Treść zadania

Twoim zadaniem jest napisanie prostej aplikacji w języku Java, składającej się dokładnie z **trzech klas**. Aplikacja ma modelować fragment rzeczywistości przypisany do Twojego wariantu.

Opis klas

1. Klasa Modelu danych (np. `Ksiazka`, `Pracownik` - zgodnie z wariantem)
   Ta klasa reprezentuje **pojedynczy obiekt** z Twojego tematu.
   Wymagania:
   - Utwórz minimum **4 pola** o różnych typach danych (np. `String`, `int`, `double`, `boolean`).
   - **Ważne:** Wszystkie pola muszą być bezwzględnie ukryte (modyfikator `private`).
   - Napisz **dwa konstruktory**:
     - domyślny (bezparametrowy) – ustawiający sensowne wartości początkowe,
     - parametryczny – przyjmujący argumenty dla wszystkich pól i inicjalizujący je (użyj słówka `this`).
     - Wygeneruj/napisz **metody dostępowe (gettery i settery)** dla każdego z pól (modyfikator `public`).
     - **Walidacja danych:** Wewnątrz minimum *jednego* settera napisz instrukcję warunkową (`if`), która zabezpieczy obiekt przed wprowadzeniem błędu (np. cena nie może być ujemna, rok produkcji nie może być z przyszłości). Jeśli podano złą wartość, wypisz w konsoli stosowny komunikat błędu i ustaw wartość domyślną.
     - Napisz jedną **metodę biznesową**, która wykonuje obliczenia na podstawie pól obiektu i zwraca wynik (np. `obliczWiek()`, `wyliczZysk()`, `czyJestWartosciowy()`).

2. Klasa Menadżera (np. `Biblioteka`, `DzialFirmy` - zgodnie z wariantem)
   Ta klasa służy do zarządzania grupą Twoich obiektów.
   Wymagania:
   - Utwórz **5-elementową tablicę** przechowującą obiekty z Twojej Klasy Modelu. Tablica musi być polem `private`.
   - Napisz **konstruktor**, w którym utworzysz konkretne obiekty i przypiszesz je do tablicy/zmiennych. Możesz wymyślić ich dane "na sztywno" w kodzie.
   - Napisz metodę `wyswietlWszystko()`, która w czytelny sposób wypisze w konsoli dane wszystkich posiadanych obiektów, korzystając z ich getterów.
   - Napisz **metodę analityczną**, która przeszuka Twoje obiekty, użyje instrukcji warunkowych i wypisze wynik. Może to być np. `znajdzNajdrozszy()`, `policzSpelniajaceWarunek()` lub `obliczSrednia()`.

3. Klasa Testowa (`Main`)
   Klasa uruchomieniowa, w której udowodnisz, że Twój kod działa.
   Wymagania:
   - Posiada metodę `public static void main(String[] args)`.
   - Tworzy jeden obiekt Twojej Klasy Menadżera.
   - Wywołuje na nim metodę wyświetlającą dane oraz metodę analityczną.
   - **Test walidacji:** Tworzy ręcznie jeden nowy obiekt Modelu, a następnie wywołuje na nim chroniony setter z *celowo błędnymi danymi*, aby zaprezentować w konsoli, że Twoje zabezpieczenie (walidacja) zadziałało i zablokowało zmianę.

Temat na dziś

GraWideo (Tytuł, RokWydania, Cena, CzyMultiplayer) + **KolekcjaGier**

Zasady oceniania (Maksymalnie 20 punktów)
Zadanie oceniane jest w skali punktowej. Aby otrzymać zaliczenie, musisz zdobyć co najmniej 10 punktów (50%).

A. Klasa Modelu (max 8 pkt)

- [2 pkt] 4 pola o odpowiednich typach i poprawne zastosowanie `private`.
- [2 pkt] Konstruktor domyślny (1 pkt) oraz parametryczny z użyciem `this` (1 pkt).
- [1 pkt] Obecność publicznych getterów i setterów dla każdego pola.
- [2 pkt] Poprawnie napisana logika walidacji w wybranym setterze (+ komunikat błędu).
- [1 pkt] Obecność działającej metody biznesowej zwracającej wyliczoną wartość.

B. Klasa Menadżera (max 7 pkt)

- [2 pkt] Utworzenie tablicy (lub 3 pól) typu obiektowego jako pole `private`.
- [1 pkt] Konstruktor inicjalizujący obiekty przykładowymi danymi.
- [2 pkt] Metoda wypisująca sformatowane dane wszystkich obiektów w konsoli.
- [2 pkt] Poprawna logika metody analitycznej (użycie pętli/ifów, porównywanie obiektów).

C. Klasa Testowa `Main` (max 5 pkt)

- [1 pkt] Poprawne utworzenie obiektu Menadżera w metodzie `main`.
- [2 pkt] Pomyślne wywołanie metod wyświetlających i analitycznych.
- [2 pkt] Pomyślne przetestowanie i udowodnienie w konsoli, że walidacja w setterze z Klasy 1 działa poprawnie.

OCENA CELUJĄCA (6.0): Jeśli zdobędziesz równe 20 punktów i chcesz ubiegać się o ocenę celującą, wdroż do swojego programu jeden z poniższych mechanizmów do wyboru:

1. Przebuduj program tak, aby zamiast danych "na sztywno", w `Main` działało interaktywne menu tekstowe (użycie klasy `Scanner`), pozwalające użytkownikowi na dodawanie nowych obiektów i ich wyświetlanie.
2. Zastosuj dynamiczną listę `ArrayList` zamiast zwykłej tablicy, aby móc dodawać nieograniczoną liczbę obiektów.
3. Napisz kod wyjątkowo "czysto": używaj wyłącznie angielskiego nazewnictwa, pełnej konwencji *camelCase* i nadpisz metodę `toString()`, zwracając idealnie sformatowany tekst za pomocą `String.format()`.