**Abstract class and methods**

### **Abstract Class**

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.

An abstract class cannot be **instantiated**, which means you are not allowed to create an **object** of it

It is used to achieve abstraction but it does not provide 100% abstraction because it can have concrete methods. But using interface we can achieve 100% abstraction since all the method in interface are abstract by default

## **Why we need an abstract class?**

Let’s say we have a class Animal that has a method sound () and the subclasses (see [**inheritance**](https://beginnersbook.com/2013/03/inheritance-in-java/)) 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 metho. 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 ensure that every animal has a sound.

Abstraction is an important feature of OOPS. It means hiding complexity and show functionality only to the user. Abstract methods are usually declared where two or more subclasses are expected to do a similar thing in different ways through different implementations. These subclasses extend the same Abstract class and provide different implementations for the abstract methods.

## **Rules**

**Note 1:**  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,

**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.

[**Interfaces**](https://beginnersbook.com/2013/05/java-interface/) on the other hand are used for 100% abstraction an abstract class has no use until unless it is extended by some other class.

1. 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.
2. It can have non-abstract method (concrete) as well

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();

}

}

# Abstract method in Java

# A method without body (no implementation) is known as abstract method. A method must always be declared in an abstract class, or in other words you can say that if a class has an abstract method, it should be declared abstract as well

public abstract int myMethod(int n1, int n2);

As you see this has no body.

## **Rules of Abstract Method**

1. Abstract methods don’t have body; they just have method signature as shown above.

2. If a class has an abstract method, it should be declared abstract, the vice versa is not true, which means an abstract class doesn’t need to have an abstract method compulsory.

3. If a regular class extends an abstract class, then the class must have to implement all the abstract methods of abstract parent class or it has to be declared abstract as well.

4.Abstract method can never be final and static.

## Example 1: abstract method in an abstract class

abstract class Sum{

public abstract int sumOfTwo(int n1, int n2);

public abstract int sumOfThree(int n1, int n2, int n3);

//Regular method

public void disp(){

System.out.println("Method of class Sum");

}

}

//Regular class extends abstract class

class Demo extends Sum{

/\* If I don't provide the implementation of these two methods, the

\* program will throw compilation error.

\*/

public int sumOfTwo(int num1, int num2){

return num1+num2;

}

public int sumOfThree(int num1, int num2, int num3){

return num1+num2+num3;

}

public static void main(String args[]){

Sum obj = new Demo();

System.out.println(obj.sumOfTwo(3, 7));

System.out.println(obj.sumOfThree(4, 3, 19));

obj.disp();

}

}

## Example 2: abstract method in interface

All the methods of an [interface](https://beginnersbook.com/2013/05/java-interface/) are public abstract by default. You cannot have concrete (regular methods with body) methods in an interface.

//Interface

interface Multiply{

//abstract methods

public abstract int multiplyTwo(int n1, int n2);

/\* We need not to mention public and abstract in interface

\* as all the methods in interface are

\* public and abstract by default so the compiler will

\* treat this as

\* public abstract multiplyThree(int n1, int n2, int n3);

\*/

int multiplyThree(int n1, int n2, int n3);

/\* Regular (or concrete) methods are not allowed in an interface

\* so if I uncomment this method, you will get compilation error

\* public void disp(){

\* System.out.println("I will give error if u uncomment me");

\* }

\*/

}

class Demo implements Multiply{

public int multiplyTwo(int num1, int num2){

return num1\*num2;

}

public int multiplyThree(int num1, int num2, int num3){

return num1\*num2\*num3;

}

public static void main(String args[]){

Multiply obj = new Demo();

System.out.println(obj.multiplyTwo(3, 7));

System.out.println(obj.multiplyThree(1, 9, 0));

}

}