Abstract class in Java

A class that is declared with abstract keyword, is known as abstract class in java. It can have abstract and non-abstract methods (method with body).

**Abstraction in Java**

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only important things to the user and hides the internal details for example sending sms, you just type the text and send the message. You don't know the internal processing about the message delivery.

Abstraction lets you focus on what the object does instead of how it does it.

**Ways to achieve Abstaction**

1) Abstract class (0 to 100%)

2) Interface (100%)

**Abstract class in Java**

A class that is declared as abstract is known as abstract class. It needs to be extended and its method implemented. It cannot be instantiated.

**Example abstract class**

abstract class A{}

**abstract method**

A method that is declared as abstract and does not have implementation is known as abstract method.

**Example abstract method**

abstract void printStatus();//no body and abstract

**Example of abstract class that has abstract method**

In this example, Bike the abstract class that contains only one abstract method run. It implementation is provided by the Honda class.

abstract class Bike{

abstract void run();

}

class Honda4 extends Bike{

void run()

{

System.out.println("running safely..");

}

public static void main(String args[]){

Bike obj = new Honda4();

obj.run();

} }

**Output:**

running safely..

**Understanding the real scenario of abstract class**

In this example, Shape is the abstract class, its implementation is provided by the Rectangle and Circle classes. Mostly, we don't know about the implementation class (i.e. hidden to the end user) and object of the implementation class is provided by the factory method.

**A factory method is the method that returns the instance of the class.**

In this example, if you create the instance of Rectangle class, draw() method of Rectangle class will be invoked.

**TestAbstraction1.java**

abstract class Shape{

abstract void draw();

}

//In real scenario, implementation is provided by others i.e. unknown //by end user

class Rectangle extends Shape{

void draw(){System.out.println("drawing rectangle");}

}

class Circle1 extends Shape{

void draw(){System.out.println("drawing circle");}

}

//In real scenario, method is called by programmer or user

class TestAbstraction1{

public static void main(String args[]){

Shape s=new Circle1();//In real scenario, object is provided through method e.g. getShape() method

s.draw();

} }

**Output:**

drawing circle

===========

**Another example of abstract class in java**

//TestBank.java

abstract class Bank{

abstract int getRateOfInterest();

}

class SBI extends Bank{

int getRateOfInterest(){return 7;}

}

class PNB extends Bank{

int getRateOfInterest(){return 7;}

}

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+" %");

}}

**Output:**

Rate of Interest is: 7 %

=====

**Abstract class having constructor, data member, methods etc.**

An abstract class can have data member, abstract method, method body, constructor and even main() method.

**TestAbstraction2.java**

//example of abstract class that have method body

abstract class Bike{

Bike(){System.out.println("bike is created");}

abstract void run();

void changeGear(){System.out.println("gear changed");}

}

class Honda extends Bike{

void run(){System.out.println("running safely..");}

}

class TestAbstraction2{

public static void main(String args[]){

Bike obj = new Honda();

obj.run();

obj.changeGear();

} }

Output:

bike is created

running safely..

gear changed

============

**Another real scenario of abstract class**

The abstract class can also be used to provide some implementation of the interface. In such case, the end user may not be forced to override all the methods of the interface.

Note: If you are beginner to java, learn interface first and skip this example.

interface A{

void a();

void b();

void c();

void d();

}

abstract class B implements A{

public void c(){System.out.println("I am C");}

}

class M extends B{

public void a(){System.out.println("I am a");}

public void b(){System.out.println("I am b");}

public void d(){System.out.println("I am d");}

}

class Test5{

public static void main(String args[]){

A a=new M();

a.a();

a.b();

a.c();

a.d();

}}

**Output**:I am a

I am b

I am c

I am d