

ABSTRACT CLASSES



What is an Abstract Class?

- There are two types of classes **Abstract class** and **Concrete class**
- If **abstract** keyword is used before the class then it is an **Abstract Class** if nothing is written before class then it is a **Concrete class**
- Object of an Abstract class cannot be created but object of Concrete class can be created
- reference of abstract class is allowed

Example program

```
//abstract class
abstract class Super
{
    Super()
    {
        System.out.println("Super");
    }
    void meth1()
    {
        System.out.println("meth1");
    }
    abstract void meth2();
}

//concrete class
class sub extends Super
{
    void meth2()
    {
        System.out.println("meth2");
    }
}

class test
{
    public static void main()
    {
        Super s1; // reference of abstract is allowed
        sub s2 = new sub();
    }
}
```

- Method which is not having a body is known as **Abstract method**, the method must be declared as abstract
- The abstract method is **undefined** method
- A class is Abstract class if at least one of the methods is abstract
- If any other class inherits abstract class then that class also becomes abstract class but to become a concrete class the subclass must override the undefined method
- A class becomes useful if it overrides all the methods of abstract class
- Abstract classes are used for imposing standards and sharing methods
- Sub classes are meant for following standards



Do's and Don'ts of Abstract Class

- An Abstract class cannot be final because if it is made final then it cannot be extended whereas **abstract class is meant for inheritance**
- An Abstract method cannot be final because if it made final then it cannot be overridden whereas **Abstract method is meant for overriding**
- Abstract Class and method can neither be final nor static
- A Sub class must override an abstract method or else it will become abstract class