Właściwości i składowe statyczne — klasa Kosc

Czas: 2 × 45 min | praca indywidualna | Wykład 1.14

Cel: Napiszesz klasę `Kosc` w stylu zadania egzaminacyjnego INF.04 — z polem statycznym liczącym obiekty, właściwościami i dokumentacją — oraz program testowy gry w kości.

Po zajęciach potrafisz:

- definiuje właściwości z prywatnym setterem i właściwości automatyczne
- używa składowej statycznej wspólnej dla wszystkich obiektów
- tworzy przeciążone konstruktory z wywołaniem `this(...)`
- dokumentuje klasę komentarzami `///` jak na egzaminie

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.

Przebieg zadania

1 Klasa Kosc

Utwórz projekt `GraWKosci` i plik `Kosc.cs` (Projekt → Dodaj klasę).

```
public class Kosc
{
    private static readonly Random Losowanie = new();
    private readonly Random _los;
    public static int LiczbaKosci { get; private set; }
    public int Wartosc { get; private set; }
    public bool Zablokowana { get; set; }

    public Kosc() : this(Losowanie) { }
    public Kosc(Random los) { _los = los; LiczbaKosci++; }
    // ...
}
```

1. Właściwość `Wartosc` (1–6, 0 przed pierwszym rzutem) z prywatnym setterem.
2. Właściwość automatyczna `bool Zablokowana { get; set; }`.
3. Statyczna właściwość `LiczbaKosci` zwiększana w konstruktorze — liczy utworzone obiekty.
4. Metoda `int Rzuc()` losuje 1–6, chyba że kość jest zablokowana; zwraca aktualną wartość.
5. Metoda `Ustaw(int wartosc)` zgłasza `ArgumentOutOfRangeException` dla wartości spoza 1–6.

2 Punktacja

Dodaj statyczną metodę `Punkty(IEnumerable<int> wartosci)`.

1. Punkty to suma oczek tych kości, których wartość wystąpiła co najmniej dwa razy (np. 3, 3, 2, 4, 2 → 10).
2. Sprawdź ręcznie trzy przykłady: brak powtórzeń (0 pkt), wszystkie równe, dwie pary.

3 Program testowy

Program w `Program.cs` działa jak w arkuszu egzaminacyjnym.

1. Pytaj o liczbę kości (3–10) aż do skutku.
2. Utwórz listę kości i wypisz wartość `Kosc.LiczbaKosci`.
3. Rzuć wszystkimi kośćmi, wypisz „Kostka 1: 3”... i „Liczba uzyskanych punktów: ...”.
4. Pytaj „Jeszcze raz? (t/n)” i powtarzaj rzut.
5. Zablokuj w kodzie jedną kość przed drugim rzutem i sprawdź, że jej wartość się nie zmienia.

4 Dokumentacja

Na egzaminie INF.04 oceniana jest dokumentacja kodu.

1. Nad klasą i każdą metodą dodaj komentarz `///` z opisem, parametrami i wartością zwracaną.
2. W nagłówku pliku wpisz autora i datę.
3. Wykonaj zrzuty ekranu działania programu.
4. Zatwierdź pracę w Git.

W sprawozdaniu umieść

- ☐ kod klasy Kosc z dokumentacją
- ☐ kod programu testowego
- ☐ zrzut ekranu z dwoma rzutami
- ☐ tabela ręcznych przykładów punktacji

Pytania kontrolne

1. Czym różni się składowa statyczna od składowej obiektu? Podaj przykład z klasy Kosc.
2. Dlaczego obiekt `Random` jest wspólny (statyczny), a nie tworzony w każdej kości?
3. Po co klasie drugi konstruktor przyjmujący `Random`?
4. Co daje prywatny setter właściwości `Wartosc`?

Kryteria oceny

Kryterium	Punkty
Właściwości i konstruktory	4
Licznik statyczny	2
Rzuc z blokowaniem, Ustaw z walidacją	4
Punktacja	3
Program testowy jak w arkuszu	4
Dokumentacja ///	3
Razem	**20**

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