POLITECHNIKA GDAŃSKA

Platformy Technologiczne
Laboratorium 10

W ramach niniejszego ćwiczenia jako zadanie wprowadzające (niepunktowane, ale niezbędne do realizacji dalszych elementów) należy:

- zaimplementować aplikację *Windows Forms* lub *Windows Presentation Foundation* prezentującą elementy kolekcji *myCars* z laboratorium nr 9 (wrap'owanej przez *BindingList<Car>*) za pomocą kontrolki *DataGridView*;
- kontrolka ma umożliwiać dodawanie, usuwanie i edycję elementów oraz sortowanie po wskazanej kolumnie;
- dodatkowo proszę zapewnić interfejs do wyszukiwania elementów kolekcji. Wykorzystać implementację *Car* i *Engine* z laboratorium nr 9 oraz rozszerzyć klasę *Engine* o implementację *IComparable* porównującą domyślnie po własności *horsePower*;
- powiązanie (bind) kontrolki z użyciem komponentu źródła *BindingSource* można zrealizować w następujący sposób:

```
BindingList<Car> myCarsBindingList = new BindingList<Car>(myCars);
BindingSource carBindingSource = new BindingSource();
carBindingSource.DataSource = myCarsBindingList;
//Przenieś kontrolkę DataGridView z Toolbox'a do aplikacji.
dataGridView1.DataSource = carBindingSource;
```

Następnie należy:

1. Napisać jedno zapytanie LINQ do kolekcji *myCars*, używając:
    - wyrażeń zapytań (query expression syntax) (0,5 pkt);
    - zapytań opartych na metodach (method-based query syntax) (0,5 pkt);

    Uwaga:
    - zapytanie ma zwrócić kolekcję *IEnumerable*, której elementy są typu anonimowego o dwóch własnościach: *engineType* i *avgHPPL*:
        - *engineType* ma wartość "diesel" dla samochodów A6 z silnikiem "TDI" a "petrol" dla pozostałych A6;
        - *avgHPPL* to średnia arytmetyczna z {*horsePower* / *displacement*} dla danego *engineType*.
    - elementy mają być posortowane malejąco po własności *avgHPPL*

    ```
    var elements = ... // Zapytanie LINQ
    foreach (var e in elements) Console.WriteLine(e.engineType + ": " + e.avgHPPL);
    ```

    - wykonanie powyższego kodu powinno wyświetlić:
        - *diesel: 95,25*
        - *petrol: 83,9015873015873*

- *przydatne źródło literaturowe:*

https://learn.microsoft.com/pl-pl/dotnet/csharp/linq/standard-query-operators

2. Nie używając wyrażeń lambda, dopisz definicje zmiennych *arg1*, *arg2* i *arg3* przed poniższym kodem:

```
myCars.Sort(new Comparison<Car>(arg1));
myCars.FindAll(arg2).ForEach(arg3);
```

   - arg1 ma być instancją delegata *Func*, która sortuje samochody malejąco po mocy silnika (1/3 pkt);
   - arg2 ma być instancją delegata *Predicate*, która zwraca *true* dla samochodów z silnikiem "TDI" (1/3 pkt);
   - arg3 ma być instancją delegata *Action*, która wyświetli każdy element w osobnym *MessageBox* (1/3 pkt).

3. Zaimplementować i zademonstrować subklasę *BindingList<T>* umożliwiającą:
   - sortowanie elementów po dowolnej własności typu *K*, pod warunkiem, że *K* implementuje *IComparable* (1 pkt);
     wskazówka:

     ```
     if (property.PropertyType.GetInterface("IComparable") != null) ...
     ```

   - wyszukanie elementu po własności typu *string* lub *Int32* (0,5 pkt);
     wskazówka: przedefiniować metody *SupportsSortingCore*, *ApplySortCore*, *SupportsSearchingCore* i *FindCore* ☺ .

4. Stworzyć:
   - funkcjonalność dodawania, usuwania i edycji elementów kolekcji poprzez *DataGridView* (0,5 pkt);
   - generyczny (niezależny od źródła danych powiązanego z *DataGridView*) GUI umożliwiający wyszukiwanie elementów w *DataGridView* po dowolnej własności typu *string* lub *Int32* (1 pkt);
     - przeciągnąć na formularz kontrolkę *ToolStrip*, a następnie "wybrać" z niej elementy *ComboBox*, *TextBox*, *ComboBox* i *Button;*
     - wszystkie własności typu *string* oraz *Int32* powinny zostać załadowane automatycznie do *ComboBox* w momencie zdarzenia *Enter*
     - szukaną wartość dla własności wskazanej w *ComboBox* wpisujemy w pole *TextBox*
     - kliknięcie przycisku *Button* uruchamia proces wyszukiwania - wywołanie metody *Find* w *BindingList*