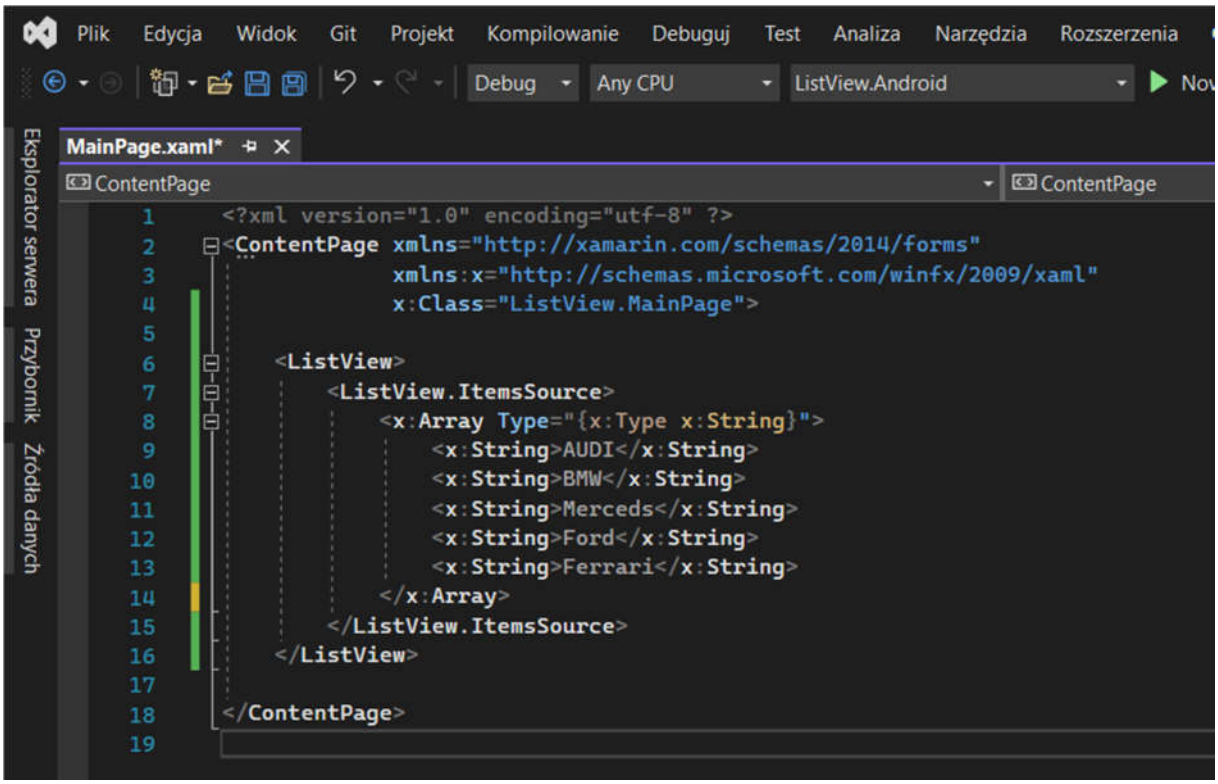
1.Do czego wykorzystuje się ListView?

2.Za co odpowiada znacznik ListView ? Jakie właściwości może obsłużyć?

3.Przepisz i przeanalizuj poniższy kod. Za co odpowiada kod w <ListView> dokonaj odpowiedniego opisu. Np. za co odpowiada ItemSource



Za co odpowiada: `<x:Array Type="{x:Type x:String}">`

4.Przedstaw efekt swojej pracy na emulatorze UWP oraz Android.

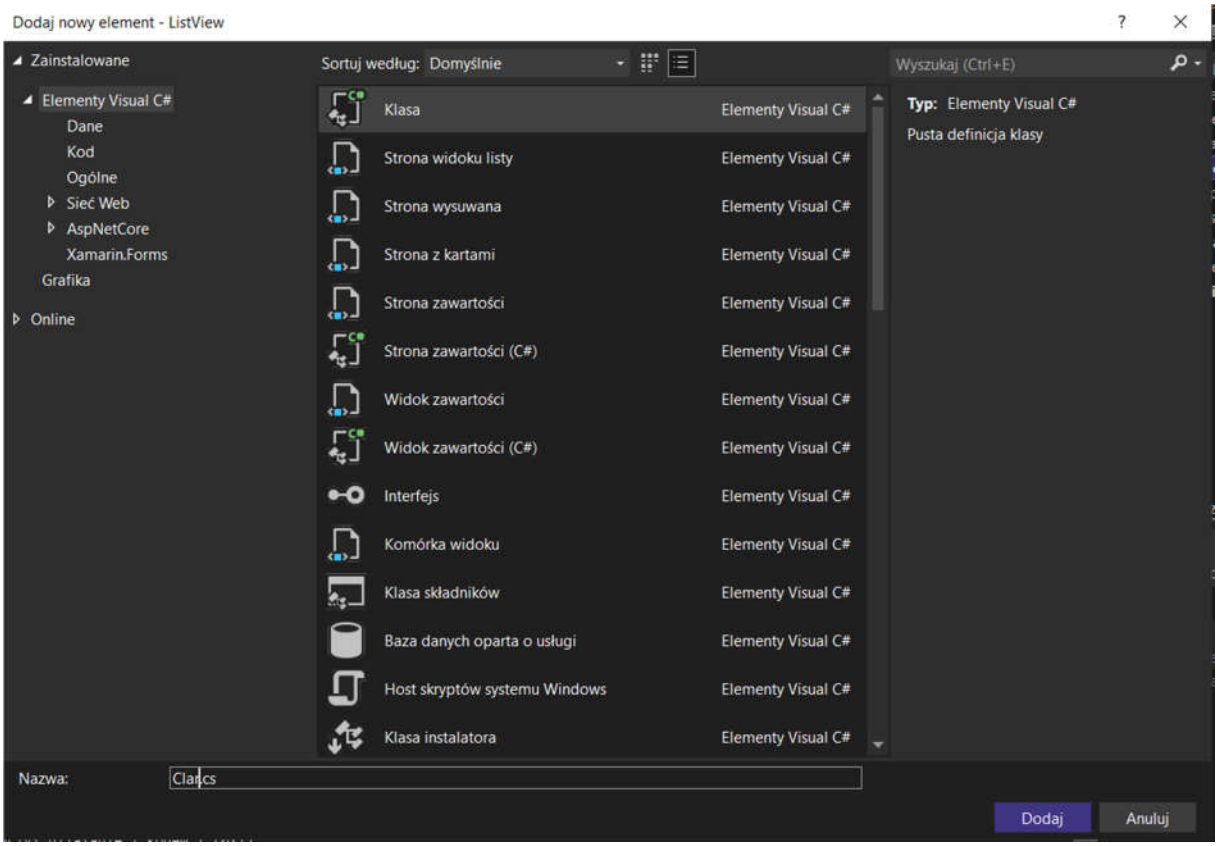
5.W znaczniku ListView możemy modyfikować ustawienia separatora, np. wyłączyć go, zmienić jego kolor.

Zanotuj jak zmienisz kolor separatora:

Za pomocą jakiego polecenia ukryjesz separator:

Dodaj kilka swoich modyfikacji do kodu. Za co one odpowiadają:

6.W kolejnych krokach postaramy się rozszerzyć naszą listę o grupy obiektów, nie tylko tekst. Na początek w SolutionExplorer dodajmy sobie nowy folder o nazwie Models, będziemy w nim przechowywać wszystkie modele danych naszej aplikacji. Tworzymy w nim nową klasę, która będzie reprezentowała nasz samochód. Nazwiemy ją Car.cs



Klasę zmieniamy na publiczną i dodajemy do niej pola Name oraz Year.

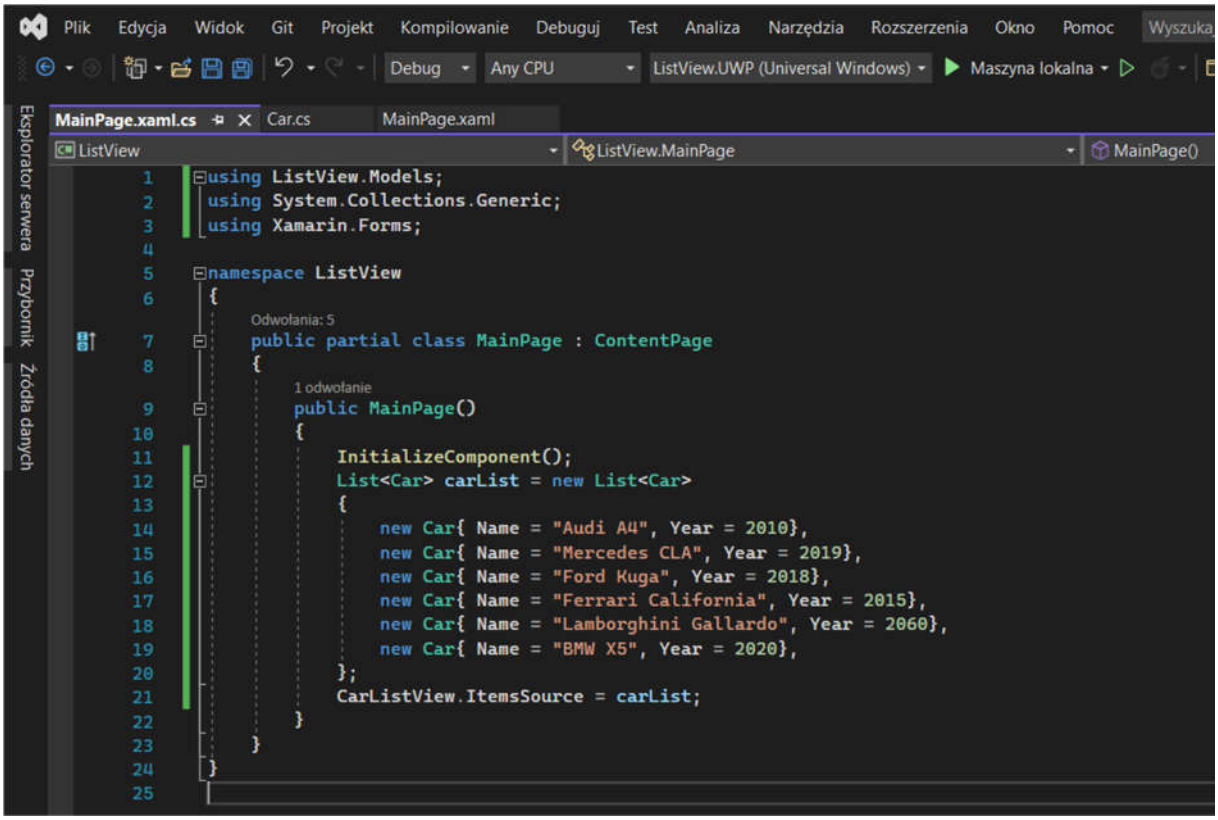
```
using System;
using System.Collections.Generic;
using System.Text;

namespace ListView.Models
{
    public class Car
    {
        public int Name { get; set; }
        public int Year { get; set; }
    }
}
```

7. Za co odpowiadają accessory? W jakim celu je stosujemy? Zanotuj długi i krótki sposób ich zapisu. Stwórz własny oryginalny przykład, aby zaprezentować ich działanie.

W znaczniku ListView kasujemy dotychczasową zawartość. Dodamy mu nazwę abyśmy mogli się do niego odwołać z CodeBehind

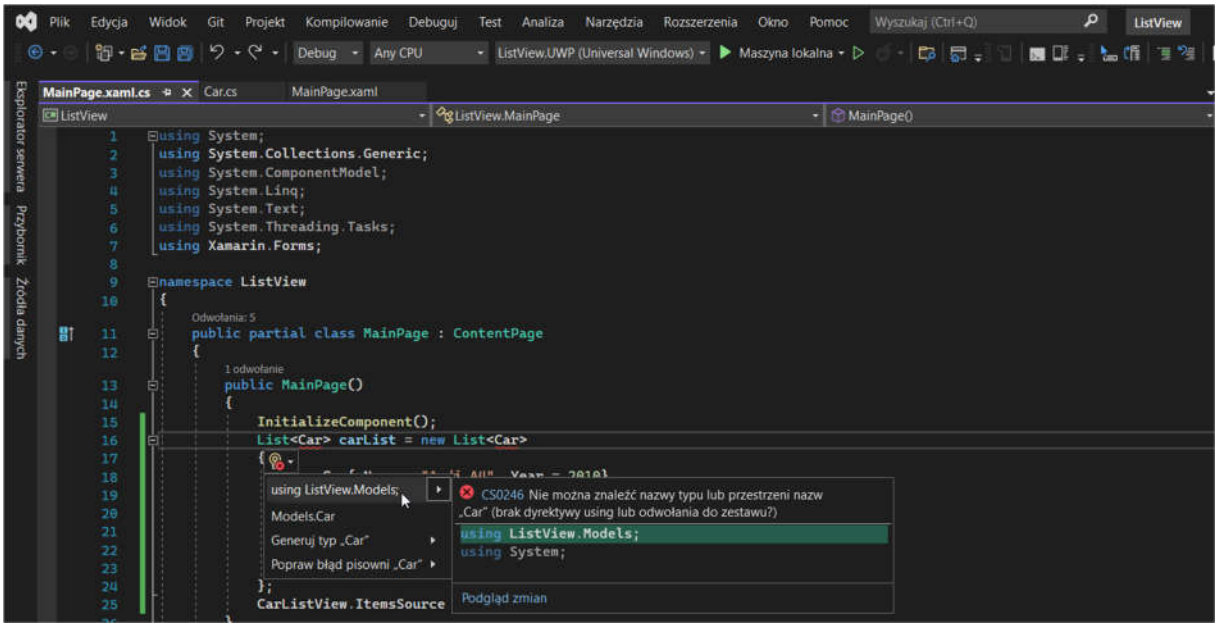
```
<ListView x:Name="CarListView">

</ListView>
```

Przechodzimy do CodeBehind MainPage.xaml.cs gdzie zadeklarujemy sobie nową listę samochodów.



Po raz kolejny przeanalizuj kod, co nowego zostało zastosowane. Dokonaj odpowiedniego opisu. Za co dokładnie odpowiada linijka 21:
CarListView.ItemsSource = carList;

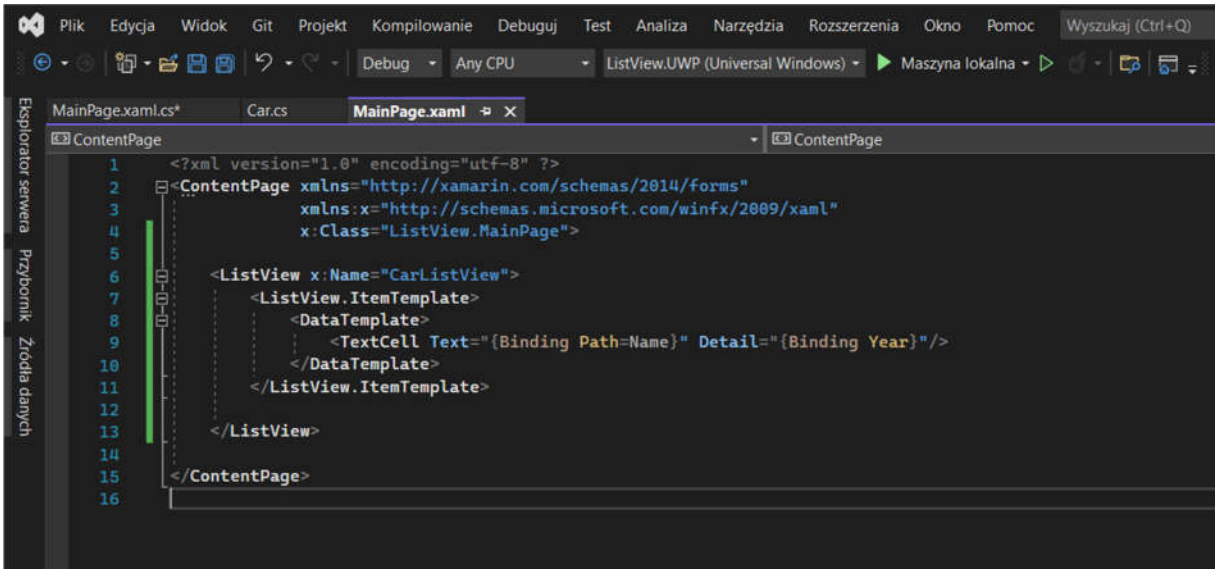
Jeśli korzystasz z VS 2022, popraw następujący błąd:



8. Aby skompilować poprawnie kod należało poprawić inny błąd na czym on polegał ?

9.Uruchom emulatory jaki efekt otrzymałeś/aś ? Czym jest to spowodowane?

10. Dokonujemy kolejenych zmian w kodzie:



Po dopisaniu kolejnych linijek kodu uruchom emulator. Jaki efekt otrzymałeś/aś?

Za co odpowiada komórka `TextCell`

Przedstaw proces Tworzenia, usuwania i odświeżania elementów listy. Można zastosować między innymi funkcjonalność „przeciągnij, aby odświeżyć”.

Jak dostosować wygląd szablonu wiersza. Platforma Xamarin.Forms zawiera cztery wbudowane typy komórek:

EntryCell — komórka z etykietą i polem do wprowadzania jednego wiersza tekstu;

SwitchCell — komórka z etykietą i przełącznikiem;

TextCell — komórka z tekstem podstawowym i uzupełniającym;

ImageCell — komórka z tekstem i obrazem.

Przetestuj każdą z nich.

Czasami zdarza się jednak, że komórki te nie pozwalają uzyskać pożądanego efektu, a nawet nie wyglądają elegancko. Ale dostępna jest również komórka typu ViewCell, którą można dostosowywać odpowiednio do własnych wymagań.

Aby dostosować wygląd wierszy kontrolki ListView, należy zastosować znaczniki ItemTemplate i DataTemplate. Wewnątrz znacznika DataTemplate muszą być umieszczone znaczniki Cell.

Za pomocą znacznika ImageCell można łatwo umieścić w jednym wierszu zdjęcie, nazwę i opis. Wewnątrz tego znacznika wiązane są właściwości obiektu stanowiącego źródło

danych wskazanego we właściwości ItemsSource: właściwość Text jest wiązana z właściwością Name, Detail ze Species, a ImageSource z ImageUrl.

Kontrolka ListView ma nagłówek i stopkę, które definiuje się odpowiednio za pomocą właściwości Header i Footer. Można im przypisać wartości bezpośrednio lub powiązać z tekstowym

źródłem danych. W efekcie u góry i u dołu kontrolki ListView automatycznie pojawią się kontrolki Label z odpowiednimi wartościami.

Oto jeden z możliwych sposobów definiowania właściwości ListView.Header lub ListView.Footer za pomocą znacznika View (widok):

<ListView.Header>

<ContentView>

<Label Text ="Prosty dostosowany nagłówek!" />

</ContentView>

</ListView.Header>

Grupowanie elementów i tworzenie listy nawigacyjnej:

Grupowanie elementów przydaje się w wielu sytuacjach. Najczęściej stosuje się je na stronach z ustawieniami do dzielenia wierszy listy na różne kategorie. Jednak grupowanie można stosować wobec wszystkich kontrolek, pomiędzy którymi istnieje zależność rodzic – potomek. W ten sposób można na przykład wyświetlać zamówienia będące grupami pozycji.

Kontrolka ListView umożliwia grupowanie wierszy. W tym celu należy jej właściwości IsGrouping Enabled nadać wartość true. Dodatkowo należy określić wartość właściwości GroupDisplayBinding definiującej klucz grupujący. Każdy element grupowanej listy musi być listą.