MVVM od podstaw — kalkulator BMI

Czas: 2 × 45 min | praca indywidualna | Wykład 2.05

Cel: Zbudujesz kalkulator BMI we wzorcu MVVM z własną klasą RelayCommand: okno bez kodu w zdarzeniach, ViewModel bez typów WPF i testy jednostkowe ViewModelu.

Po zajęciach potrafisz:

- dzieli aplikację na View i ViewModel
- implementuje ICommand z CanExecute i CanExecuteChanged
- ustawia DataContext w XAML
- testuje ViewModel testami MSTest bez uruchamiania okna

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 Infrastruktura

Utwórz projekt WPF `Bmi` z folderami `Mvvm` i `ViewModels`.

1. `Mvvm/ObservableBase.cs` — z zadania 1.
2. `Mvvm/RelayCommand.cs` — `Execute`, `CanExecute`, `CanExecuteChanged`, `NotifyCanExecuteChanged()`.

2 ViewModel

`BmiViewModel` — wszystkie dane i operacje.

```
public string Masa
{
    get => _masa;
    set { if (SetProperty(ref _masa, value)) ObliczCommand.NotifyCanExecuteChanged(); }
}
```

1. Właściwości `Masa`, `Wzrost` (string), `Wynik`, `Kategoria` (prywatny setter).
2. Komenda `ObliczCommand` aktywna tylko dla masy 20–300 kg i wzrostu 100–250 cm; zmiana pól wywołuje `NotifyCanExecuteChanged`.
3. Metody statyczne `ObliczBmi` i `KategoriaBmi` (progi 18,5 / 25 / 30) oraz komenda `WyczyscCommand`.
4. Liczby parsuj z kulturą pl-PL (przecinek dziesiętny).

3 Widok

Okno bez obsługi zdarzeń.

1. `<Window.DataContext><vm:BmiViewModel /></Window.DataContext>`.
2. Pola z `UpdateSourceTrigger=PropertyChanged`, przyciski z `Command`, wynik i kategoria w TextBlock.
3. Sprawdź, że `MainWindow.xaml.cs` zawiera tylko konstruktor z `InitializeComponent()`.

4 Testy jednostkowe

Projekt MSTest z odwołaniem do projektu `Bmi`.

1. Test: bez danych komenda nieaktywna; dla 70 kg i 175 cm wynik „BMI 22,9” i „waga prawidłowa”.
2. Test parametryzowany CanExecute (min. 5 przypadków) i granice kategorii (18,4 / 18,5 / 25 / 30).
3. Test, że zmiana masy zgłasza CanExecuteChanged, i test Wyczyść.
4. Zrzut Eksploratora testów. Zatwierdź pracę w Git.

W sprawozdaniu umieść

- ☐ kod RelayCommand i BmiViewModel
- ☐ XAML okna
- ☐ kod testów i zrzut Eksploratora testów
- ☐ uzasadnienie: dlaczego ViewModel da się testować bez okna

Pytania kontrolne

1. Za co odpowiada View, a za co ViewModel?
2. Skąd przycisk wie, że ma się wyszarzyć?
3. Dlaczego projekt testów może używać ViewModelu, a nie może używać okna?
4. Co trzeba zmienić, aby dodać przycisk „Pokaż normy”?

Kryteria oceny

Kryterium	Punkty
RelayCommand	3
BmiViewModel z CanExecute	6
Widok bez kodu zdarzeń	4
Testy (min. 10 przypadków)	5
Pytania	2
Razem	**20**

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