Delegaty

- Z klasy Car usuń referencję do klasy CarFactory zachowując jej poprawne działanie
- Użyj delegatów zamiast bezpośrednich odwołań do CarFactory
- Przykładowy konstruktor klasy Car
  - Car(string name, Action countCarUsages, Func<int> speedLimitFunc)

Zdarzenia (Events)

- Z klasy Car usuń referencję do klasy CarFactory zachowując jej poprawne działanie
- Użyj eventów, żeby zliczać wywołanie metody Drive
- W klasie Car dodaj event EventHandler CarUsed, którego wywołanie zwiększać będzie counter

Zdarzenia (Events)

- Zaimplementuj klasę TimeTicker, która wywołuje event regularnie według zadanego kwantu czasu
  - event Tick
    - Delegate void TimeHandler(object obj, DataTimeEventArgs e)
    - DataTimeEventArgs posiada właściowość TriggeredDateTime typu DateTime
  - Start(int milisecondsInterval) – Thread.Sleep(int), Task.Delay(int).Wait()
  - Stop()

Metody rozszerzeń

- Zaimplementuj metodę rozszerzoną GetFirstOrEmptyIfNullOrEmpty dla obiektów IEnumerable<string>
- Zaimplementuj metodę rozszerzoną zliczającą liczbę słów w stringu
- Zaimplementuj metodę rozszerzoną konwertującą stringa do enuma
    - public static T ToEnum<T>(this string value, bool ignorecase)

Refleksja

- Wypisz składowe assembly
- Aplikacja powinna na wejściu dostać assembly do załadowania
- Na początku wypisze podstawowe informacje o Assembly
- Na podstawie danego assembly wypisze wszystkie dostępne publiczne klasy wraz z sygnaturami:
    - Konstruktorów
    - Metod
    - Właściwości
- Klasy pogrupowane będą po namespace

Plugin

- Stwórz prostą aplikację, która sprawdza obecność dostępnych pakietów w danym folderze i wykonuje ich kod
    - Directory.GetFiles(".", "*.dll")
- Dodawane pakiety muszą implementować wspólny interfejs (IPlugin)
- Po znalezieniu pasującego pakietu, metoda interfejsu jest wykonywana
    - TypeInfo.ImplementedInterfaces
- Na konsoli wypisz nazwę każdego znalezionego pakietu
    - Assembly.CreateInstance().GetText()

Atrybuty

- Stwórz atrybut DescAttribute, który umozliwia dodawać opis do klasy, metody oraz właściwości
- Stwórz atrybut TestMethodAttribute, który zdefiniować można na metodzie
  - Takich atrybutów można zdefiniować kilka w jednej metodzie
  - Do atrybutu można przekazać parametry wywołania metody
- Stwórz atrybut TestClassAttribute, który można przypisać do klasy

- Napisz metodę która odpala każdą metodę z klasy oznaczoną atrybutem TestMethodAttribute, wypisuje jej opis i wynik każdej

Dekompilacja plików

- Użyj programu ILSpy.exe (lub ILDasm.exe) do podejrzenia dowolnego skompilowanego programu pod .NET
    - Jakie elementy kodu są widoczne
    - Jakie elementy są niejawne

Debugowanie

- Przeanalizuj StackTrace
- Podłącz się z debuggerem VS pod plik wykonywalny
    - Co dają pliki.pdb?
- Release vs Debug – jakie są różnice

Client-Server (WCF)

- Napisać aplikację typu klient-serwer bazującą na WCF
- Wykorzystać httpBinding

- Klient wysyła id jako parametr
- Serwer odpowiada zwracając obiekt User dla zadanego id

Client-Server (REST)

- Utwórz nowy kontroler, którego:
  - metoda będzie zwracała string “Hello World”
  - będzie zwracała przekazany string wraz z liczbą znalezionych w nim wyrazów
- Utwórz klienta, który:
  - Wykona obie metody I wynik wypisze na konsoli

Client-Server (gRPC)

- Utwórz nowy serwer, którego metoda:
  - będzie zwracała string “Hello World”
  - będzie zwracała obiekt User dla zadanego Id
- Utwórz klienta, który:
  - wykona obie metody i wynik wypisze na konsoli

Security – szyfrowanie symetryczne

- Zaimplementuj klasę wykorzystującą algorytm szyfrowania synchronicznego (AES, lub DES)
- Klasa powinna posiadać dwie metody publiczne
    - Encrypt
    - Decrypt

Wzorce projektowe – Singleton

- Stwórz klasę LanguageManager implementującą wzorzec Singleton
- Powstać może tylko jedna instancja klasy
- Klasa posiada dwie metody publiczne
    - void SetLanguage(string language)
    - CultureInfo GetLanguage()
- static GetInstance()

Wzorce projektowe – Observer

- Dana jest kolekcja 10 portów,
  - port ma dwie właściwości: IsOpened oraz Number
- Zaimplementuj sytuację gdzie program antywirusowy śledzi stan portu (IsOpened) i każdą zmianę jego stanu zapisuje do pliku jako nowy log
  - Może być kilka programów antywirusowych

Wyrażenia Regularne

- Napisz własne wyrażenia regularne które:
  - Znajdują wyrazy, które zaczynają się z dużej litery.
  - Określają prawidłowy adres email
    - qwerty123@domain.pl (*login@domena.rozszerzenie*)
  - Podmieniają wszystkie pliki z rozszerzeniem .doc lub .xls na pliki z rozszerzeniem odpowiednio .docx oraz .xlsx (dane wejściowe: somedoc.doc somexls.xls, semiCapital.XLs)
  - Dla adresu email definiują jego części składowe (wypisz je): użytkownik, domena, globalna domena

Wyrażenia Regularne

- Napisz własne wyrażenia regularne które:
  - Podmienia wszystkie duże litery na małe, a małe na duże (qweRTY123 => QWErty123)

Wyrażenia Regularne

- Porównaj efektywność znajdowania pasującego tekstu w różnych wywołaniach klasy Regex
- Użyj jednego tekstu wejściowego do znalezienia elementu. Sprawdź rezultat dla 1, 10 i 100 powtórzeń szukania.
- Sprawdź z:
    - Statycznego wywołania Regex.Match()
    - Inicjalizacji klasy: var r = new Regex()
    - Opcji z kompilacją Regex: var r = new Regex(Option.Compile)

info: https://docs.microsoft.com/pl-pl/dotnet/standard/base-types/best-practices

Wątki – Ex

- Przepisz klasę CriticalSectionUnsafe aby uniknąć “race condition”
  - Użyj lock
  - Użyj klasy Interlocked

Zadania - Ex1

- Użyj klasy Task do implementacji sytuacji gdzie zdanie "Hello {Name}!!!" jest wyświetlane na konsoli co sekundę
- Kiedy jakiś przycisk na klawiaturze jest wciśnięty, przestań wypisywać "Hello {Name}!!!“, wypisz raz “Bye” i czekaj na dalszy input od użytkownika

- użyj CancellationToken.

Zadania - Ex2

- Napisz algorytm wykonujący równolegle następujące zadanie
- Każde zadanie sprawdza czy liczba, którą otrzyma w parametrze metody jest liczbą pierwszą (Task<bool>)
  - Var tasks = new List<Task<bool>>()
- Wygeneruj 50 liczb, górna granica wyszukiwania może być ustalana losowo między 10 a 300
- Zlicz wszystkie powodzenia próby ustalenia liczby pierwszej, a wartość zwróć przez zadanie.
- Zlicz sumę dla rezultatów zakończonych powodzeniem i wypisz tę sumę.
- Użyj statycznej metody z klasy Task do synchronizacji. (WaitAll lub WhenAll)

Zadania – Ex3

- Napisz prostą metodę która rzuca wyjątkiem.
- Złap i obsłuż wyjątek wypisując go do konsoli. W tym celu użyj:
  - Klasy Thread
  - Klasy Task.Wait()
  - async/await
  - BeginInvoke/EndInvoke na delegatach
  - Klasy Parallel

Zadania async/await – Ex4 (WinForms)

- Odczytaj plik
- Wypisz jego zawartość na TextBox
- Użyj składni async/await
- Użyj metody ReaderStream.ReadToEndAsync

Button, OpenFileDialog, TextBox

Parallels – Ex

- Get a list of files in C:\Temp\ folder
- Using Parallel.Foreach
    - Print all file names with their size to the console

PLINQ - Ex

- List all files in folder C:\Temp, get each file size and write it to console
- Write total sum of file sizes
- Write average file size
- Write count of files
- Use PLINQ