# Abstract Class in Java with example

A class that is declared using "**abstract**" keyword is known as abstract class. It can have abstract methods(methods without body) as well as concrete methods (regular methods with body). A normal class(non-abstract class) cannot have abstract methods. In this guide we will learn what is a abstract class, why we use it and what are the rules that we must remember while working with it in Java.

An abstract class can not be **instantiated**, which means you are not allowed to create an **object** of it. Why? We will discuss that later in this guide.

## Why we need an abstract class?

Lets say we have a class `Animal` that has a method `sound()` and the subclasses(see inheritance) of it like `Dog`, `Lion`, `Horse`, `Cat` etc. Since the animal sound differs from one animal to another, there is no point to implement this method in parent class. This is because every child class must override this method to give its own implementation details, like `Lion` class will say "Roar" in this method and `Dog` class will say "Woof".

So when we know that all the animal child classes will and should override this method, then there is no point to implement this method in parent class. Thus, making this method abstract would be the good choice as by making this method abstract we force all the sub classes to implement this method( otherwise you will get compilation error), also we need not to give any implementation to this method in parent class.

Since the `Animal` class has an abstract method, you must need to declare this class abstract.

Now each animal must have a sound, by making this method abstract we made it compulsory to the child class to give implementation details to this method. This way we ensures that every animal has a sound.

## Abstract class Example

```
//abstract parent class
abstract class Animal{
   //abstract method
   public abstract void sound();
}
//Dog class extends Animal class
public class Dog extends Animal{

   public void sound(){
        System.out.println("Woof");
   }
   public static void main(String args[]){
        Animal obj = new Dog();
        obj.sound();
```

```
    }
}
```

Output:

```
Woof
```

Hence for such kind of scenarios we generally declare the class as abstract and later **concrete classes** extend these classes and override the methods accordingly and can have their own methods as well.

# Abstract class declaration

An abstract class outlines the methods but not necessarily implements all the methods.

```
//Declaration using abstract keyword
abstract class A{
   //This is abstract method
   abstract void myMethod();

   //This is concrete method with body
   void anotherMethod(){
      //Does something
   }
}
```

# Rules

**Note 1:** As we seen in the above example, there are cases when it is difficult or often unnecessary to implement all the methods in parent class. In these cases, we can declare the parent class as abstract, which makes it a special class which is not complete on its own.

A class derived from the abstract class must implement all those methods that are declared as abstract in the parent class.

**Note 2:** Abstract class cannot be instantiated which means you cannot create the object of it. To use this class, you need to create another class that extends this this class and provides the implementation of abstract methods, then you can use the object of that child class to call non-abstract methods of parent class as well as implemented methods(those that were abstract in parent but implemented in child class).

**Note 3:** If a child does not implement all the abstract methods of abstract parent class, then the child class must need to be declared abstract as well.

**Do you know?** Since abstract class allows concrete methods as well, it does not provide 100% abstraction. You can say that it provides partial abstraction. Abstraction is a process where you show only “relevant” data and “hide” unnecessary details of an object from the user.

Interfaces on the other hand are used for 100% abstraction (See more about abstraction here).
You may also want to read this: Difference between abstract class and Interface in Java

### Why can't we create the object of an abstract class?

Because these classes are incomplete, they have abstract methods that have no body so if java allows you to create object of this class then if someone calls the abstract method using that object then What would happen?There would be no actual implementation of the method to invoke.
Also because an object is concrete. An abstract class is like a template, so you have to extend it and build on it before you can use it.

### Example to demonstrate that object creation of abstract class is not allowed

As discussed above, we cannot instantiate an abstract class. This program throws a compilation error.

```
abstract class AbstractDemo{
   public void myMethod(){
      System.out.println("Hello");
   }
   abstract public void anotherMethod();
}
public class Demo extends AbstractDemo{

   public void anotherMethod() {
        System.out.print("Abstract method");
   }
   public static void main(String args[])
   {
      //error: You can't create object of it
      AbstractDemo obj = new AbstractDemo();
      obj.anotherMethod();
   }
}
```

Output:

```
Unresolved compilation problem: Cannot instantiate the type AbstractDemo
```

Note: The class that extends the abstract class, have to implement all the abstract methods of it, else you have to declare that class abstract as well.

## Abstract class vs Concrete class

A class which is not abstract is referred as **Concrete class**. In the above example that we have seen in the beginning of this guide, `Animal` is a abstract class and `Cat`, `Dog` & `Lion` are concrete classes.

**Key Points:**

1. An abstract class has no use until unless it is extended by some other class.
2. If you declare an **abstract method** in a class then you must declare the class abstract as well. you can't have abstract method in a concrete class. It's vice versa is not always true: If a class is not having any abstract method then also it can be marked as abstract.
3. It can have non-abstract method (concrete) as well.

I have covered the rules and examples of abstract methods in a separate tutorial, You can find the guide here: Abstract method in Java
For now lets just see some basics and example of abstract method.
1) Abstract method has no body.
2) Always end the declaration with a **semicolon**(;).
3) It must be overridden. An abstract class must be extended and in a same way abstract method must be overridden.
4) A class has to be declared abstract to have abstract methods.

**Note:** The class which is extending abstract class must override all the abstract methods.

# Example of Abstract class and method

```
abstract class MyClass{
   public void disp(){
     System.out.println("Concrete method of parent class");
   }
   abstract public void disp2();
}

class Demo extends MyClass{
   /* Must Override this method while extending
    * MyClas
    */
   public void disp2()
   {
       System.out.println("overriding abstract method");
   }
   public static void main(String args[]){
       Demo obj = new Demo();
       obj.disp2();
   }
}
```

Output:

```
overriding abstract method
```