

1. What do you mean by 'abstract' keyword? WAP in which your class contain more than one abstract method.

Abstract keyword is a non-access modifier keyword which is used to apply abstraction to any class or method.

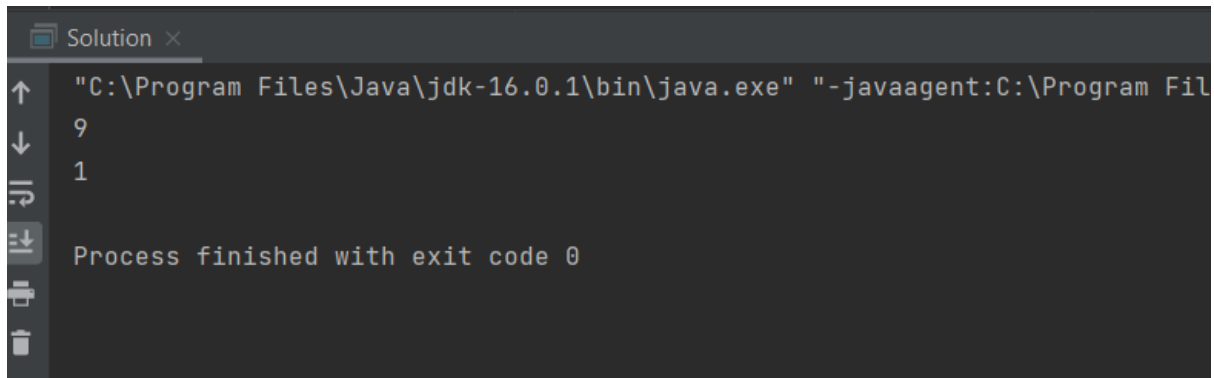
When a class is abstracted, one cannot create its instance in any of the method, to use the features of an abstract class another class must inherit it.

Whereas when a method is abstracted, we can't add a body to the method and to use that method the child class must override it.

## Code

```
class Solution {
    public static void main(String[] args) {
        SecondClass secondClass = new SecondClass();//Object of child
        class also calls the method of parent class
        //Even when it is
        abstract
        System.out.println(secondClass.sum(5,4));
        System.out.println(secondClass.difference(5,4));
    }
}
abstract class NewClass{
    //abstract methods can't have a body
    abstract int sum(int a,int b);
    abstract int difference(int a,int b);
}
class SecondClass extends NewClass{
    @Override
    //abstract methods must be overridden in the child class
    int sum(int a,int b){
        return a+b;
    }
    @Override
    int difference(int a,int b){
        return Math.abs(a-b);
    }
}
```

**Output: -**



```
Solution x
"C:\Program Files\Java\jdk-16.0.1\bin\java.exe" -javaagent:C:\Program Fil
9
1
Process finished with exit code 0
```

**2. What would be the behavior if this() and super() used in a method?**

The keyword this() behaves as an instance of the same class and creates an object with all the variables of the same class. Basically it acts as the constructor of the class called in that same class but **not recursively**.

Whereas super() behaves as the instance of the super class and creates an object with all the variables of the super class of that child class. Basically, it acts as the constructor of the parent class in the child class.

**3. I would like to set data as a = 2 and b = 3, so I came up with the following code. But it won't give the required output. Check and update the code so it would give the output as per the need.**

```
class Account{
    int a;
    int b;
    public void setData(int a, int b) {
        a = a;
        b = b;
    }
    public void showData( ) {
        System.out.println( " Value of A =" + a );
        System.out.println( " Value of B =" + b );
    }
    public static void main ( String args[] ) {
```

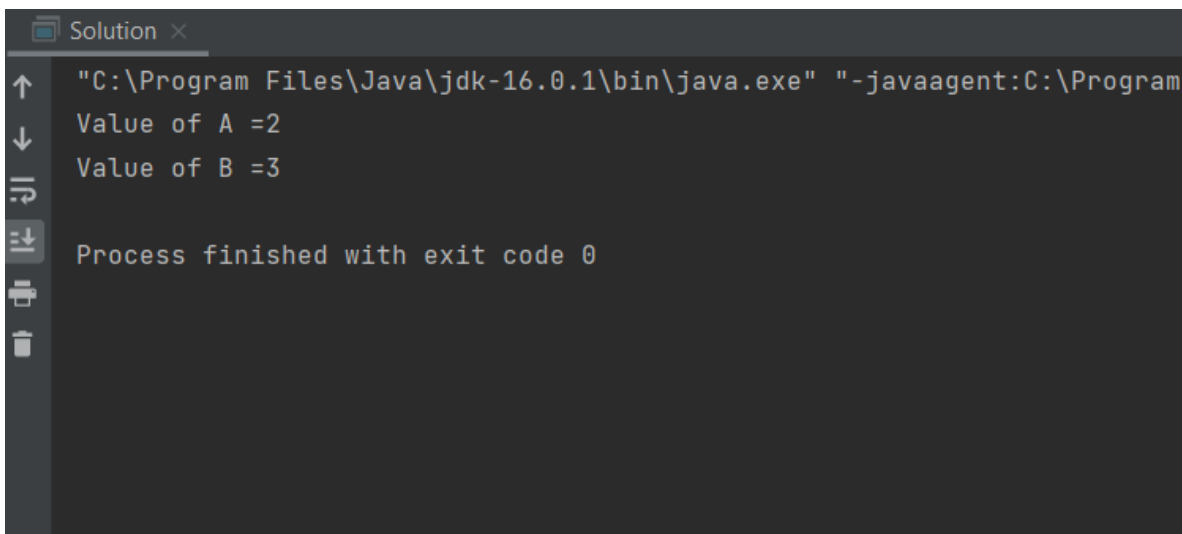
```
Account obj = new Account ();  
  
obj.setData( 2, 3);  
  
obj.showData( );  
  
}  
  
}
```

In the method setData() we are naming the parameters as same as the variables, therefore the variables a and b remains uninitialized resulting in the null or 0 as the output.

We must use this keyword before a = a and b = b so that the variables a and b get themselves a value as per the parametes of setData() method.

```
public void setData(int a, int b) {  
  
    this.a = a;  
  
    this.b = b;  
  
}
```

## Output



```
Solution x  
"C:\Program Files\Java\jdk-16.0.1\bin\java.exe" "-javaagent:C:\Program  
Value of A =2  
Value of B =3  
Process finished with exit code 0
```

### 4. When do static members of a class get initialized? How can we call non-static methods from static methods in java? Explain with code.


Static members are initialized at the time of the execution of the code, as the entire code have a single copy of the static variables or objects.

We can call a not static method from a static method by creating an object of the class.

### Example with code:

```
class Solution{
    int a = 10;
    int b = 20;
    public void showData( ) {
        System.out.println( "Value of A =" + a );
        System.out.println( "Value of B =" + b );
    }
    public static void main ( String args[] ) {
        Solution obj = new Solution();//object of the class is created
        // non-static method is called from a static method
        obj.showData( );
    }
}
```

### Output



```
Solution x
"C:\Program Files\Java\jdk-16.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\I
Value of A =2
Value of B =3
Process finished with exit code 0
```

### 5. Show the difference between this() and super() with help of a code.

this()	super()
<pre>class Solution{     int a;     int b;     Solution(){         this(10,20);// acting as the second constructor of this class         System.out.println(a);         System.out.println(b);     }     Solution(int a,int b){         this.a = a;         this.b = b;     }     public static void main ( String args[] ) {         Solution obj = new Solution();//object of the class is created     } }</pre>	<pre>class Solution extends Parent{     Solution(){         super(10,20);//Acting as the constructor of the parent class         System.out.println(a);         System.out.println(b);     }     public static void main ( String args[] ) {         Solution obj = new Solution();//object of the class is created     } } class Parent{     int a;     int b;     Parent(int a,int b){         this.a = a;         this.b = b;     } }</pre>

	<pre>} }</pre>
this() keyword calls the constructor of the same class.	super() keyword calls the constructor of the parent class.