Практическая работа №10

Шаблон dispose

Цель работы: изучить работу с неуправляемыми ресурсами в языке C#.

Теоретическая часть

Сборщик мусора связан с управляемыми объектами. Он не знает, как освободить ресурсы, связанные с неуправляемыми объектами. Если в классе существует ссылка на неуправляемые ресурсы, при удалении последней ссылки на класс, неуправляемый объект не будет уничтожен. Операционная система не сможет очистить ресурсы до тех пор, пока приложение не завершится.

Шаблон `dispose` является шаблоном проектирования, позволяющим высвободить неуправляемые ресурсы, используемые классом, контролируемо и своевременно. Реализация в типе этого шаблона будет способствовать тому, что приложения будут хорошо работать и не сохранять неуправляемые ресурсы дольше, чем это необходимо.

Интерфейс `IDisposable` определяет единственный метод `Dispose`, не принимающий никаких параметров. Метод `Dispose` должен освободить все неуправляемые ресурсы, принадлежащие объекту. Он также должен освободить все ресурсы, принадлежащие его базовым типам, вызовом метода `Dispose` родительского типа.

Когда реализуется поддержка интерфейса `IDisposable`, предполагается, что после завершения работы с объектом метод `Dispose` должен вручную вызываться пользователем этого объекта, прежде чем объектной ссылке будет позволено покинуть область видимости.

Благодаря этому объект может выполнять любую необходимую очистку неуправляемых ресурсов без попадания в очередь финализации и без

ожидания того, когда сборщик мусора запустит содержащуюся в классе логику финализации.

При создании собственных классов, ссылаемых на неуправляемые типы, необходимо реализовать интерфейс `IDisposable`. Для любого создаваемого напрямую объекта, если он поддерживает интерфейс `IDisposable`, следует всегда вызывать метод `Dispose`.

Если нужно гарантировать, чтобы метод `Dispose` вызывался всегда, можно включить его в качестве части процесса завершения, выполняемого сборщиком мусора. Для этого, можно добавить деструктор для своего класса и вызвать в нем метод `Dispose`.

Практическая часть

1) Откройте существующий файл (используйте `FileClass` из ЛР 10) и обнулите ссылку в переменной экземпляра. Создайте новый экземпляр класса для работы с текстовым файлом, укажите тот же путь к файлу (без использования методов `Close` и `Dispose`).
2) Реализуйте в разработанном классе интерфейс `IDisposable`.
3) Повторите п.1, но теперь используйте метод `Dispose`.
4) Создайте экземпляр класса для работы с новым текстовым файлом.
5) Сохраните в файл строку и освободите ресурсы с применением `try/finally`.
6) Откройте существующий файл, выведите его содержимое в консоль и освободите ресурсы с применением `using`.
7) Изучите работу сборщика мусора на примере методов `GC.GetTotalMemory` и `GC.Collect`. Нужно получить информацию о размере кучи (за исключением фрагментации), затем создать несколько объектов (любого типа), снова проверить

размер кучи, запустить сборку мусора и снова получить размер кучи.