Темы урока

Распространенные интерфейсы .NET

IDisposable

Интерфейс IDisposable предоставляет механизм для освобождения неуправляемых ресурсов. Например, при работе с файловой системой или базами данных.

Пример: **L14_C01_interface_IDisposable**.

Конструкция using

Вместо явного использования метода Dispose() можно воспользоваться конструкцией using.

Пример: **L14_C02_interface_IDisposable_using**

Самостоятельная работа

Показать задание (на слайде).
Решение: **L14_C03_interface_IDisposable_using_SW**

IEnumerable

Предоставляет перечислитель, который поддерживает простой перебор элементов в указанной коллекции.

Требует реализации двух перегрузок метода GetEnumerator().

Благодаря его реализации мы можем перебирать значения последовательности в цикле foreach.

Пример: **L14_C04_interface_IEnumerable**

Самостоятельная работа

Показать задание (на слайде).
Решение: **L14_C05_interface_IEnumerable_SW**

Статические члены в обычных классах

Статические методы, свойства, поля

- Кроме обычных полей, методов, свойств класс может иметь статические поля, методы, свойства. Статические поля, методы, свойства относятся ко всему классу.
- Для обращения к подобным членам класса не обязательно создавать экземпляр класса.
- Статические члены могут иметь доступ к другим членам класса, только если они тоже статические.
- На уровне памяти для статических полей будет создаваться участок в памяти, который будет общим для всех объектов класса. При этом память для статических переменных выделяется даже в том случае, если не создано ни одного объекта этого класса.

Статический конструктор

Кроме обычных конструкторов у класса также могут быть статические конструкторы. Статические конструкторы имеют следующие отличительные черты:

- Статические конструкторы не должны иметь модификатор доступа и не принимают параметров,
- Как и в статических методах, в статических конструкторах нельзя использовать ключевое слово this для ссылки на текущий объект класса и можно обращаться только к статическим членам класса,
- Статические конструкторы нельзя вызвать в программе вручную. Они выполняются автоматически при самом первом создании объекта данного класса или при первом обращении к его статическим членам (если таковые имеются)

Базовый пример: **L14_C06_static_members_in_regular_classes_demo**

Примеры:

- **L14_C07_static_members_in_regular_classes**
- **L14_C08_static_members_in_regular_classes_final**

Самостоятельная работа

Добавьте в класс ErrorList

- публичное статическое свойство **OutputPrefixFormat** типа string, которое будет позволять просматривать и задавать префиксную строку для вывода
- **статический конструктор** в котором бы определялось начальное значение свойства OutputPrefixFormat как дата и время по заданному формату согласно примеру:
  - April 7, 2019 (7:55 PM)
- обычный публичный метод void **WriteToConsole()**, который бы выводил в консоль все сообщения об ошибках вместе с префиксом, формирующимся через `DateTime.Now.ToString(OutputPrefixFormat)`

Пример решения **L14_C09_static_members_in_regular_classes_SW**.

Статические классы

Вводная часть, демонстрация

Статические классы объявляются с модификатором static и могут содержать только статические поля, свойства и методы. Например, если бы класс имел бы только статические переменные, свойства и методы, то его можно было бы объявить как статический.

Рассмотреть пример **L14_C10_static_class_demo**.

В C# показательными примерами статических классов являются классы

- Console, который используется для вывода данных в консоль
- Math, который применяется для различных математических операций

Базовый пример: **L14_C10_static_class_demo**

Примеры:

- **L14_C11_static_class**
- **L14_C12_static_class_final**

Самостоятельная работа

Сделайте класс ErrorList статическим.

Пример решения **L14_C13_static_class_SW**.

Синглтон

Одиночка (Singleton, Синглтон) - порождающий паттерн, который гарантирует, что для определенного класса будет создан только один объект, а также предоставит к этому объекту точку доступа.
Когда надо использовать Синглтон? Когда необходимо, чтобы для класса существовал только один экземпляр
Синглтон позволяет создать объект только при его необходимости. Если объект не нужен, то он не будет создан. В этом отличие синглтона от глобальных переменных.
Классическая реализация данного шаблона проектирования на C# выглядит следующим образом (см. следующий слайд)

Рассмотреть пример **L14_C14_singleton_demo**.

Домашнее задание

Реализовать **паттерн синглтон** в классах с прошлого домашнего задания:

- **ConsoleLogWriter**,
- **FileLogWriter**,
- **MultipleLogWriter**.