**Abstract Class**

Abstract Class is a class which contains atleast one Pure Virtual function in it. Abstract classes are used to provide an Interface for its sub classes. Classes inheriting an Abstract Class must provide definition to the pure virtual function, otherwise they will also become abstract class.

**Characteristics of Abstract Class**

1. Abstract class cannot be instantiated, but pointers and refrences of Abstract class type can be created.
2. Abstract class can have normal functions and variables along with a pure virtual function.
3. Abstract classes are mainly used for Upcasting, so that its derived classes can use its interface.
4. Classes inheriting an Abstract Class must implement all pure virtual functions, or else they will become Abstract too.

**Pure Virtual Functions**

Pure virtual Functions are virtual functions with no definition. They start with **virtual** keyword and ends with= 0. Here is the syntax for a pure virtual function,

virtual void f() = 0;

**Example of Abstract Class**

class Base //Abstract base class

{

public:

**virtual void show() = 0;**//Pure Virtual Function

};

classDerived:public Base

{

public:

void **show**()

{ cout<< "Implementation of Virtual Function in Derived class"; }

};

int main()

{

Base obj; //Compile Time Error

Base \*b;

Derived d;

b = &d;

b->show();

}

Output : Implementation of Virtual Function in Derived class

In the above example Base class is abstract, with pure virtual **show()** function, hence we cannot create object of base class.

**Why can't we create Object of Abstract Class ?**

When we create a pure virtual function in Abstract class, we reserve a slot for a function in the VTABLE(studied in last topic), but doesn't put any address in that slot. Hence the VTABLE will be incomplete.

As the VTABLE for Abstract class is incomplete, hence the compiler will not let the creation of object for such class and will display an errror message whenever you try to do so.

**Pure Virtual definitions**

* Pure Virtual functions can be given a small definition in the Abstract class, which you want all the derived classes to have. Still you cannot create object of Abstract class.
* Also, the Pure Virtual function must be defined outside the class definition. If you will define it inside the class definition, complier will give an error. Inline pure virtual definition is Illegal.

class Base //Abstract base class

{

public:

**virtual void show() = 0;**//Pure Virtual Function

};

void Base :: show() //Pure Virtual definition

{

cout<< "Pure Virtual definition\n";

}

classDerived:public Base

{

public:

void**show**()

{ cout<< "Implementation of Virtual Function in Derived class"; }

};

int main()

{

Base \*b;

Derived d;

b = &d;

b->show();

}

**Output :**

Pure Virtual definition

Implementation of Virtual Function in Derived class