Lista z ObservableCollection — ToDo

Czas: 2 × 45 min | praca indywidualna | Wykład 2.02

Cel: Zbudujesz listę zadań z dodawaniem, usuwaniem i oznaczaniem wykonanych, korzystając z ObservableCollection, szablonu elementu i licznika aktualizowanego automatycznie.

Po zajęciach potrafisz:

- wyświetla ObservableCollection w ListBox przez ItemsSource
- projektuje DataTemplate z wiązaniami do elementu
- reaguje na zmiany wewnątrz elementów kolekcji
- bezpiecznie usuwa wiele elementów z kolekcji

Sprzęt i materiały

- komputer z Windows 11 i dostępem do internetu
- pendrive lub konto w repozytorium do zapisania projektu

Oprogramowanie

- **Visual Studio Community 2026 (obciążenie „Programowanie aplikacji klasycznych platformy .NET")** — darmowe dla uczniów i nauczycieli; w pracowni zainstalowane visualstudio.microsoft.com/pl/downloads
- **.NET 10 SDK (LTS)** — instalowany razem z Visual Studio; sprawdzenie: `dotnet --version` dotnet.microsoft.com/pl-pl/download/dotnet/10.0

Bezpieczeństwo

- Zapisuj projekt w swoim folderze (np. `C:\Users\<login>\source\repos`), a na koniec zajęć skopiuj go na pendrive lub wyślij do repozytorium.
- Nie uruchamiaj programów i pakietów z nieznanych źródeł; pakiety NuGet instaluj tylko te, które wskazuje zadanie.
- Rób przerwy dla oczu (zasada 20-20-20) — praca z projektantem okien wymaga długiego patrzenia w ekran.

Przebieg zadania

1 Model

Utwórz projekt WPF `ToDo`.

1. Klasa `Zadanie` (dziedziczy po ObservableBase): `Tytul`, `Termin`, `Wykonane` z powiadomieniem.
2. Klasa `ListaZadan`: `ObservableCollection<Zadanie> Zadania`, `NowyTytul`, `NowyTermin`, `Wybrane`.
3. Metody `Dodaj()` (odrzuca pusty tytuł, czyści pole), `UsunWybrane()`, `UsunWykonane()` (zwraca liczbę usuniętych).
4. Właściwość `DoZrobienia` — liczba niewykonanych.

2 Licznik na żywo

Licznik musi się zmieniać także po zaznaczeniu CheckBoxa w liście.

```
foreach (var z in Zadania.Where(z => z.Wykonane).ToList())
    Zadania.Remove(z);
```

1. Podłącz `Zadania.CollectionChanged` i w nim subskrybuj `PropertyChanged` nowych elementów (i odłączaj usuniętych).
2. Przy zmianie `Wykonane` zgłaszaj `OnPropertyChanged(nameof(DoZrobienia))`.
3. W `UsunWykonane` usuwaj z kopii listy (`.ToList()`) — sprawdź, co się dzieje bez kopii.

3 Okno

Interfejs.

1. Górny pasek: TextBox tytułu, DatePicker terminu, przycisk Dodaj (IsDefault).
2. ListBox z `ItemTemplate`: CheckBox, tytuł, termin w formacie dd.MM.yyyy w kolorze szarym.
3. Dolny pasek: licznik „Do zrobienia: N”, przyciski Usuń zaznaczone i Usuń wykonane.
4. Przyciski obsłuż w code-behind jedną linijką wywołującą metodę modelu (w zadaniu 6 zamienimy je na komendy).

4 Testy ręczne

Sprawdź działanie.

1. Dodaj 5 zadań, zaznacz 2 jako wykonane, usuń wykonane — licznik i lista muszą się zgadzać.
2. Spróbuj dodać pusty tytuł i tytuł z samych spacji.
3. Zatwierdź pracę w Git.

W sprawozdaniu umieść

- ☐ kod klas Zadanie i ListaZadan
- ☐ XAML z DataTemplate
- ☐ opis błędu przy usuwaniu bez kopii
- ☐ zrzut okna z kilkoma zadaniami

Pytania kontrolne

1. Dlaczego List<T> nie wystarcza dla ListBox?
2. Co jest DataContextem wewnątrz DataTemplate?
3. Dlaczego licznik wymaga subskrypcji PropertyChanged elementów?
4. Jaki wyjątek zgłasza modyfikacja kolekcji w pętli foreach?

Kryteria oceny

Kryterium	Punkty
Klasy Zadanie i ListaZadan	5
Licznik reagujący na zmiany elementów	4
Okno z DataTemplate	5
Usuwanie (pojedyncze i wykonanych)	3
Pytania	3
Razem	**20**

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).