Programowanie Java
Anna Gogolińska
Zestaw 7

Zad 1. Napisz klasę *Kwadrat*, która będzie zawierać pole rzeczywiste oznaczającą długość boku oraz metody: getter i setter dla długości boku, *obliczObwod(), obliczPole(), powieksz(),* ewentualnie konstruktor/konstruktory. Klasę napisz tak, aby zabezpieczyć się przed podaniem nieprawidłowych danych np. ujemną długością boku. Napisz klasę zawierającą testy sprawdzające poprawność działania metod. W testach użyj poprawnych oraz niepoprawnych wartości.

Zad 2. Napisz klasę *NumbersProvider*, która będzie czytać zawartość pliku tekstowego (nazwa pliku może być *liczby.csv*, czytanie może być w konstruktorze). Plik ten ma mieć w każdej linijce trzy liczby zmiennoprzecinkowe, oddzielone znakiem ';'. Klasa ma zawierać trzy listy liczb zmiennoprzecinkowych, liczby z pierwszej kolumny mają być wczytane do pierwszej listy, liczby z drugiej kolumny do drugiej listy, liczby z trzeciej kolumny do trzeciej listy. Klasa ma zawierać metody getter dla każdej z list, poza tym również ma zawierać trzy metody zwracające długość każdej z list. Napisz klasę *CalculateAvgs*, której polem będzie obiekt klasy *NumbersProvider* (*NumberProvider* ma być przekazany w konstruktorze i ustawiany bądź stworzyć dla niego setter). Klasa ta ma zwierać metody: *calculateAvg()*, która będzie zwracać średnią arytmetyczną liczb z pierwszej listy pobranej z klasy *NumbersProvider*, *calculateGeometric(),* która będzie zwracać średnią geometryczną liczb z drugiej listy pobranej z klasy *NumbersProvider* oraz *calculateHermonic(),* która będzie zwracać średnią harmoniczną liczb z trzeciej listy pobranej z klasy *NumbersProvider* (podczas liczenia średnich użyć również metod zwracających ilość elementów list z *NumbersProvider*). Napisz klasę testową dla *CalculateAvgs*, gdzie obiekt klasy *NumbersProvider* będzie „mockowany". Napisz testy dla każdej z metod liczenia średnich.

Zad 3. Napisz klasę *Magazyn* zawierającą tablicę *n* liczb całkowitych (*n* ma być podawane w konstruktorze). Tablica ta ma obrazować stan magazynu, indeksy w tablicy będą oznaczać id produktów, a wartość w tablicy ilość danego produktu (*tab[i] = a* znaczy że produktu o id *i* jest *a* w magazynie). Napisz metody: *dodajProduct(int id, int ile),* która będzie zwiększać ilość danego produktu, *wydajProduct(int id, int ile),* która będzie zmniejszać ilość danego produktu, *zerujProduct(int id),* która będzie zwracać całkowitą ilość danego produktu i zerować jego ilość w magazynie, *zwrocProduct(int id),* która będzie zwracać ilość danego produktu w magazynie, ale nie zmieni jego ilości, *zwrocWszystkie()*, która zwróci łączną ilość wszystkich produktów w magazynie. W pisaniu metod uwzględnij nieprawidłowe sytuacje – np. pobieranie z magazynu większej ilości produktu niż jest na stanie. Napisz klasę testową dla *Magazyn*, testująca poprawność jej metod. Może dobrym pomysłem będzie zaczęcie od testów?

Zadanie domowe

Zad 4. Gra w życie (https://pl.wikipedia.org/wiki/Gra_w_%C5%BCycie). Gra odbywa się w turach, na kwadratowej planszy – wielość planszy ma być ustawiana w konstruktorze. Każde pole planszy może być "żywe" albo "martwe". Stan pola może zmienić się po turze w zależności od stanu jego sąsiadów:

1. Jeśli żywa komórka ma mniej niż 2 żywych sąsiadów ginie z samotności
2. Jeżeli żywa komórka ma 2 albo 3 żywych sąsiadów przeżyje do następnej tury
3. Jeżeli żywa komórka ma więcej niż 3 żywych sąsiadów ginie z przeludnienia
4. Jeśli martwa komórka ma dokładnie 3 żywych sąsiadów, ożywa.

Przejście do kolejnej tury odbywa się w jednym kroku, czyli określając aktualny stan danej komórki bierze się pod uwagę stany jej i jej sąsiadów z poprzedniej tury (a nie być może częściowo uaktualnione stany z obecnej tury).

Napisz program implementujący grę w życie. Ma on zawierać klasę *Board* zawierającą tablicę reprezentującą planszę – można stworzyć w razie potrzeby również pomocnicze klasy.
Klasa Board ma zwierać metody:

- Wyliczającą dla danej komórki (jej współrzędne podane jako argumenty) ilość jej żywych sąsiadów.
- Dla podanej komórki (współrzędne podanej jako argument) zmieniającą jej stan w zależności od ilości żywych sąsiadów (ma używać poprzedniej metody).
- Metodę w której będzie znajdować się pętla przebiegająca po wszystkich komórkach i aktualizująca ich stan.

Napisz klasę testową BoardTest, która będzie testować działanie metod klasy *Board*. Należy napisać przynajmniej 7 testów:

- Przynajmniej 3 testy dla metody liczącej ilość sąsiadów: dla komórki ze środka planszy, z rogu oraz z brzegu (ale nie rogu) planszy.
- Przynajmniej cztery testy dla metody wyliczającej nowy stan komórki (zgodnie z zasadami 1-4).

Można odpisać więcej testów (np. testy metod z podanymi błędnymi współrzędnymi, wyliczanie nowego stanu dla komórek z różnych miejsc planszy).