Destructor

We will cover the following

- What is a destructor?
- Advantages of destructor
- How does java Destructor work?
- How finalize() method works as a Destructor?
- Example of Destructor

What is a Destructor?

In Java, when we create an object of the class. It occupies space in the memory (heap). If we do not delete these objects, they will remain there in the memory and occupy some space that is not useful from programming aspects. In order to resolve this problem, we use the concept of **destructor**.

In this section, we will look over the alternate option to the **destructor in Java**. We will also learn how we can use the **finalize()** method as a destructor.
The **destructor** is just the opposite of the constructor. The constructor is used to initialize java objects, while the destructor is used to destroy the object in order to release the resource and memory occupied by the object.

You need to remember that **there is no concept of destructor in Java**. Instead of the destructor, Java provides an alternative as the garbage collector works the same as the destructor in any other programming language. The garbage collector is a thread/program that runs on the Java Virtual Machine. It automatically destroys/deletes the unused objects (objects no longer used in the program) and frees up the memory space. The programmer need not worry about memory management manually. It can be error-prone, vulnerable, and may lead to a memory leak.

Advantages of Destructor

- It releases the resources occupied by the object.
- No need to call explicitly, it is automatically invoked at the end of the execution of the program.
- it cannot be overloaded because does not accept any parameter.

How does java destructor work?

When we create the object, it occupies the space in the heap memory area. In the program, threads use these objects. If the thread no longer uses the objects, it becomes eligible for deletion/garbage collection. The memory occupied by that object gets available for new objects created in the program. when the garbage collector destroys the object from the heap, the JRE(Java Runtime Environment) calls the finalize() method to close the connections such as network connection and database connection.

How finalize() Method works as a Destructor

It is difficult for the programmers to forcefully execute the garbage collector to destroy the object from the heap. But Java provides an alternative method to do the same thing. The Java Object class (parent class of all the classes in Java) provides the **finalize()** method that works the same as the destructor in other programming languages. The syntax of the finalize() method is given below:

Syntax:

```
1. protected void finalize throws Throwable()
2. {
3. //resources to be close eg. DB connection or network connection
4. }
```

It is not exactly a destructor, but it provides extra security. It ensures the use of external resources in the program, like closing the file, etc., before closing the program. We can call it explicitly by using the method itself or invoking the method predefined in the java **System.runFinalizersOnExit(true)**.

- It is the protected method of the Object class defined in Java.lang package.
- It can be called only once in the program.
- We have to call finalize() method explicitly if we want to override the method in the program.
- The gc() is a method of JVM executed by the Garbage Collector in Java. It gets invoked when the heap memory is full and requires more memory for new objects that are being created in the memory.
- the JVM ignores all the exceptions that occur by the finalize() method except the unchecked exceptions,

Example of Destructor

DestructorExample.java

```java
public class DestructorExample
{

    public static void main (String[]args) {

    DestructorExample de = new DestructorExample ();

    de.finalize ();

    de = null;

    System.gc ();

    System.out.println ("Inside the main() method");

    }

    protected void finalize () {

    System.out.println ("Object is destroyed by the Garbage Collector");

    }
}

```

input

```
Object is destroyed by the Garbage Collector
Inside the main() method
Object is destroyed by the Garbage Collector


...Program finished with exit code 0
```