Zadanie 1

Definiowanie własnych obiektów iterowalnych

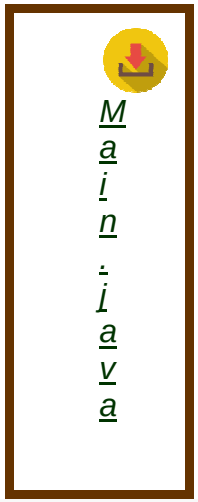
Ciąg Collatza (znany też jako „hailstone sequence” lub ciąg Ulama) to ciąg liczb naturalnych rozpoczynający się od dowolnej liczby a_0, którego kolejne wyrazy obliczane są według zasady

$$a_{n+1} = \begin{cases} a_n/2 & \text{dla } a_n \text{ parzystych} \\ 3 * a_n + 1 & \text{dla } a_n \text{ nieparzystych} \end{cases}$$

Istnieje hipoteza, że taki ciąg zawsze dojdzie do liczby 1 (i potem będzie już periodyczny: `1,4,2,1,4,2,1,4,...`). Została ona sprawdzona aż do astronomicznie wielkich liczb, ale do tej pory nie udało się jej udowodnić.
Na przykład, jeśli rozpoczniemy od liczby 5, otrzymamy ciąg `[5,16,8,4,2,1,...]`, a rozpoczynając od 7 otrzymamy już ciąg dłuższy:`[7,22,11,34,17,52,26,13,40,20,10,5,16,8,4,2,1,...]`.
Zadanie polega na stworzeniu klasy **Hailstone**, której obiekty reprezentują pojedynczy ciąg Collatza. Konstruktor pobiera liczbę startową (a_0), o której można założyć, że zawsze będzie większa od 1. Sam obiekt jest iterowalny, czyli implementuje interfejs **Iterable** i w każdej iteracji zwraca kolejne elementy ciągu, poczynając od wartości startowej. Iteracja powinna kończyć się po zwróceniu, jako ostatniego elementu, liczby 1.
Implementacja klasy **Hailstone**

- ***może*** zawierać statyczną klasę wewnętrzną, jeśli będzie potrzebna;
- ***nie może*** tworzyć żadnych tablic, ani używać żadnych kolekcji z bibliotek Jawy.

Utworzoną klasę przetestuj za pomocą następującego programu:




```
public class Main {
    public static void main(String... args) {
        int ini = 77031, count = -1, maxel = 0;
        Hailstone hailstone = new Hailstone(ini);
        for (int h : hailstone) {
            if (h > maxel) maxel = h;
            ++count;
        }
        System.out.println(ini + " " + count + " " + maxel);
    }
}
```

Powinien on wypisać, w jednej linii, oddzielone spacjami, trzy liczby: wartość startową (***ini***; w tym przykładzie 77031), ilość kroków wykonanych do osiągnięcia jedynki (***count***), oraz największy wyraz tego ciągu (***maxel***). Na przykład dla wartości startowej 10, ciąg, aż do uzyskania jedynki, zawierałby elementy `[10 5 16 8 4 2 1]`, a zatem trzy liczby, które wtedy wypisałby program miałyby wartości `10 6 16`.

Zad 2 Napisać prosty kalkulator dla liczb typu BigDecimal.
Obliczenia mają być podawane jako **argumenty wiersza poleceń** w postaci:

liczba1 op liczba2

gdzie op jeden ze znaków +,- (minus), * (mnożenie), / (dzielenie), a pomiędzy liczba1, op i liczba2 występuje jeden lub więcej białych znaków.

Obliczenia zrealizować w klasie **Calc** jako metodę **String doCalc(String cmd)**, zwracająca napisową reprezentację wyniku (uzyskanej liczby) lub napis "Invalid command to calc", jeśli wystąpią jakiekolwiek błędy.

Następująca klasa Main::

```
public class Main {

  public static void main(String[] args) {
    Calc c = new Calc();
    String wynik = c.doCalc(args[0]);
    System.out.println(wynik);
  }

}
```

po uruchomieniu winna wyprowadzić na konsolę wynik obliczenia (np. jesli podano jako argument wiersza poleceń "1 / 2" , to wynikiem powinine być napis 0.5.

Jeśli liczba wynikowa nie ma dokładnej reprezentacji (jak np. wynik dzielenia 1/3), to wynik powinien być pokazany z dokładnością co najmniej 7 miejsc dziesiętnych.

Uwaga 1: klasy Main nie wolno modyfikować i musi ona prawidłowo działać.

Uwaga 2: w zadnej z klas programu nie wolno używać instrukcji if, ani switch, ani operatora warunkowego, ani instrukcji for, ani isntrukcji while.