Interfejsy i polimorfizm — figury

Czas: 2 × 45 min | praca indywidualna | Wykład 1.16

Cel: Zdefiniujesz interfejs `IFigura` i kilka implementujących go klas, a następnie posortujesz listę figur według pola dzięki interfejsowi `IComparable<T>`.

Po zajęciach potrafisz:

- definiuje i implementuje interfejs
- wykorzystuje interfejs `IComparable<T>` do sortowania
- waliduje dane w konstruktorze
- korzysta z interfejsu jako typu zmiennej i elementu listy

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 Interfejs i pierwsze klasy

Utwórz projekt `Figury`.

1. Interfejs `IFigura` z właściwością `string Nazwa` i metodami `double Pole()`, `double Obwod()`.
2. Klasy `Prostokat(a, b)` i `Kolo(r)` implementujące interfejs.
3. Klasa `Kwadrat(a)` dziedzicząca po `Prostokat` — nadpisz tylko nazwę (właściwość `virtual` w klasie bazowej).

2 Trójkąt z walidacją

Klasa `Trojkat(a, b, c)`.

1. W konstruktorze sprawdź warunek trójkąta (suma dwóch boków większa od trzeciego); w przeciwnym razie `ArgumentException`.
2. Pole oblicz wzorem Herona: p = obwód/2, P = √(p(p−a)(p−b)(p−c)).
3. Sprawdź: trójkąt 3-4-5 ma pole 6; boki 1, 2, 10 — wyjątek.

3 Sortowanie i raport

Lista różnych figur.

```
interface IFigura : IComparable<IFigura>
{
    string Nazwa { get; }
    double Pole();
    double Obwod();
    int IComparable<IFigura>.CompareTo(IFigura? inna) => Pole().CompareTo(inna?.Pole() ?? 0);
}
```

1. Rozszerz interfejs: `interface IFigura : IComparable<IFigura>` i zaimplementuj porównywanie po polu (szablon).
2. Utwórz `List<IFigura>` z co najmniej 6 figurami i wywołaj `figury.Sort()`.
3. Wypisz tabelę: nazwa, pole, obwód (2 miejsca po przecinku), sumę pól i figurę o największym obwodzie (`MaxBy`).
4. Posortuj listę także po obwodzie malejąco, używając `Sort((a, b) => b.Obwod().CompareTo(a.Obwod()))`.

4 Nowa figura

Sprawdź elastyczność rozwiązania.

1. Dodaj klasę `Romb(d1, d2)` lub `Szesciokat(a)`.
2. Sprawdź, czy raport działa bez zmian w pętli.
3. Zatwierdź pracę w Git.

W sprawozdaniu umieść

- ☐ kod interfejsu i klas
- ☐ tabela figur posortowana po polu
- ☐ obliczenia kontrolne dla trzech figur
- ☐ wyjaśnienie, dlaczego dodanie nowej figury nie wymagało zmian w raporcie

Pytania kontrolne

1. Czym różni się interfejs od klasy abstrakcyjnej?
2. Do czego służy interfejs `IComparable<T>`?
3. Dlaczego `Kwadrat` dziedziczy po `Prostokat`, a `Kolo` tylko implementuje interfejs?
4. Co zwraca metoda `CompareTo` i jak interpretujemy wynik?

Kryteria oceny

Kryterium	Punkty
Interfejs i klasy Prostokat, Kwadrat, Kolo	5
Trojkat z walidacją i wzorem Herona	4
Sortowanie IComparable i lambdą	4
Raport (suma, MaxBy)	2
Nowa figura	2
Pytania	3
Razem	**20**

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