StudentScoresAnalysis – Opis programu i omówienie kodu

Autor: Adrian Kasprzak

Cel programu

Celem programu jest utworzenie i analiza macierzy ocen 10 studentów, gdzie każdy student ma 5 ocen. Program dla każdego studenta:

- generuje losowe oceny z zakresu 2–5,
- oblicza średnią ocen,
- wyznacza ocenę minimalną i maksymalną,
- prezentuje wyniki w formie czytelnej tabeli na konsoli.

Struktura projektu

- **IStudentResult.java** – interfejs opisujący wynik analizy ocen studenta (gettery na oceny, średnią, min, max).
- **StudentScoresAnalysis.java** – główna klasa z metodą `main()`, zawiera logikę tworzenia i analizy ocen.
- **StudentResult.java** – klasa modelowa do przechowywania ocen jednego studenta oraz wyników analizy (średnia, min, max).
- **ArrayHelpers.java** – klasa narzędziowa z metodami statycznymi do analizy tablic (średnia, min, max).

Opis działania krok po kroku

1. **Inicjalizacja stałych i generatora liczb losowych**
    - `STUDENTS = 10` – liczba studentów.
    - `SUBJECTS = 5` – liczba ocen dla każdego studenta.
    - Obiekt klasy `Random` do generowania losowych ocen.

2. **Tworzenie tablicy wyników**
    - Tworzy się tablicę `StudentResult[] results` o rozmiarze 10 – jeden obiekt na każdego studenta.
3. **Losowanie ocen i analiza**
    - Dla każdego studenta generowana jest tablica 5 ocen losowych (od 2 do 5).
    - Za pomocą klasy `ArrayHelpers` wyliczane są: średnia, minimalna i maksymalna ocena.
    - Tworzony jest obiekt `StudentResult` przechowujący oceny oraz wyniki analizy i dodawany do tablicy wyników.
4. **Wyświetlanie wyników**
    - Nagłówek tabeli z nazwami kolumn jest drukowany formatowanym napisem (`printf`).
    - Dla każdego studenta drukowana jest jedna linia: numer, tablica ocen, średnia (zawsze 2 miejsca po przecinku), min, max.

Opis użytych klas, interfejsów, metod i typów danych

Interfejs `IStudentResult`

- Definiuje wymagany zestaw metod (tzw. getterów), które każda klasa opisująca wynik analizy ocen studenta powinna udostępniać:
    - `int[] getGrades();` – zwraca tablicę ocen studenta,
    - `double getAverage();` – zwraca średnią ocen,
    - `int getMinGrade();` – zwraca ocenę minimalną,
    - `int getMaxGrade();` – zwraca ocenę maksymalną.
- Pozwala na elastyczność i możliwość podmiany implementacji modelu bez zmiany reszty kodu analizującego wyniki studentów.

Klasa `StudentScoresAnalysis`

- Zawiera punkt startowy programu: metodę `main()`.
- Odpowiada za całą logikę analizy ocen i wyświetlanie tabeli wyników.

Klasa `StudentResult`

- Model danych dla jednego studenta.
- Przechowuje:
  - tablicę ocen,
  - średnią ocen,
  - ocenę minimalną,
  - ocenę maksymalną.
- Umożliwia dostęp do tych danych przez gettery.

Klasa `ArrayHelpers`

- Zbiór metod statycznych do pracy z tablicami:
  - `average(int[] grades)` – liczy średnią ocen,
  - `min(int[] grades)` – znajduje najmniejszą ocenę,
  - `max(int[] grades)` – znajduje największą ocenę.

Typy danych

- `int[]` – tablica ocen studenta (liczby całkowite).
- `StudentResult[]` – tablica wyników wszystkich studentów.
- `double` – średnia ocen.
- `Random` – klasa do losowania liczb.

Użyte pętle i formatowanie

- Zagnieżdżone pętle `for`:
  - Pierwsza pętla – po wszystkich studentach.
  - Druga pętla – po wszystkich ocenach danego studenta.

- Formatowanie `printf` – do czytelnego drukowania tabeli wyników.

Warunki sterujące

- Brak złożonych instrukcji warunkowych – całość logiki opiera się na prostym sumowaniu oraz wywołaniu funkcji pomocniczych.

Przykładowy fragment kodu

```
for (int i = 0; i < STUDENTS; i++) {
    int[] grades = new int[SUBJECTS];
    for (int j = 0; j < SUBJECTS; j++) {
        grades[j] = rand.nextInt(4) + 2; // Losuje 2-5
    }
    double avg = ArrayHelpers.average(grades);
    int min = ArrayHelpers.min(grades);
    int max = ArrayHelpers.max(grades);

    results[i] = new StudentResult(grades, avg, min, max);
}
```

Sposób budowania i uruchamiania programu

1. Kompilacja

Skompiluj wszystkie pliki razem, będąc w folderze z plikami `.java`:

```
javac StudentsScoresAnalysis.java StudentsScoresAnalysis.java
```

Powstaną pliki `.class` dla każdej klasy/interfejsu.

2. Uruchomienie

Program uruchamiasz komendą:

```
java StudentsScoresAnalysis
```

Najważniejsze zalety rozwiązania

- Automatyzacja analizy ocen i czytelna prezentacja danych.
- Przykład czytelnego podziału na klasy: główną, modelową i narzędziową.
- Łatwość rozbudowy kodu (np. o dodatkowe analizy, inne liczby studentów/ocen).

Podsumowanie

Program demonstruje, jak zorganizować analizę prostych danych liczbowych z podziałem na klasy i wykorzystaniem narzędzi do pracy z tablicami. Dzięki temu kod jest przejrzysty, łatwy w utrzymaniu i gotowy do rozbudowy w przyszłości.