What is abstract class?

* *Class which has abstract keyword is called abstract class*

The rules for abstract methods and abstract classes are:

* A class can be marked as abstract with out containing any abstract method. But if a class has even one abstract method, then the class has to be an abstract class.

abstract class A  
{  
    // Valid, even with out any abstract methods  
}  
  
class B // Invalid, class B should be abstract, since it has abstract method.  
{  
    abstract void method1();  
}

* An abstract class can have one or more abstract methods.

abstract class C  
{  
    abstract void method1();  
  
    abstract double method2(int x, int y);  
  
    abstract boolean method3(char z);  
}

* An abstract class can have both abstract and non abstract (or concrete) method.

abstract class D  
{  
    void method1()  
    {  
        System.out.println("I am a concrete method");  
    }  
  
    abstract double method2(int x, int y);  
  
    int method3(double z)  
    {  
        System.out.println("I am also a concrete method");  
    }  
  
    abstract boolean method4(char z);  
}

* The abstract method should not have method body. Even empty flower braces { } are not allowed.

abstract class A  
{  
    abstract void method1(); // Valid  
  
    abstract void method2() {} // Invalid - since it has method body  
  
}

* Any sub-class extending from an abstract class should either implement all the abstract methods of the super-class or the sub-class itself should be marked as abstract.

abstract class A  
{  
    abstract void method1();  
  
    abstract void method2();  
  
}  
  
class B extends A  
{  
    // Invalid since B does not implement the abstract methods  
}  
  
abstract class C extends A  
{  
    // Valid since C is marked as abstract, even though the abstract methods are not implemented,  
}  
  
class D extends A  
{  
    void method1()  
    {  
        System.out.println("Method1 implemented here.");  
    }  
  
    // Invalid, class D should be marked as abstract, since method2 is not implemented.  
}  
  
abstract class E extends A  
{  
    void method1()  
    {  
        System.out.println("Method1 implemented here.");  
    }  
  
    // Even though method2 is not implemented, class D is marked as abstract, so it is Valid.  
}  
  
  
class F extends A  
{  
    // Valid since both methods are implemented here.  
    void method1()  
    {  
        System.out.println("Method1 implemented here.");  
    }  
  
    void method2()  
    {  
        System.out.println("Method2 implemented here.");  
    }  
}

* If an abstract class contains multiple methods, it is not necessary that all the methods of the abstract class are implemented in the immediate sub-class. Few of them can be implemented in sub-sub-classes or any where else in the sub-class hierarchy. But for a class to be concrete, all the abstract methods in its super-class must be implemented.

abstract class X  
{  
    abstract void method1();  
    abstract void method2();  
}  
  
abstract class Y extends X  
{  
    void method1()  
    {  
        System.out.println("Method1 implemented here.");  
    }  
}  
  
class Z extends Y  
{  
    void method2()  
    {  
        System.out.println("Method2 implemented here.");  
    }  
}

* It is not necessary to add the abstract methods only in the super most class, we can add more abstract methods in the sub-classes.

abstract class X  
{  
    abstract void method1();  
}  
  
abstract class Y extends X  
{  
    abstract void method2();  
}  
  
class Z extends Y  
{  
    void method1()  
    {  
        System.out.println("Method1 from class X implemented here.");  
    }  
  
    void method2()  
    {  
        System.out.println("Method1 from class Y implemented here.");  
    }  
}