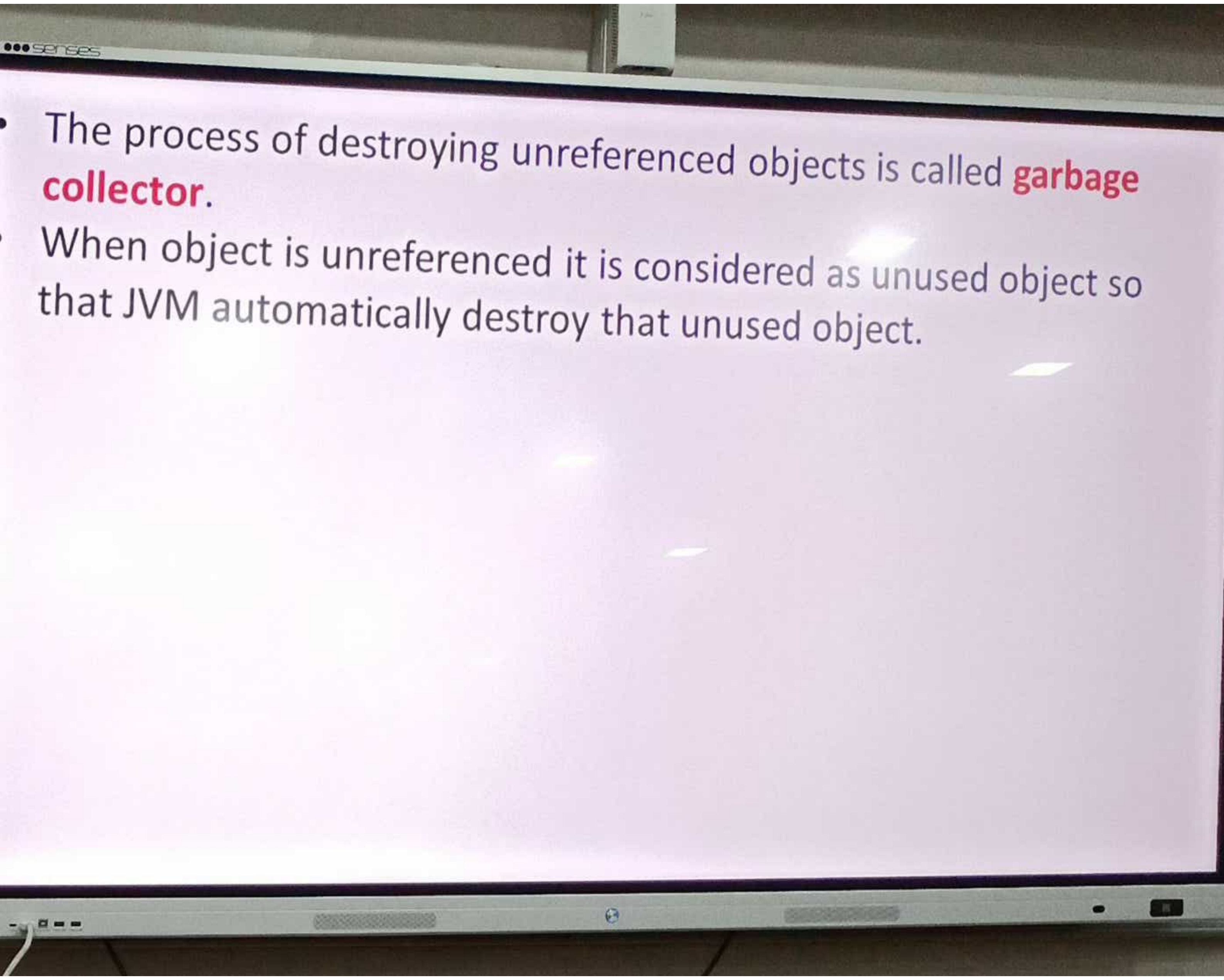
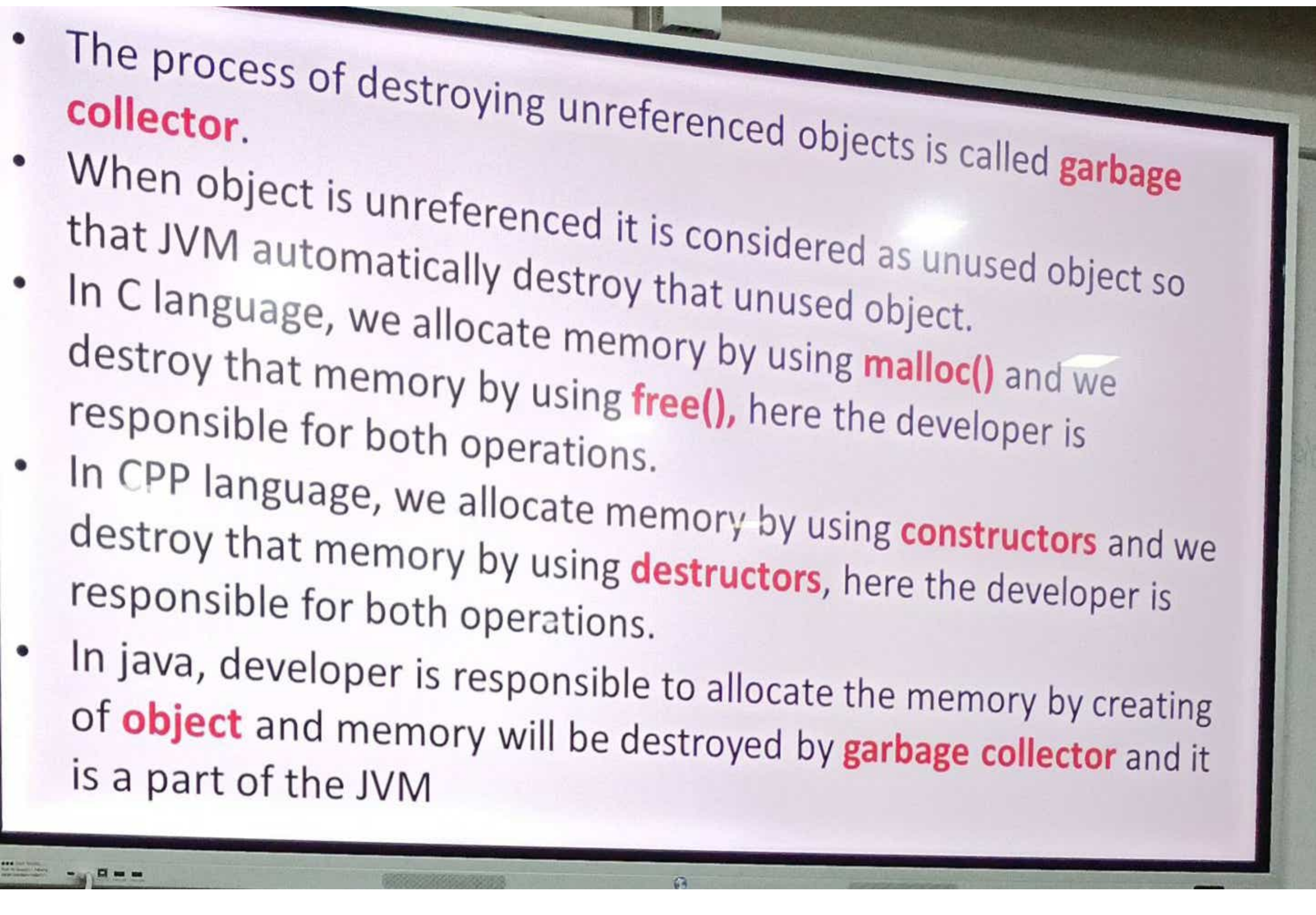
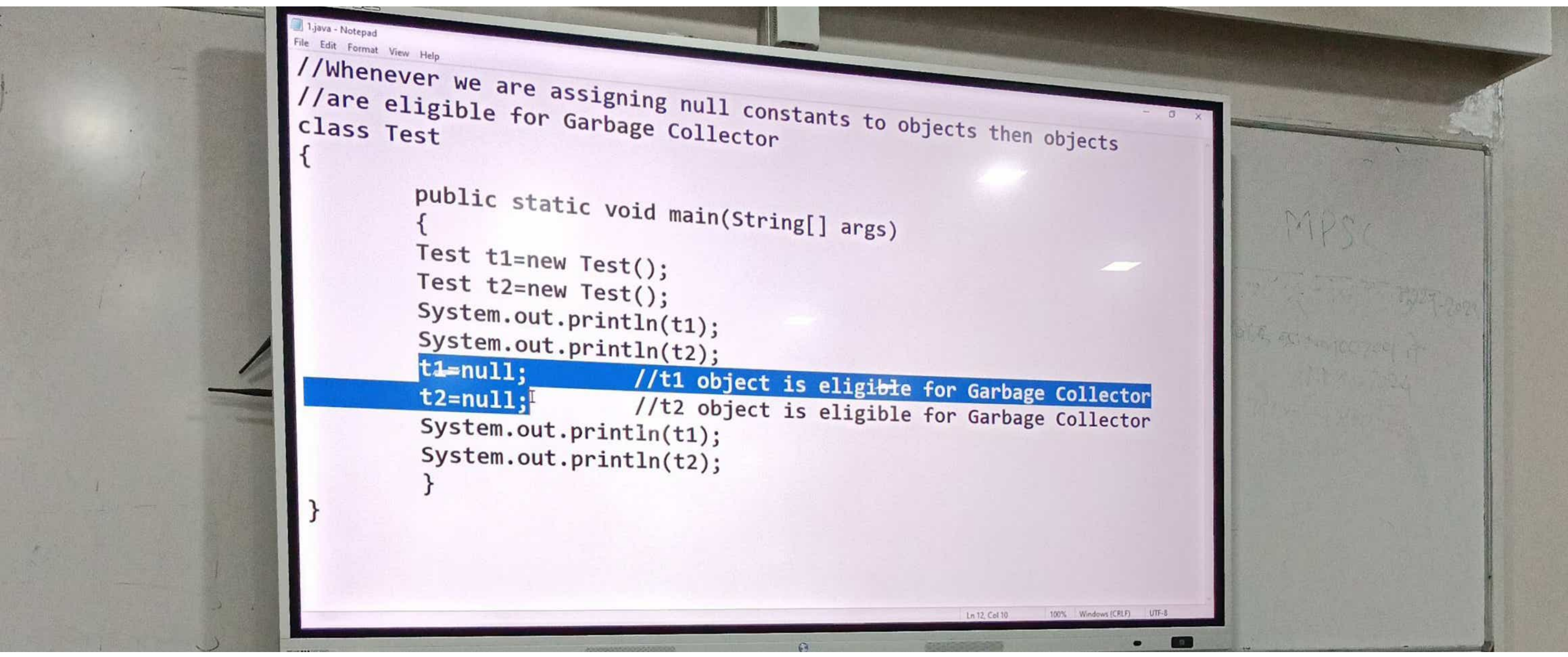


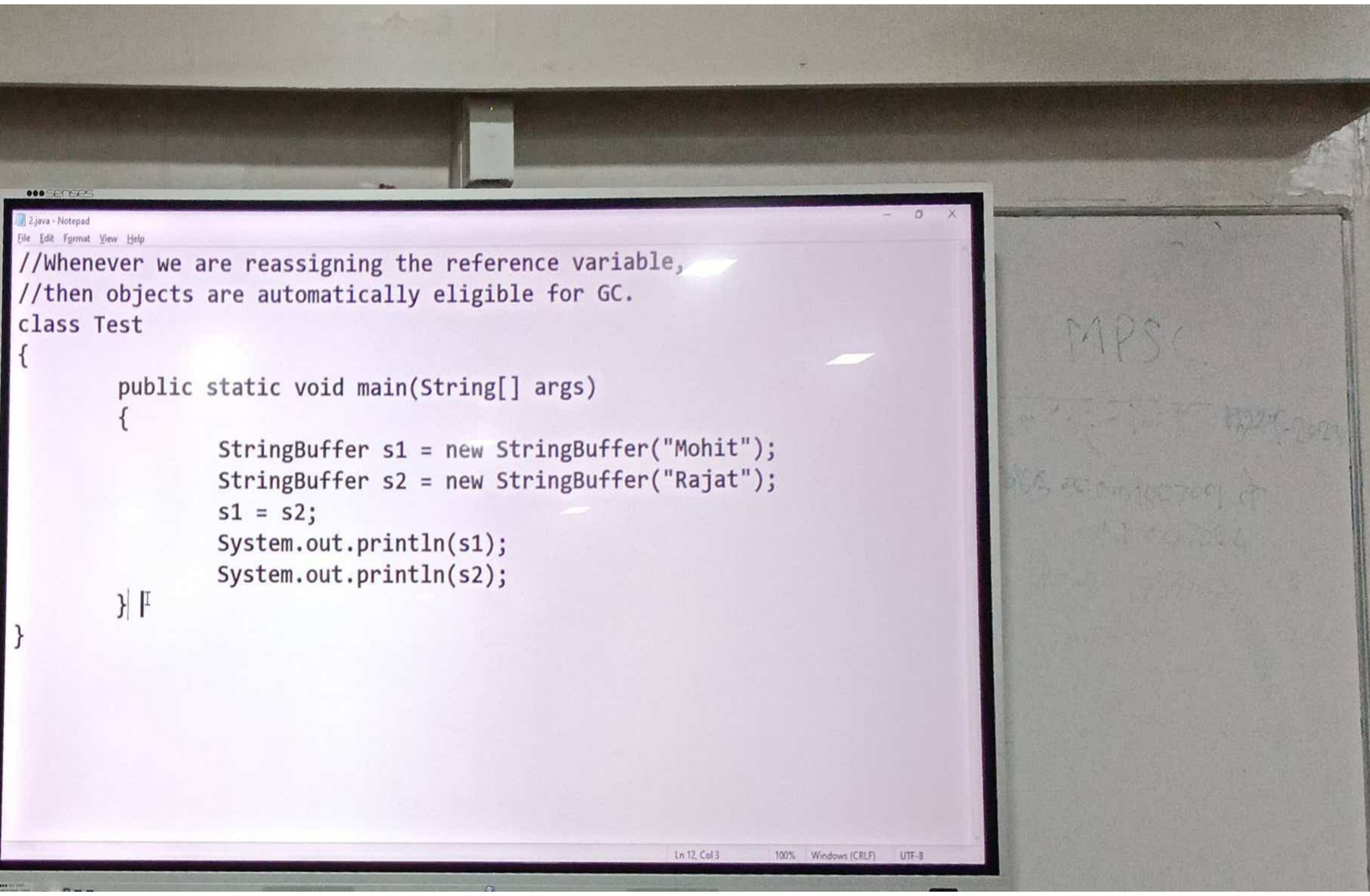
- The process of destroying unreferenced objects is called **garbage collector**.
- When object is unreferenced it is considered as unused object so that JVM automatically destroy that unused object.



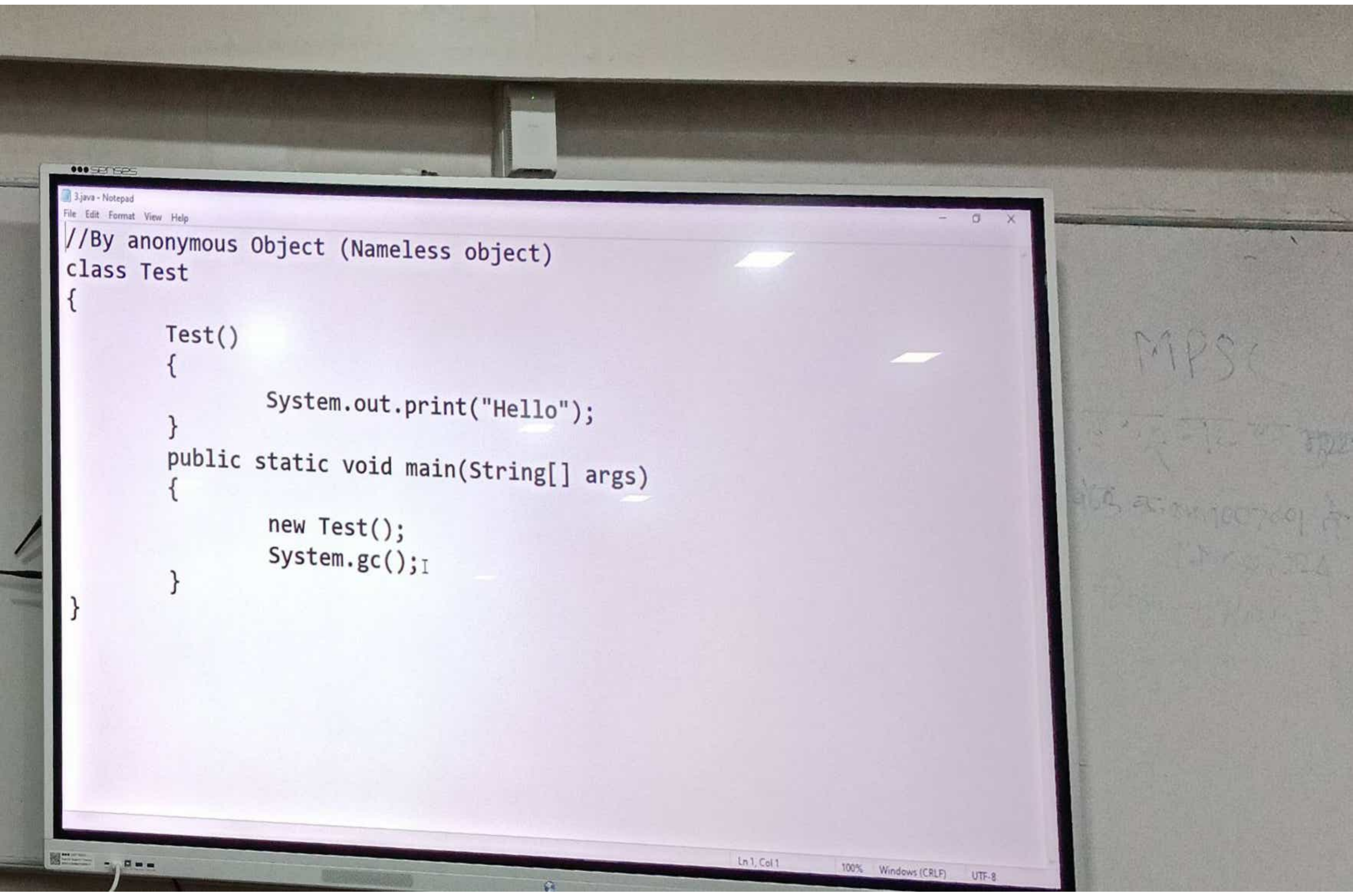
- The process of destroying unreferenced objects is called **garbage collector**.
- When object is unreferenced it is considered as unused object so that JVM automatically destroy that unused object.
- In C language, we allocate memory by using **malloc()** and we destroy that memory by using **free()**, here the developer is responsible for both operations.
- In CPP language, we allocate memory by using **constructors** and we destroy that memory by using **destructors**, here the developer is responsible for both operations.
- In java, developer is responsible to allocate the memory by creating of **object** and memory will be destroyed by **garbage collector** and it is a part of the JVM



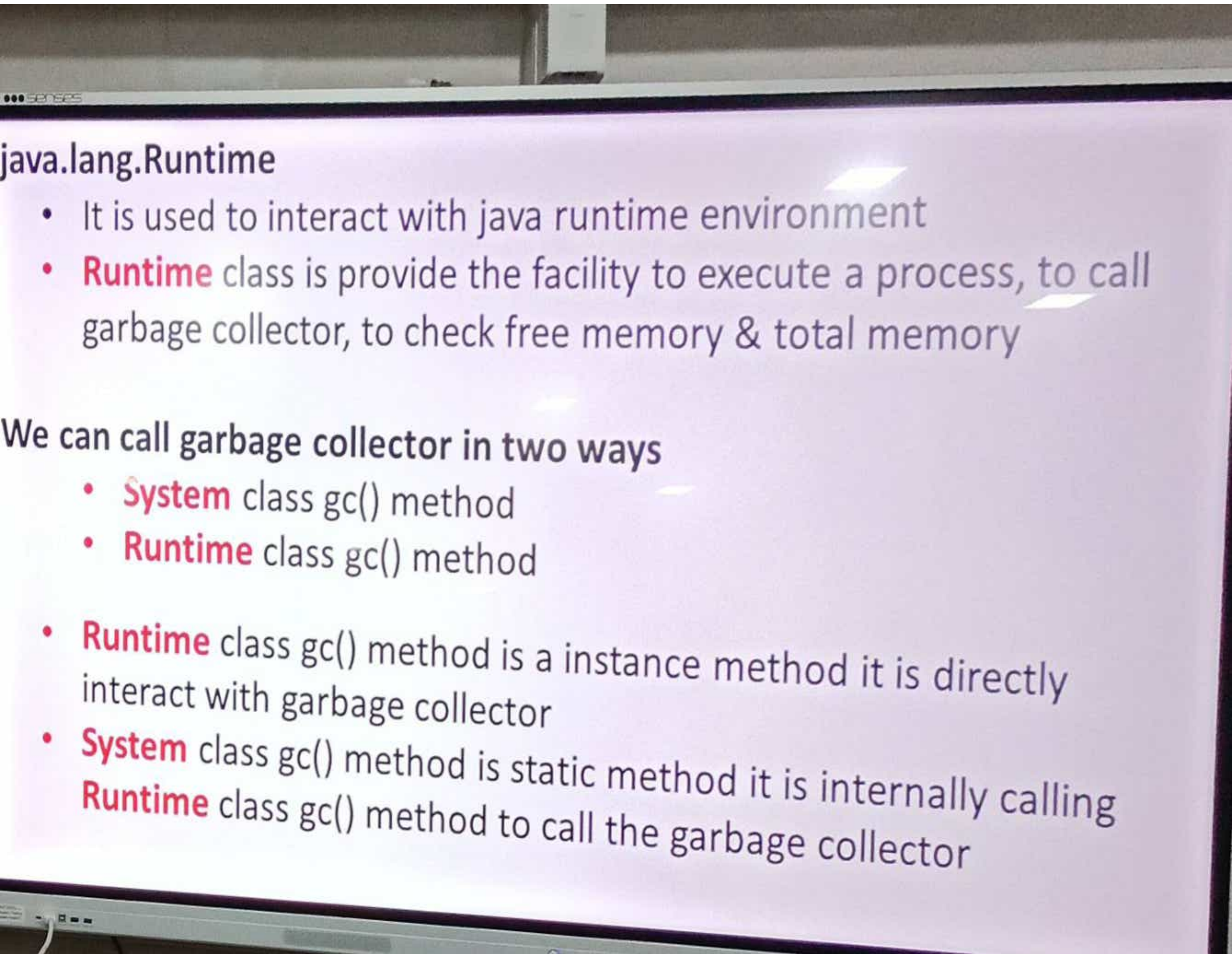
```
//Whenever we are assigning null constants to objects then objects
//are eligible for Garbage Collector
class Test
{
        public static void main(String[] args)
        {
        Test t1=new Test();
        Test t2=new Test();
        System.out.println(t1);
        System.out.println(t2);
        t1=null;        //t1 object is eligible for Garbage Collector
        t2=null;        //t2 object is eligible for Garbage Collector
        System.out.println(t1);
        System.out.println(t2);
        }
}
```



```
//Whenever we are reassigning the reference variable,
//then objects are automatically eligible for GC.
class Test
{
	public static void main(String[] args)
	{
		StringBuffer s1 = new StringBuffer("Mohit");
		StringBuffer s2 = new StringBuffer("Rajat");
		s1 = s2;
		System.out.println(s1);
		System.out.println(s2);
	}
}
```



```
//By anonymous Object (Nameless object)
class Test
{
        Test()
        {
                System.out.print("Hello");
        }
        public static void main(String[] args)
        {
                new Test();
                System.gc();
        }
}
```



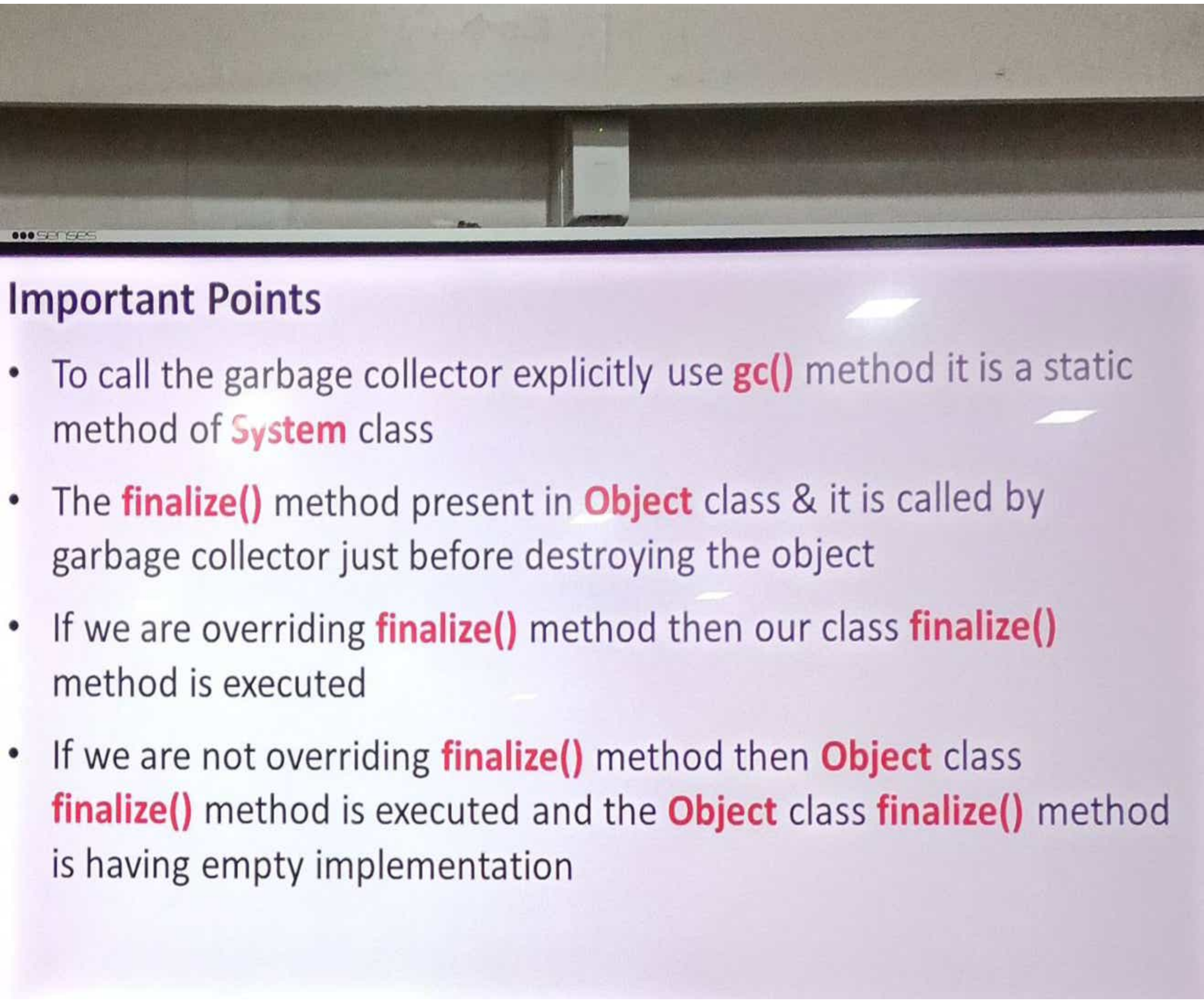
java.lang.Runtime

- It is used to interact with java runtime environment
- **Runtime** class is provide the facility to execute a process, to call garbage collector, to check free memory & total memory

We can call garbage collector in two ways

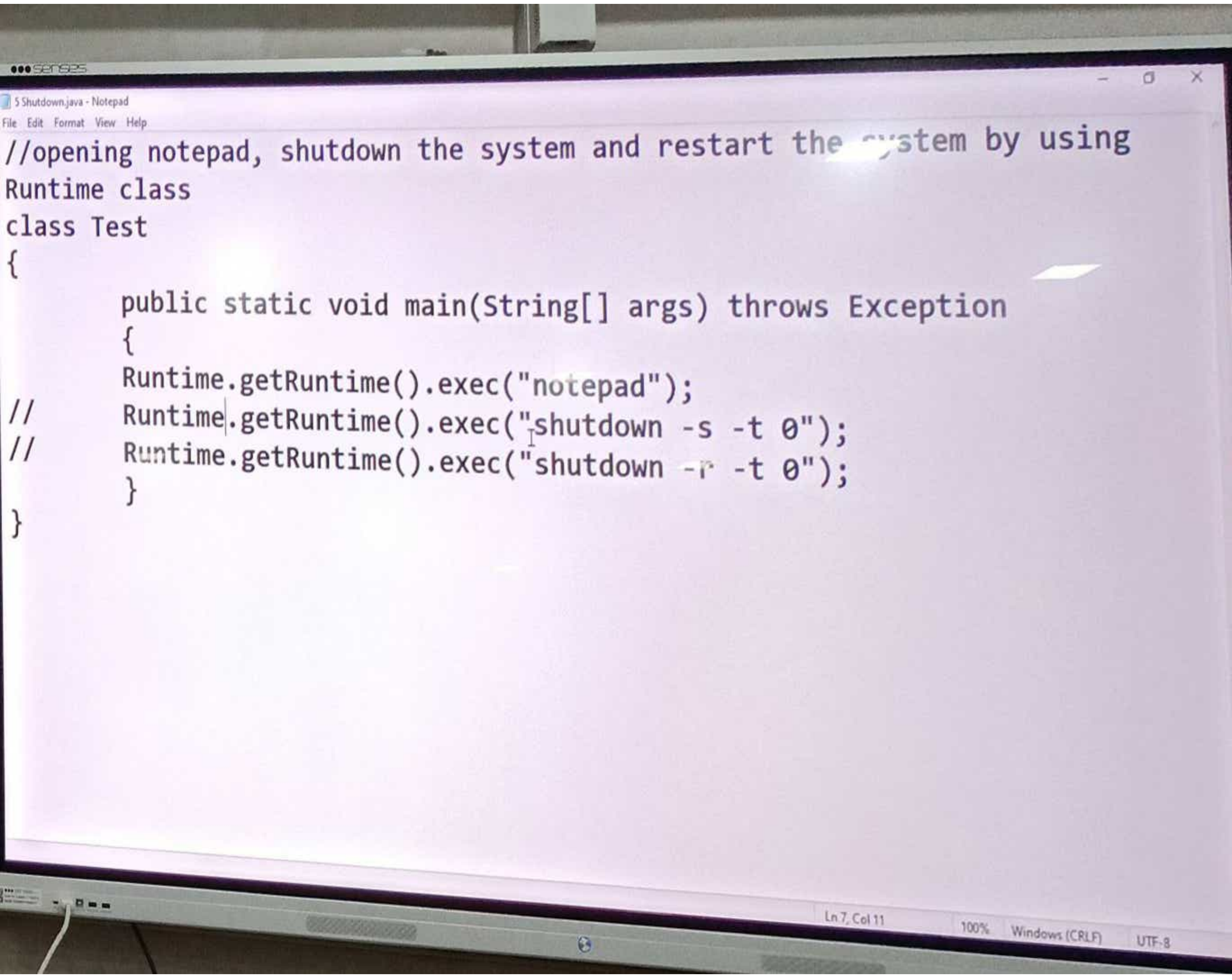
- **System** class gc() method
- **Runtime** class gc() method

- **Runtime** class gc() method is a instance method it is directly interact with garbage collector
- **System** class gc() method is static method it is internally calling **Runtime** class gc() method to call the garbage collector



Important Points

- To call the garbage collector explicitly use **gc()** method it is a static method of **System** class
- The **finalize()** method present in **Object** class & it is called by garbage collector just before destroying the object
- If we are overriding **finalize()** method then our class **finalize()** method is executed
- If we are not overriding **finalize()** method then **Object** class **finalize()** method is executed and the **Object** class **finalize()** method is having empty implementation



```
//opening notepad, shutdown the system and restart the system by using
Runtime class
class Test
{
	public static void main(String[] args) throws Exception
	{
	Runtime.getRuntime().exec("notepad");
//	Runtime.getRuntime().exec("shutdown -s -t 0");
//	Runtime.getRuntime().exec("shutdown -r -t 0");
	}
}
```