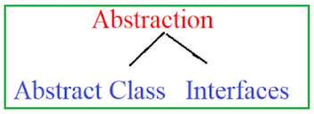
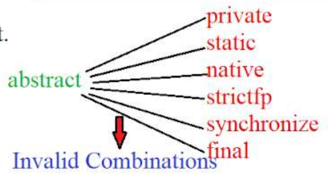
Abstraction: It is a process of hiding the implementation details from the user & showing only necessary details to the user.

abstract Method: **<AccessModifier>abstract<ReturnType><MethodName>();**

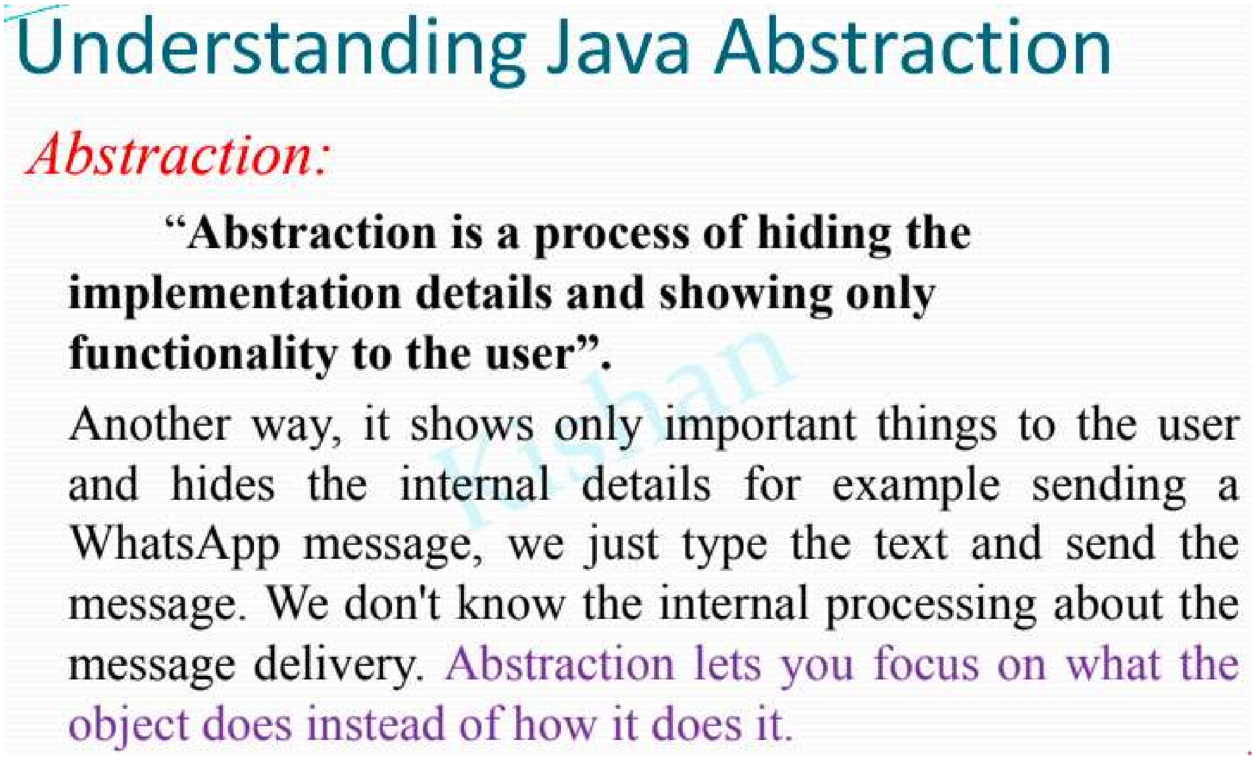


1) A method which is declared as abstract with **abstract** keyword is known as abstract method.
2) An **abstract** method always ends with a semicolon **;**
3) **For an abstract method there will not be any method body.**
4) Implementation for an abstract method should be given in the next class by using **M**ethod **O**verriding concept.

abstract Class:



1) A class which is declared as abstract with **abstract** keyword is known as abstract class.
2) Inside an **abstract class** we can have **BOTH** normal methods and abstract methods.
3) It is not mandatory to write at least one abstract method inside an abstract class.
4) Writing abstract methods inside an **abstract class** is always **OPTIONAL.**
5) **If we are writing an abstract method inside a normal class** then 100% that class **should** be declared as abstract class otherwise we will be getting an compile time error.
6) Inside an abstract class we can write main(), constructors, normal methods, abstract methods including static methods also.
7) abstract class can't be instantiated means **we cannot create an object for abstract class.**
8) If we are **inheriting** an **abstract class** then in the **child class 100%** we need to provide implementation **(Method body) for all the abstract methods** which are **present in the abstract class** otherwise we will be getting an compile time error in the child class.
9) If we are not overriding **all** the abstract methods present in the abstract class, then in the child class we will be getting an **C.E.**
10) In the child class **if we don't want to provide implementation for the abstract methods** present in the abstract class then we need to **make our child class also as abstract.**

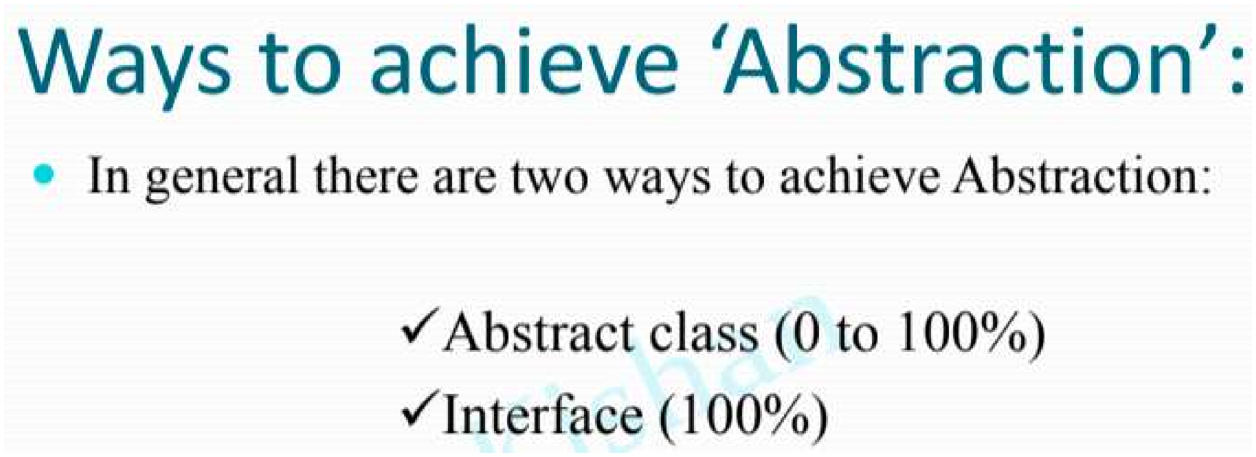


Understanding Java Abstraction

Abstraction:

"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 a WhatsApp message, we just type the text and send the message. We 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.

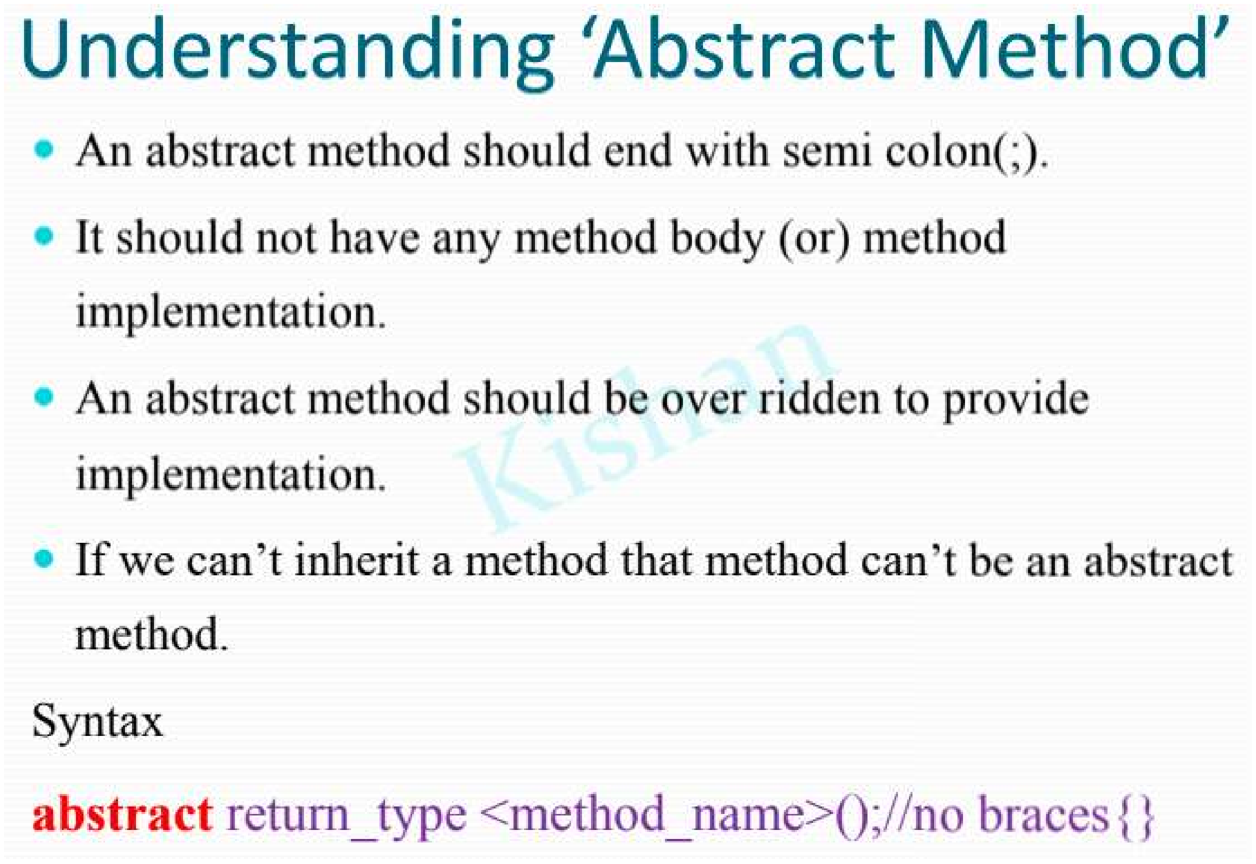


Ways to achieve ‘Abstraction’:

- In general there are two ways to achieve Abstraction:

  ✓Abstract class (0 to 100%)

  ✓Interface (100%)

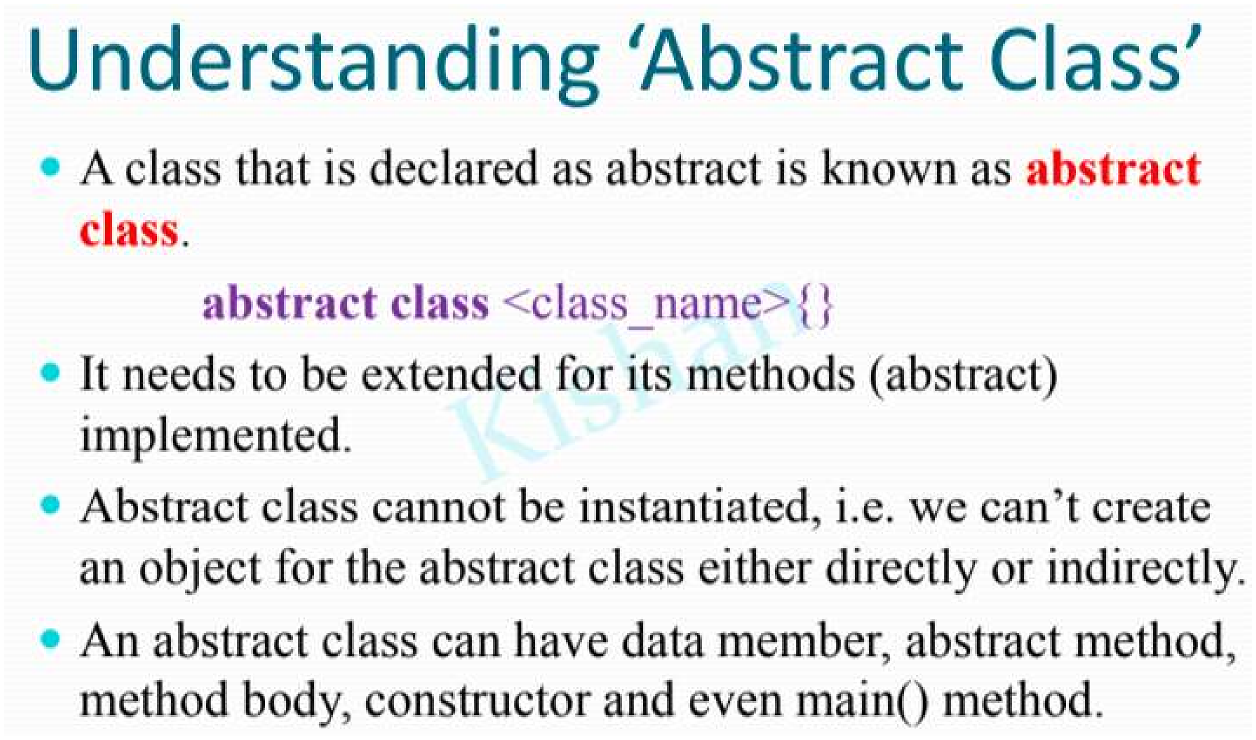


Understanding ‘Abstract Method’

- An abstract method should end with semi colon(;).
- It should not have any method body (or) method implementation.
- An abstract method should be over ridden to provide implementation.
- If we can’t inherit a method that method can’t be an abstract method.

Syntax

```
abstract return_type <method_name>();//no braces{}
```

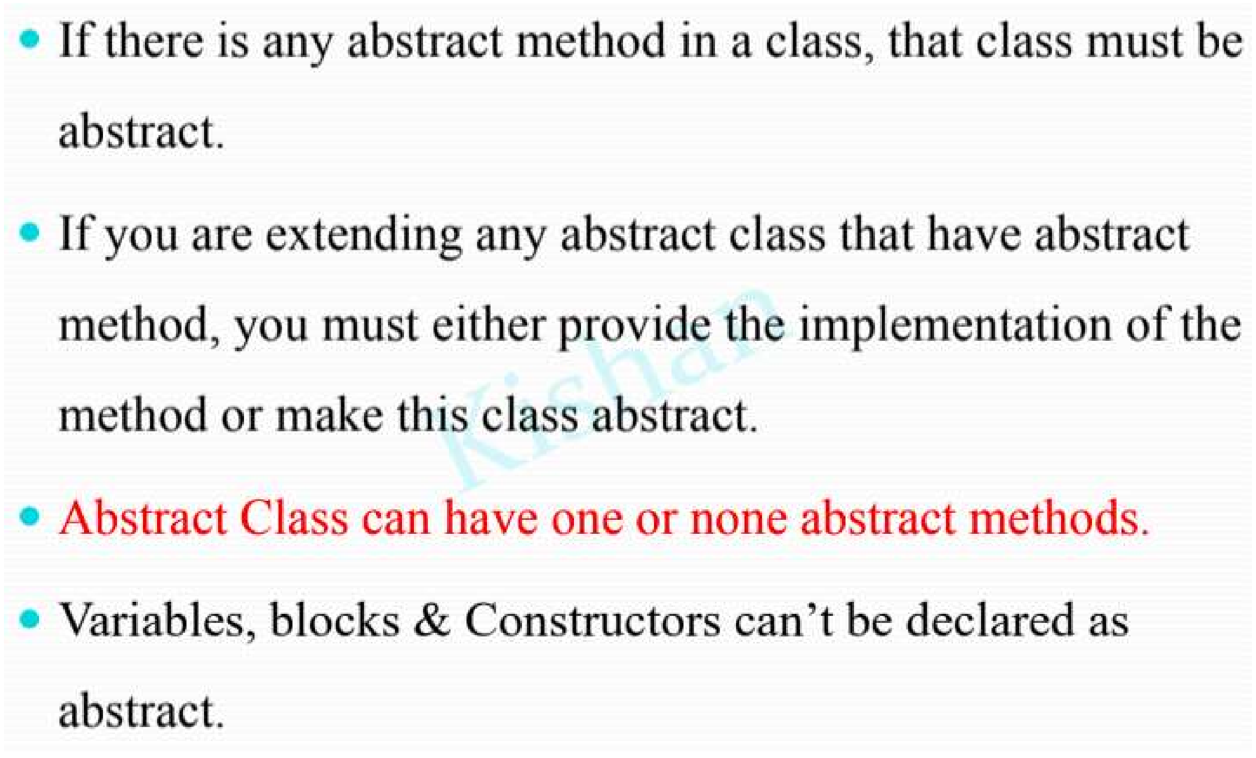


Understanding 'Abstract Class'

- A class that is declared as abstract is known as **abstract class**.

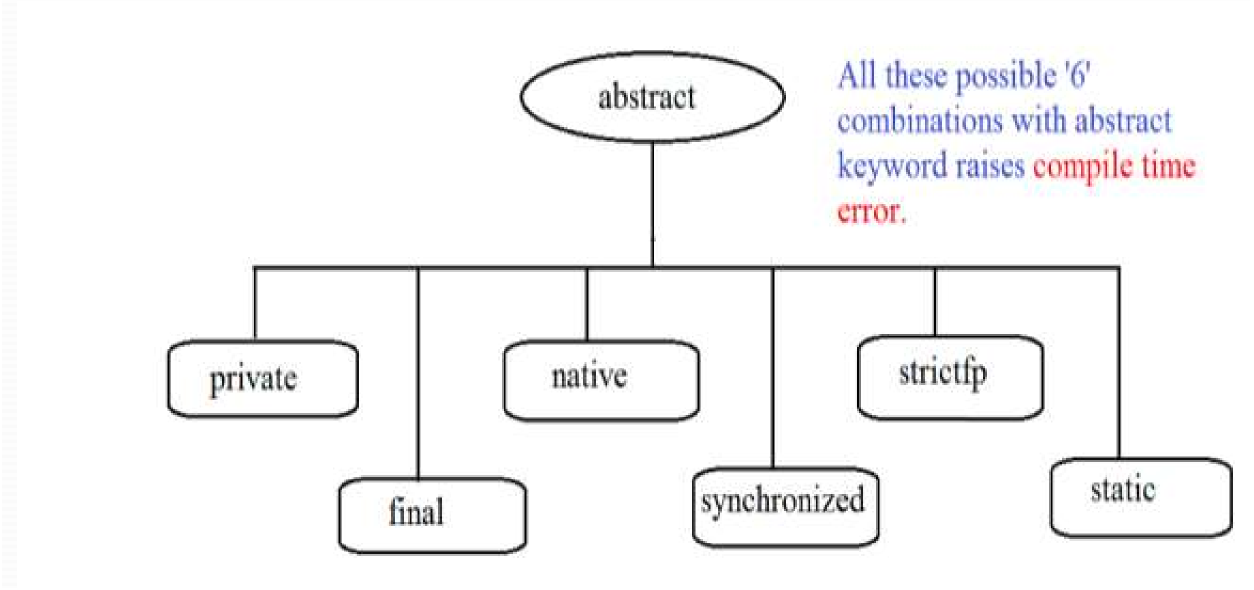
  abstract class <class_name>{}

- It needs to be extended for its methods (abstract) implemented.
- Abstract class cannot be instantiated, i.e. we can't create an object for the abstract class either directly or indirectly.
- An abstract class can have data member, abstract method, method body, constructor and even main() method.



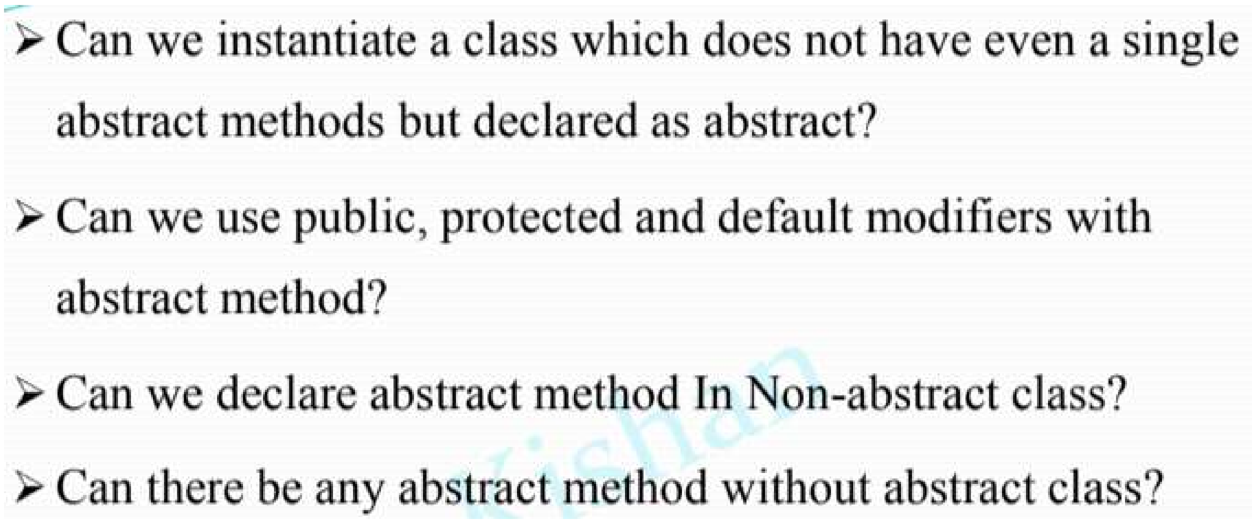
- If there is any abstract method in a class, that class must be abstract.
- If you are extending any abstract class that have abstract method, you must either provide the implementation of the method or make this class abstract.
- Abstract Class can have one or none abstract methods.
- Variables, blocks & Constructors can't be declared as abstract.

Invalid combinations with 'abstract'


abstract
All these possible '6' combinations with abstract keyword raises compile time error.
private
final
native
synchronized
strictfp
static

Questions on Abstract Class

- Can abstract class have constructors?
- Can abstract class be final in Java?
- Can you create instance of abstract class?
- Abstract class must have only abstract methods.(T/F)?
- Can abstract class contains main method in Java?
- Can main method be abstract?
- Is it compulsory for a class which is declared as abstract to have at least one abstract method?
- Can we use "abstract" keyword with constructor?



- Can we instantiate a class which does not have even a single abstract methods but declared as abstract?
- Can we use public, protected and default modifiers with abstract method?
- Can we declare abstract method In Non-abstract class?
- Can there be any abstract method without abstract class?

What is an abstract method?

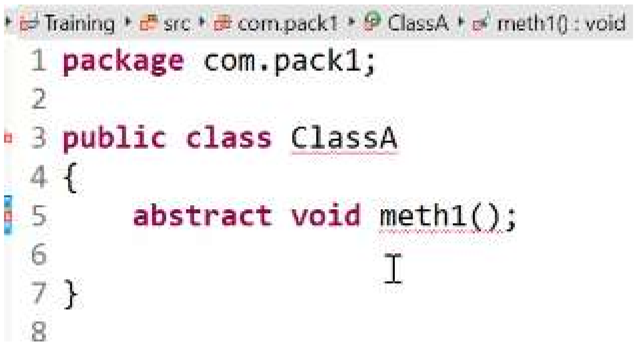
A method which is declared as abstract method which as abstract keyword.

<access modifier>abstract<return type><method name>();

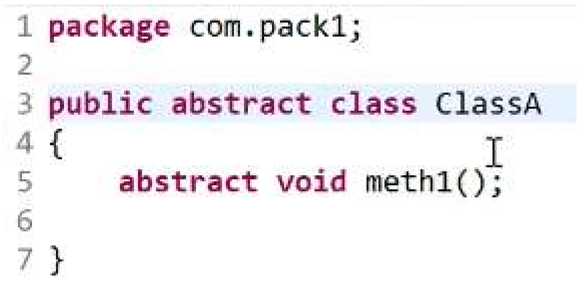
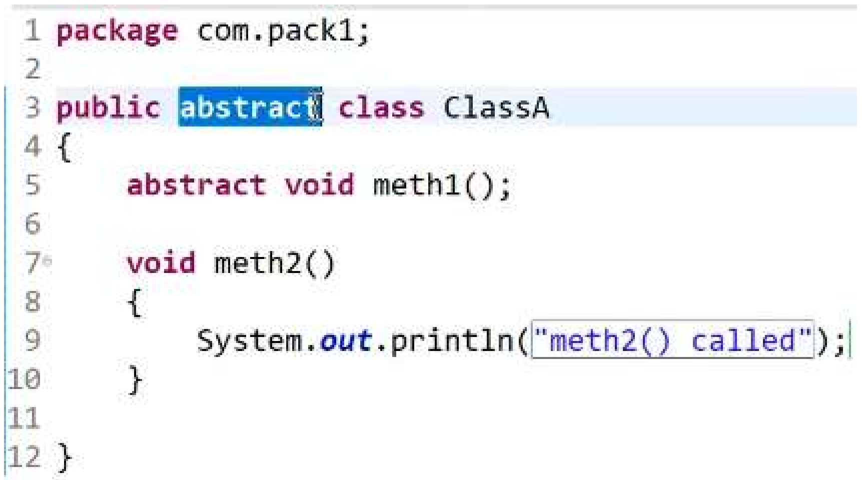
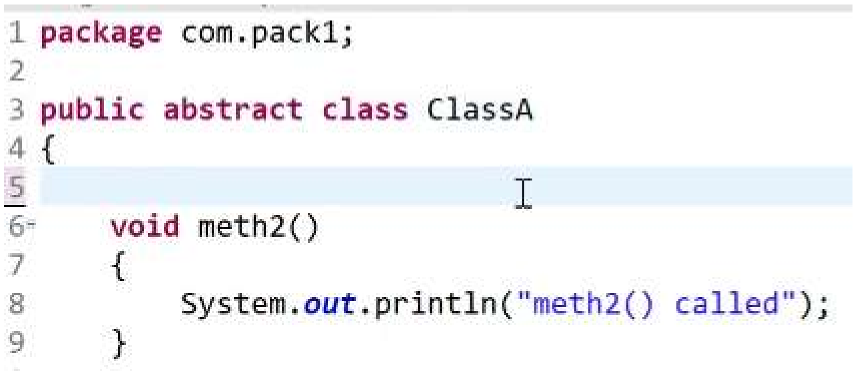
Abstract method does not have any method body.

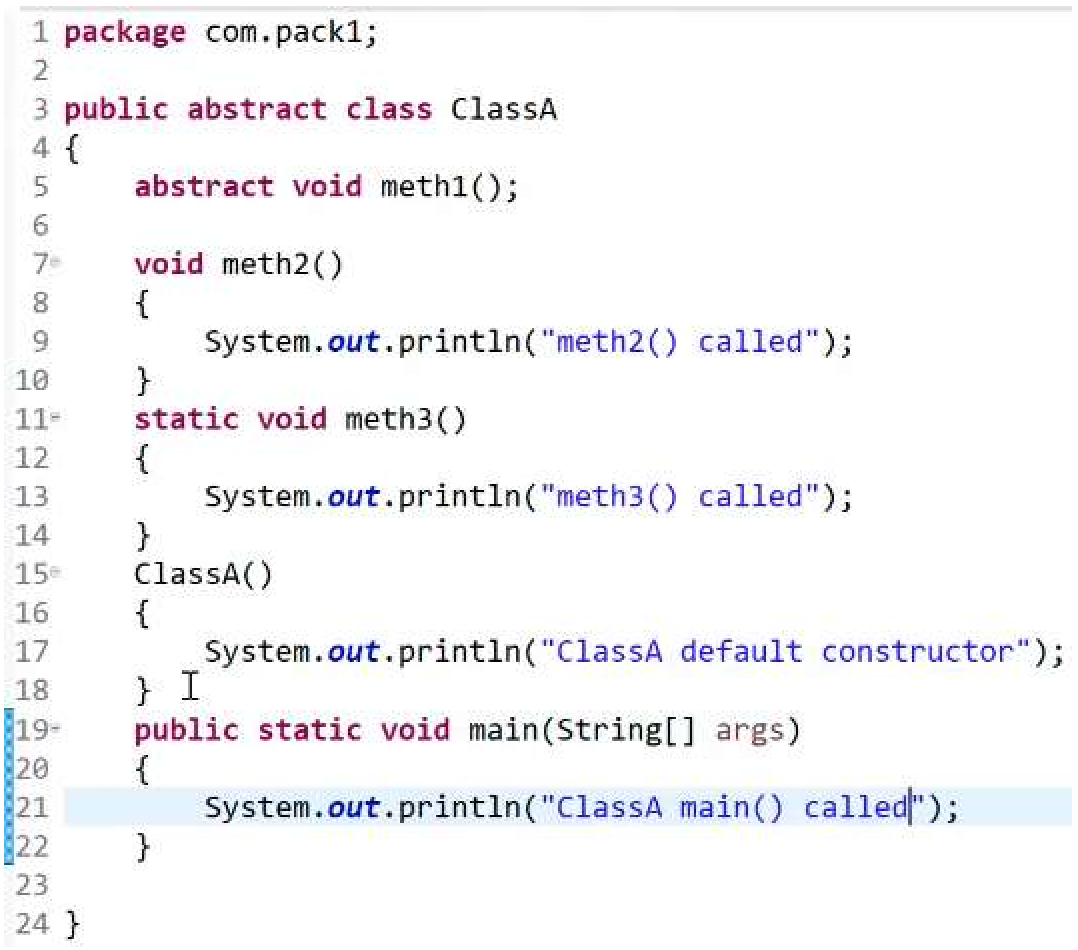
Implementation of an abstract method should be provided in the next class using method-overriding.

100% we need to override an abstract method.



```
package com.pack1;

public class ClassA
{
    abstract void meth1();

}

```


```java
package com.pack1;

public abstract class ClassA
{
    abstract void meth1();

}
```

```java
package com.pack1;

public abstract class ClassA
{
    abstract void meth1();

    void meth2()
    {
        System.out.println("meth2() called");
    }

}
```

```java
package com.pack1;

public abstract class ClassA
{

    void meth2()
    {
        System.out.println("meth2() called");
    }
```

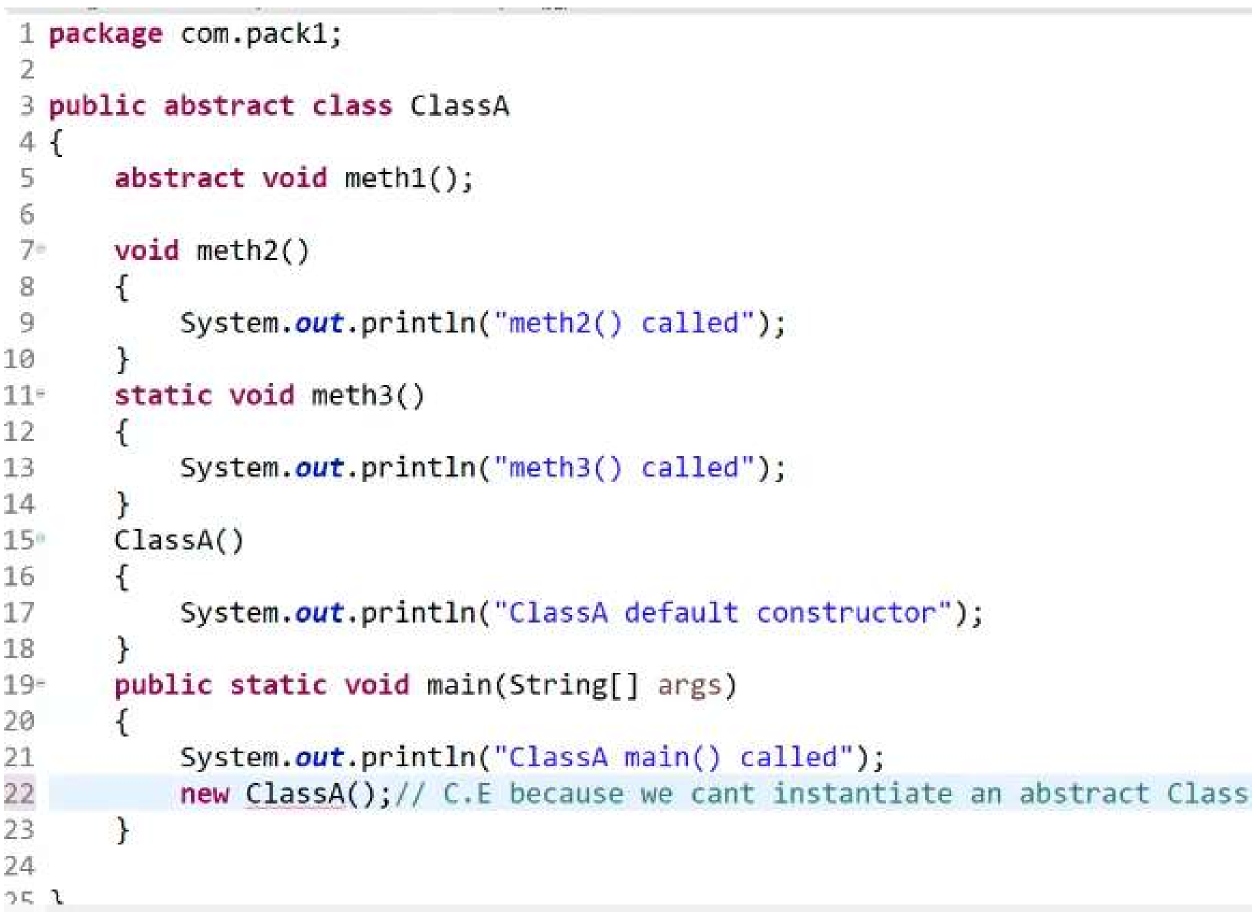


```java
package com.pack1;

public abstract class ClassA
{
    abstract void meth1();

    void meth2()
    {
        System.out.println("meth2() called");
    }
    static void meth3()
    {
        System.out.println("meth3() called");
    }
    ClassA()
    {
        System.out.println("ClassA default constructor");
    }
    public static void main(String[] args)
    {
        System.out.println("ClassA main() called");
    }

}
```

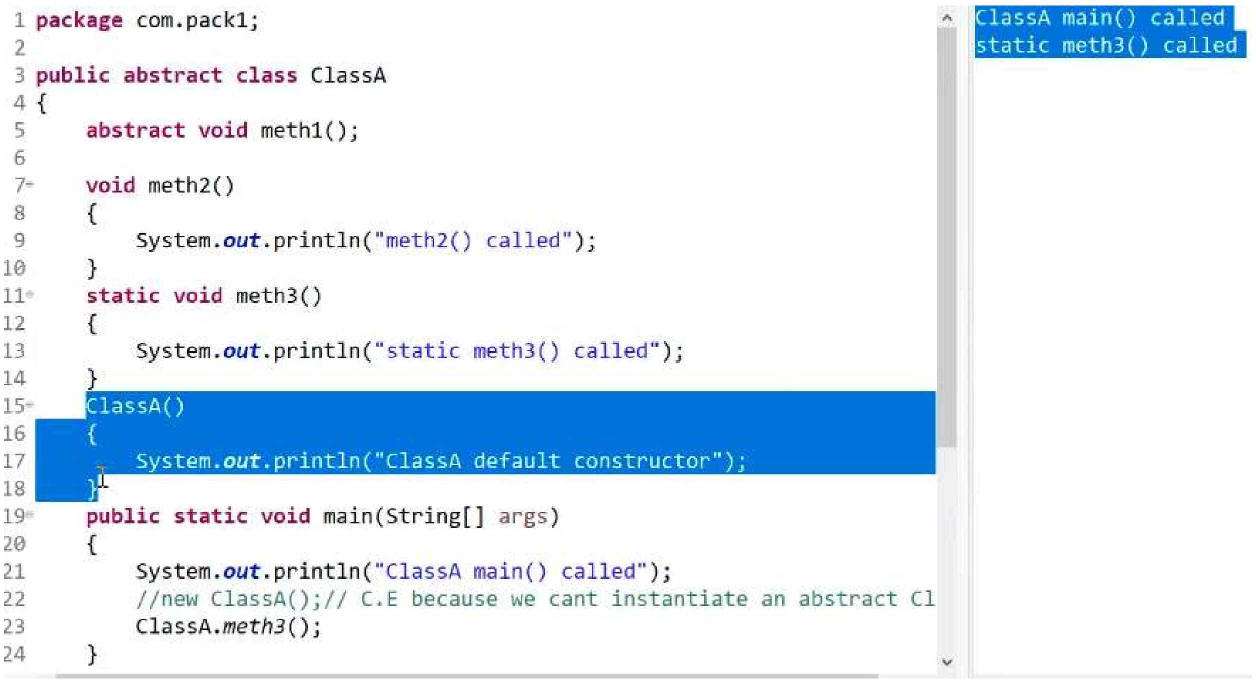
We cannot create an object for an abstract class

If we are writing all normal methods in an abstract class, then what is the use of making the class an abstract class

Abstract class can’t be instantiated

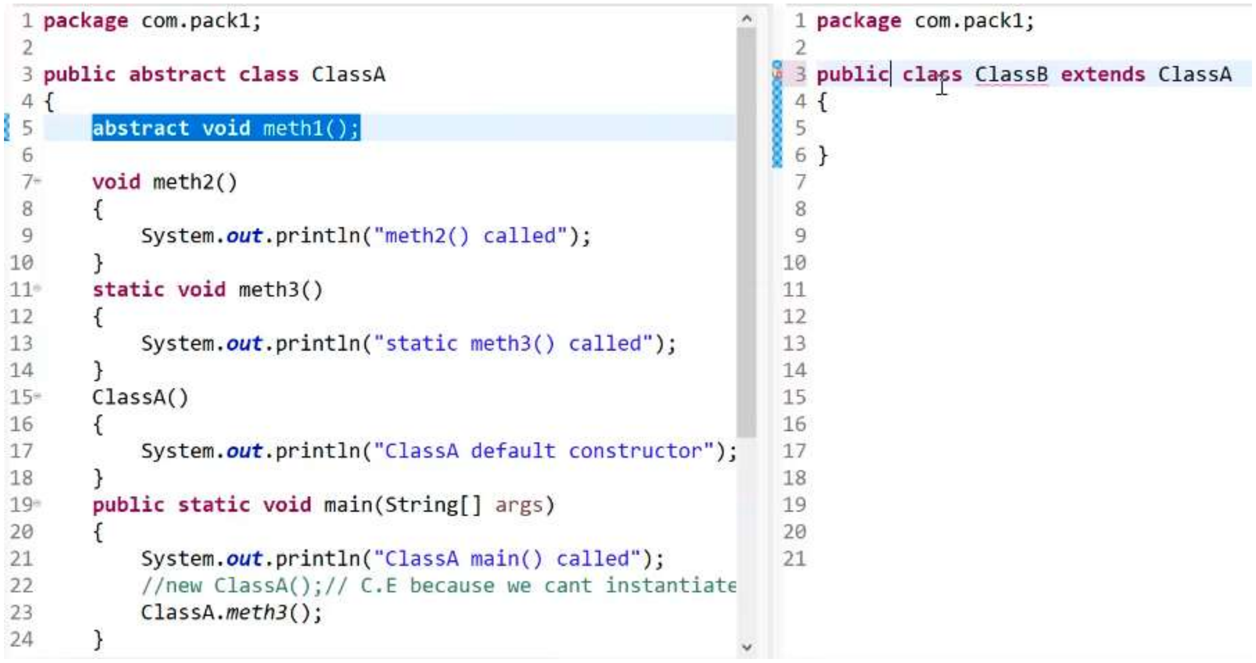


```
package com.pack1;

public abstract class ClassA
{
    abstract void meth1();

    void meth2()
    {
        System.out.println("meth2() called");
    }
    static void meth3()
    {
        System.out.println("meth3() called");
    }
    ClassA()
    {
        System.out.println("ClassA default constructor");
    }
    public static void main(String[] args)
    {
        System.out.println("ClassA main() called");
        new ClassA();// C.E because we cant instantiate an abstract Class
    }

}
```



```
package com.pack1;

public abstract class ClassA
{
    abstract void meth1();

    void meth2()
    {
        System.out.println("meth2() called");
    }
    static void meth3()
    {
        System.out.println("static meth3() called");
    }
    ClassA()
    {
        System.out.println("ClassA default constructor");
    }
    public static void main(String[] args)
    {
        System.out.println("ClassA main() called");
        //new ClassA();// C.E because we cant instantiate an abstract Cl
        ClassA.meth3();
    }
```

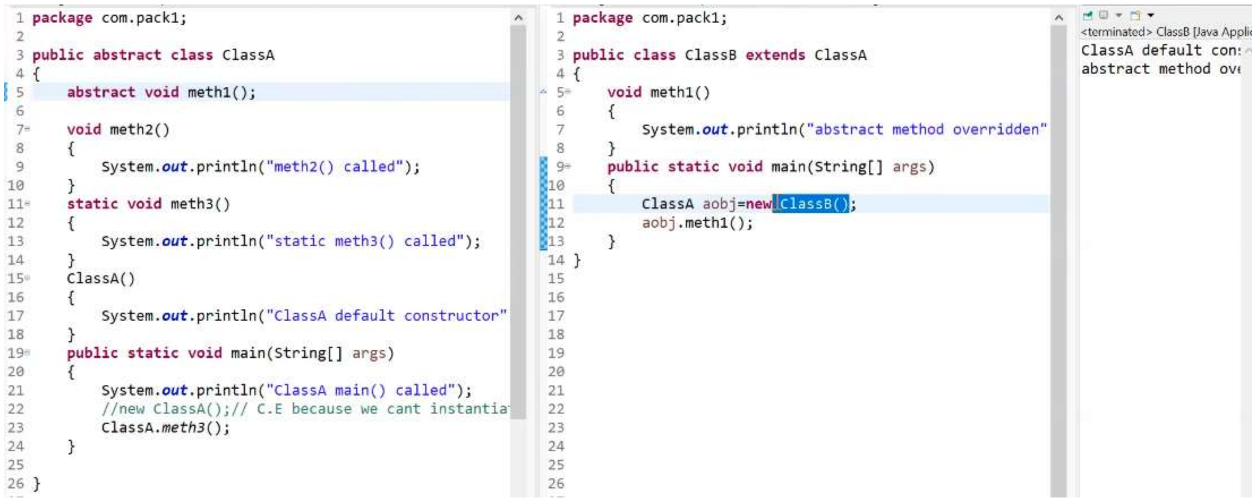
```
ClassA main() called
static meth3() called
```



```java
package com.pack1;

public abstract class ClassA
{
    abstract void meth1();

    void meth2()
    {
        System.out.println("meth2() called");
    }
    static void meth3()
    {
        System.out.println("static meth3() called");
    }
    ClassA()
    {
        System.out.println("ClassA default constructor");
    }
    public static void main(String[] args)
    {
        System.out.println("ClassA main() called");
        //new ClassA();// C.E because we cant instantiate
        ClassA.meth3();
    }
```

```java
package com.pack1;

public class ClassB extends ClassA
{

}
```

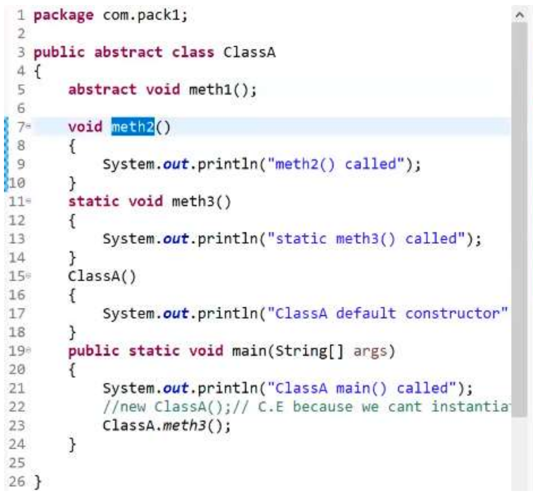
Whenever we are inheriting an abstract class into a normal class if that abstract class has any abstract methods, then 100%, we need to provide implementation(overriding) for all the abstract methods in child class overwise we are getting an error.



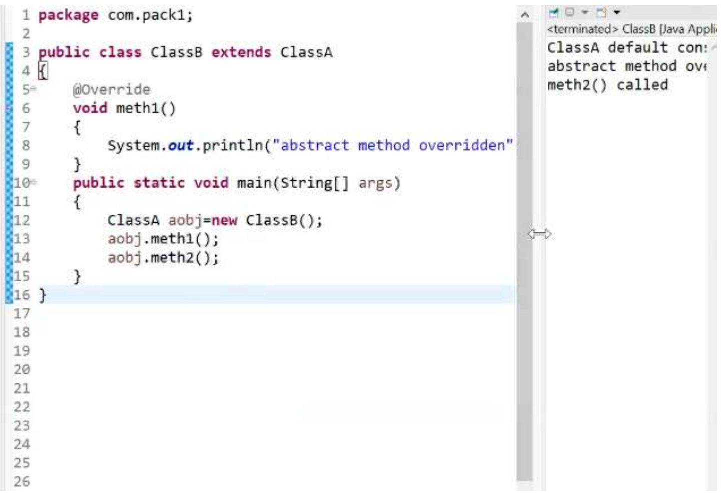
```java
package com.pack1;

public abstract class ClassA
{
    abstract void meth1();

    void meth2()
    {
        System.out.println("meth2() called");
    }
    static void meth3()
    {
        System.out.println("static meth3() called");
    }
    ClassA()
    {
        System.out.println("ClassA default constructor"
    }
    public static void main(String[] args)
    {
        System.out.println("ClassA main() called");
        //new ClassA();// C.E because we cant instantia
        ClassA.meth3();
    }

}
```

```java
package com.pack1;

public class ClassB extends ClassA
{
    void meth1()
    {
        System.out.println("abstract method overridden"
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassB();
        aobj.meth1();
    }
}
```

```
<terminated> ClassB [Java Applic
ClassA default con:
abstract method ove
```

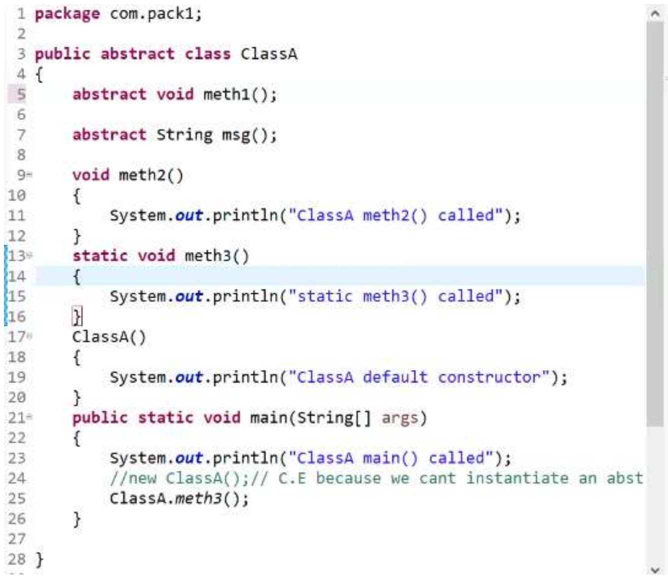


```java
package com.pack1;

public abstract class ClassA
{
    abstract void meth1();

    void meth2()
    {
        System.out.println("meth2() called");
    }
    static void meth3()
    {
        System.out.println("static meth3() called");
    }
    ClassA()
    {
        System.out.println("ClassA default constructor"
    }
    public static void main(String[] args)
    {
        System.out.println("ClassA main() called");
        //new ClassA();// C.E because we cant instantia
        ClassA.meth3();
    }

}
```

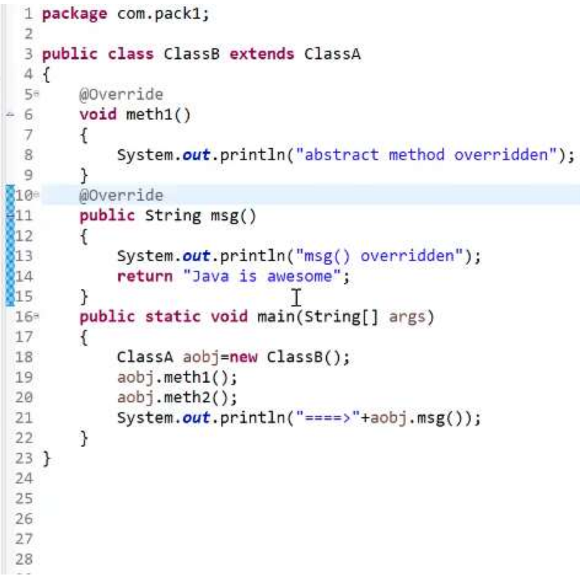


```java
package com.pack1;

public class ClassB extends ClassA
{
    @Override
    void meth1()
    {
        System.out.println("abstract method overridden"
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassB();
        aobj.meth1();
        aobj.meth2();
    }
}
```

```
ClassA default con:
abstract method ove
meth2() called
```



```java
package com.pack1;

public abstract class ClassA
{
    abstract void meth1();

    abstract String msg();

    void meth2()
    {
        System.out.println("ClassA meth2() called");
    }
    static void meth3()
    {
        System.out.println("static meth3() called");
    }
    ClassA()
    {
        System.out.println("ClassA default constructor");
    }
    public static void main(String[] args)
    {
        System.out.println("ClassA main() called");
        //new ClassA();// C.E because we cant instantiate an abst
        ClassA.meth3();
    }

}
```



```java
package com.pack1;

public class ClassB extends ClassA
{
    @Override
    void meth1()
    {
        System.out.println("abstract method overridden");
    }
    @Override
    public String msg()
    {
        System.out.println("msg() overridden");
        return "Java is awesome";
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassB();
        aobj.meth1();
        aobj.meth2();
        System.out.println("====>"+aobj.msg());
    }
}
```