Abstract Class in Java

- A class that is declared with abstract keyword, is known as abstract class in java.
- It needs to be extended and its method implemented. It cannot be instantiated.
- It can have abstract and non-abstract methods (method with body).
- Abstract classes cannot be instantiated, but they can be subclassed.
- When an abstract class is subclassed, the subclass usually provides implementations for all of the abstract methods in its parent class. However, if it does not, then the subclass must also be declared abstract.
- If a class includes abstract methods, then the class itself *must* be declared abstract

Example of abstract class that has abstract method

```
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();
}
}
```

```
abstract class Shape{
abstract void draw();
}

class Rectangle extends Shape{
void draw(){System.out.println("drawing rectangle");}
}

class Circle extends Shape{
void draw(){System.out.println("drawing circle");}
}

class TestAbstraction1{
public static void main(String args[]){
Shape s=new Circle();
s.draw();
}
}
```

```
abstract class Bank{
abstract int getRateOfInterest();
}
class SBI extends Bank{
int getRateOfInterest(){return 7;}
}
class PNB extends Bank{
int getRateOfInterest(){return 8;}
}

class TestBank{
public static void main(String args[]){
Bank b;
b=new SBI();
System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");
b=new PNB();
System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");
}}
```

```
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();
 }
}
```

```
public abstract Animal {
public void eat(int food) {
 // method for food
}
public void sleep(int hours) {
 //method for sleep
 }
 public abstract void makeNoise();
}

  public Dog extends Animal {
  public void makeNoise()
  { System.out.println ("Bark! Bark!");
   } }

   public Cow extends Animal {
   public void makeNoise() {
   System.out.println ("Moo! Moo!"); } }
```

```
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();
}}
```

Difference between abstract class and interface

1. Abstract class and interface both are used to achieve abstraction where we can declare the abstract methods.
2. Abstract class and interface both can't be instantiated.

Abstract class	Interface
1) Abstract class can **have abstract and non-abstract** methods.	Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also.
2) Abstract class **can have final, non-final, static and non-static variables**.	Interface has **only static and final variables**.
4) Abstract class **can provide the implementation of interface**.	Interface **can't provide the implementation of abstract class**.
5) The **abstract keyword** is used to declare abstract class.	The **interface keyword** is used to declare interface.

6) **Example:**
public abstract class Shape{
public abstract void draw();
}
	Example:
	public interface Drawable{
	void draw();
	}

7) We can define public, protected, and private concrete methods	All methods that you declare or define are public