abstract keyword in java

Difficulty Level : Easy Last Updated : 30 May, 2018

abstract is a non-access modifier in java applicable for classes, methods but **not** variables. It is used to achieve abstraction which is one of the pillar of Object Oriented Programming(OOP). Following are different contexts where *abstract* can be used in Java.

Abstract classes

The class which is having partial implementation(i.e. not all methods present in the class have method definition). To declare a class abstract, use this general form :

```
abstract class class-name{
    //body of class
}
```

Due to their partial implementation, we cannot instantiate abstract classes.Any subclass of an abstract class must either implement all of the abstract methods in the super-class, or be declared abstract itself.Some of the predefined classes in java are abstract. They depends on their sub-classes to provide complete implementation. For example, java.lang.Number is a abstract class. For more on abstract classes, see abstract classes in java

Abstract methods

Sometimes, we require just method declaration in super-classes.This can be achieve by specifying the **abstract** type modifier. These methods are sometimes referred to as *subclasser responsibility* because they have no implementation specified in the super-class. Thus, a subclass must override them to provide method definition. To declare an abstract method, use this general form:

```
abstract type method-name(parameter-list);
```

As you can see, no method body is present.Any concrete class(i.e. class without abstract keyword) that extends an abstract class must overrides all the abstract methods of the class.

Important rules for abstract methods:

- Any class that contains one or more abstract methods must also be declared abstract
- The following are various **illegal combinations** of other modifiers for methods with respect to *abstract* modifier :
  1. final
  2. abstract native
  3. abstract synchronized
  4. abstract static
  5. abstract private
  6. abstract strictfp

Consider the following java program, that illustrate the use of *abstract* keyword with classes and methods.

```
// A java program to demonstrate
// use of abstract keyword.

// abstract with class
abstract class A
{
    // abstract with method
    // it has no body
    abstract void m1();

    // concrete methods are still allowed in abstract classes
    void m2()
    {
        System.out.println("This is a concrete method.");
    }
}

// concrete class B
class B extends A
{
    // class B must override m1() method
    // otherwise, compile-time exception will be thrown
    void m1() {
        System.out.println("B's implementation of m2.");
    }


}

// Driver class
public class AbstractDemo
{
```

```
    public static void main(String args[])
    {
        B b = new B();
        b.m1();
        b.m2();
    }
}
```

Output:

```
B's implementation of m2.
This is a concrete method.
```

Note : Although abstract classes cannot be used to instantiate objects, they can be used to create object references, because Java's approach to run-time polymorphism is implemented through the use of super-class references. Thus, it must be possible to create a reference to an abstract class so that it can be used to point to a subclass object.

abstract and final

In java, you will never see a class or method declared with both final and abstract keywords. For classes, final is used to prevent inheritance whereas abstract classes depends upon their child classes for complete implementation. In cases of methods, final is used to prevent overriding whereas abstract methods needs to be overridden in sub-classes.

This article is contributed by **Gaurav Miglani**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.