Testy ViewModeli

Czas: 2 × 45 min | praca indywidualna | Wykład 2.18

Cel: Napiszesz testy jednostkowe ViewModeli z zadań 1, 5 i 15 — z powiadomieniami, komendami i atrapami usług — i ocenisz pokrycie kodu.

Po zajęciach potrafisz:

- testuje właściwości, CanExecute i Execute ViewModelu
- sprawdza zgłaszanie PropertyChanged
- zastępuje usługi atrapami przez interfejsy
- analizuje pokrycie kodu i uzupełnia brakujące testy

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

Przygotowanie przed zajęciami

- Projekty z zadań 1, 5 i 15.

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.

Przebieg zadania

1 Projekt testów

Rozwiązanie z projektami aplikacji.

1. Dodaj projekt MSTest (TargetFramework `net10.0-windows`, jeśli odwołuje się do projektu WPF).
2. Odwołania do projektów z zadań 1, 5 i 15 (lub kopie klas ViewModeli do biblioteki klas).

2 Model i BMI

Testy klas bez usług.

```
var zmienione = new List<string?>();
m.PropertyChanged += (s, e) => zmienione.Add(e.PropertyName);
m.Kilometry = 42.195;
CollectionAssert.Contains(zmienione, nameof(PrzelicznikModel.Mile));
```

1. PrzelicznikModel: zmiana km zgłasza Mile i Metry; ta sama wartość — brak powiadomienia; ujemne → 0; °C → °F (DataRow).
2. BmiViewModel: CanExecute dla min. 5 przypadków, wynik dla 70 kg i 175 cm, granice kategorii, Wyczyść.

3 Atrapy

OcenyViewModel z zadania 15.

1. Klasa `AtrapaDialogu(bool odpowiedz)` zapamiętuje komunikaty i pytania.
2. Repozytorium: `PamiecOcenyRepo` z zadania 15.
3. Testy: wczytanie danych w konstruktorze, komenda nieaktywna bez wyboru, zapis oceny i komunikat, ocena 1 z potwierdzeniem i bez (DataRow).

4 Pokrycie

Test → Analizuj pokrycie kodu dla wszystkich testów.

1. Zapisz procent pokrycia klas ViewModeli.
2. Znajdź niepokrytą gałąź i dopisz test.
3. Zrzut wyników. Zatwierdź pracę w Git.

W sprawozdaniu umieść

- ☐ kod klas testowych i atrap
- ☐ zrzut Eksploratora testów
- ☐ zrzut pokrycia przed i po dopisaniu testu
- ☐ wniosek: czego testy nie sprawdzają

Pytania kontrolne

1. Dlaczego atrapa jest lepsza od prawdziwej bazy w teście jednostkowym?
2. Jak sprawdzić, że właściwość zgłosiła PropertyChanged?
3. Co oznacza 100% pokrycia i czego nie gwarantuje?
4. Jak nazywasz testy i dlaczego?

Kryteria oceny

Kryterium	Punkty
Testy modelu przelicznika	4
Testy BmiViewModel	4
Atrapy i testy OcenyViewModel	7
Pokrycie i dopisany test	3
Pytania	2
Razem	**20**

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