Pliki CSV i JSON — dziennik wydatków

Czas: 2 × 45 min | praca indywidualna | Wykład 1.20

Cel: Wczytasz dziennik wydatków z pliku CSV z pominięciem błędnych wierszy, obliczysz sumy w kategoriach, zapiszesz raport oraz wyeksportujesz dane do JSON i odczytasz je z powrotem.

Po zajęciach potrafisz:

- odczytuje plik tekstowy wiersz po wierszu i dzieli wiersze na pola
- przetwarza liczby i daty z jawnie podaną kulturą
- zapisuje raport do pliku CSV
- serializuje i deserializuje listę obiektów do JSON z polskimi znakami

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

- Pobierz plik `wydatki.csv` z folderu `pliki` przy tym zadaniu.

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 Odczyt CSV

Utwórz projekt `Wydatki` i dodaj do niego plik `wydatki.csv`; we właściwościach pliku ustaw **Kopiuj do katalogu wyjściowego: Kopiuj, jeśli nowszy**.

1. Rekord `Wydatek(DateOnly Data, string Kategoria, string Opis, decimal Kwota)`.
2. Czytaj plik `File.ReadLines`; pomiń nagłówek i puste wiersze.
3. Dziel wiersz `Split(';')`; datę czytaj `DateOnly.TryParseExact(p[0], "yyyy-MM-dd", out var d)`, kwotę `decimal.TryParse` z kulturą pl-PL.
4. Wiersz błędny pomiń i wypisz jego numer — w pliku jest jeden celowo błędny wiersz.
5. Wypisz liczbę wczytanych wydatków i błędnych wierszy.

2 Analiza i raport

Sumy w kategoriach.

1. Pogrupuj wydatki po kategorii, policz sumę i udział procentowy; posortuj malejąco.
2. Wypisz tabelę w konsoli (kwoty `:C`, udział `:P1`) i sumę całkowitą.
3. Zapisz raport do `raport.csv` (`File.WriteAllLines`) z nagłówkiem `kategoria;suma;udział`.
4. Otwórz `raport.csv` w Excelu lub LibreOffice Calc — czy kolumny rozdzieliły się poprawnie?

3 JSON

Eksport i import.

```
var opcje = new JsonSerializerOptions
{
    WriteIndented = true,
    Encoder = JavaScriptEncoder.Create(UnicodeRanges.All)   // polskie znaki bez \u0105
};
File.WriteAllText("wydatki.json", JsonSerializer.Serialize(wydatki, opcje));
```

1. Zapisz listę wydatków do `wydatki.json` metodą `JsonSerializer.Serialize` z opcjami poniżej.
2. Otwórz plik JSON w Visual Studio i sprawdź, czy polskie znaki są czytelne. Usuń opcję `Encoder` i porównaj.
3. Odczytaj plik `JsonSerializer.Deserialize<List<Wydatek>>` i sprawdź zgodność `SequenceEqual`.
4. Zatwierdź pracę w Git (bez plików wynikowych z folderu `bin`).

W sprawozdaniu umieść

- ☐ kod programu
- ☐ zrzut tabeli sum
- ☐ zawartość `raport.csv` i fragment `wydatki.json`
- ☐ porównanie JSON z opcją Encoder i bez niej

Pytania kontrolne

1. Dlaczego przy odczycie liczb z pliku warto podać kulturę jawnie?
2. Czym różni się `File.ReadAllLines` od `File.ReadLines`?
3. Gdzie program szuka pliku podanego bez ścieżki i dlaczego trzeba ustawić „Kopiuj do katalogu wyjściowego”?
4. Czym różni się format CSV od JSON? Kiedy który wybrać?

Kryteria oceny

Kryterium	Punkty
Odczyt CSV z obsługą błędnych wierszy	6
Sumy, udziały i raport CSV	5
Eksport i import JSON z polskimi znakami	5
Pytania	4
Razem	**20**

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