GC

If a object does not have a ref variable, then its eligible for garbage collection

1. Nullifying the ref variable

Integer a = Integer.valueOf(“10”);

A= null;

- ➔ Now Integer.valuOf(10)- is eligible for GC

Connection = null;

2. ReAssigning the ref variable to some other object. Will make the existing object with no ref

Integere s1 = Integer.valueOf(“10”);

S1= Integer.valueOf(“20”);

3. Local variables will be destroyed after method executed is completed. Objects created inside a method

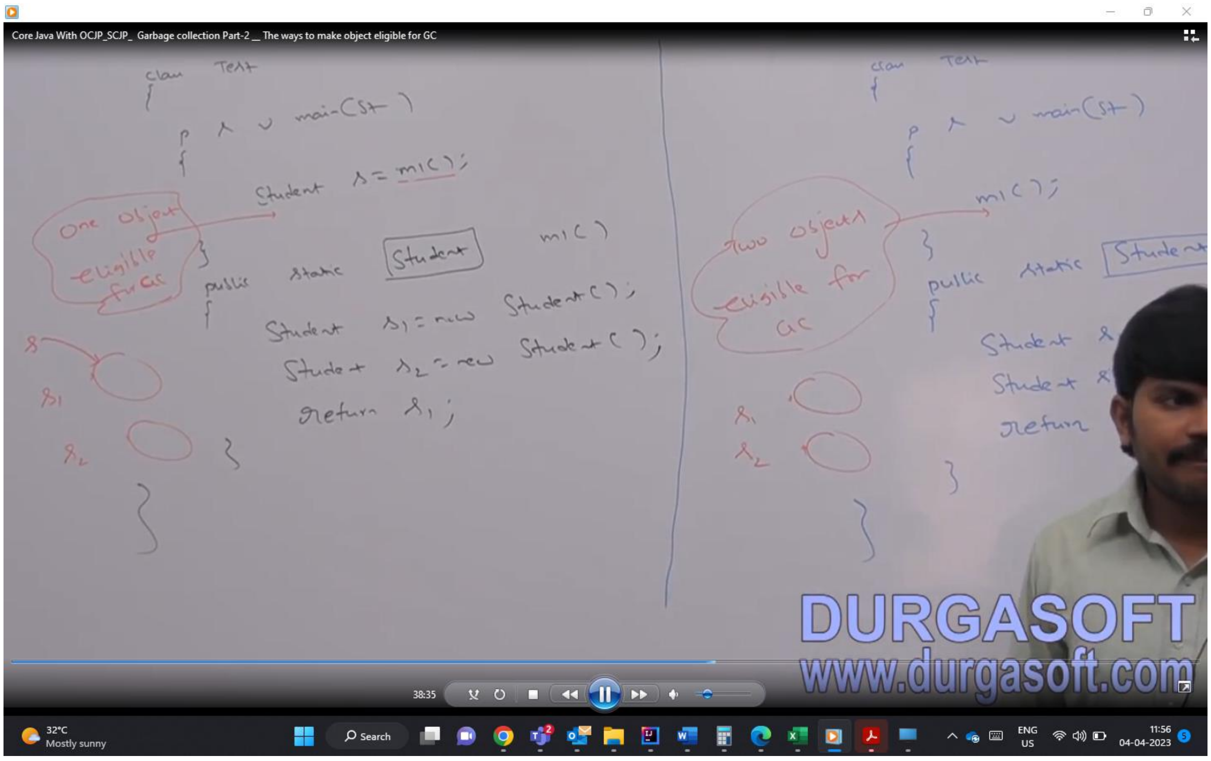
Public void m1(){

Integer m = Integer.valueOf(“10”);

}

If this method returns an object and the return value is assigned to a diff variable , a new ref is maintained there so that object will not be available for GC

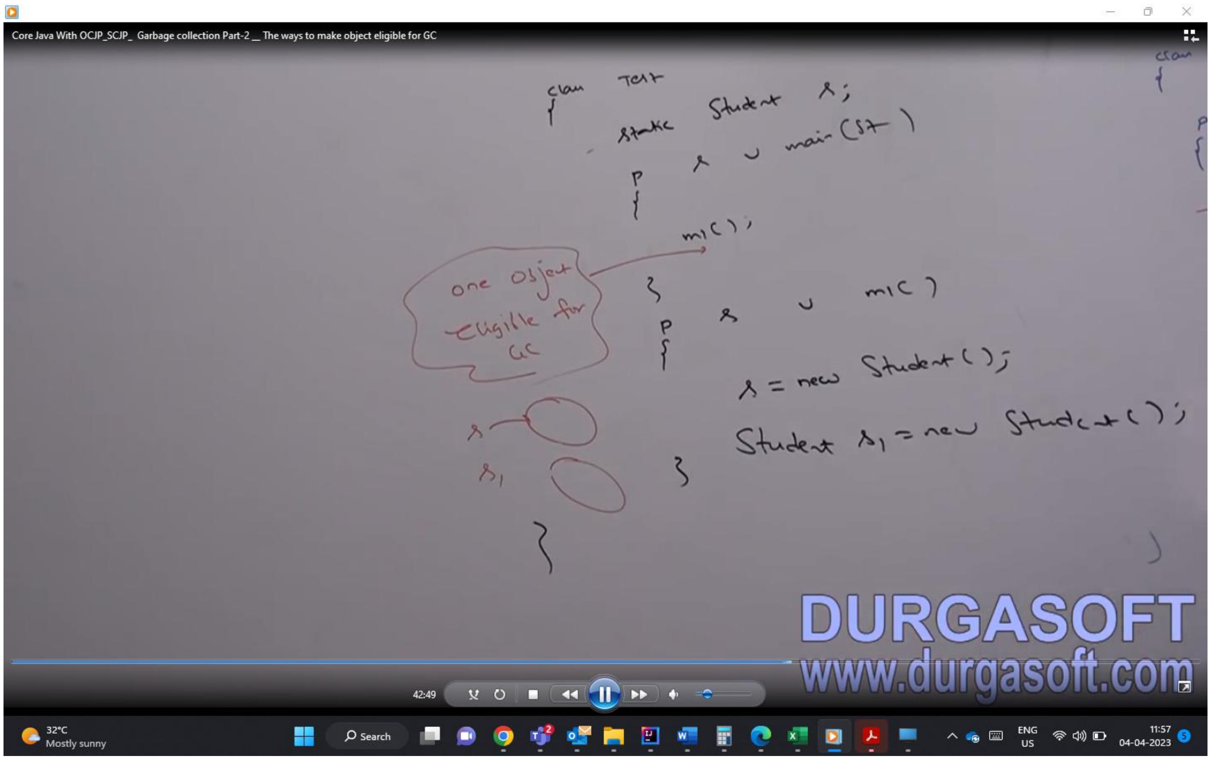
If the returned object is not assigned to any variable then its available for GC



Core Java With OCJP_SCJP_ Garbage collection Part-2 _ The ways to make object eligible for GC

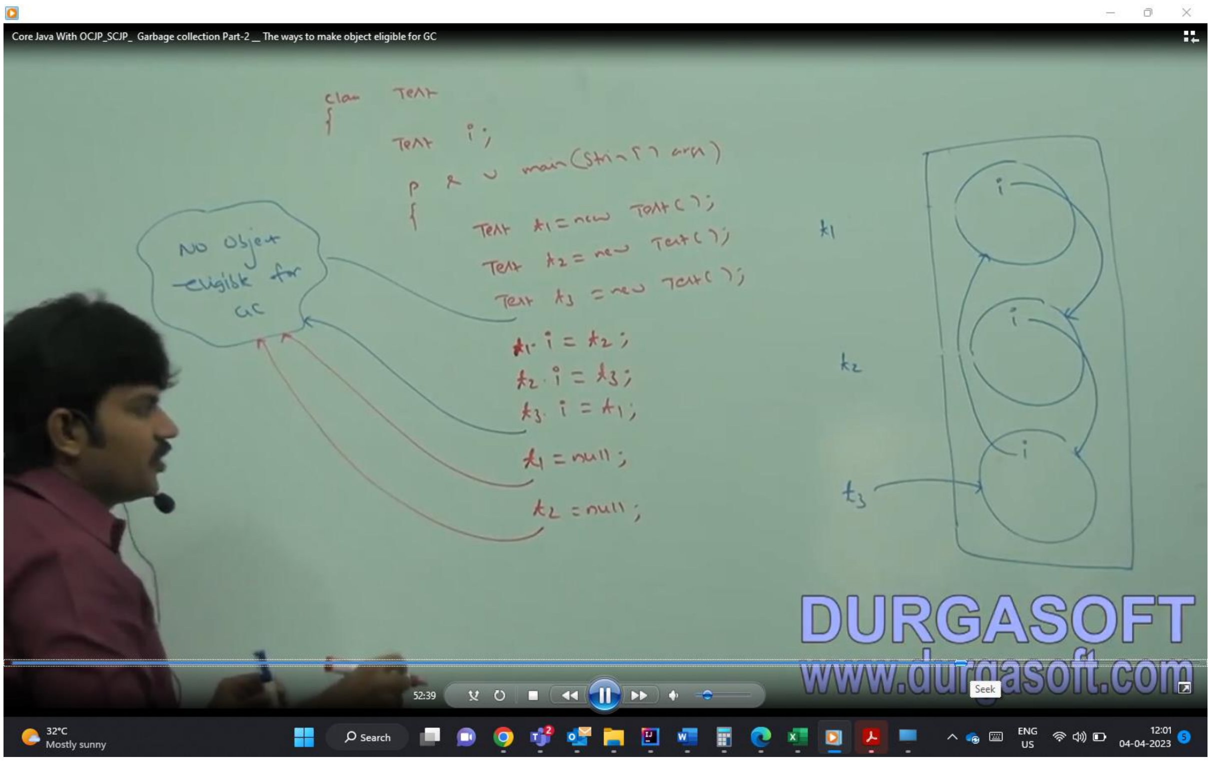
```
class Test
{
    p s v main(St )
    {
        Student s = m1();
    }
    public static Student m1()
    {
        Student s1 = new Student();
        Student s2 = new Student();
        return s1;
    }
}
```

Here static variable s holds the ref after m1 execution. 1 object s1 is available for GC



Core Java With OCJP_SCJP_ Garbage collection Part-2 _ The ways to make object eligible for GC

```
class Test
{
    static Student s;
    p s v main(St )
    {
        m1();
    }
    p s v m1()
    {
        s = new Student();
        Student s1 = new Student();
    }
}
```


Core Java With OCJP_SCJP_ Garbage collection Part-2 __ The ways to make object eligible for GC

```
class Test
{
    Test i;
    p s v main(String[] args)
    {
        Test t1 = new Test();
        Test t2 = new Test();
        Test t3 = new Test();
        t1.i = t2;
        t2.i = t3;
        t3.i = t1;
        t1 = null;
        t2 = null;
```

when t1 = null

T3.i still has ref to that object – so no GC

When t2 is also made null

T3.i has ref to t1Object and t1.i object has ref to t2 , so no GC

When t3 is made null

Ti object has no ref – so its available of GC, when t1 is GCed t2 is freed, and t3 is freeded

So all three available for GC

This is called Island of Isolation

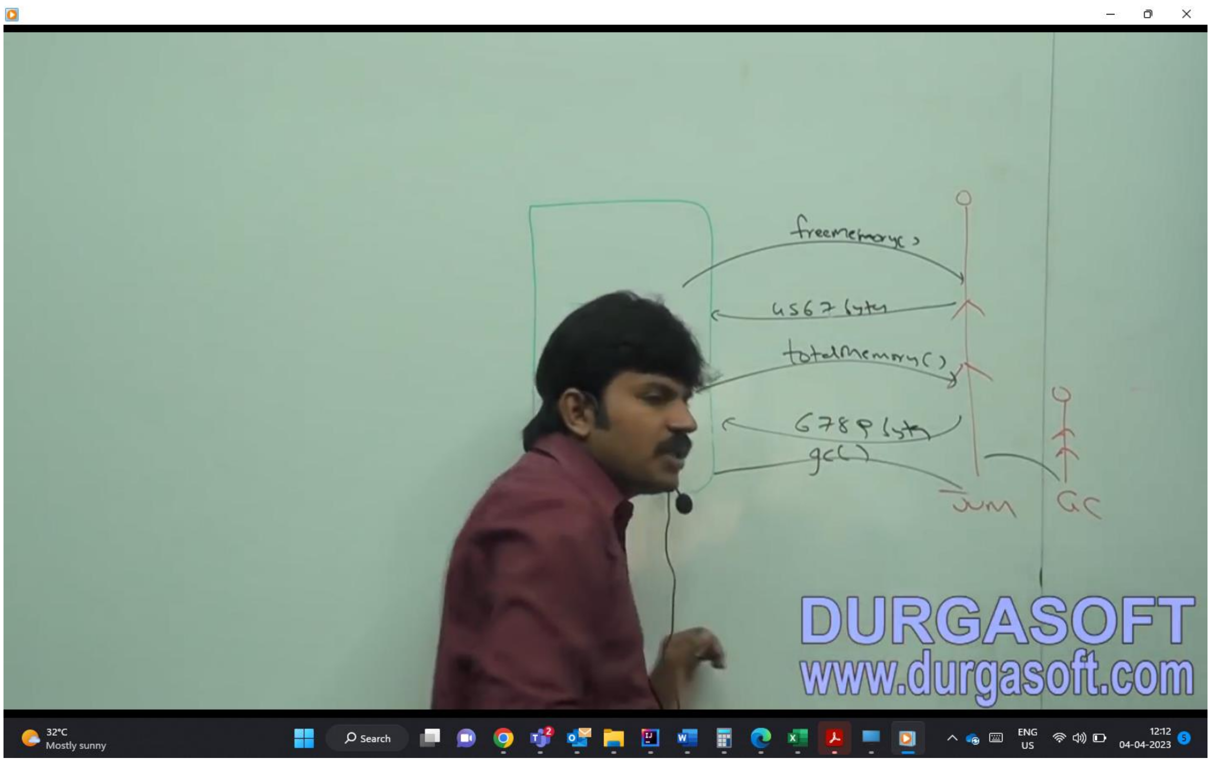
An object will not be destroyed immediately after GC, JVM will call GC only when needed. Its based on JVM beh, like if its configured to run GC when low memory etc

We can request JVM to run GC , but JVM accepts or not depends on JVM. Most of the time it accepts

By using System class GC method

System.gc();

Ask JVM to check how much of free memory is there or total memory . User can find whether memory is enough of not and can request System.gc()

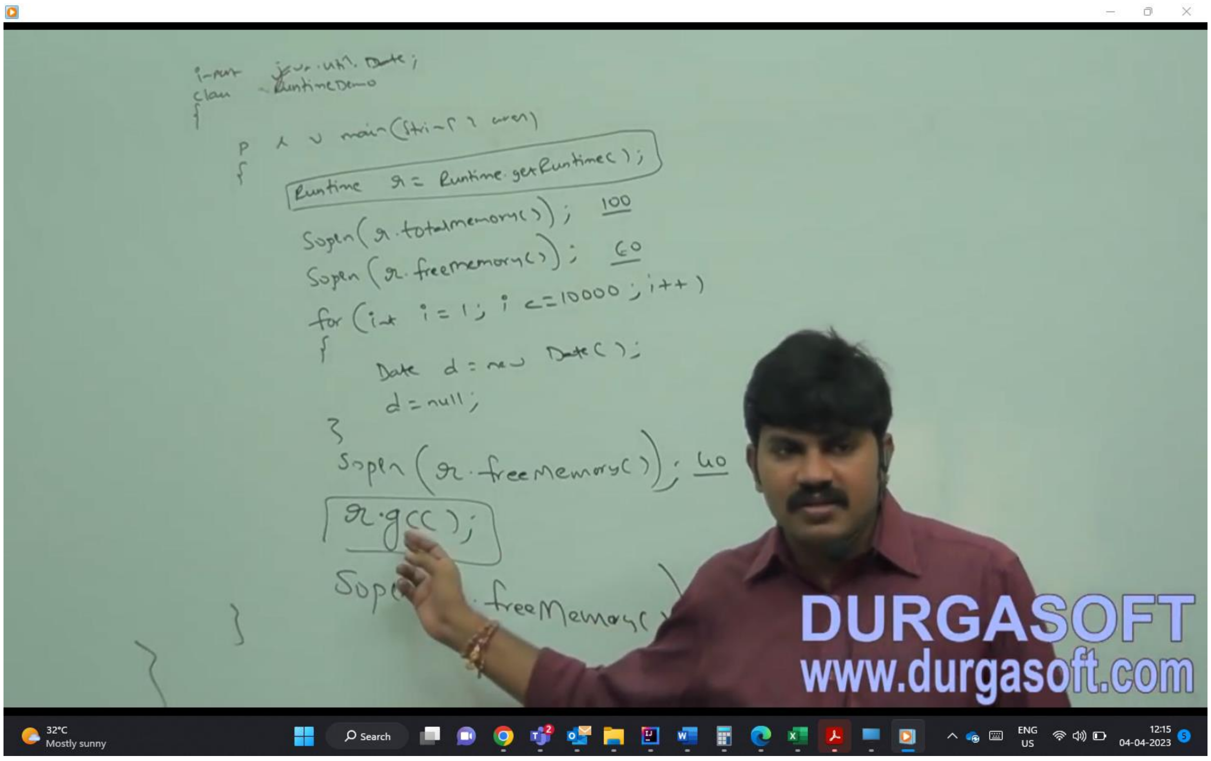


Runtime r = Runtime.getRuntime();

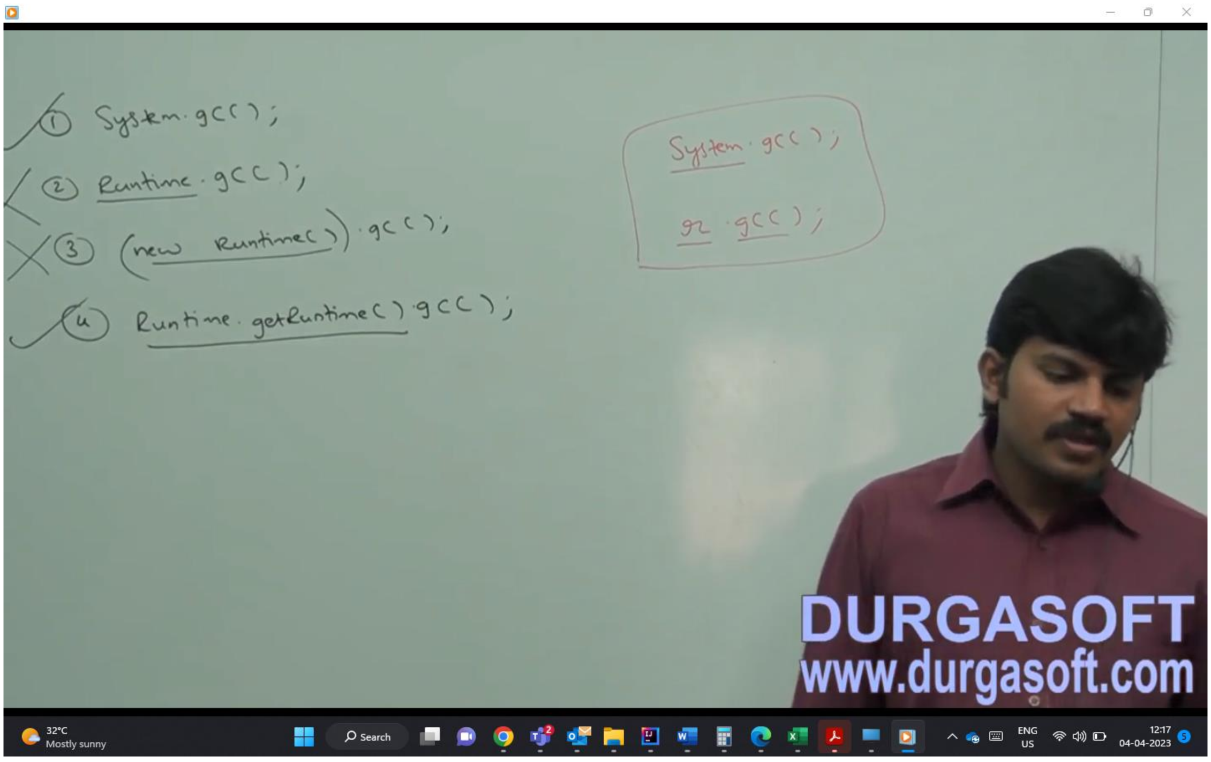
r.totalMemory()

r.freeMemory()

gc()



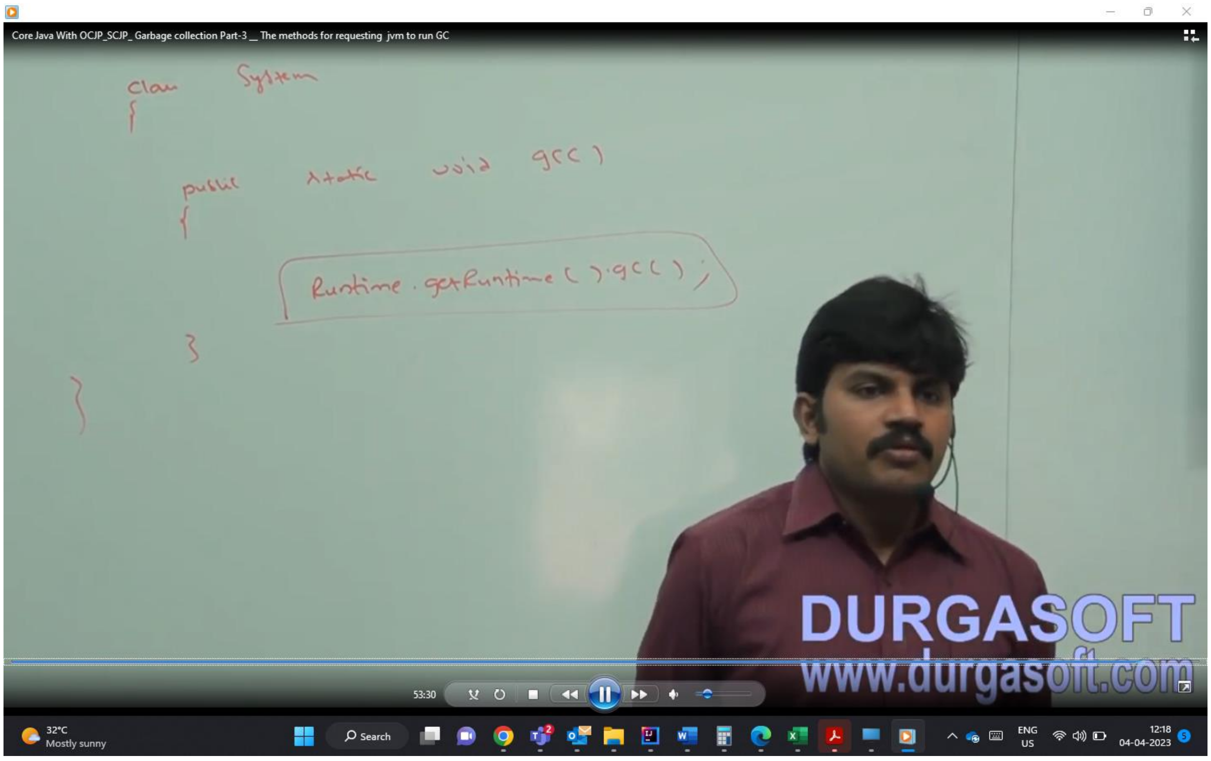
```
import java.util.Date;
class RuntimeDemo
{
  p s v main(String[] args)
  {
    Runtime r = Runtime.getRuntime();
    Sopln(r.totalMemory());  100
    Sopln(r.freeMemory());  60
    for (int i = 1; i <= 10000; i++)
    {
      Date d = new Date();
      d = null;
    }
    Sopln(r.freeMemory());  40
    r.gc();
    Sop[illegible].freeMemory(
  }
}
```



```
1 System.gc();
2 Runtime.gc();
3 (new Runtime()).gc();
4 Runtime.getRuntime().gc();

System.gc();
r.gc();
```

GC method present in system class is a static method whereas the GC present in Runtime class is instance method

Highly recommended to use RunTime.gc to run GC

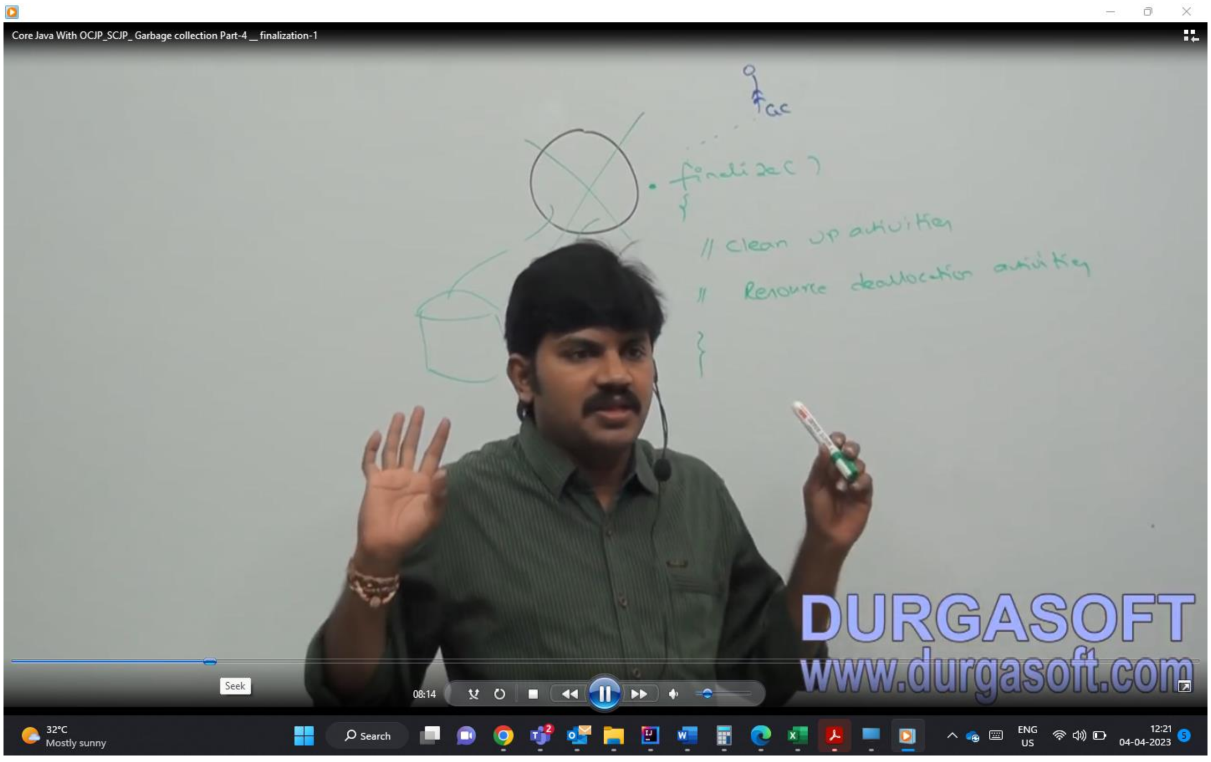

Core Java With OCJP_SCJP_ Garbage collection Part-3 __ The methods for requesting jvm to run GC
DURGASOFT
www.durgasoft.com

```
class System
{
    public static void gc()
    {
        Runtime.getRuntime().gc();
    }
}
```

System Class GC method internally calls RunTime class GC method

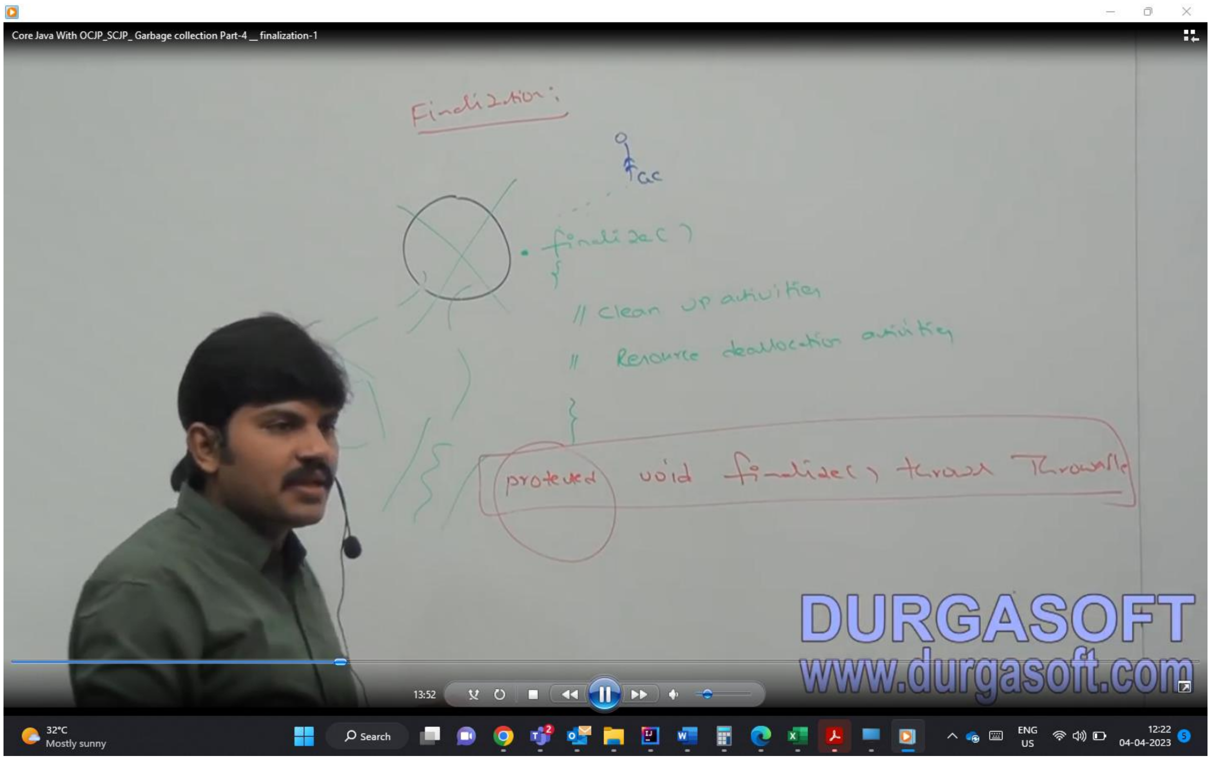
Finalization()

Before destroying if there are any dependencies , like close connections etc. Then u can specify that in finalization


Core Java With OCJP_SCJP_ Garbage collection Part-4 __ finalization-1
GC
DURGASOFT
www.durgasoft.com

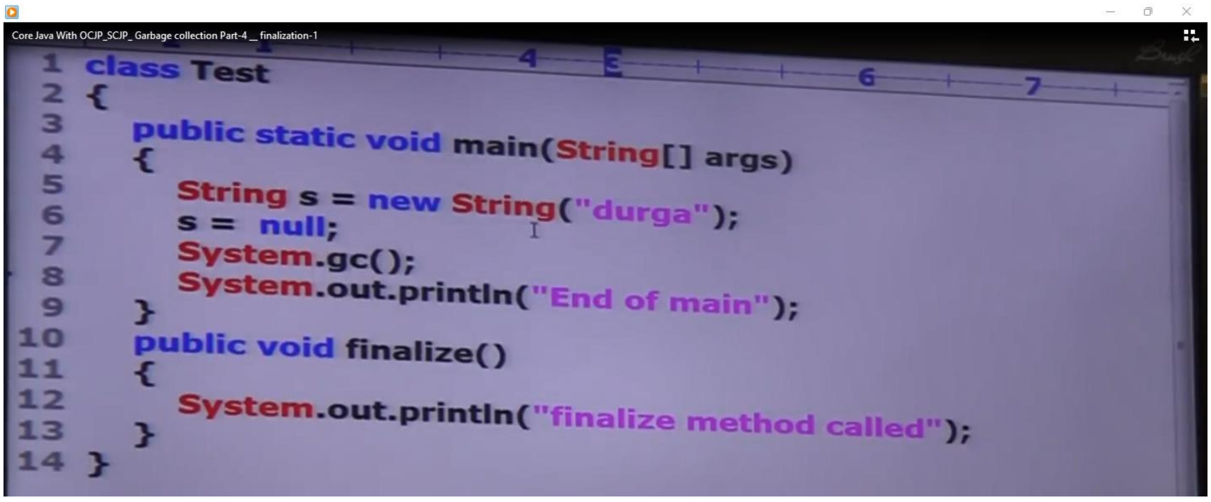
```
finalize()
{
    // clean up activities
    // Resource deallocation activities
}
```

GC calls finalize method in Object class to perform finalization


Core Java With OCJP_SCJP_ Garbage collection Part-4 _ finalization-1
Finalization:
GC
finalize()
{
// clean up activities
// Resource deallocation activities
}
protected void finalize() throws Throwable
DURGASOFT
www.durgasoft.com

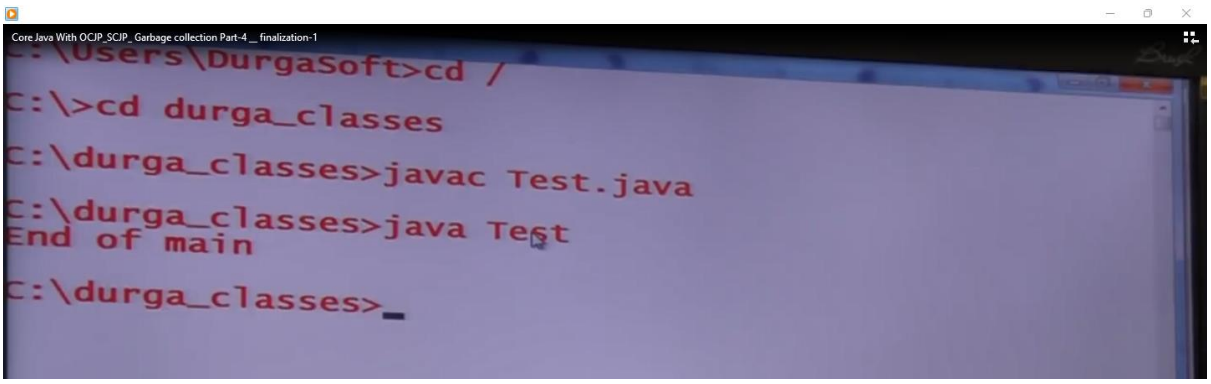
Protected method highly recommended to override

Protected method to public overriding.


Core Java With OCJP_SCJP_ Garbage collection Part-4 _ finalization-1

```
class Test
{
    public static void main(String[] args)
    {
        String s = new String("durga");
        s =  null;
        System.gc();
        System.out.println("End of main");
    }
    public void finalize()
    {
        System.out.println("finalize method called");
    }
}
```

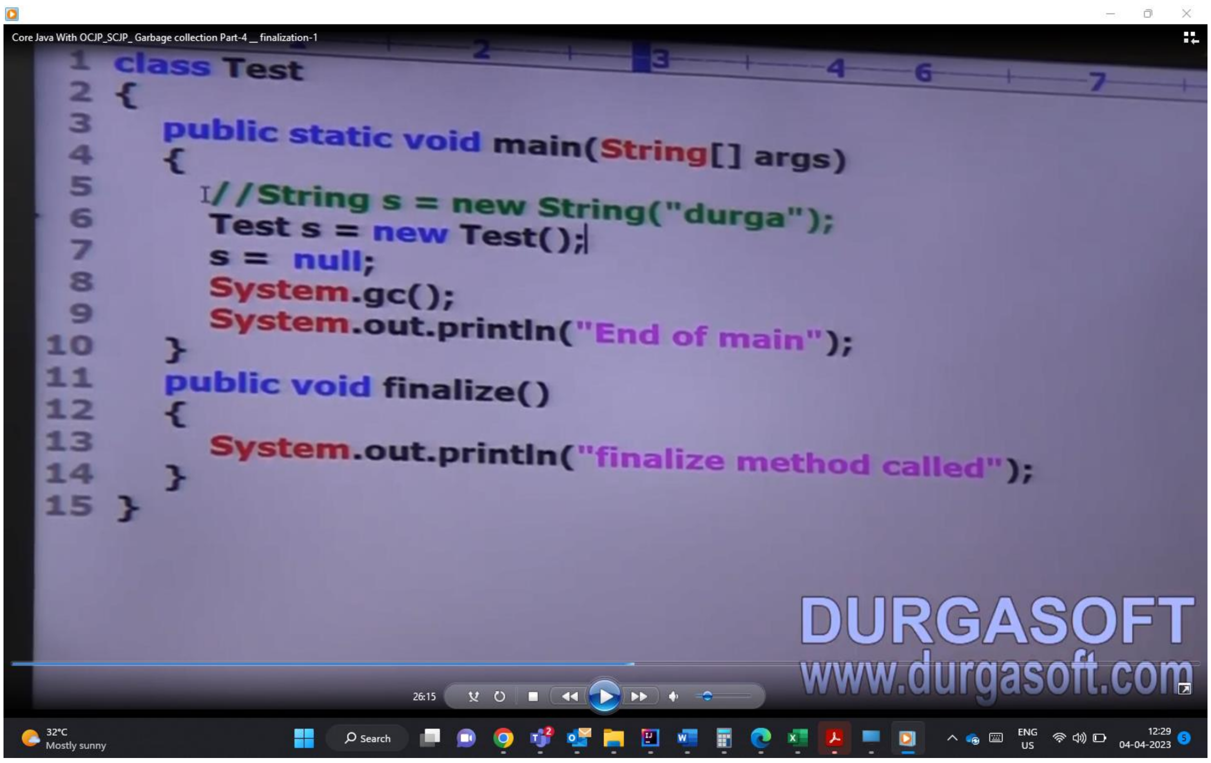
Here we have String Object only the objects available in this program are available for GC. Here there is no Test class object created only String class object is being created, When GC is being called , String Class finalize method will be called not test class finalize. There is no implementation for String class finalize so the program just printed “End of Main”



Core Java With OCJP_SCJP_ Garbage collection Part-4 _ finalization-1

```
C:\Users\DurgaSoft>cd /

C:\>cd durga_classes

C:\durga_classes>javac Test.java

C:\durga_classes>java Test
End of main

C:\durga_classes>_
```

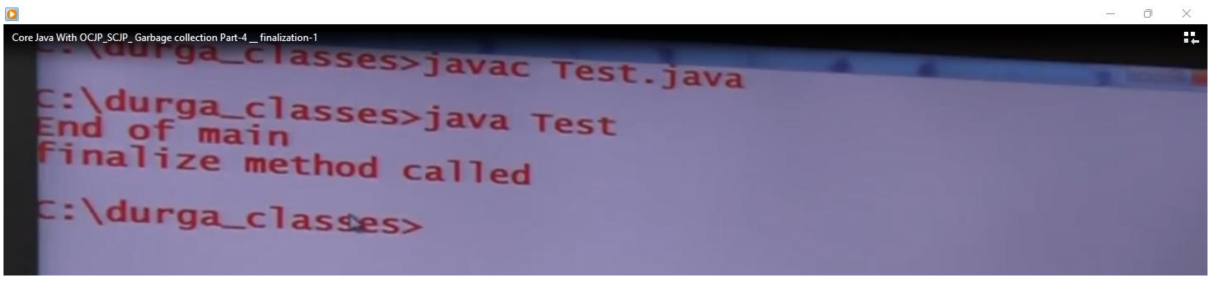
In the below class the finalize method of test class will be called and will print finalize method called string therea



Core Java With OCJP_SCJP_ Garbage collection Part-4 _ finalization-1

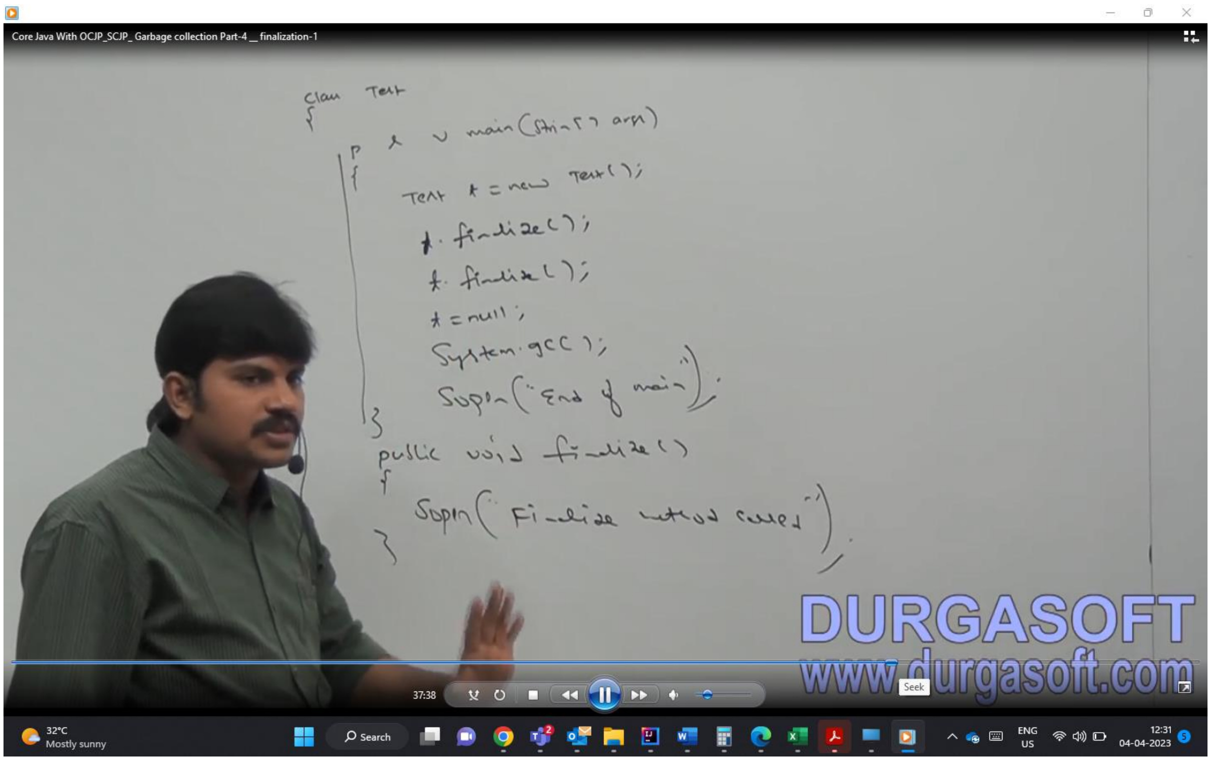
```java
class Test
{
    public static void main(String[] args)
    {
        //String s = new String("durga");
        Test s = new Test();
        s =  null;
        System.gc();
        System.out.println("End of main");
    }
    public void finalize()
    {
        System.out.println("finalize method called");
    }
}
```

Sometime end of main will be followed by finalize method called depending on GC may or may not respond immediatyely


Core Java With OCJP_SCJP_ Garbage collection Part-4 _ finalization-1

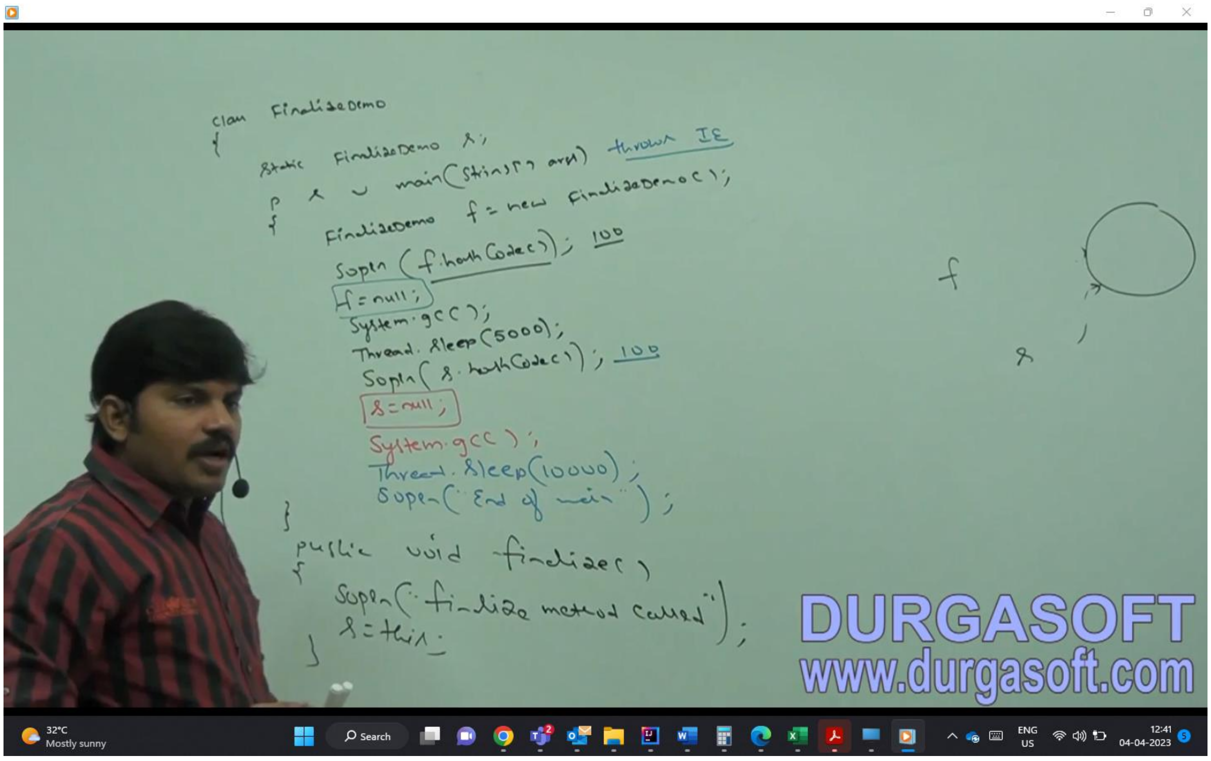
```
C:\durga_classes>javac Test.java

C:\durga_classes>java Test
End of main
finalize method called

C:\durga_classes>
```

We can call finalize explicitly and can be executed as a normal method but the object will not be destrotyed. At the end System.GC will call finalize and performs GC

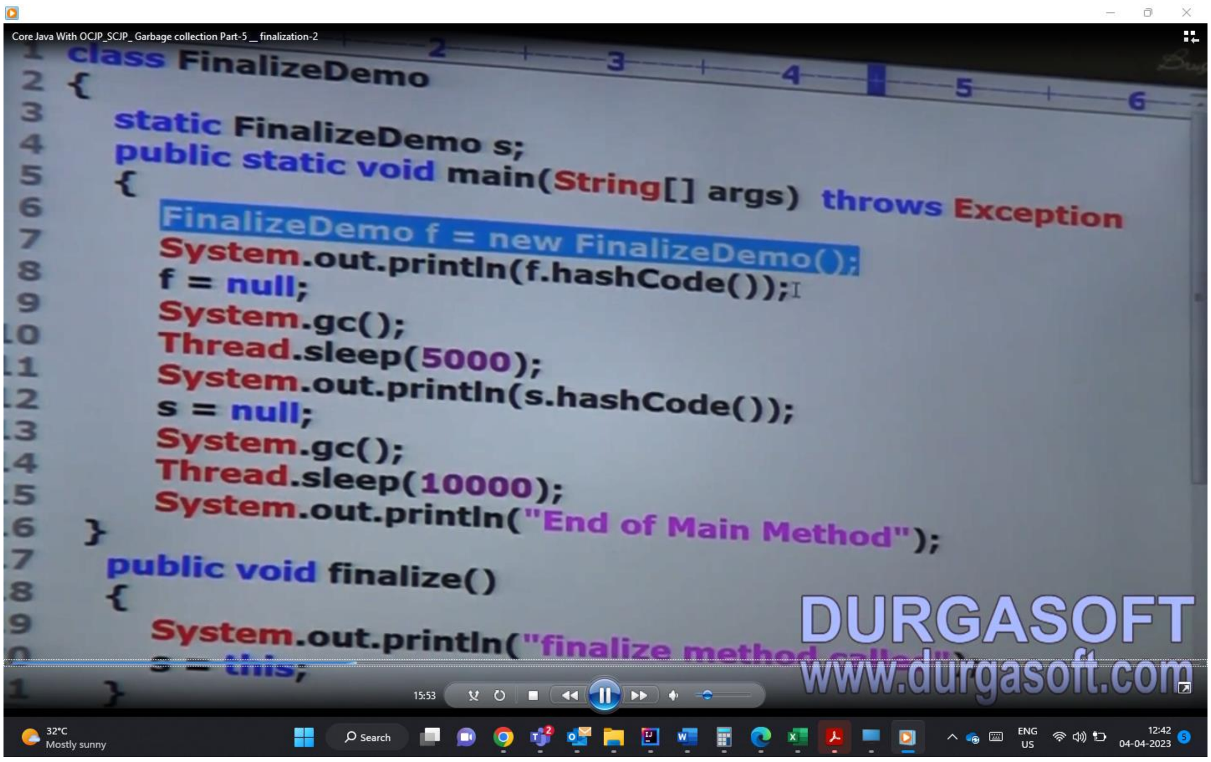

Core Java With OCJP_SCJP_ Garbage collection Part-4 _ finalization-1
DURGASOFT
www.durgasoft.com
37:38
Seek

```
class Test
{
    p s v main(String[] args)
    {
        Test t = new Test();
        t.finalize();
        t.finalize();
        t = null;
        System.gc();
        Sopln("End of main");
    }
    public void finalize()
    {
        Sopln("Finalize method called");
    }
```

For any object , GC can be called only once, second time it will not call the finalize method it will directly destroy it



```
class FinalizeDemo
{
    static FinalizeDemo s;
    p s v main(String[] args) throws IE
    {
        FinalizeDemo f = new FinalizeDemo();
        Sopln(f.hashCode()); 100
        f = null;
        System.gc();
        Thread.sleep(5000);
        Sopln(s.hashCode()); 100
        s = null;
        System.gc();
        Thread.sleep(10000);
        Sopln("End of main");
    }
    public void finalize()
    {
        Sopln("finalize method called");
        s = this;
    }
```



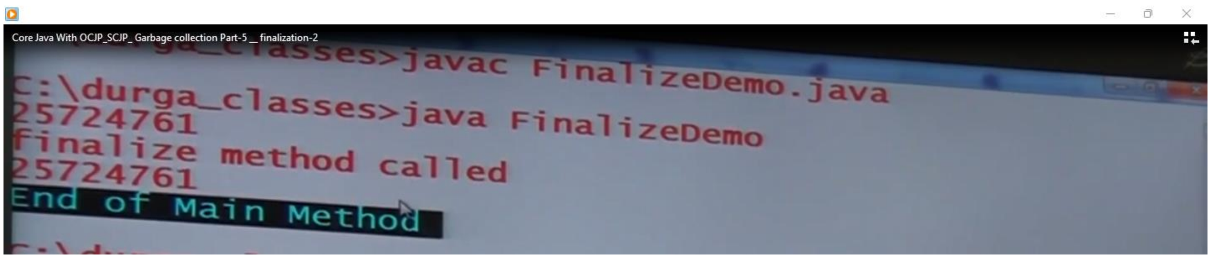
Core Java With OCJP_SCJP_ Garbage collection Part-5 _ finalization-2

```
1  class FinalizeDemo
2  {
3      static FinalizeDemo s;
4      public static void main(String[] args) throws Exception
5      {
6          FinalizeDemo f = new FinalizeDemo();
7          System.out.println(f.hashCode());
8          f = null;
9          System.gc();
10         Thread.sleep(5000);
11         System.out.println(s.hashCode());
12         s = null;
13         System.gc();
14         Thread.sleep(10000);
15         System.out.println("End of Main Method");
16     }
17     public void finalize()
18     {
19         System.out.println("finalize method called");
20         s = this;
21     }
```

First time when Line 9 will call GC and finalize method but here s is assigned to the object f (this). Then the object will not be destroyed as it is assigned to s, s will have the same hascode as f.

Now again when gc is called in line 13 it will not call finalize on object for s as it has already called

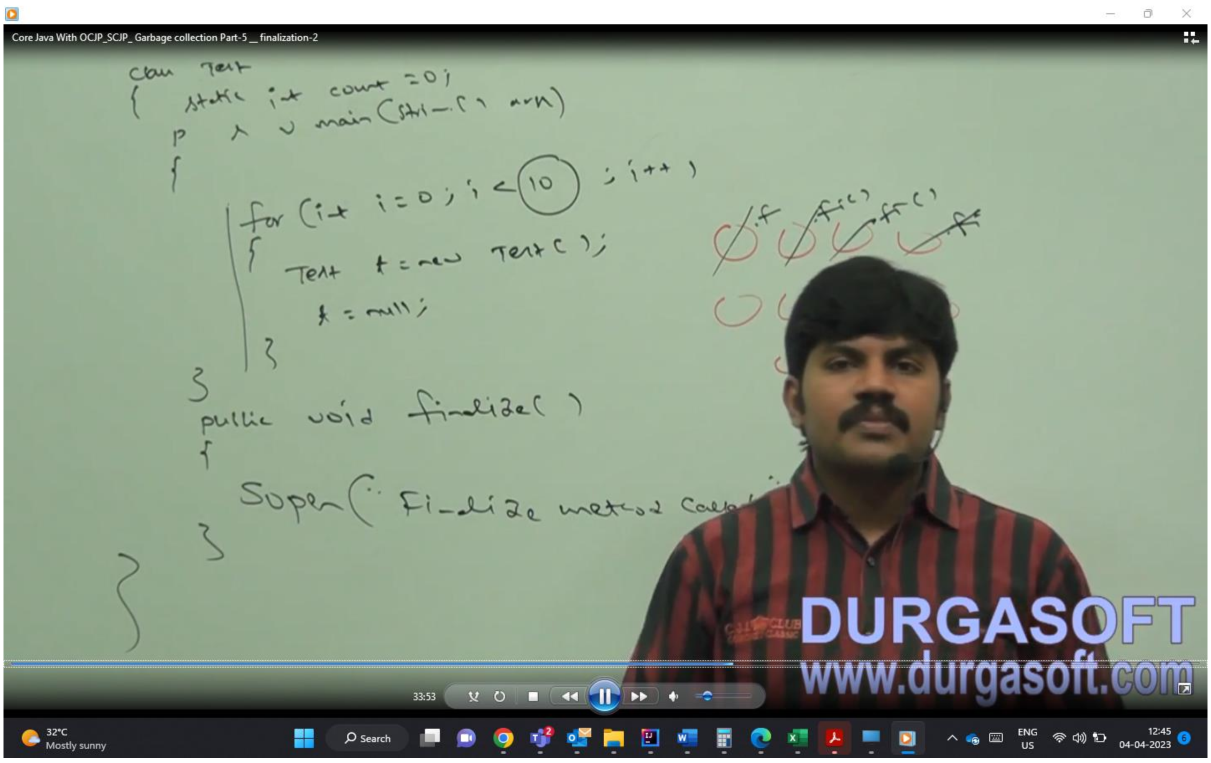
The object is eligible for GC twice but Garbage collector has called finalize only once


Core Java With OCJP_SCJP_ Garbage collection Part-5 _ finalization-2

```
asses>javac FinalizeDemo.java
C:\durga_classes>java FinalizeDemo
25724761
finalize method called
25724761
End of Main Method
```

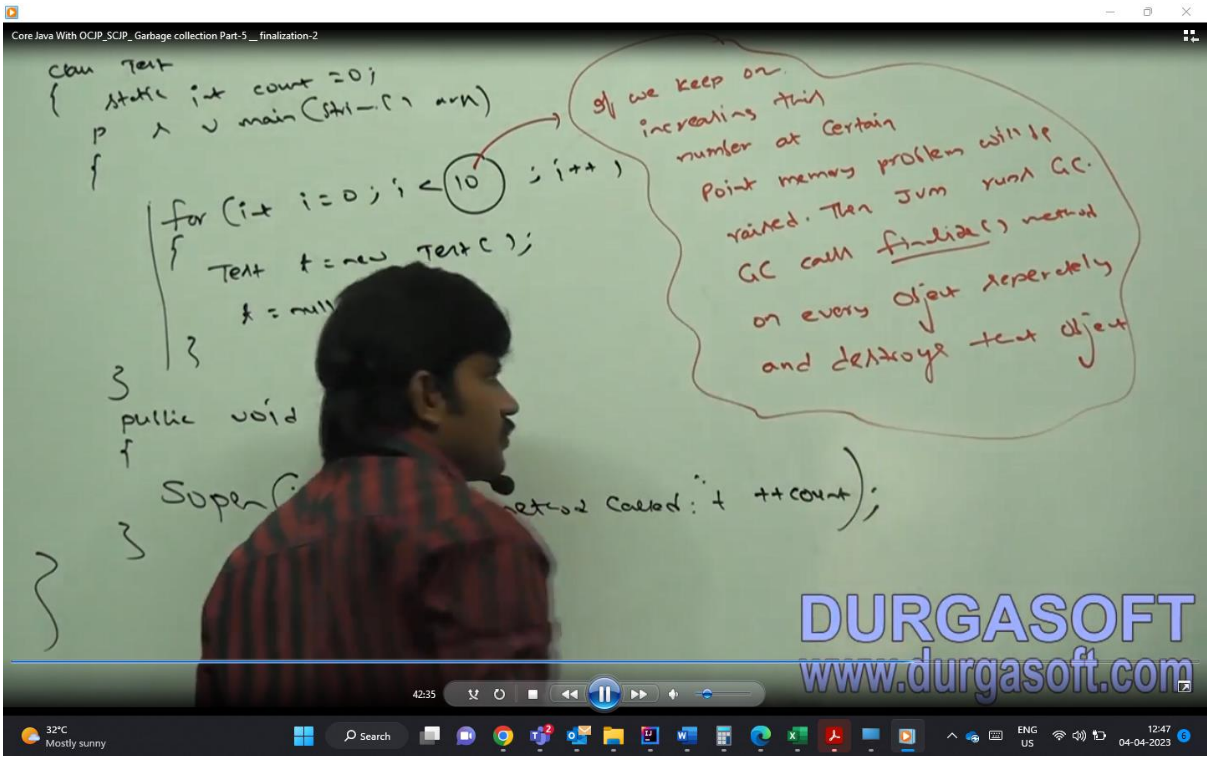
In the below eg , this code creates test object and its assigned null and available for GC

If we keep running the program sometime other JVM will run GC and will print the finalize method for the object


Core Java With OCJP_SCJP_ Garbage collection Part-5 _ finalization-2
DURGASOFT
www.durgasoft.com
33:53

```
class Test
{
  static int count =0;
  p s v main(String[] args)
  {
    for(int i=0;i<10;i++)
    {
      Test t=new Test();
      t=null;
    }
  }
  public void finalize()
  {
    Sopln("Finalize method Called
  }
}
```

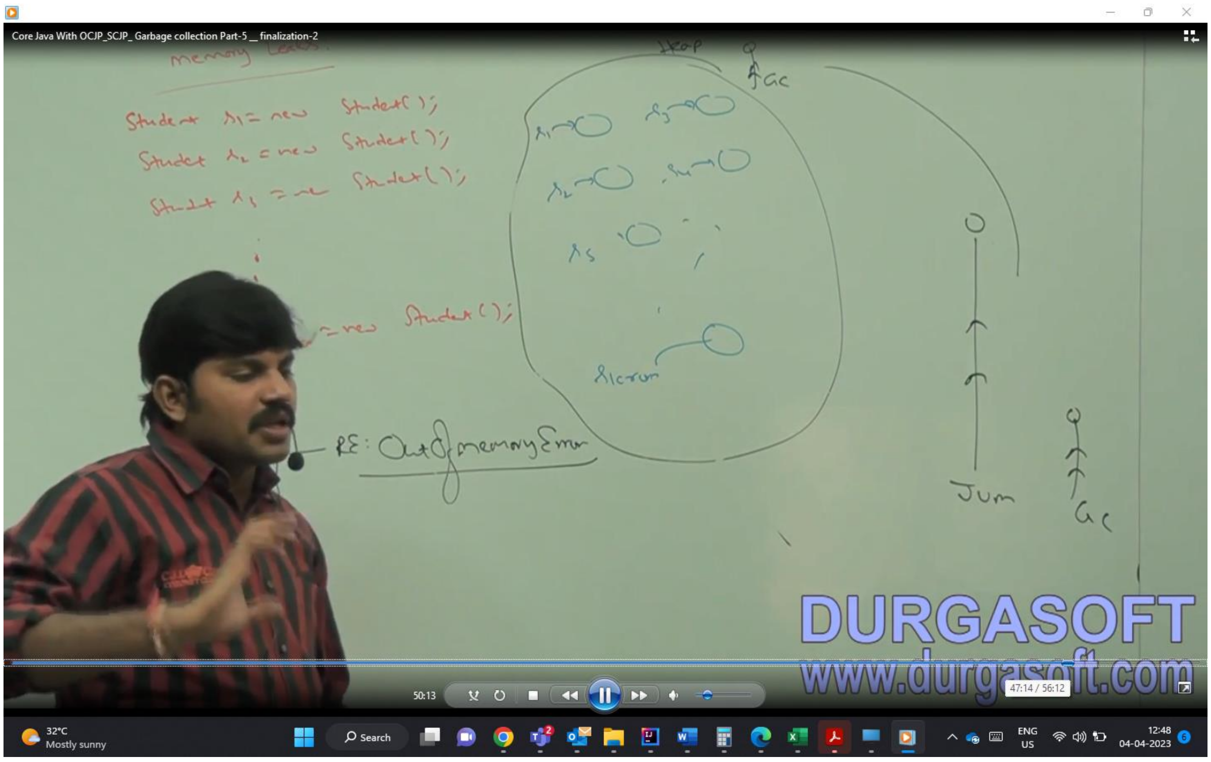
1. Programs run with low memory then JVM runs GC

2. 

Core Java With OCJP_SCJP_ Garbage collection Part-5 __ finalization-2

```
class Test
{
    static int count = 0;
    p s v main(String[] args)
    {
        for (int i = 0; i < 10 ; i++)
        {
            Test t = new Test();
            t = null;
        }
    }
    public void
    {
        Super( ... method Called: + ++count);
    }
}
```

If we keep on increasing this number at certain point memory problem will be raised. Then Jvm runs GC. GC calls finalize() method on every object seperately and destroys test object



Core Java With OCJP_SCJP_ Garbage collection Part-5 __ finalization-2

```
Student s1 = new Student();
Student s2 = new Student();
Student s3 = new Student();
        :
        = new Student();
```

RE: OutOfMemoryError

If there is memory crunch and no object is available for GC then OutOfMemory Error