Rekordy i wyliczenia — kawiarnia

Czas: 2 × 45 min | praca indywidualna | Wykład 1.17

Cel: Zamodelujesz zamówienia w kawiarni za pomocą rekordów i typów wyliczeniowych, z kontrolą dozwolonych zmian statusu zamówienia.

Po zajęciach potrafisz:

- definiuje rekord pozycyjny i tworzy jego zmodyfikowaną kopię wyrażeniem `with`
- porównuje rekordy według wartości
- stosuje typ wyliczeniowy `enum` w wyrażeniu `switch`
- pilnuje poprawnej kolejności stanów obiektu

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 Wyliczenia i rekord

Utwórz projekt `Kawiarnia`.

1. `enum Rozmiar { Maly, Sredni, Duzy }` i `enum StatusZamowienia { Nowe, WPrzygotowaniu, Gotowe, Wydane }`.
2. Rekord `record Pozycja(string Nazwa, decimal CenaBazowa, Rozmiar Rozmiar)` z właściwością `Cena` = cena bazowa + dopłata za rozmiar.
3. Dopłaty (0 / 2 / 4 zł) zwracaj w statycznej metodzie z wyrażeniem `switch` na `Rozmiar`.
4. Wypisz rekord metodą `Console.WriteLine(pozycja)` — co wypisuje domyślny `ToString` rekordu?

2 Kopie i porównania

Sprawdź cechy rekordów.

1. Utwórz duże latte jako kopię średniego: `menu[1] with { Rozmiar = Rozmiar.Duzy }`.
2. Porównaj `==` dwa rekordy o tych samych danych i dwa o różnych. Zapisz wyniki.
3. Zamień `record` na `class` i powtórz porównanie. Co się zmieniło? Przywróć `record`.

3 Zamówienie

Klasa `Zamowienie` z numerem, listą pozycji i statusem.

```
public void ZmienStatus(StatusZamowienia nowy)
{
    if (nowy != Status + 1)
        throw new InvalidOperationException($"Nie można przejść ze statusu {Status} do
{nowy}.");
    Status = nowy;
}
```

1. Metoda `Dodaj(Pozycja p)` — działa tylko dla statusu `Nowe`.
2. Metoda `Suma()` — suma cen pozycji.
3. Metoda `ZmienStatus(StatusZamowienia nowy)` pozwala tylko na przejście do następnego statusu; inaczej `InvalidOperationException`.
4. W programie złóż zamówienie z trzech pozycji, przeprowadź je przez wszystkie statusy i spróbuj cofnąć status.

4 Podsumowanie

Raport kawiarni.

1. Wypisz wszystkie wartości `Rozmiar` pętlą po `Enum.GetValues<Rozmiar>()` z liczbą (`(int)r`) i dopłatą.
2. Wypisz paragon zamówienia z pozycjami i sumą.
3. Zatwierdź pracę w Git.

W sprawozdaniu umieść

- ☐ kod programu
- ☐ wyniki porównań rekordów i klas
- ☐ zrzut paragonu
- ☐ zrzut komunikatu przy próbie cofnięcia statusu

Pytania kontrolne

1. Czym rekord różni się od klasy przy porównaniu `==`?
2. Do czego służy wyrażenie `with`?
3. Jakie wartości liczbowe mają kolejne elementy `enum` i jak je zmienić?
4. Dlaczego status zamówienia lepiej zapisać jako `enum` niż jako `string`?

Kryteria oceny

Kryterium	Punkty
Wyliczenia i rekord z ceną	4
Kopie with i porównania (z wnioskami)	4
Zamówienie z kontrolą statusu	6
Raport	3
Pytania	3
Razem	**20**

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