The Object Class and its Methods: -

Object Class in Java:-

Object Class in Java is the topmost class among all the classes in Java. We can also say that the Object class in Java is the parent class for all the classes. It means that all the classes in Java are derived classes and their base class is the Object class. **Object** class is present in **java.lang** package.

All the classes directly or indirectly inherit from the Object class in Java.

Ex:-public class Object{

//Object class body

/*

1. toString()
2. hashCode()
3. equals(Object o)
4. clone()
5. finalize()
6. getClass()
7. wait()
8. wait(long timeout)
9. wait(long timeout,int nanos)
10. notify()
11. notifyAll()

*/

}

/*Consider the example-

class Vehicle{
//body of class Vehicle
}

Here we can see that class Vehicle is not inheriting any class, but it inherits the Object class. It is an example of the direct inheritance of object class.

Now, look at this example-

class Vehicle{
//body of class Vehicle
}
class Car extends Vehicle{
//body of class Car
}

Here we can see that the Car class directly inherits the Vehicle class. The extends keyword is used to establish inheritance between two classes. But we have seen in the above example that if we define a class without inheriting it from some other class, it directly inherits the Object class. Since Car inherits the Vehicle and the Vehicle inherits the Object class, Car indirectly inherits the Object class. */

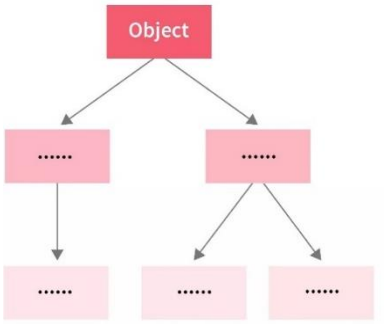

Fig.1.Object Class

Methods of Object Class in Java

The Object class is parent class for all the classes so that the methods are by default available to every java class automatically.

There are various methods of the Object class. They are inherited by other classes.

//Java.lang.Object

12.toString()
13.hashCode()
14.equals(Object o)
15.clone()
16.finalize()
17.getClass()
18.wait()
19.wait(long timeout)
20.wait(long timeout,int nanos)
21.notify()
22.notifyAll()

1.toString():-

This method is used to return a string representation of the object. If we try to print the object of any class, then the JVM internally invokes the toString() method. Override the toString() in an inherited class as it'll aid in a better string representation of the object. If we do not override it, it will give the *hexadecimal representation* of the object.

Syntax:-public String toString();

/*Object Class Method Implementation:-

```
public String toString()
{
return getclass().getName()+"@"+Integer.toHexString(hashcode());
}
```

How Object Class is inherited in sub class is like

```
public String toString()
{
Return super.toString();
}*/
```

Example Program 1:-

```
package objectclass;
public class toStringClass {
   public static void main(String args[])
   {
      toStringClass s=new toStringClass();
      System.out.println(s);
      System.out.println(s.toString());
   }
}
```

O/p:-

objectclass.toStringClass@19e0bfd

objectclass.toStringClass@19e0bfd

Example Program 2(Override toString Method):-

```
package objectclass;

public class toStringClass {

    public String toString()

    {

    return "hello";

    }

   public static void main(String args[])

    {

        toStringClass s=new toStringClass();

        System.out.println(s);

        System.out.println(s.toString());

    }

}
```

O/P:-

hello

hello

2.hashCode():-

A unique id that is assigned to an object is known as hashCode of object. this method is a Native method. A native method allows us to use code from another language like C, C++ in Java. In Java, we cannot find the memory address of objects, so the hashCode() method is written in C or C++ which fetches the object's address in Java.

Syntax:-public int hashCode();

/*How Object Class is inherited in sub class is like

```
Public int hashCode()
{
return super.hashCode();
}*/
```

Example Program1:-

```
package objectclass;
public class hashCodeClass {
public static void main(String args[])
    {
        hashCodeClass s=new hashCodeClass();
        System.out.println(s.hashCode());
    }
}
```

O/P:- 27134973

Example Program 2(Override hashCode( ) Method):-

```
package objectclass;
public class hashCodeClass {
@Override
    public int hashCode()
    {
    return 123;
    }
  public static void main(String args[])
    {
        hashCodeClass s=new hashCodeClass();
```

```
        System.out.println(s.hashCode());
    }
}
```

O/P:-123

3.equals( ):-

We can use this method to check the equality of an object. This method compares two objects and returns whether they are equal or not.

Syntax:-public Boolean equals(Object o);

/*Object Class Method Implementation:-

```
public Boolean equals(object obj)
{
return (this==obj);
}*/
```

Example Program1:-

```
package objectclass;
public class EqualsMethod {
   public static void main(String args[])
   {
        EqualsMethod s=new EqualsMethod();
        EqualsMethod s1=new EqualsMethod();
         System.out.println(s.equals(s1));
   }
}
```

o/p:-

false

4.finalize():-

Finalize() is the method of Object class. This method is called just before an object is garbage collected. finalize() method overrides to dispose system resources, perform clean-up activities and minimize memory leaks.

Syntax

protected void finalize() **throws** Throwable

Example Program1:-

```
public class JavafinalizeExample1 {
    public static void main(String[] args)
    {
       JavafinalizeExample1 obj = new JavafinalizeExample1();
       System.out.println(obj.hashCode());
       obj = null;
       // calling garbage collector
       System.gc();
       System.out.println("end of garbage collection");
    }
    @Override
    protected void finalize()
    {
       System.out.println("finalize method called");
    }
        }
```

5.getClass():-

It returns the class object of "this" object and is used to get the actual runtime class of the object. It can also be used to get metadata of this class. The returned Class object is the object that is locked by static synchronized methods of the represented class. As it is final so we don't override it.

Syntax:-public final Class getClass();

```
public class Test {
    public static void main(String[] args)
    {
        Object obj = new String("Anusha");
        Class c = obj.getClass();
        System.out.println("Class of Object obj is : "
                           + c.getName());
    }
}
```

o/p:-
Class of Object obj is : java.lang.String

6.Clone():-

Creates and returns a copy of this object.

Syntax:-

Protected object clone() throws CloneNotSuppotedException

Example Program:-

```
package objectclass;
public class CloneMethod implements Cloneable{
   int a1;
   String s1;
   CloneMethod(int a,String s)
   {
     a1=a;
     s1=s;
   }
   void show()
   {
      System.out.println(a1+" "+s1);
   }
   public static void main(String args[]) throws CloneNotSupportedException
   {
      CloneMethod m=new CloneMethod(1,"User");
      CloneMethod m1=(CloneMethod) m.clone();
      m.show();
      m1.show();
   }
}
```

O/p:-
1 User
1 User

7.notify()

wakes up single thread, waiting on this object's monitor.
Syntax:- public final void notify()

8.notifyAll()

wakes up all the threads, waiting on this object's monitor.
public final void notifyAll()

9.wait()

causes the current thread to wait, until another thread notifies (invokes notify() or notifyAll() method).
Syntax:-public final void wait()throws InterruptedException

10.wait(long timeout)

causes the current thread to wait for the specified milliseconds, until another thread notifies (invokes notify() or notifyAll() method).
Syntax:-public final void wait(long timeout)throws InterruptedException

11.wait(long timeout,int nanos)
causes the current thread to wait for the specified milliseconds and nanoseconds, until another thread notifies (invokes notify() or notifyAll() method).
Syntax:-public final void wait(long timeout,int nanos)throws InterruptedException

Example Program :-

```
public class AllThreadMethods {
   public static void main(String args[]) throws InterruptedException
   {
      Task t=new Task();
      t.start();
      synchronized (t)
      {
         System.out.println("trying to call wait method");
         t.wait(1000);
         System.out.println("Main thread got notification");
      }
   }
}
class Task extends Thread{
   @Override
   public void run()
   {
      synchronized(this){
      for(int i=1;i<=10;i++)
      {
         System.out.println("output: "+i);
      }
      System.out.println("child Thread Sending notification");
      this.notifyAll();
   }
   }
}
```

o/p:-

```
trying to call wait method
output: 1
output: 2
output: 3
output: 4
output: 5
output: 6
output: 7
```

output: 8
output: 9
output: 10
child Thread Sending notification
Main thread got notification