Laboratorium 4

Cel: Budowa klas, tablice obiektów

1. Napisz program, który tworzy klasę Baza1, która posiada metody: *wypelnij_tablice(); wyswietl_tablice(); wyszukaj_w_tablicy();* i powinien działać, jak program Baza.java. dołączony w postaci źródłowej do laboratorium, podany również dalej w postaci listingu. Klasa Baza1 posiada atrybuty: int N –rozmiar tablicy podawany z klawiatury, int ile=0-liczba danych w tablicy, na początku równa 0 oraz tablice elementów typu Osoba o nazwie Dane. W metodach, które należy zdefiniować, należy używać atrybutów klasy w sposób bezpośredni. Ciało tych metod należy zdefiniować wg wskazówek przy komentarzach do funkcji.

```
//*************************************************************************
import javax.swing.*;
import java.util.*;

class Osoba
 { String nazwisko;
   float srednia;
   String uwagi;
   static int ile_obiektow = 0;

   public void Inicjuj()
    { ile_obiektow++;}
   public  void Nadaj_nazwisko(String lan)
    { nazwisko=lan;}
   public String Podaj_nazwisko()
    { return nazwisko;}
   public  void Nadaj_uwagi(String lan)
    { uwagi=lan;}
   public String Podaj_uwagi()
     { return uwagi;}
   public  void Nadaj_srednia(float srednia_)
    { srednia=srednia_;}
  public float Podaj_srednia()
     {return srednia;}
  public void Wstaw()
       { String S;
         Inicjuj();
         S = JOptionPane.showInputDialog(null, "Podaj nazwisko");
         Nadaj_nazwisko(S);
         S = JOptionPane.showInputDialog(null, "Podaj srednia");
         Nadaj_srednia(Float.parseFloat(S));
         S =JOptionPane.showInputDialog(null, "Podaj uwagi");
         Nadaj_uwagi(S);
       }
public void Wyswietl()
   {String napis="";
    napis+="\n Nazwisko: "+nazwisko;
    napis+="\n Średnia: "+srednia;
    napis+="\n Uwagi: "+uwagi;
    napis+="\n Liczba osób jest równa "+ile_obiektow;
    JOptionPane.showMessageDialog(null, napis);
   }
 public boolean Szukaj(String s)
   { return nazwisko.equals(s);
   }
 }
```

//Klasa Baza1 powinna mieć następującą postać:

```
public class Baza1
 { int N=2, ile;
   Osoba Dane[]=null;
   public void wypelnij_tablice()    //na podstawie instrukcji przy case ‘1’ w metodzie main klasy Baza
    {     }
   public void wyswietl_tablice()   //na podstawie instrukcji przy case ‘2’  w metodzie main klasy Baza
     {     }
   public void wyszukaj_w_tablicy()   //na podstawie instrukcji przy case ‘3’  w metodzie main klasy Baza
      {      }
   static public  void main(String args[])
     {
      String s;
      char ch;
      Baza1 baza=new Baza1();
      do
       { s=JOptionPane.showInputDialog(null, "Podaj wybor"
         +" \n1- Wypelnij tablice,"
         +"\n2 - Wyswietl dane osob"
         +"\n3 - Wyszukaj osobe i wyswietl jej dane"
         +"\nk - Koniec programu");
         ch = s.charAt(0);
         switch(ch)
          { case '1' : baza.wypelnij_tablice();  break;
            case '2' : baza.wyswietl_tablice();   break;
            case '3' : baza.wyszukaj_w_tablicy(); break;
             case 'k' :  JOptionPane.showMessageDialog(null, "Koniec programu"); break;
           default :   JOptionPane.showMessageDialog(null,"Zla opcja");
          }
       }while (ch != 'k') ;
     System.exit(0);
  }
 }
//**************************************************************************
```

Klasa Baza ma postać następującą

```
//definicja klasy Osoba jak wyżej

public class  Baza
 {static  int N;
   static public  void main(String args[])
     { Osoba Dane[]=null;              //tablica Osob nie jest składową klasy
       int ile=0;
      String s;
      char ch;
      do
      { s=JOptionPane.showInputDialog(null, "Podaj wybor"
        +" \n1 – Wypelnij tablice,"
        +"\n2 - Wyswietl dane osob"
        +"\n3 - Wyszukaj osobe i wyswietl jej dane"
        +"\nk - Koniec programu");
        ch = s.charAt(0);
        switch(ch)
         { case '1' : s=JOptionPane.showInputDialog(null,"Podaj rozmiar tablicy");
                      N=Integer.parseInt(s);
                      Dane=new Osoba[N];
                      for (ile=0; ile<Dane.length;ile++)
                        { Dane[ile]=new Osoba();
                          Dane[ile].Wstaw();}
                       break;
           case '2' :  if (Dane!=null)
                         for (int i=0; i<ile;i++)
                              Dane[i].Wyswietl();
                       break;
           case '3' :  if (Dane==null) break;
```

```
                    s = JOptionPane.showInputDialog(null,"Podaj nazwisko");
                    for (int i=0; i<ile;i++)
                       { if (Dane[i].Szukaj(s))
                              Dane[i].Wyswietl();    }
                    break;
            case 'k' :   JOptionPane.showMessageDialog(null, "Koniec programu"); break;
            default :   JOptionPane.showMessageDialog(null,"Zla opcja");
           }
         }while (ch != 'k') ;
        System.exit(0);
      }
     }
```

//**********

//**********

2. Uzupełnij metody klasy Baza1 o metodę sortującą wg nazwiska*: sortuj*_tablicę*_wg_nazwiska().* Należy wykorzystać metody do sortowania bąbelkowego z programu 2 z laboratorium3. Funkcja babelki teraz nazywa się *sortuj_tablicę_wg_nazwiska()*. Wywoływana w niej metoda *porownaj_zamien* ma teraz postać

```
    void  porownaj_zamien(int a, int b)         //  przekazanie indeksów tablicy
        { if (Dane[b].porownaj_nazwisko(Dane[a]))
            zamien(a, b);
        }                                                                    }
```

W tym celu należy zdefiniować dodatkową metodę *porownaj_nazwisko* w klasie Osoba. Oznacza to że porównanie atrybutów klasy Osoba do sortowania powinna wykonywać metoda klasy Osoba, gdyż *nazwisko* jest jej atrybutem (**zasada hermetyzacji**)! Metoda ta powinna zwracać wynik **true,** gdy nazwisko w obiekcie Dane[b] powinno być alfabetycznie przez nazwiskiem w obiekcie Dane[a], a w przeciwnym razie **false**.

```
public boolean porownaj_nazwisko (Osoba p)
{ if ( nazwisko.compareTo(p.nazwisko)<0) //metoda compareTo w klasie String działa tak funkcja strcmp w C/C++
    return true;
  else return false; }
```

//**********

3. Uzupełnij klasę Baza1 o metodę *sortowanie_wg_sredniej*, wg zasad podanych w punkcie 2.

//**********

4. Uzupełnij klasę Baza1 o metodę *usun_wg_nazwiska*, która ma następujący algorytm:
   4.1. Sprawdź, czy istnieje tablica *(Dane!=null*) – jeśli nieprawda, metoda kończy działanie
   4.2. Sprawdź, czy tablica nie jest pusta (ile!=0) – jeśli nieprawda, metoda kończy działanie
   4.3. Podaj z klawiatury szukane nazwisko (tak jak w metodzie *wyszukaj_w_tablicy*)
   4.4. Uruchom pętlę wyszukującą w tablicy (tak jak w metodzie *wyszukaj_w_tablicy*); jeśli zostanie znaleziony element (metoda z klasy *Szukaj* z klasy *Osoba*) to należy usunąć ten element z tablicy na jeden z podanych sposobów:
      4.4.1. wstawić w miejsce znalezionego elementu obiekt ostatni o indeksie *ile-1* i zmniejszyć liczbę elementów *ile* o 1 (*ile*--)
      4.4.2. zsunąć elementy i zmniejszyć liczbę elementów o 1 (ile--)
   4.5. Należy we wszystkich metodach (oprócz metody *wypelnij_tablice*, która ustawia wartość zmiennej *ile*) używać do określania liczby zmiennej *ile*, która zawiera aktualna liczbę elementów, która może być mniejsza niż rozmiar tablicy *Dane.length*

//**********

5*. Zmodyfikuj wstawianie do tablicy w następujący sposób:
   5.1. Gdy tablicy nie ma (*Dane=null*), program powinien zażądać od użytkownika rozmiar tablicy i utworzyć tablicę referencji do klasy *Osoba* o podanym z klawiatury rozmiarze.
   5.2. Na żądanie użytkownika powinien program wstawiać po jednej osobie wstawiając ją na pozycję o indeksie *ile*, gdy *ile* jest mniejsze od *Dane.length.* Jeśli dana zostanie wstawiona, należy ile powiększyć o 1 (np. ile++).
   5.3. Jeśli tablica jest wypełniona, należy utworzyć nową tablicę o większych rozmiarach (*Osoba pom[] = new [N+delta]; N+=delta*) i skopiować do niej elementy z zapełnionej tablicy (*Pom ← Dane*) i przypisać jej tablicę nowo utworzoną (*Dane=Pom*)- to pozwoli usunąć z pamięci niepotrzebną starą tablicę z danymi (*Dane)*

6*. Dodaj pewne komunikaty ułatwiając użytkownikowi śledzenie przebiegu programu, zabezpieczenia itp