Worksheet – Resource Management

Lernziele:

Sie verstehen die Funktionsweise des Finalizers und können diese erklären.

Aufgabe 1 – Finalizer

In der folgenden Klasse wird mit dem Destruktor auf Nummer sicher gegangen, dass das zu Beginn der Methode geöffnete File zum Schluss auch wieder geschlossen wird:

```
class LogFileReader
{
    private TextReader logFile = null;

    public LogFileReader(string fileName)
    {
        //open file for reading
        logFile = new StreamReader(fileName);
    }

    ~LogFileReader()
    {
        Console.WriteLine($"Instance { GetHashCode() } of "
                        + $"class <{ GetType() }> is finalizing");

        // make sure file is closed, when object is garbage collected
        logFile.Dispose();
    }
}
```

Testen Sie den Code, in dem Sie eine neue Anwendung "DisposableApp" schreiben und die obige Klasse verwenden. Beurteilen Sie die Implementierung, insbesondere unter den folgenden Aspekten:

- Wird das angestrebte Ziel erreicht?
- Zu welchem Zeitpunkt wird die Datei wieder geschlossen?
- Wie sicher/robust ist die Implementierung?
- Wie könnte die Methode sicherer und robuster gemacht werden?

Hinweis: Sie können eine Garbage Collection mittels `GC.Collect()` forcieren.

Aufgabe 2 – Finalization

Angenommen Sie haben in Ihrer Applikationsklasse aus 1 eine statische Variable des Typs `LogFileReader` definiert:

```
static public LogFileReader globalReference = null;
```

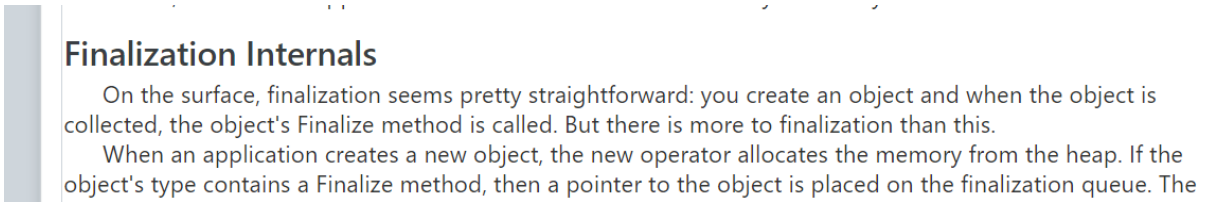
Untersuchen Sie in den folgenden Aufgaben welche Auswirkungen es hat, wenn Sie in der Methode `~LogFileReader()` folgendes Statement hinzufügen?

```
MyApp.globalReference = this;
```

Nehmen wir nun an, dass das Objekt nicht mehr reachable ist (nicht mehr referenziert wird).

1. Wird das `LogFileReader`-Objekt freigegeben? Warum? Wie nennt man das Phänomen?
2. Können Sie das Objekt danach weiterverwenden? Warum/Warum nicht?
3. Angenommen, `MyApp.globalReference` wird wieder auf `null` gesetzt. Wird der Speicher nun freigegeben? Wird der Destruktor ausgeführt? Warum? Wie lässt sich das Verhalten ändern?

Hinweis: Beachten und Lesen Sie dazu den Abschnitt Finalization Internals im Artikel http://web.archive.org/web/20150519133448/https://msdn.microsoft.com/en-us/magazine/bb985010.aspx



Finalization Internals

On the surface, finalization seems pretty straightforward: you create an object and when the object is collected, the object's Finalize method is called. But there is more to finalization than this.

When an application creates a new object, the new operator allocates the memory from the heap. If the object's type contains a Finalize method, then a pointer to the object is placed on the finalization queue. The