

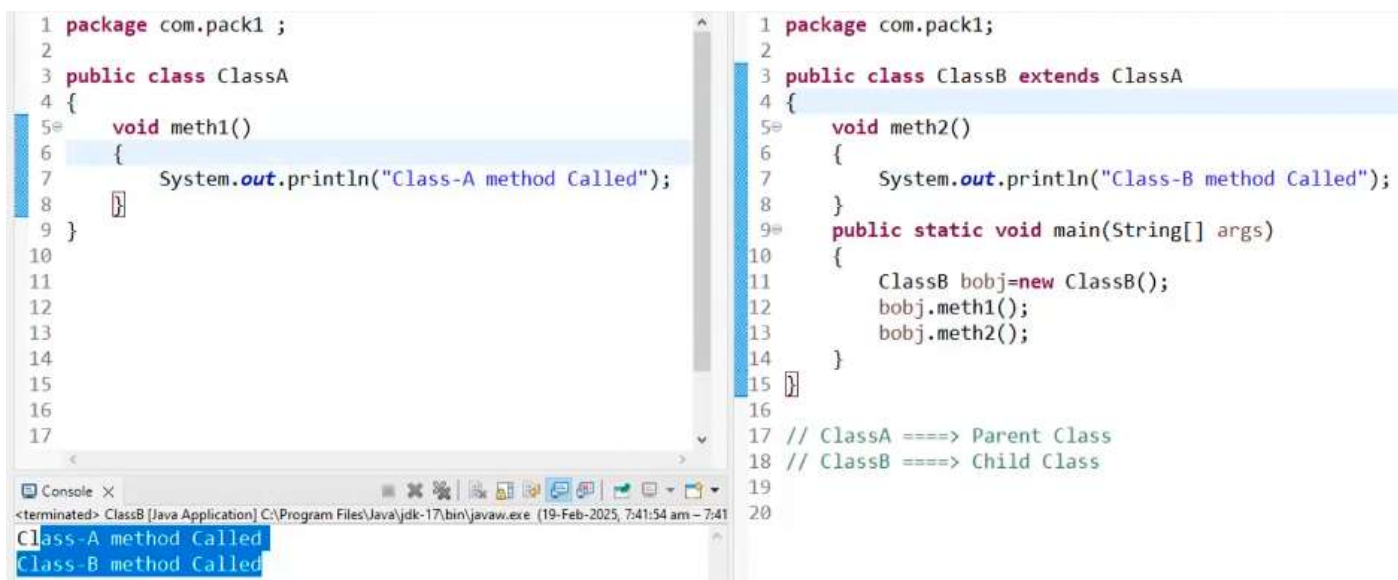
Today our topic is not inheritance and small introduction requires

Inheritance

It means acquiring the properties of one class into another class. Both the classes belong to the same package

Types of inheritance

1. Single inheritance
2. Multi-level inheritance
3. Hierarchical inheritance
4. Hybrid inheritance
5. Multiple inheritance



The screenshot displays a Java IDE with two source files and a console window. The left file, `ClassA.java`, defines a package `com.pack1` and a class `ClassA` with a method `meth1()` that prints "Class-A method Called". The right file, `ClassB.java`, defines the same package and a class `ClassB` that extends `ClassA`. `ClassB` has a method `meth2()` that prints "Class-B method Called" and a `main` method that creates an instance of `ClassB` and calls both `meth1()` and `meth2()`. The console window at the bottom shows the output of running `ClassB`, displaying both "Class-A method Called" and "Class-B method Called", which demonstrates that `ClassB` inherits the `meth1()` method from `ClassA`.

```
1 package com.pack1 ;
2
3 public class ClassA
4 {
5     void meth1()
6     {
7         System.out.println("Class-A method Called");
8     }
9 }
10
11
12
13
14
15
16
17
```

```
1 package com.pack1;
2
3 public class ClassB extends ClassA
4 {
5     void meth2()
6     {
7         System.out.println("Class-B method Called");
8     }
9     public static void main(String[] args)
10    {
11        ClassB bobj=new ClassB();
12        bobj.meth1();
13        bobj.meth2();
14    }
15 }
16
17 // ClassA ==> Parent Class
18 // ClassB ==> Child Class
19
20
```

Console X
<terminated> ClassB [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (19-Feb-2025, 7:41:54 am - 7:41:54 am)
Class-A method Called
Class-B method Called

Extends are different—both classes belong to the same package.

Imports are different and used

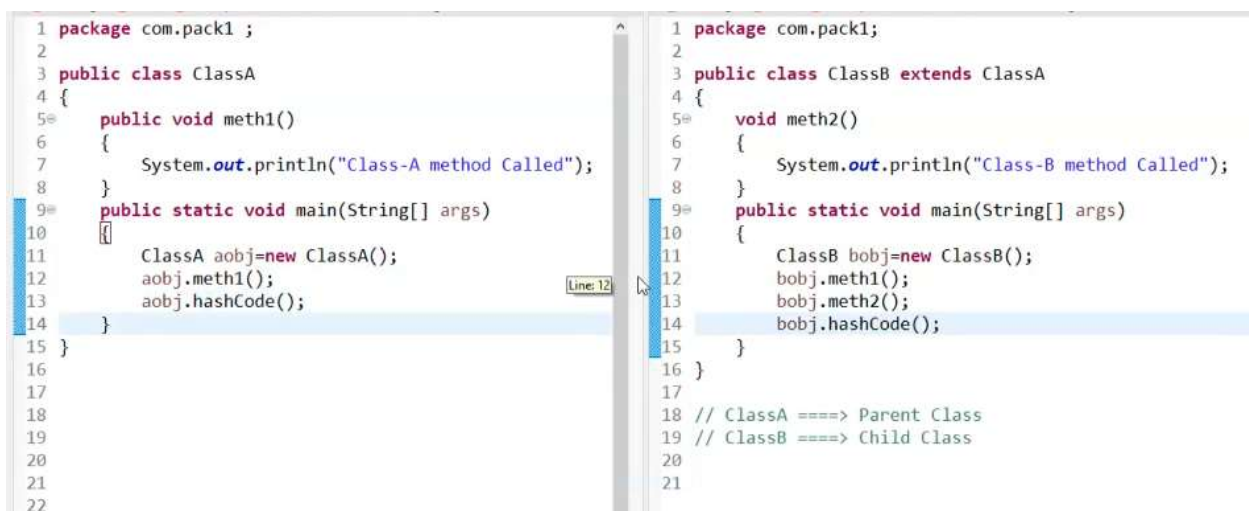
If both the classes are present in different packages

For every java class it may be predefined, or user defined one class by default act as a parent class. As a matter of fact, that class is the first class in the java hierarchy. Every class acts as a child's class for that class and that class name is object class.

In object class there are 11 methods

All these 11 methods can be accessed by every other class

Object class is the parent class for all the classes with help of every other class object we can call, we can access the methods which are present in object class.



```
1 package com.pack1 ;
2
3 public class ClassA
4 {
5     public void meth1()
6     {
7         System.out.println("Class-A method Called");
8     }
9     public static void main(String[] args)
10    {
11        ClassA aobj=new ClassA();
12        aobj.meth1();
13        aobj.hashCode();
14    }
15 }
16
17
18
19
20
21
22
```

```
1 package com.pack1;
2
3 public class ClassB extends ClassA
4 {
5     void meth2()
6     {
7         System.out.println("Class-B method Called");
8     }
9     public static void main(String[] args)
10    {
11        ClassB bobj=new ClassB();
12        bobj.meth1();
13        bobj.meth2();
14        bobj.hashCode();
15    }
16 }
17
18 // ClassA =====> Parent Class
19 // ClassB =====> Child Class
20
21
```

```
1 package com.pack1;
2
3 public class ClassA extends Object
4 {
5     public void meth1()
6     {
7         System.out.println("Class-A method Called");
8     }
9     public static void main(String[] args)
10    {
11        ClassA aobj=new ClassA();
12        aobj.meth1();
13        aobj.hashCode();
14        aobj.nextInt();
15    }
16 }
17
18
19
20
21
22
```

```
1 package com.pack1;
2
3 public class ClassB extends ClassA
4 {
5     void meth2()
6     {
7         System.out.println("Class-B method Called");
8     }
9     public static void main(String[] args)
10    {
11        ClassB bobj=new ClassB();
12        bobj.meth1();
13        bobj.meth2();
14        bobj.hashCode();
15    }
16 }
17
18 // ClassA ==> Parent Class
19 // ClassB ==> Child Class
20
21
```

Above line error because it is not present in object class

Package

It consists of similar types of classes, interfaces and Enum

In java there are 2 types of packages

1.Predefined packages

2.User defined packages

Nearly there are 5000 predefined packages are there

In all these 5000 predefined packages one package by default imported into every java program. The name the package is **java.lang package**

Java.lang package consist of String, System, integer, thread, runnable.

Object class is present in the java.lang package

Understanding object class

Object- class is the first class in the java class hierarchy.

Object class by default acts as a parent class for every java class (predefined or user defined classes).

As object class is parent class for all the classes with the help of every other class object, we can call the methods which are present in object class.

In object-class there are 11 methods present.

Object class is present in java.lang package which is by default imported into every java program.

Understanding Object class

- Object class, is present in the java.lang package, is the first class in the java class hierarchy.
- Every class either Predefined or User Defined is the sub class for Object class.
- Object class has "11" important methods, As this is the super class for all the classes we can use (Override) those methods in all the class.
- In those "11" methods there are "5" final methods for which we can't provide override.

Method Name	Description
public int hashCode()	Returns a hash code value for the object
protected void finalize() throws Throwable	Called by the garbage collector
public boolean equals(Object obj)	Used for comparing two Objects
protected Object clone() throws CloneNotSupportedException	Creates exact copy of the object
public String toString()	Returns a string representation of the object.
public final Class getClass()	Returns present class reference
public final void notify()	All these methods are used in java multithreading, which plays a crucial role in synchronization.
public final void notifyAll()	
public final void wait()	
public final void wait(long timeout)	
public final void wait(long timeout, int nanos)	

hashCode()

- The return type for hashCode() is int.
- It is going to return a unique identification number for your Class Object.
- It is not going to return the address locations of the object.


```
1 package com.pack1 ;
2
3 public class ClassA
4 {
5     public static void main(String[] args)
6     {
7         ClassA aobj1=new ClassA(); // 1st Object
8         ClassA aobj2=new ClassA(); // 1st Object
9
10        System.out.println("aobj1 hashCode() value : "+aobj1.hashCode());
11        System.out.println("aobj2 hashCode() value : "+aobj2.hashCode());
12
13    }
14 }
15
16 /*
17  hashCode():
18  -----
19  1) The return type for hashCode() is int.
20  2) It is going to return a Unique identification number for your Class Object
21  3) It is not going to return the address locations of the Object.
22
23  */
```

aobj1 hashCode() value : 1072408673
aobj2 hashCode() value : 885284298

To see the hashCode

press and hold the ctrl button and click on hashCode

```
1 package com.pack1 ;
2
3 public class ClassA
4 {
5     public static void main(String[] args)
6     {
7         ClassA aobj1=new ClassA(); // 1st Object
8         ClassA aobj2=new ClassA(); // 1st Object
9
10        System.out.println("aobj1 hashCode() value : "+aobj1.hashCode());
11        System.out.println("aobj2 hashCode() value : "+aobj2.
12
13    }
14 }
15
16 /*
17  hashCode():
18  -----
19  1) The return type for hashCode() is int.
20  2) It is going to return a Unique identification number for your Class Object
21  3) It is not going to return the address locations of the Object.
22
23  */
```

Open Declaration
Open Implementation
Open Call Hierarchy

```

26 package java.lang;
27
28 import jdk.internal.vm.annotation.IntrinsicCandidate;
29
30 /**
31  * Class {@code Object} is the root of the class hierarchy.
32  * Every class has {@code Object} as a superclass. All objects,
33  * including arrays, implement the methods of this class.
34  *
35  * @see    java.lang.Class
36  * @since  1.0
37  */
38 public class Object {
39
40     /**
41      * Constructs a new object.
42      */
43     @IntrinsicCandidate
44     public Object() {}
45
46     /**
47      * Returns the runtime class of this {@code Object}. The returned
48      * {@code Class} object is the object that is locked by {@code
49      * static synchronized} methods of the represented class.

```

equals()

- The return type for equal() is Boolean(either true or false).
- It is going to compare the address locations of the objects.
- If both the objects are present in the same address locations, we will be getting true otherwise false.

```

4 {
5     public static void main(String[] args)
6     {
7         ClassA aobj1=new ClassA(); // 1st Object
8         ClassA aobj2=new ClassA(); // 1st Object
9
10        System.out.println("aobj1 hashCode() value : "+aobj1.hashCode());
11        System.out.println("aobj2 hashCode() value : "+aobj2.hashCode());
12
13        System.out.println("-----");
14
15        System.out.println(aobj1.equals(aobj2)); I
16
17    }
18 }
19
20 /*
21 hashCode():
22 -----
23 1) The return type for hashCode() is int.
24 2) It is going to return a Unique identification number for your Class Object
25 3) It is not going to return the address locations of the Object.
26
27 equals():
28 -----

```

aobj1 hashCode() value : 1072408673
aobj2 hashCode() value : 885284298

false

```

4 {
5     public static void main(String[] args)
6     {
7         ClassA aobj1=new ClassA(); // 1st Object
8         ClassA aobj2=new ClassA(); // 1st Object
9
10        System.out.println("aobj1 hashCode() value : "+aobj1.hashCode());
11        System.out.println("aobj2 hashCode() value : "+aobj2.hashCode());
12
13        System.out.println("-----");
14
15        System.out.println(aobj1.equals(aobj2)); // f
16        System.out.println(aobj1.equals(new ClassA()));//f
17        System.out.println(aobj2.equals(aobj2));//t
18        System.out.println(new ClassA().equals(new ClassA()));
19
20        I
21    }
22 }
23
24 /*
25 hashCode():
26 -----
27 1) The return type for hashCode() is int.

```

<terminated> ClassA [Java Application] C:\Program Files\Java\jdk-17\bin
aobj1 hashCode() value : 1072408673
aobj2 hashCode() value : 885284298

false
false
true
false

In the above case we have created 5 objects, how we know?

Every **new** key word is going to create new object


```

1 package com.pack1 ;
2
3 public class ClassA
4 {
5     public static void main(String[] args)
6     {
7         ClassA aobj1=new ClassA(); // 1st Object
8         ClassA aobj2=new ClassA(); // 1st Object
9
10        System.out.println("aobj1 hashCode() value : "+aobj1.hashCode());
11        System.out.println("aobj2 hashCode() value : "+aobj2.hashCode());
12
13        System.out.println("-----");
14
15        System.out.println(aobj1.equals(aobj2)); // f
16        System.out.println(aobj1.equals(new ClassA())); //f
17        System.out.println(aobj2.equals(aobj2)); //t
18        System.out.println(new ClassA().equals(new ClassA())); //f
19    }
20 }
21

```

```

aobj1 hashCode() value : 1072408673
aobj2 hashCode() value : 885284298
-----
false
false
true
false

```