Testy jednostkowe MSTest

Czas: 2 × 45 min | praca indywidualna | Wykład 1.21

Cel: Utworzysz projekt testów MSTest dla klas `Kosc` (zadanie 15) i `Matematyka` (zadanie 10), korzystając z testów parametryzowanych i testów wyjątków.

Po zajęciach potrafisz:

- tworzy projekt testowy i dodaje odwołanie do testowanego projektu
- pisze testy według wzorca Arrange–Act–Assert
- stosuje `[DataRow]` do testów parametryzowanych
- testuje zgłaszanie wyjątków `Assert.ThrowsExactly<T>`

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

- Przygotuj projekt z klasą `Kosc` z zadania 15 i klasę `Matematyka` z zadania 10.

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 Struktura rozwiązania

Biblioteka klas + testy.

1. Utwórz rozwiązanie z projektem **Biblioteka klas** `GraLogika` (.NET 10) i skopiuj do niego `Kosc.cs` oraz `Matematyka.cs` (klasa `public static`).
2. Dodaj projekt **Projekt testowy MSTest** `GraLogika.Tests`.
3. W projekcie testów: Dodaj → Odwołanie do projektu → `GraLogika`.
4. Otwórz **Eksplorator testów** (Test → Eksplorator testów) i uruchom przykładowy test z szablonu.

2 Testy klasy Kosc

Klasa `KoscTests` z atrybutem `[TestClass]`.

```
[TestMethod]
[DataRow(new[] { 3, 3, 2, 4, 2 }, 10)]
[DataRow(new[] { 1, 2, 3, 4, 5 }, 0)]
public void Punkty_SumujePowtarzajaceSieWartosci(int[] wartosci, int oczekiwane)
{
    Assert.AreEqual(oczekiwane, Kosc.Punkty(wartosci));
}

[TestMethod]
[DataRow(0)] [DataRow(7)]
public void Ustaw_WartoscSpozaZakresu_ZglaszaWyjatek(int w)
{
    var kosc = new Kosc();
    Assert.ThrowsExactly<ArgumentOutOfRangeException>(() => kosc.Ustaw(w));
}
```

1. Test: 1000 rzutów daje wartości 1–6 (użyj `new Kosc(new Random(1))` — wynik powtarzalny).
2. Test: zablokowana kość nie zmienia wartości.
3. Test: nowa kość ma wartość 0; konstruktor zwiększa `LiczbaKosci`.
4. Test parametryzowany `Ustaw` dla 0, 7 i −1 — oczekiwany `ArgumentOutOfRangeException`.
5. Test parametryzowany `Punkty` dla co najmniej 4 zestawów (szablon).

3 Testy klasy Matematyka

Klasa `MatematykaTests`.

1. `Nwd` dla 4 par, w tym z zerem i liczbą ujemną.
2. `CzyPierwsza` dla 2, 97, 1, 91 i −7.
3. Zgodność `Silnia` i `SilniaRek` dla 0, 1, 5, 15.
4. `MinMax` — dwa asercje w jednym teście.

4 Test, który wykrywa błąd

Sprawdź, czy testy naprawdę działają.

1. Wprowadź celowo błąd do `Kosc.Punkty` (np. `>= 3` zamiast `>= 2`) i uruchom testy.
2. Zrób zrzut Eksploratora testów z czerwonymi testami, popraw błąd i zrób zrzut z zielonymi.
3. Zatwierdź pracę w Git.

W sprawozdaniu umieść

- ☐ struktura rozwiązania (zrzut Eksploratora rozwiązań)
- ☐ kod klas testowych
- ☐ zrzut Eksploratora testów: testy niezaliczone i zaliczone
- ☐ lista przypadków testowych

Pytania kontrolne

1. Co oznacza wzorzec Arrange–Act–Assert?
2. Dlaczego w testach kości przekazujemy `Random` z ziarnem?
3. Po co projekt logiki jest osobną biblioteką klas?
4. Co powinien sprawdzać dobry test — przypadki typowe, graniczne czy błędne? Uzasadnij.

Kryteria oceny

Kryterium	Punkty
Struktura rozwiązania i odwołanie	3
Testy Kosc (min. 6, w tym DataRow i wyjątek)	6
Testy Matematyka (min. 4)	4
Wykrycie celowego błędu (zrzuty)	3
Pytania	4
Razem	**20**

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