ABSTRACT CLASSES AND METHODS

Abstract class

A class that is declared as abstract is known as **abstract class**. It can have abstract and non-abstract methods (method with body). It needs to be extended and its method implemented. It cannot be instantiated.

Syntax:

```
abstract class classname
{
}
```

Abstract method

A method that is declared as abstract and does not have implementation is known as abstract method. The method body will be defined by its subclass.

Abstract method can never be final and static. Any class that extends an abstract class must implement all the abstract methods declared by the super class.

Note:

A normal class (non-abstract class) cannot have abstract methods.

Syntax:

abstract returntype functionname (); //No definition

Syntax for abstract class and method:

```
modifier abstract class className
  {
   //declare fields
   //declare methods
   abstract dataType methodName();
}
modifier class childClass extends className
{
dataType methodName()
{
}
}
```

Example 1

```
//abstract parent class
abstract class Animal
{
  //abstract method
  public abstract void sound();
}
//Lion class extends Animal class
public class Lion extends Animal
{
  public void sound()
{
    System.out.println("Roars");
  }
  public static void main(String args[])
{
    Animal obj = new Lion();
    obj.sound();
  }
}
```

Output:

```
Roars
```

In the above code, Animal is an abstract class and Lion is a concrete class.

Example 2

```
abstract class Bank
{
abstract int getRateOfInterest();
}
class SBI extends Bank
{
int getRateOfInterest()
{
   return 7;
}
}
class PNB extends Bank
{
```

```
int getRateOfInterest()
{
   return 8;
}
}
public class TestBank
{
public static void main(String args[])
{
Bank b=new SBI();//if object is PNB, method of PNB will be invoked
int interest=b.getRateOfInterest();
System.out.println("Rate of Interest is: "+interest+" %");
b=new PNB();
System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");
}
}
```

Output:

```
Rate of Interest is: 7 %
Rate of Interest is: 8 %
```

Abstract class with concrete (normal) method

Abstract classes can also have normal methods with definitions, along with abstract methods.

Sample Code:

```
abstract class A
{
 abstract void callme();
 public void normal()
 {
  System.out.println("this is a normal (concrete) method.");
 }
}
public class B extends A
{
 void callme()
 {
    System.out.println("this is an callme (abstract) method.");
   }
   public static void main(String[] args)
   {
    B b = new B();
    b.callme();
    b.normal();
   }
  }
```

Output:

```
  this is an callme (abstract) method.
  this is a normal (concrete) method.
```

Observations about abstract classes in Java

1. **An instance of an abstract class cannot be created; But, we can have referencesof abstract class type though.**

Sample Code:

```
abstract class Base
{
   abstract void fun();
}
class Derived extends Base
{
   void fun()
{
System.out.println("Derived fun() called");
}
}
public class Main
{
   public static void main(String args[])
{
      // Base b = new Base(); Will lead to error
      // We can have references of Base type.
     Base b = new Derived();
```

```
        b.fun();
      }
    }
```

Output:

```
Derived fun() called
```

2. **An abstract class can contain constructors in Java. And a constructor of abstract class is called when an instance of a inherited class is created.**

Sample Code:

```
abstract class Base
{
   Base()
   {
      System.out.println("Within Base Constructor");
   }
   abstract void fun();
}
class Derived extends Base
{
   Derived()
   {
      System.out.println("Within Derived Constructor");
   }
   void fun()
   {
      System.out.println(" Within Derived fun()");
   }
}
public class Main
{
   public static void main(String args[])
   {
     Derived d = new Derived();
   }
}
```

Output:

```
Within Base Constructor
Within Derived Constructor
```

3. **We can have an abstract class without any abstract method. This allows us to create classes that cannot be instantiated, but can only be inherited.**

Sample Code:

```
abstract class Base
{
   void fun()
   {
      System.out.println("Within Base fun()");
   }
}
class Derived extends Base
{
}
public class Main
{
   public static void main(String args[])
   {
      Derived d = new Derived();
      d.fun();
   }
}
```

Output:

```
Within Base fun()
```

4. **Abstract classes can also have final methods (methods that cannot be overridden).**

Sample Code:

```
abstract class Base
{
   final void fun()
   {
        System.out.println("Within Derived fun()");
    }
}
class Derived extends Base
{
}
public class Main
{
    public static void main(String args[])
    {
        Base b = new Derived();
        b.fun();
    }
}
```

Output:

```
Within Derived fun()
```