**Finalization – Part\_02**

* **Case\_03:**

Even though, object eligible for GC multiple times but GC calls finalize method only once.

Example:

class FinalizeDemo{

static FinalizeDemo s;

public static void main(String[] args) throws InterruptedException{

FinalizeDemo f = new FinalizeDemo();

System.out.println(f.hashCode());

f=null;

System.gc();

Thread.sleep(5000);

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

s=null;

System.gc();

Thread.sleep(10000);

System.out.println(“End of main”);

}

public void finalize(){

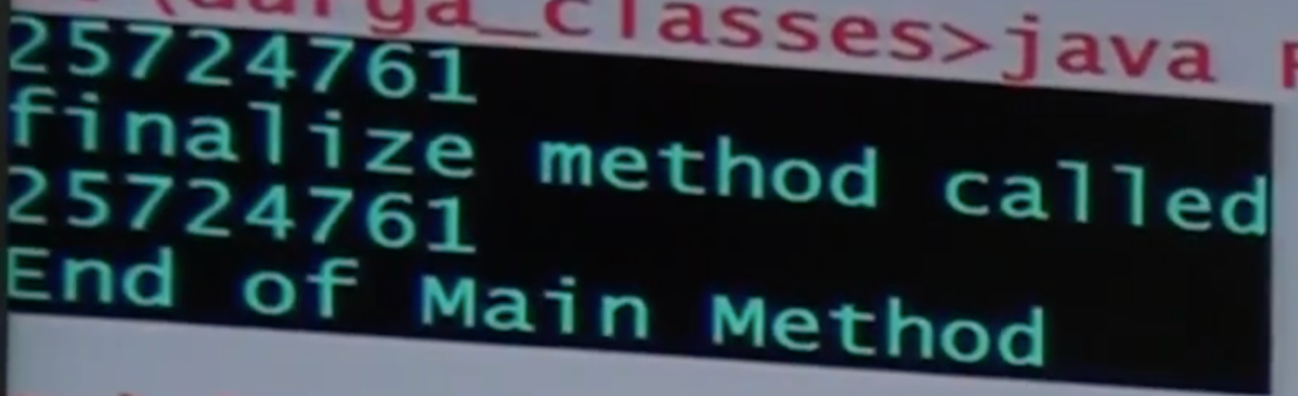
System.out.println(“Finalize method called”);

s=this;

}

}

Output:



Note:

In the above program even though object eligible for GC multiple times but GC calls finalize method only once.

* **Case\_04:**

We can’t expect exact behavior of garbage collector it is varied from JVM to JVM, hence for the following questions we can’t provide exact answers.

1. When exactly JVM runs GC?
2. In which order GC identifies eligible objects?
3. In which order GC destroys eligible objects?
4. Whether GC destroys all eligible or not?
5. What is the algorithm followed by GC etc?

Note:

1. Whenever program runs with low memory, then JVM runs GC, but we can’t expect exactly at what time.
2. Most of the GC follows standard algorithm “Mark and sweep” algorithm. It doesn’t mean every GC follows the same algorithm.

Example:

class Test{

static int count = 10;

public static void main(String[] args){

for(int i=0;i<10;i++){

Test t = new Test();

t=null;

}

}

public void finalize(){

System.out.println(“Finalize method called:”+ ++count);

}

}

Note:

If we keep on increasing this number (i value), at certain point memory problem will be raised, then JVM runs GC.

GC calls finalize method on every object separately and destroys that object.

* **Case\_05: Memory Leaks:**

The objects which are not used in our program and which are not eligible for GC such type of useless objects is called memory leaks. In our program if memory leaks present, then the program will be terminated by raising OutOfMemoryError.

Hence, if an object no longer required it is highly recommended to make that object eligible for GC.

The following are various 3rd party memory management tools to identify memory leaks:

HP OVO

HP JMeter

JProbe

Patrol

IBM Tivoli