

Programowanie obiektowe II

Laboratorium nr 5

Stefan Sokołowski

Zadanie 1 (o implementowaniu interfejsu):

Przez *sumę funkcji rzeczywistych* (czyli z `double` do `double`) f i g rozumie się funkcję, która każdemu argumentowi rzeczywistemu przypisuje sumę wartości funkcji f i funkcji g na tym argumencie:

$$(f + g)(x) \stackrel{\text{def}}{=} f(x) + g(x)$$

Program

`iis.ans-elblag.pl/~stefan/Dydaktyka/ProgObiII/Prog05/P5.Cwiczenia/Suma.java`
definiuje interface `Funkcja`, a następnie powinien obliczać wartość sumy funkcji

$$f(x) \stackrel{\text{def}}{=} x^2 \quad \text{oraz} \\ g(x) \stackrel{\text{def}}{=} 2 \cdot x$$

na argumencie podanym z linii rozkazowej przy wywołaniu programu. Proszę wpisać do programu definicję funkcji `sumaFcji` takiej, żeby efektem było istotnie sumowanie funkcji.

Proszę to tak zrobić, żeby funkcji `sumaFcji` można było użyć do sumowania dowolnych funkcji rzeczywistych, a nie tylko dwóch podanych w zadaniu. W programie nie wolno zmieniać niczego, należy tylko dopisać funkcję `sumaFcji`.

Zadanie 2 (o implementowaniu interfejsu):

W wykładzie ¹ omówiona jest klasa abstrakcyjna `Figura` oraz sposób liczenia „pękatości” figur (metoda `bmi`).

Proszę przerobić tę klasę abstrakcyjną na interfejs, a następnie napisać jego implementację, definiując

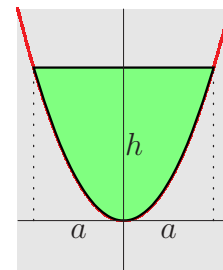
1. **koło**;
2. **prostokąt** (dowolny, niekoniecznie kwadrat);
3. **trójkąt** (dowolny, niekoniecznie równoboczny);

¹`iis.ans-elblag.pl/~stefan/Dydaktyka/ProgObiII/Slajdy/03a.pdf`, str. 4.

4. odcinek paraboli

Uwaga: pole odcinka paraboli jest równe $\frac{4}{3}ah$, a długość łuku paraboli (to jeszcze nie jest cały obwód!) wynosi

$$\frac{a^2}{2h \sinh\left(\frac{2h}{a}\right)}$$



(wg <http://mathworld.wolfram.com/ParabolicSegment.html>).

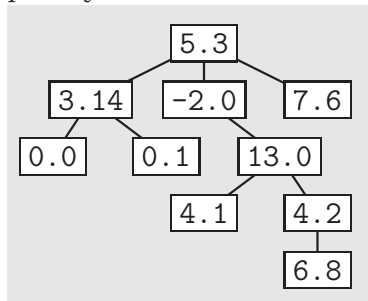
Program główny należy tak napisać, żeby w kółko pytał o wybór którejs z tych czterech figur, następnie o charakteryzujące ją dane, po czym drukował jej „pękatość”. Metoda ją licząca jest już zdefiniowana w interfejsie `Figura` i nie należy jej definiować dla każdej konkretnej figury na nowo.

Zadanie 3 (o generyczności z rekursją):

Generyczne drzewo, w którym parametr generyczny określa typ elementów mogących stać w węzłach, definiuje się jako parę (ℓ, d) , gdzie ℓ to jakiś element w korzeniu drzewa, a d to lista (`ArrayList`) drzew będących „dziećmi” korzenia. Taka definicja podana jest w

`iis.ans-elblag.pl/~stefan/Dydaktyka/ProgObiII/Prog05/P5_Cwiczenia/Drzewa.java`

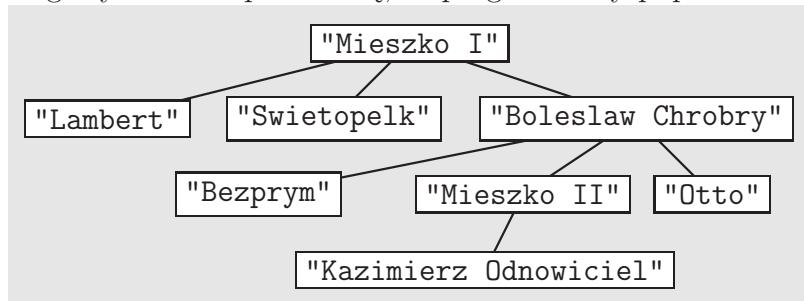
Proszę rozszerzyć klasę `Drz<T>` w tym pliku do klasy `Drzewo<T>` zawierającej dodatkowo metody liczenia liczby węzłów i liczby pokoleń danego drzewa — por. rysunek na następnej stronie. W programie głównym proszę skonstruować drzewa z tego rysunku i upewnić się, że program liczy poprawnie.



`Drzewo<Double>`

węzły: 10

pokolenia: 5



`Drzewo<String>`

węzły: 8

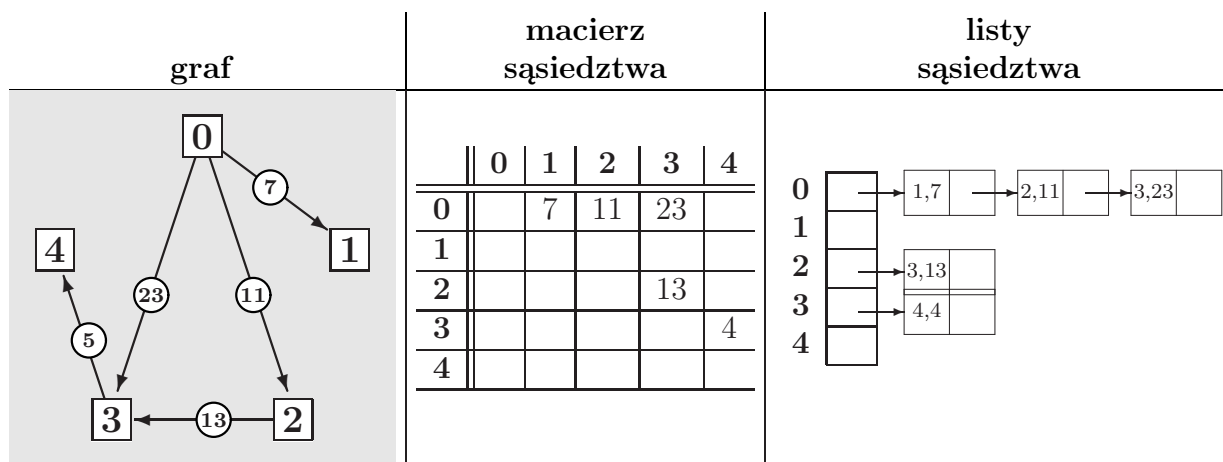
pokolenia: 4

Zadanie 4 (domowe specjalne):

Proszę napisać klasę zawierającą metody konwersji między dwiema reprezentacjami grafu skierowanego:

- **macierzą sąsiedztwa** — pole na przecięciu wiersza k z kolumną n jest wypełnione, o ile w grafie jest strzałka z wierzchołka k do wierzchołka n ; i wtedy stoi tam etykieta napisana na strzałce;
- **tablicą list sąsiedztwa** — w wierszu n znajduje się lista zawierająca po jednej parze dla każdej strzałki wychodzącej z wierzchołka n ; w każdej parze znajduje się docelowy wierzchołek oraz etykieta na strzałce.

Poniżej podany jest przykład grafu, jego list sąsiedztwa i jego macierzy sąsiedztwa.



Uwaga:

Pierwszej osobie, która przysła mi (na s.sokolowski@ans-elblag.pl) poprawne rozwiązanie tego zadania, przysługują punkty do zaliczenia laboratorium. Liczba tych punktów zależy od oryginalności i elegancji rozwiązania.