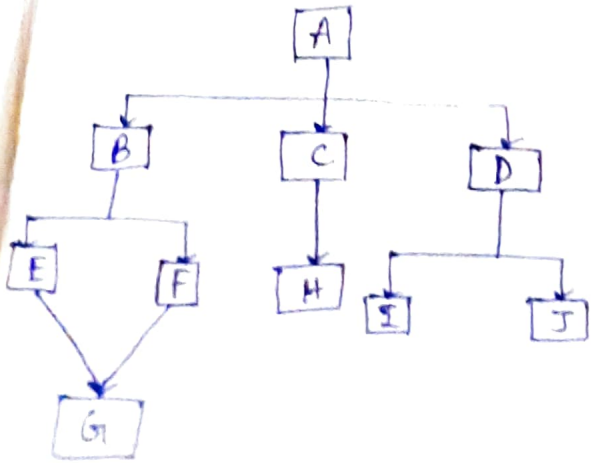
Multiple Inheritance.



Hierarchichal Inheritance.



Hybrid Inheritance.



S.L.I.
- A → B
- A → C
- A → D
- B → E
- B → F
- C → H
- D → I
- D → J
- E → G
- F → G

M.L.I.
- A → B → E → G.
- A → B → F → G.
- A → C → H
- A → D → I
- A → D → J.

M.I
- E, F → G

Hierarchichal I.
- A → B, C, D
- B → E, F.
- D → I, J

Access Specifier: Access Specifier is used to access a data member of a class i.e. variable & function within the scope of the class as well as outside the scope of the class.

a) **Private:** If we declare a variable and function as private then that variable and function can't be accessible outside the scope of the class.

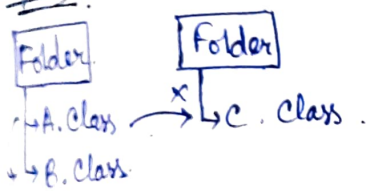
Syntax:

```
class A
{
  private int x;
}
```

→ class -র বাইরে accessible নয়

b) **Default:** Default variable can be accessible within the same directory either through inheritance or object creation, but it can't be accessible outside the scope of the directory.

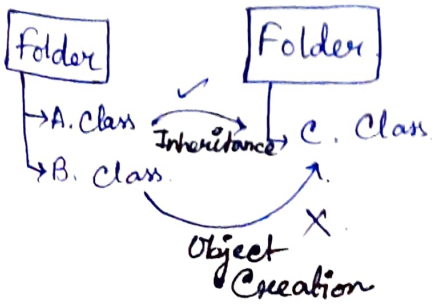
Syntax:



একই folder -র মধ্যে যতগুলোই variable থাকুক না কেন default variable access করতে পারে, কিন্তু আলাদা ~~variable~~ folder এ accessible নয়

c) **Protected:** If we declare a variable as protected then it can be accessible within the same folder as well as different folder if inheritance is there, but it can't be accessible through object creation.

Syntax:



d) **Public**: It is accessible anywhere within the program within the same folder as well as outside the ~~floor~~ folder.

Syntax: