Programowanie Java
Anna Gogolińska
Zestaw 2

Zadania 1-4 dotyczą jednego projektu.

Zad 1. Utworzyć plik *Figura.java* z definicją klasy zawierającej pola zmiennoprzecinkowe: *pole*, *obwod* oraz *wymiar*, a także jedno pole *nazwa* typu String. Klasa powinna zawierać bezargumentowy konstruktor inicjujący pola wartościami 0 oraz metody *set* i *get* dla pól. Ponadto, należy zaimplementować metodę *print()*, wypisującą na ekran (standardowe wyjście) nazwę figury i wartość jej pola oraz obwodu. W metodzie *main()* utworzyć obiekt klasy Figura, ustawić wartości dla jego pól i wypisać informację na ekran.

Zad 2. Dopisać do klasy Figura metodę *setValues(-, -)* z dwoma parametrami, ustawiającymi wartość pól: *pole* i *obwod*. Dopisać także metodę *toString(),* która będzie zwracała obiekt klasy String zawierający nazwę figury wraz z wartościami jej pola i obwodu, w przypadku gdy są one większe od zera. W metodzie *main()* utwórz dwa obiekty klasy Figura dla kwadratu i koła. Przetestuj metody tworząc obiekt, modyfikując jego pola i wypisując go na ekran bez jawnego wywołania metody *toString()* (*System.out.println(obiekt)*).

Zad 3. Dopisać przeciążoną metodę *setValues*(-) tylko z jednym parametrem, która będzie ustawiać wartość pola *wymiar.* Następnie zaimplementować kolejną przeciążoną metodę *setValues*(-) z argumentem typu String zmieniającą pole *nazwa*. Zastanów się, dlaczego nie możesz zrobić analogicznych metod dla pól *pole* i *obwod*. Zaproponuj własne rozwiązanie dla tych pól z wykorzystaniem kolejnej, dwuargumentowej metody *setValues*(-, -), której drugim argumentem będzie zmienna boolowska.

Zad 4. Do klasy Figura dopisać całkowitoliczbowe pole statyczne *licznik* zainicjowane na 0 oraz metody *set* i *get* dla tego pola. Stworzyć dwa obiekty typu Figura i sprawdzić czy zmiana wartości pola *licznik* w jednym zaowocuje zmianą wartości pola *licznik* w drugim. Zmodyfikuj tak klasę, aby każdy jej obiekt zawierał własny unikalny identyfikator liczbowy, którego wartość będzie wypisywana w ramach działania metody *toString()* (poza innymi wartościami wypisywanymi przez tą metodę, wykorzystać istnienie pola statycznego).

Zadania samodzielne

Zad 5. Dopisać przeciążony, jednoargumentowy konstruktor klasy *Figura*, który będzie przyjmował początkową wartość pola *wymiar.* Dodać metodę o nazwie *isInside(-,-)* sprawdzającą czy dany punkt znajduje się wewnątrz figury. Metoda powinna zwracać wartość boolowską, a przyjmować współrzędne punktu na płaszczyźnie (rzeczywiste). Należy założyć, że:

- figura będzie reprezentować koło,
- środek tego koła znajduje się zawsze w początku układu współrzędnych,
- *wymiar* będzie reprezentował promień koła.

Następnie, zaimplementować drugą klasę *MonteCarlo*, w której należy utworzyć obiekt klasy *Figura* o wymiarze równym 1. Klasa *MonteCarlo* ma zawierać metody/metodę w której losowane będą 1000 razy współrzędne punktów z przedziału [0,1]x[0,1] (klasa *java.util.Random* i metoda *Random.nextDouble()*). Następnie zliczana ma być liczba punktów znajdujących się wewnątrz figury. Otrzymany wynik należy pomnożyć przez 4 i podzielić przez liczbę losowań, a następnie zwrócić. Co przypomina otrzymana wartość? Obliczenia powtórzyć dla 10000 (liczbę losować dobrze ustawić jako pole lub argument metody).

Zad 6.
Utworzyć klasę *SumyPrzedzialu*, która w konstruktorze pobierać będzie tablicę liczb oraz ewentualnie wykonywać potrzebne obliczenia. Klasa ta powinna dysponować metodą *calculateSum(-,-)* zwracającą sumę z dowolnego, ciągłego przedziału w tablicy wejściowej, jej argumentami powinien być początek przedziału i jego koniec (numeracja od 1). Dodatkowo metoda ta powinna działać w czasie liniowym. Spróbuj napisać drugą metodę *SumyPrzedzialuS,* która będzie działać w czasie stałym.

Pobierz dane ze standardowego wejścia. Pierwsza linia to ilość liczb.
Druga linia to tablica liczb oddzielonych spacjami (sprawdź metodę *split()*).
Kolejne linie to pary liczb określających zakres.
Dla każdej z tych par wyświetl sumę liczb z tablicy z podanego zakresu.

Przykład

Wejście:
5
1 5 2 3 1
1 2
2 3
3 5

Wyjście:
6
7
6