8 Abstract Classes

In some real life applications, we have a single base class and a number of different derived classes. In such situations, if we have created a base class and want to ensure that no object of the base class is created later, we can make the base class as abstract. An **abstract class** is one that is not used to create objects. It is only used as base class for other classes.

Therefore, if the objects of a class cannot be instantiated, it is called an abstract class. The **abstract** keyword in a class indicates that the class cannot be instantiated and is an abstract class. Abstract classes are always **public** or friendly.

Syntax of declaration of an abstract class is

```
public abstract class class_name
{
    ...
        // class members
    ...
}
```

8.8.1 Characteristics of Abstract Class

Characteristics of abstract class are :

1. You cannot create an object of the abstract class.
2. An abstract class must be inherited.
3. An abstract class is always public or friendly.
4. An abstract class can contain abstract as well as non-abstract members.
5. Abstract class must contain at least one abstract method.
6. If any class inherits an abstract class then it must implements all the abstract methods of the abstract class.
7. Abstract class can't be static.
8. A class can't be both abstract and final.

8.8.2 Abstract Methods

Similar to abstract classes, we can also create abstract methods. When an mehtod declaration includes the modifier **abstract**, the method is said to be an **abstract method.**

Abstract methods have no implementation in the abstract class, so the method definition is followed by a semicolon. Therefore, an abstract method does not have method body. Abstract methods can only be declared in abstract classes.

The declaration of an abstract method in an abstract class is given below :

```
abstract class class_name
{
        abstract void method_name (parameter-list);
}
```

Example 8.12

Program to illustrate abstract method & class.

```
//abstract class
abstract class sample

{
    abstract void display();
}

// Derived class
class derived extends sample

{
    void display() // implementation of abstract method

    {
        System.out.println("Demo of Abstract Method and Class");
    }
}

class abstractdemo

{
    public static void main(String args[])

    {
        derived x = new derived();
        x.display();
    }
}
```

Output :

Demo of Abstract Method and Class

8.8.2.1 Characteristics of an Abstract Method

Characteristics of an abstract method are :

1. It cannot have implementation.
2. It is always public, default or protected.
3. It can be declared only in abstract classes.

4. It cannot take either **static** or **virtual** modifiers.
4. When we implement abstract methods in derived class, it must be overridden.
5. We can not declare abstract constructors.
6.

Example 8.13

Write a Java program to find the rate of interest of a particular bank like SBI, ICICI, PNB using abstract class.

```
abstract class bank

{
  abstract int rateofinterest( );
}
class SBI extends bank

 {
   int rateofinterest( )
    {
       return 7;
    }
 }

 class ICICI extends bank

 {
   int rateofinterest( )
    {
       return 8;
    }
 }

 class PNB extends bank
 {
   int rateofinterest( )
    {
       return 8;
    }
 }
```

```
class bankinterest
{
  public static void main (String arg [ ])
  {
    // create SBI object
    bank obj1 = new SBI( );
    int interest = obj1. rateofinterest( );
    System.out.println("Rate of Interest of SBI bank is " + interest + "% ");

    // create ICICI object
    bank obj2 = new ICICI( );
    int interest1 = obj2.rateofinterest( );
    System.out.println("Rate of Interest of ICICI bank is " + interest1 + "% ");
  }
}
```

Output :
Rate of Interest of SBI bank is 7%
Rate of Interest of ICICI bank is 8%

8.9 Final Keyword

The final keyword can be used as

1. final variables (to declare constants, already discussed)
2. final classes (to avoid inheritance)
3. final methods (to avoid method overriding)

8.9.1 Final Classes

A **final class** impiles that the class cannot be used as a base class. Once you have declared a class as final, no other class can inherit that class. Therefore, the final keyword in a class is used to indicate that a class cannot be inherited.

Syntax is :

```
final class fclass
{
    ...
    ...  // class members
}
```

In this example, **fclass** is name of final class. If we try the following statement to attempt to derive a class from the fclass, we get an error message.

Example 8.14

```
class derived extends fclass
{
    ...
        //class members
    ...
}
```

When we execute the above mentioned statements, we shall get an error message.

The **main purpose** of using final classes in a program is to take away the inheritance feature from the user. It also allows the Java compiler to perform some optimizations when a method of a final class is invoked. A **final** class can not be an **abstract** class.

8.9.2 Final Methods

Similar to final classes, we can declare some or all of class methods as **final**. The **final** keyword is used in a method declaration to indicate that the method cannot be overridden by subclasses.

Syntax is :

```
final returntype mname (parameters list)
{
    ...
        // body of  method
    ...
}
```

Example 8.15

```
class vehicle

{
  final void condition( )
  {
     System.out.println("running ");
  }
}
```

```
class bike extends vehicle
{
   void condition( )
   {
      System.out.println("Good condition ");
   }
}

class finalmethoddemo
{
   public static void main(String args[ ])
   {
      bike obj = new bike( );
      obj.condition( );
   }
}
```

Output :

Compile time error