Dispose () VS Finalize ()

# Dispose()

**Dispose** is usually called by your code. The resources are freed instantly when you call it. People forget to call the method. **using() {}** statement comes handy. When your program finishes the execution of the code inside the {}, it will call Dispose method automatically.

# Finalize()

**Finalize()** is not called by your code. It is meant to be called by the Garbage Collector (GC). That means the resource might be freed anytime in future whenever GC decides to do so. When GC does its work, it will go through many Finalize methods.

If you have heavy logic in this, it will make the process slow. It may cause performance issues for your program. So be careful about what you put in there.

Finalizers should always be protected, not public or private so that the method cannot be called from the application's code directly and at the same time, it can make a call to the base.Finalize().

Preferably write most of the destruction logic in Dispose.

|  |  |  |
| --- | --- | --- |
| **Sl** | **Dispose()** | **Finalize()** |
| 1 | Called by the Code. |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |