	default	private	protected	public
Same Class	Yes	Yes	Yes	Yes
Same package subclass	Yes	No	Yes	Yes
Same package non-subclass	Yes	No	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

—----------

-

JAVA.LANG Package

—----------

-

- **Whether it is a simple or complex program, The most common class and interfaces are Encapsulated or Grouped into separate Packages called java.lang Package.**

—----------

-

Ex:

```
//java.lang package will be automatically imported.
// Each and Every class is a subclass of Object in Java.
// And That Object is Present in the Java.lang Package.
Class Test
{
	// String Class is in Java.Lang package
	public static void main(String args[])
	{
		//System Class is in Java.Lang package
		// "Hello world" String Object is in Java.Lang package
		System.out.println("Hello world");
	}
}
```

—--
-

java.lang.Object

-

Everyclass in java is a subclass of Object.
Object has Root for java class.

—----------
-

Why is Object only called as root for all Classes in Java ?

—----------
-

- For any Java Classes, like student, string or stringBuffer class the most commonly required Methods are defined in a separate class called Object.
- hashCode(), Equal() is some methods on Objects(parent) and applicable for all Java Classes.
- All Classes can Access Object Methods (hashCode(),equals(),getClass() ... etc.)
- String Object methods like concat only used for String Objects not for StringBuffer, StringBuilder, Student Objects(like : Concat), But Objects Methods Are accessible for all Objects like [HashCode()] is Object Methods can access by all Objects.

-

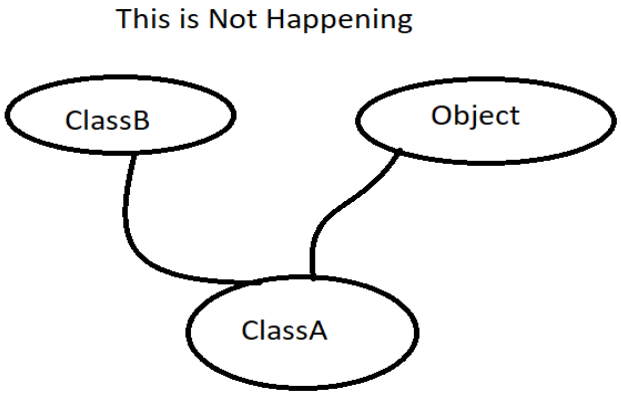
- The Most Commonly required methods for every Java class(Where it is predefined class or customized class) are defined in separate class which is nothing but Object Class
- Every class Of java is a child class of Object either directly or indirectly so Object class methods Available to every java class.
- Hence the Object class is considered as the root of all java classes.

-Ex:

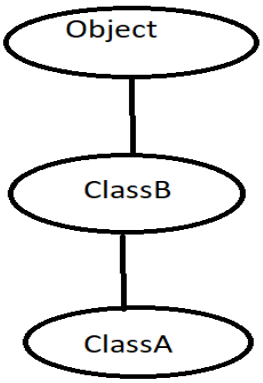
ClassA

|

ClassB extends classA (ClassB is not Multiple inheritance (Not Supported In JAVA)



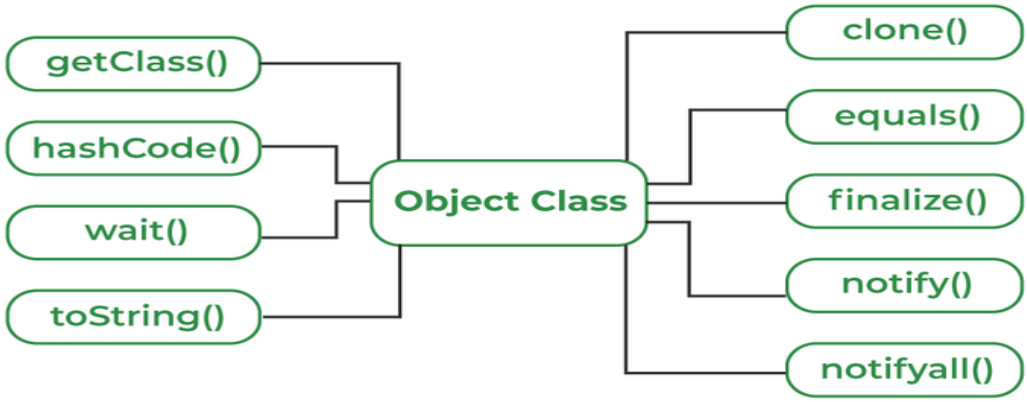
Correct Syntax: Object indirect subclass



If your Class is Not Extending any class then our class is a Direct subclass of Object else It is an indirect subclass.

Directly or indirectly java won't support multiple Inheritance.

<u>The Object Class has 11 Methods.</u>



public final Class getClass()	returns the Class class object of this object. The Class class can further be used to get the metadata of this class.
public int hashCode()	returns the hashcode number for this object.
public boolean equals(Object obj)	compares the given object to this object.
protected Object clone() throws CloneNotSupportedException	creates and returns the exact copy (clone) of this object.
public String toString()	returns the string representation of this object.
public final void notify()	wakes up a single thread, waiting on this object's monitor.

public final void notifyAll()	wakes up all the threads, waiting on this object's monitor.
public final void wait(long timeout)throws InterruptedException	causes the current thread to wait for the specified milliseconds, until another thread notifies (invokes notify() or notifyAll() method).
public final void wait(long timeout,int nanos)throws InterruptedException	causes the current thread to wait for the specified milliseconds and nanoseconds, until another thread notifies (invokes notify() or notifyAll() method).
public final void wait()throws InterruptedException	causes the current thread to wait, until another thread notifies (invokes notify() or notifyAll() method).
protected void finalize()throws Throwable	is invoked by the garbage collector before the object is being garbage collected.

```
class Test
{
    public static void main(String args[]) throws Exception
    {
        Class c = Class.forName("java.lang.Object");
        Method[] m = c.getDeclaredMethods();
        for(Method m1 : m)
            System.out.println(m1.getName());
    }}
```

toString:

- To string form will be formed.
- If our class doesn't contain the toString method then Object class toString() Executed.
- In all collection classes in String class StringBuffer and stringBuilder classes to string method is overridden for meaningful String Representation hence it is Highly recommended to override toString method in our class also.

HashCode():

- For Every Object a unique number generated by JVM which is nothing but hashcode, hashcode don't represent address of Object, JVM will use hashcode while saving the object into hashing related data structures like hashtable, hashmap etc. the main advantage of saving object based on hashcode is search operation will become easy, the most powerful search algorithm is hashing.
- Public native int hashcode();
- Hashcode of Object has generated by address of Object link(address*2*7.5 and something.) It doesn't mean the hashcode represents the address.
- We can Override the hashcode() based on our requirement.

  Improper Hashcode:
  - Return 100; all Objects as 100 hashcode().

  Proper HashCode():
  - Return "roll number" , All Objects as different hashcode().

toString VS HashCode

- If we are giving chance to Object class to string () it will internally call hashcode()
- If we are overriding toString() then our toString () may not call hashcode().

```java
public class Demo
{
        int i;
        Demo(int i)
        {
            this.i = i;
        }
        public String toString()
        {
            return i+"";
        }
        public int hashCode()
        {
            return i;
        }

    public static void main(String args[])
    {
        Demo d = new Demo( i:10);
        Demo d1 = new Demo( i:100);
        System.out.println( x:d); //10
        System.out.println( x:d1);//100
        //System.out.println(d); //Demo@a (Hexa Form)
        //System.out.println(d1);//Demo@64
//          System.out.println(d); // Output - Demo@568db2f2
    }
}
```

Equals() :-

- We can use equal() to check equality of two Object (obj1.equals(obj2))
- If your class doesn't contain equals() then the Object class equals() will be executed.

```java
public class Demo
{
    String name ;
    int roll;
    Demo(String name,int roll)
    {
        this.name = name;
        this.roll = roll;
    }
    public static void main(String args[])
    {
        Demo d = new Demo( name: "chandu", roll: 10);
        Demo d1 = new Demo( name: "Madhu", roll: 100);
        Demo d2 = new Demo( name: "chandu", roll: 10);
        Demo d3 = d;

       System.out.println( x: d.equals( obj: d3));
       System.out.println( x: d.equals( obj: d1));
       System.out.println( x: d.equals( obj: d2));
    }
}
```

- In the above example Object class Equals() got executed which is meant for reference Comparison(Address Comparison)
- If 2 references point to the same Object then it returns True.
- Based on Our requirement we can Override equals() for Content Comparison.
- While overriding equals()for content comparison we have to take care following
- What is the meaning of equality (whether we have to check only names or roll)
- If we are passing different types of Object our equals method should not raise classCastException we have to handle to return false.
- If we are passing null argument the our equals() should not raise NullPointerException we have to handle NullPointerException to return False;

Default of equals() is reference comparison we can override for content comparison

Example:

—----------

```
String s= new String("Chandu");
String s1= new String("Chandu");

s.equals(s1); //true (Content)
S == s1     //False
```

Example:

—----------

```
StringBuffer s= new StringBuffer("Chandu");
StringBuffer s1= new StringBuffer("Chandu");

s.equals(s1); //false (reference)
S == s1     //False
```

getClass():

- We can use getClass() to get runtime class definition of an Object
- Public final Class getClass()
- By this class Class Object we can access class level properties full Qualified name of the class, methods information, COnstructor Information.

```java
import java.lang.reflect.*;
class Demo
{
    public static void main(String args[])
    {
        Object o = new String( original: "Chandu");
//          Object o1 = new Integer(10);
        Class c = o.getClass();
        System.out.println( x: c);
        Method[] m = c.getDeclaredMethods();
        for( Method x : m)
            System.out.println( x: x.getName());
    }
}
```

Example2:

To display vendor specific connection interface implemented class name

—----------

Connection con= driverManger.GetConnection();

S.o.p(con.GetClass().GetName());

—----------

- After loading every .class file JVM will create an Object type Java.lang.class in the heap Area.
- Programmers can Use this class Object to get Class Level information.
- We can use Getclass method very frequently in reflections.

Finalize();

- Finalize before Just destroying an Object Garbage collector calls finalize () to perform cleanup activities once finalize complete the garbage collector destroys Object.

wait(); notify(); notifyall();

Wecan use these methods for interthread communication , the thread which is executing updation ,it is responsible to call wait(), then immediately the thread enters into waiting state, the thread which is responsible for updation,After performing updation the bread can call Notify method the waiting thread can get notification and continue its execution with those updates.