AdvancedCalculator – Opis programu i architektury

Autor: Adrian Kasprzak

Cel programu

Celem projektu jest stworzenie zaawansowanego kalkulatora matematycznego w języku Java, zorientowanego obiektowo (OOP), który pokazuje:

- **stosowanie interfejsów,**
- **dziedziczenie (rozszerzanie funkcjonalności),**
- oraz obsługę wyjątków i sytuacji brzegowych.

Struktura projektu

- **ICalculator.java** – interfejs, który definiuje podstawowe i zaawansowane operacje matematyczne (dodawanie, odejmowanie, mnożenie, dzielenie, potęgowanie, pierwiastkowanie).
- **BasicCalculator.java** – klasa bazowa, która implementuje interfejs i realizuje wszystkie operacje matematyczne BEZ obsługi wyjątków.
- **AdvancedCalculator.java** – klasa dziedzicząca po BasicCalculator, nadpisuje wybrane metody, dodaje obsługę błędów (np. dzielenie przez zero, pierwiastkowanie z liczby ujemnej, 0^0, liczby zespolone w potęgowaniu).
- **Main.java** – program testowy, zawiera metodę `main`, w której tworzony jest kalkulator i uruchamiane przykładowe operacje, także błędne.

Opis działania programu

1. **Tworzony jest obiekt AdvancedCalculator** (przez interfejs `ICalculator` – pokazuje polimorfizm).

2. **Wywoływane są przykładowe operacje matematyczne:**
    - dodawanie, odejmowanie, mnożenie, dzielenie, potęgowanie, pierwiastkowanie.
    - także przypadki specjalne: dzielenie przez zero, 0^0, pierwiastkowanie z liczby ujemnej, potęgowanie liczby ujemnej do niecałkowitej potęgi.
3. **Dla przypadków błędnych wyświetlane są komunikaty, a funkcje zwracają Double.NaN** (nie liczby).
4. **Wyniki są drukowane na konsoli – widzisz od razu, które operacje są poprawne, a które nie.**

Opis użytych klas, interfejsów i typów danych

Interfejs `ICalculator`

- Deklaruje metody: `add`, `subtract`, `multiply`, `divide`, `power`, `sqrt` (wszystko na `double`).
- Pozwala na polimorficzne używanie kalkulatora – można łatwo podmienić implementację.

Klasa `BasicCalculator`

- Implementuje interfejs – wszystkie metody robią czyste operacje matematyczne, bez obsługi błędów (np. dzielenie przez zero daje "Infinity", a sqrt(-4) daje "NaN").

Klasa `AdvancedCalculator`

- Dziedziczy po `BasicCalculator`.
- Nadpisuje metody `divide`, `sqrt`, `power` i obsługuje przypadki brzegowe (np. dzielenie przez zero, pierwiastkowanie z liczby ujemnej, 0^0, niecałkowita potęga liczby ujemnej).
- W przypadku błędu wypisuje komunikat na konsoli i zwraca `Double.NaN`.
- Pozostałe metody dziedziczy bez zmian.

Klasa `Main`

- Zawiera punkt startowy programu: `public static void main(String[] args)`.
- Tworzy kalkulator i uruchamia testowe operacje (wyświetla też komunikaty o błędach).

Sposób budowania i uruchamiania programu

1. Kompilacja

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

```
javac *.java
```

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

2. Uruchomienie

Program uruchamiasz komendą:

```
java Main
```

UWAGA: dlaczego tu jest inaczej niż w poprzednich zadaniach?

- W tym projekcie **punkt wejścia** (main) jest TYLKO w pliku `Main.java`.
- Pozostałe klasy są wyłącznie do obsługi logiki i narzędzi matematycznych.
- W innych zadaniach (np. matrix, analiza ocen) mogłeś mieć wiele programów startowych lub każdą klasę uruchamianą oddzielnie. Tu jest **jeden, wspólny**

program testowy.

- Pokazuje to dobry styl architektury większego programu — jedno „wejście”, reszta to narzędzia.

Przykładowy wynik działania

```
--- Test kalkulatora zaawansowanego ---
5 + 3 = 8.0
10 - 4 = 6.0
6 * 7 = 42.0
20 / 4 = 5.0
8 / 0 = NaN
Błąd: nie można dzielić przez zero!
2 ^ 8 = 256.0
0 ^ 0 = NaN
Błąd: 0 do potęgi 0 jest nieokreślone!
(-4) ^ 0.5 = NaN
Błąd: nieobsługiwane potęgowanie liczby ujemnej do niecałkowitej potęgi!
sqrt(9) = 3.0
sqrt(-16) = NaN
Błąd: nie można obliczyć pierwiastka z liczby ujemnej!
```

Najważniejsze zalety rozwiązania

- Prawidłowa architektura OOP: interfejs, dziedziczenie, helpersy statyczne
- Wydzielony punkt wejścia (main) i czytelny kod testujący funkcje
- Obsługa wyjątków i sytuacji brzegowych z komunikatami
- Możliwość łatwej rozbudowy (np. inne operacje, własny interfejs użytkownika)

Podsumowanie

Program AdvancedCalculator to kompletna, architektura OOP w Javie:

- Pokazuje praktyczne zastosowanie interfejsów, dziedziczenia
- Zawiera czytelną strukturę projektu i wygodny program testowy
- Jasno rozdziela logikę matematyczną od wejścia/wyjścia programu