Zadanie zaliczeniowe 1: Współbieżna kostka

Wprowadzenie

Kostka Rubika to sześcian złożony z ruchomych sześciennych bloków.

Ściany kostki dzielą się kolorowe kwadraty, należące do poszczególnych bloków. Kwadraty uporządkowane są w wiersze i kolumny. W ułożonej kostce wszystkie kwadraty danej ściany pokryte są jednym kolorem, dla każdej ściany innym.

Blok należy do trzech prostopadłych warstw kostki. Bloki można przemieszczać, obracając warstwy o wielokrotność kąta prostego.

Konstrukcja kostki pozwala na jednoczesne obracanie warstw, które nie mają wspólnych bloków.

Polecenie

- Zaimplementuj kostkę umożliwiającą współbieżne wykonywanie operacji obrócenia warstwy i pokazania aktualnego stanu kostki (7 pkt).
- Napisz klasę testów jednostkowych kostki (3 pkt).

Rozwiązanie ma być zgodne z poniższą specyfikacją.

Implementacja kostki

W rozwiązaniu zadania jest pakiet `concurrentcube` z implementacją kostki za pomocą klasy:

```
package concurrentcube;

import java.util.function.BiConsumer;
...

public class Cube {

    public Cube(int size,
            BiConsumer<Integer, Integer> beforeRotation,
            BiConsumer<Integer, Integer> afterRotation,
            Runnable beforeShowing,
            Runnable afterShowing) {
        ...
    }

    public void rotate(int side, int layer) throws InterruptedException {
        ...
    }

    public String show() throws InterruptedException {
        ...
    }

    ...

}
```

- Konstruktor `Cube(size, beforeRotation, afterRotation, beforeShowing, afterShowing)` tworzy ułożoną kostkę rozmiaru `size`. Parametr `size` oznacza długość boku kostki mierzoną liczbą kwadratów.

  Argumenty typu `BiConsumer<Integer, Integer>` i `Runnable` to akcje wywoływane podczas operacji na kostce. Zakładamy, że nie korzystają one z metod klasy `Cube`.

  Akcja `beforeRotation` może np. uruchomić ramię robota manipulującego fizyczną kostką a `afterRotation` może zaczekać, aż robot zakończy obracanie warstwy.

- Metoda `rotate(side, layer)` obraca o kąt prosty, zgodnie z ruchem wskazówek zegara, warstwę kostki o numerze `layer`, patrząc od strony ściany `side`.

  Argument `side` przyjmuje jedną z wartości: `0` (góra), `1` (lewo), `2` (przód), `3` (prawo), `4` (tył), `5` (dół).

  Argument `layer`, dla kostki o rozmiarze `size`, przyjmuje wartość od `0` do `size - 1`.

  Wszystkie zlecone obroty są wykonywane, nawet, jeśli anulują nawzajem swój wynik.

  Bezpośrednio przed obrotem warstwy, gdy wątek doczeka się na możliwość wykonania tej operacji, metoda wywołuje akcję `beforeRotation` a bezpośrednio po obrocie wywołuje akcję `afterRotation`. Obu akcjom przekazuje jako argumenty wartości `side` i `layer`.

- Metoda `show()` pokazuje stan kostki w postaci napisu złożonego z cyfr od `'0'` do `'5'`.

  Cyfry reprezentują kolory kwadratów na ścianach. W ułożonej kostce kolor ściany numer `s` jest reprezentowany przez cyfrę o wartości `s`.

  Opis kostki składa się z opisów ścian, uporządkowanych w kolejności rosnącego numeru. Dla każdej ściany podaje kolory kwadratów w poszczególnych wierszach, od góry do dołu, a w ramach wiersza, w kolumnach, od lewej do prawej.

  Dla ściany lewej, przedniej, prawej i tylnej, kolory kwadratów mają kolejność taką, jaką zobaczymy po obrocie całej kostki o wielokrotność kąta prostego od strony ściany górnej lub dolnej.

  Dla ściany górnej i dolnej kolejność jest taka, jaką zobaczymy po obrocie całej kostki o wielokrotność kąta prostego od strony ściany lewej lub prawej.

  Bezpośrednio przed zbudowaniem opisu kostki, gdy wątek doczeka się na możliwość wykonania tej operacji, metoda wywołuje akcję `beforeShowing` a bezpośrednio po zbudowaniu opisu wywołuje akcję `afterShowing`.

Oprócz klasy `Cube`, w rozwiązaniu mogą być inne definicje, zarówno w pakiecie `concurrentcube`, jak i w pakietach pomocniczych.

Współbieżność implementacji

Rozwiązanie nie dopuszcza do zagłodzenia wątku.

Wykonanie metod `rotate(side, layer)` i `show()` nie jest wewnętrznie zrównoleglone. Całą operację realizuje wątek, który metodę wywołał.

Implementacja umożliwia współbieżne wykonywanie tych operacji na kostce, które ze sobą nie kolidują. Nie pozwala na pokazanie kostki podczas obrotu ani na współbieżne obroty warstw mających wspólny blok.

Wynik metody `show()` jest zgodny ze scenariuszem, w którym obroty są wykonywane, w pewnej określonej kolejności, sekwencyjnie i niepodzielnie.

W przypadku przerwania wątku czekającego na wykonanie operacji, metody `rotate(side, layer)` i `show()` zgłaszają wyjątek `InterruptedException`. Po przerwaniu stan kostki pozostaje poprawny. Pozostałe wątki mogą normalnie kontynuować pracę.

Testy jednostkowe

W pakiecie `concurrentcube` rozwiązania, oprócz klasy `Cube` jest też klasa testów jednostkowych `CubeTest`, używająca JUnit w wersji 5.

Testy sprawdzają poprawność i współbieżność rozwiązania oraz obsługę przerwań.

Komentarze metod testujących wskazują, jakiego aspektu implementacji test dotyczy.