mgr inż. Sławomir Aleksander Dańczak
inż. Mateusz Burzyński

Warszawa, 04.03.2021 r.

GUI

Ćwiczenia powtórzeniowe z zakresu PPJ i wstępu z GUI

1. Bank

Należy napisać program, który symuluje operacje na kontach bankowych.
Konta są obiektami klasy ***Account***. zaś każde konto ma aktualny stan (***balance***) i można:

- wpłacać pieniądze (metoda ***deposit()***)
- wypłacać pieniądze (metoda ***withdraw()***)
- przelewać na inne konto (metoda ***transfer(...)***)
- dodawać do stanu konta odsetki w skali rocznej(metoda ***addInterest()***).

Należy zabezpieczyć się przed niedopuszczalnymi wartościami wpłat i wypłat (np. ujemne wartości lub wypłaty przekraczające stan konta).
Stopa oprocentowania jest wspólna dla wszystkich kont (***statyczna***) i ustalana za pomocą metody ***setInterestRate(...).***
Obiekty klasy ***BankCustomer*** reprezentują konta klientów banku, które są reprezentowane przez obiekty klasy ***Person***.

Należy dostarczyć takich definicji w/w klas, aby poniższy kod:

```
public class BankingTest {

  public static void main(String[] args) {

    Person janP = new Person("Jan"),
           alaP = new Person("Ala");

    BankCustomer jan = new BankCustomer(janP);
    BankCustomer ala = new BankCustomer(alaP);

    jan.getAccount().deposit(1000);
    ala.getAccount().deposit(2000);
    jan.getAccount().transfer(ala.getAccount(), 500);
    ala.getAccount().withdraw(1000);

    System.out.println(jan);
    System.out.println(ala);

    Account.setInterestRate(4.5);
    jan.getAccount().addInterest();
    ala.getAccount().addInterest();

    System.out.println(jan);
    System.out.println(ala);
  }
}
```

wyprowadził na konsoli następujący, oczekiwany wynik:

```
Klient: Jan stan konta 500.0
Klient: Ala stan konta 1500.0
Klient: Jan stan konta 522.5
Klient: Ala stan konta 1567.5
```

2. *Kolekcje liczb*

Napisać program, który z pliku ***tab.txt*** wczytuje do kolekcji liczby całkowite, które są rozdzielone dowolnymi białymi znakami.
Wśród zapisanych wartości należy znaleźć wartość maksymalną oraz wszystkie indeksy kolekcji gdzie taka wartość występuje.

Program powinien wypisywać na konsoli:

- w pierwszym wierszu - wszystkie liczby, rozdzielone spacjami
- w drugim wierszu - wartość maksymalną,
- w trzecim wierszu - indeksy pod którymi w kolekcji znajduje się wartość maksymalna

Przykład dla pliku zawierającego liczby:

```
1 5 5 3
-1 2 5 4
```

Otrzymamy na konsoli następujący wynik:

```
1 5 5 3 -1 2 5 4
5
1  2 6
```

3. *Pacjenci*

Należy zdefiniować klasy z poniższego programu:

```
public class Test {

  public static void main(String[] args) {

    Pacjent[] pacjenci = { new ChoryNaGlowe("Janek"),
                           new ChoryNaNoge("Edzio"),
                           new ChoryNaDyspepsje("Marian")
                         };

    for (Pacjent p : pacjenci) {
      System.out.println("Pacjent:     " + p.nazwisko() + '\n' +
                         "Chory na:    " + p.choroba() + '\n' +
                         "Zastosowano: " + p.leczenie() +"\n\n"
                         );
    }
  }
}
```

Tak, aby otrzymać w wyniku następujący wydruk:

```
Pacjent:     Janek
Chory na:    głowa
Zastosowano: aspiryna
Pacjent:     Edzio
Chory na:    noga
Zastosowano: gips
Pacjent:     Marian
Chory na:    dyspepsja
Zastosowano: węgiel
```

Uwaga: metody ***choroba()*** i ***leczenie()*** muszą być **wywoływane polimorficznie**.

4. * *Ciąg Collatza - Zadanie dla chętnych do dłuższego i głębszego przemyślenia*

Ciąg *Collatza* (znany też jako „*hailstone sequence*" lub ciąg *Ulama*) – jest to ciąg liczb naturalnych rozpoczynający się od dowolnej liczby a_0, którego kolejne wyrazy obliczane są według wzoru:

$$a_{n+1} = \begin{cases} {}^{a_n}/_2 & dla\ a_n\ parzystych \\ 3 * a_n + 1 & dla\ a_n\ nieparzystych \end{cases}$$

W ciągu *Collatza* istnieje hipoteza, że taki ciąg zawsze dojdzie do liczby o wartości 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 w rzeczywistości udowodnić.
Przykłady liczb i wynikających ciągów Collatza:

- Dla liczby 5, otrzymamy ciąg [5,16,8,4,2,1,...]
- Dla liczby 7 otrzymamy ciąg [7,22,11,34,17,52,26,13,40,20,10,5,16,8,4,2,1,...].

Postaraj się stworzyć klasę ***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***.
W implementacja klasy ***Hailstone*** nie może tworzyć żadnych tablic, ani używać kolekcji.

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);
    }
}
```

Powyższy przykład powinien wypisać, w jednej linii, oddzielone spacjami, trzy liczby:

- wartość startową - ***ini***,
- liczbę kroków wykonanych do osiągnięcia jedynki - ***count***,
- największy wyraz tego ciągu - ***maxel***

Wynikiem uzyskanym na konsoli jest w tym wypadku:

```
77031 350 21933016
```

*5. * Coctail sort*

Stwórz 100-elementową tablicę przechowującą wartości całkowite z zakresu 10-100. Wyświetl podaną tablicę, następnie zaimplementuj sortowanie koktajlowe (Cocktail sort).

Film obrazujący działanie sortowania koktajlowego:
https://www.youtube.com/watch?v=njClLBoEbfI