Zadanie 4 *4, 5 listopada 2015 r.*

KURS JĘZYKA JAVA

LISTOWA IMPLEMENTACJA KOLEJEK PRIORYTETOWYCH

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Zadanie 1.

W pakiecie `narzedzia` zdefiniuj klasę `Para`, która będzie przechowywać pary klucz–wartość, gdzie klucz jest identyfikatorem typu `String` a skojarzona z nim wartość to liczba rzeczywista typu `double`. Klucz powinien być polem publicznym ale niemodyfikowalnym, a wartość polem ukrytym, które można odczytać za pomocą gettera `czytaj()` i zmodyfikować tylko za pomocą settera `zapisz(double)`.

```
public class Para implements Comparable<Para>, Cloneable
{
    public final String klucz;
    private double wartość;
    // ...
}
```

W klasie tej zaimplementuj porównywanie par (poprzez porównywanie wartości), klonowanie pary (wystarczy klonowanie płytkie), porównywanie par za pomocą `equals(Object)` (sprwdzenie identyczności kluczy) oraz metodę `toString()` do prezentacji zawartości pary w postaci łańcucha znakowego.

Zadanie 2.

W pakiecie `narzedzia` zdefiniuj interfejs `Priorytetowy` realizujący podstawowe operacje na kolejce priorytetowej:

```
public interface Priorytetowy
{
    void wstaw (Para p);
    Para największy ();
    Para usuńNajwiększy ();
    int rozmiar();
}
```

Metoda `wstaw(Para)` ma wstawiać nową parę do kolekcji (o ile para o takim samym kluczu jeszcze w kolekcji nie występuje). Metoda `największy()` ma udostępniać referencję do pary o największej wartości a metoda `usuńNajwiększy()` ma dodatkowo usuwać tą parę z kolekcji. Wartością zwracaną przez metodę `rozmiar()` ma być liczba wszystkich par w kolekcji.

Zadanie 3.

W pakiecie `narzedzia` zdefiniuj publiczną klasę `ListaNieuporz`, która będzie implementacją *nieuporządkowanej listy jednokierunkowej* przechowującej w węzłach referencje do obiektów typu `Para`. Zadaniem tej listy ma być realizowanie operacji kolejkowych, klasa ta ma więc implementować interfejs `Priorytetowy`. Dodatkowo lista powinna się powielać, dlatego należy zaimplementować w niej interfejs `Cloneable` (zaimplementuj klonowanie głębokie).

```
public class Lista implements Priorytetowy, Cloneable
{
    protected class Wezel implements Comparable<Wezel>, Cloneable
    {
        // ...
    }
    protected Wezel początek;
    // ...
}
```

Klasa `Lista` ma być klasą opakowująca dla homogenicznej struktury zbudowanej z węzłów. Pojedynczy węzeł, zdefiniowany jako klasa `Wezel`, powinien być niepubliczną klasą wewnętrzną w klasie `Lista`. W klasie `Węzeł` zaimplementuj rekurencyjne metody do wstawiania par klucz–wartość oraz wyszukiwania i usuwania pary o największej wartości. Klasa węzła ma przejąć na siebie cały ciężar implementacji operacji kolejkowych (chociaż nie musi implementować interfejsu `Priorytetowy`). Dodatkowo węzły muszą być porównywalne (interfejs `Comparable<Wezel>`) co jest potrzebne do znajdowania maksimum oraz klonowalne (interfejs `Cloneable`) co będzie konieczne przy głębokim klonowaniu całej struktury listowej.

Zadanie 4.

W pakiecie `narzedzia` zdefiniuj publiczną klasę `ListaUporz`, która będzie implementacją *uporządkowanej listy jednokierunkowej* przechowującej w węzłach referencje do obiektów typu `Para`. Klasa ta ma dziedziczyć po klasie `ListaNieuporz` i ma odróżniać się od klasy bazowej tym, że będzie ona przehowywała pary w sposób uporządkowany od największej do najmniejszej wartości.

Zadanie 5.

Na koniec napisz krótki program testowy, sprawdzający działanie obu list. Najpierw stwórz listę uporządkowaną i wczytaj do niej dane ze standardowego wejścia, aż do zamknięcia strumienia (klawisze *Ctrl–Z*). Następnie sklonuj tą listę i posortują ją algorytmem podobnym do *heap–sort*: wyciągamy z listy wejściowej po kolei wszystkie elementy zaczynając od największego i przenosimy je do listy wynikowej, która ma być listą nieuporządkowaną (w liście nieuporządkowanej wstawienie elementu polega na umieszczeniu go na początku listy). W swoim programie wypisz metodą `toString()` listę z wczytanymi danymi oraz posortowaną listę wyjściową.

Uwaga.

Program należy napisać, skompilować i uruchomić w środowisku zintegrowanym *NetBeans*! Do swojego programu dopisz komentarze dokumentacyjne i wygeneruj na ich podstawie dokumentację.