Remainig Methods Of Object class.
----
1) protected native Object clone() throws CloneNotSupportedException :

It is a predefined non static method of Object class.

It is mainly used to create a duplicate copy of original object.

We can only perform clone() method operation, If the object is cloneable object that means that particular class must implements from java.lang.Cloneable interface which is a marker interface.

It is a protected method so we need to override the method and called the Object class clone() method.

It throws a checked exception i.e. java.lang.CloneNotSupportedException so handling is compulsory.

It follows deep copy concept so if we modify one object then another object remains unchanged.

clone() is method of Object class but not Cloneable interface, It is marker interface.

Steps we need to follow to perform clone operation :
----
1) The class must implements Cloneable interface
2) Override clone method from Object class [throws OR try catch]
3) In this Overridden method call Object class clone method [super.clone()]
4) Perform downcasting at the time creating duplicate object
5) Two different objects are created [deep copy]

//Program how to create Duplicate Object for Customer

```java
package com.ravi.clone;

class Customer implements Cloneable
{
    private Integer id;
    private String name;

    public Customer(Integer id, String name)
    {
        super();
        this.id = id;
        this.name = name;
    }


    public void setId(Integer id)
    {
        this.id = id;
    }

    public void setName(String name)
    {
        this.name = name;
    }


    @Override
    public Object clone() throws CloneNotSupportedException
    {
        return super.clone();
    }


    @Override
    public String toString()
    {
        return "Customer [id=" + id + ", name=" + name + "]";
    }
}

public class CustomerCloneCopy
{
    public static void main(String[] args)
    {
        Customer c1 = new Customer(111, "Scott");
        System.out.println(c1);

        try
        {
         Customer c2 =  (Customer) c1.clone();
         System.out.println(c2);

         System.out.println("After Modification Object c1 data");
         c1.setId(999);
         c1.setName("Raj");;

         System.out.println(c1);
         System.out.println(c2);

        }
         catch(CloneNotSupportedException e)
        {
              System.err.println("Object cloning is not possible :"+e);
        }
    }

}
```

Note : The main purpose of Marker interafce to privide additional information regarding Object like Object is cloneable OR Serializable OR Randomly Accessible, Without these information, JVM will not allow to perform operation on these objects.

---

proteced void finalize() throws Throwable :

It is a predefined non static method of Object class.

Garbage Collector automatically call this method just before an object is eligible for garbage collection to perform clean-up activity.

Here clean-up activity means closing the resources associated with that object like file connection, database connection, network connection and so on we can say resource de-allocation.

Note :- a) JVM calls finalize method only one per object.

b) This method is deprecated from java 9V.

c) Calling finalize() method on a particular object is an indication that Object is eligible for GC

```java
package com.ravi.finalize;

record Product(Integer id, String name)
{
    @Override
    public void finalize()
    {
        System.out.println("Product object is eligible for GC");
    }
}

public class FinalizeDemo
{
    public static void main(String[] args) throws InterruptedException
    {
        Product p1 = new Product(111, "Laptop");
        System.out.println(p1);

        p1 = null;

        System.gc();

        Thread.sleep(3000);

        System.out.println(p1);

    }

}
```

** What is the difference between final, finally and finalize()

Final
----
final it is modifier.it is used to provide some kind of restriction, by using final modifier we can declare our class, method, and field/variable as a final
- if we declare a class as final then we cannot extend OR inherits that class
- if we declare a method as a final then we cannot Override that method
- if we declare field/variable as final then re-assignment of that field /variable is not possible

Finally

- it is block always associated with try, catch block to maintain clean-up activity
- the speciality of this block is it will be always executed irrespective of exception. whether exception is raised or not
- it is specially used for handling resources, our resources are files, database, network resourse, and so on ...
that resources we should compulsory closed inside finally block

finalize()
----
finalize it is a Object class method. with the help of JVM garbage collector automatically call this method just before an Object is eligible for garbage collection to perform clean-up activity

---

public final void wait() throws InterruptedException :-

It is a predefined non static method of Object class. We can use this method from synchronized area only otherwise we will get java.lang.IllegalMonitorStateException.

It will put a thread into temporarly waiting state and it will release the Object lock, It will remain in the waiting state till another thread sends a notification message on the same object, After getting the notification the waiting thread will move from WAITING to BLOCKED state and now it is waiting for Object lock to re-enter inside synchronized area. Once the lock is available, It will move to RUNNABLE State to get processor time.

It throws a checked Exception i.e InterruptedException.

public native final void notify() :-

It will wake up the single thread that is waiting on the same object.It will not release the lock , once synchronized area is completed then only lock will be released.

Once a waiting thread will get the notification from the another thraed using notify()/notifyAll() method on the same object then the waiting thread will move to Blocked state and once it will get the lock then it will come to Runnable state, now It depends upon the Thread scheduler to schedule this thread for execution.

public native final void notifyAll() :-

It will wake up all the threads which are waiting on the same object.It will not release the lock , once synchronized area is completed then only lock will be released.

//Programs :
----
1) WAP to show wait(), notify() and notifyAll() must be used from synchronized area.

```java
package com.ravi.itc;

public class ITCDemo1
{
    public static void main(String[] args)
    {
        System.out.println("Main Thread started!!!");
        try
        {
            Object obj = new Object();
            obj.wait();
        }
        catch(InterruptedException e)
        {
            System.err.println("Thread is Interrupted");
        }
    }
}
```

Why wait(), notify() and notifyAll() methods are defined in Object class but not in Thread class ?

ANS : All these methods must be used from **synchronized area only** that means the current thread must be the owner of the lock and lock is available with Object class so all these methods are defined in Object class but not in Thread class.