LABORATORIUM 9

Cel: Zapis pojemników obiektów do pliku za pomocą mechanizmu serializacji realizowanego za pomocą klas **ObjectOutputStream** oraz **ObjectInputStream**. W zadaniach wykorzystuje się dwa typy pojemników: typu Collection rozróżniając wśród nich pojemniki typu List (ArrayList oraz LinkedList) oraz Set (HashSet, TreeSet) oraz typu Map (HashMap oraz TreeMap)

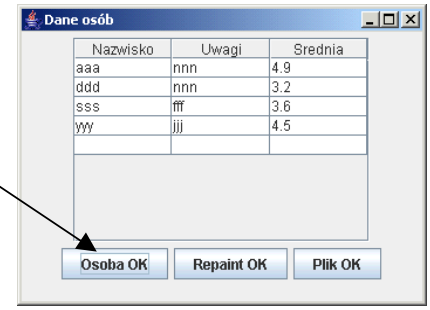
1. Uzupełnij zadanie 1 z laboratorium 8 o zapis do pliku oraz odczyt z pliku za pomocą strumieni **ObjectOutputStream** oraz **ObjectInputStream.** W projekcie *pliki1* (patrz załączony kod do instrukcji) pokazano, jak zapisywać i odczytywać pojemnik typu *Vector<Dane1> dane*. W zadaniu tym obiekt klasy *DaneOsob* w metodzie ***odczytaj_z_pliku()*** próbuje w bloku ***try*** odczytać pojemnik ***Vector <Dane1> dane*** z pliku ***dane.dat*** z katalogu bieżącego. Jeśli taki pojemnik jeszcze nie powstał, wystąpi wyjątek *IOException* i sterowanie w programie jest przekazane do bloku ***catch.*** W bloku ***catch*** obiekt typu *DaneOsob* tworzy w pamięci pusty pojemnik i przekazuje jego referencję do obiektu typu *Okno* (interfejs graficzny użytkownika). W interfejsie znajduje się przycisk „Plik OK", którego obsługa umożliwia zapis pojemnika *dane* do pliku *dane.dat* w katalogu bieżącym. Uzupełnij program 1 z laboratorium 8 o obsługę odczytu pliku *dane.dat*, wywołaną za pomocą dodanego przycisku „Odczyt Ok" i wypełnienie zawartością pliku pojemnik *dane* w następujący sposób:
   Metoda ***public void odczyt_z_pliku()*** w klasie *Panel* (zobacz również *metodę public void odczytaj_z_pliku()* w klasie *DaneOsob)*
   - //przygotowanie strumieni **FileInputStream** oraz **ObjectInputStream** do odczytu z pliku **dane.dat**
   - dane.clear(); //czyści zawartość pojemnika z danych
   - ArrayList<Dane1> pom= (ArrayList<Dane1>)plik.readObject(); // odczyt pojemnika z danymi z pliku
   - dane.addAll(pom); //wstawienie odczytanych danych do bieżącego pojemnika,
   - //zamknięcie strumieni

Należy uzupełnić obsługę zdarzeń o obsługę naciśnięcia przycisku „Odczyt OK" w klasie *Okno,* w której w chwili naciśnięcia tego przycisku wywoływana jest metoda z klasy Panel *public void odczyt_z_pliku()* (obsługa podobna jak przycisku „Zapisz OK" w klasie *Okno).* W klasie *Dane1* musi pojawić w liście **implementation** interfejs ***Serializable*** – podobnie jak w podanym przykładzie.
Uwaga: podobnie należy uzupełnić programy 2,3,4 z laboratorium 8 .

```
import java.io.*;
import java.lang.*;
import java.util.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.table.*;
import java.awt.*;
import java.awt.event.*;

class Panel extends JPanel implements ListSelectionListener
 { JTable tabela;
   DefaultTableModel model;
   Vector <Dane1> dane;
   int wiersz;

 public Panel(Vector <Dane1> v)
 { super();
    dane=v;
    final String[] columnNames = {"Nazwisko","Uwagi","Srednia"};
    model=new DefaultTableModel(columnNames,0);
    tabela= new JTable(model);
    model.addRow(new Vector<String>(3));
    tabela.setPreferredScrollableViewportSize(new Dimension(250,150));
    tabela.setSelectionMode(ListSelectionModel.SINGLE_SELECTION );
    ListSelectionModel selekcja = tabela.getSelectionModel();
    selekcja.addListSelectionListener(this);
    JScrollPane suwak = new JScrollPane(tabela);
    add(suwak);
   }

 public void valueChanged(ListSelectionEvent e)
   { if (e.getValueIsAdjusting()) return;
       ListSelectionModel lsm = (ListSelectionModel)e.getSource();
      if (!lsm.isSelectionEmpty())
        { wiersz = lsm.getMinSelectionIndex();
          System.out.println("Wiersz " + wiersz+ " jest wybrany.");  }
    }
```

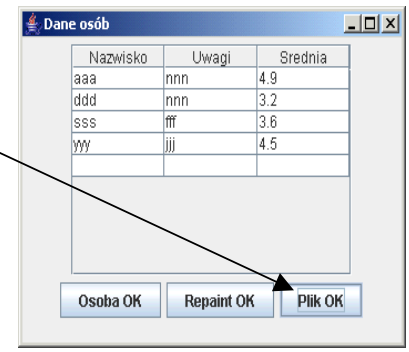
```
public void wyswietl()
 {    Iterator iterator = dane.iterator();
      model.setRowCount(0);
      int i=0;
      while(iterator.hasNext())
        { Dane1 t=(Dane1)iterator.next();
          Vector <String>vv=new Vector<String>(3);
          model.addRow(vv);
          model.setValueAt(t.Podaj_nazwisko(),i,0);
          model.setValueAt(t.Podaj_uwagi(),i,1);
          model.setValueAt(""+t.Podaj_srednia(),i,2);
          i++;
        }
    Vector <String>vv=new Vector<String>(3);
    model.addRow(vv);
 }
```

3) Stan tabeli po starcie programu, gdy nie było pliku dane.dat w katalogu bieżącym – wynik wprowadzania danych do tabeli




```
 void obslugaactionPerformed()
  { if(tabela.isRowSelected(wiersz))
      { String nazwisko=(String)model.getValueAt(wiersz, 0); //pobranie z wybranych
        String uwagi=(String)model.getValueAt(wiersz, 1); // komórek łańcuchy
        String srednia=(String)model.getValueAt(wiersz, 2);
        if(nazwisko!=null&&uwagi!=null&&srednia!=null&& !nazwisko.equals("")&& !uwagi.equals("")  &&!srednia.equals(""))
          { try
            { Float.parseFloat(srednia);
              Dane1 d=Dane1.Wstaw(nazwisko, uwagi, srednia);
              dane.add(d);
              Collections.sort(dane);
              wyswietl();
             } catch(NumberFormatException e)
               { JOptionPane.showMessageDialog(this, "Wprowadz poprawną średnią");}
          }
      }
  }
```

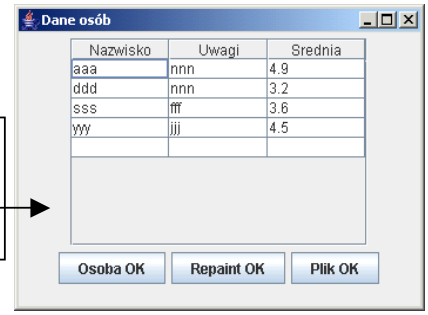
4) Stan tabeli po starcie programu, gdy nie było pliku dane.dat w katalogu bieżącym – uruchomiono zapis do pliku




```
 public void zapisz_do_pliku()
   { try
     {   FileOutputStream p = new FileOutputStream ("dane.dat");
         ObjectOutputStream plik = new ObjectOutputStream (p);
         plik.writeObject(dane);
         plik.close();
      } catch(IOException e)
        { System.out.println("Nie udalo sie zapisac danych do pliku"); }
   }
 }
//*************************************************************************************************
class Okno extends JFrame implements ActionListener
{   Panel  panel;    //składniki interfesju graficznego użytkownika
    JButton osobaOK ;
    JButton repaintOK ;
    JButton plikOK;

   public Okno(Vector <Dane1>dane)
   { super("Dane osób");
     setSize(350,250);
     osobaOK=new JButton("Osoba OK");
     osobaOK.addActionListener(this);
     repaintOK=new JButton("Repaint OK");
     repaintOK.addActionListener(this);
     plikOK=new JButton("Plik OK");
     plikOK.addActionListener(this);
     setDefaultLookAndFeelDecorated(true);
     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     panel = new Panel(dane);
     panel.setOpaque(true);
     panel.add(osobaOK);
     panel.add(repaintOK);
     panel.add(plikOK);
     setContentPane(panel);
     panel.wyswietl();          //zapis danymi z pliku, jeśli odczytano
    setVisible(true);  }        // plik lub pusta tabele, gdy brak pliku
```

2) Stan tabeli po starcie programu, gdy nie było pliku dane.dat w katalogu bieżącym

6) Stan tabeli po starcie programu, gdy był plik dane.dat w katalogu bieżącym



```
 public void actionPerformed(ActionEvent evt)
   { Object zrodlo=evt.getSource();
     if (zrodlo==osobaOK)
        panel.obslugaactionPerformed();
     else
      if (zrodlo==repaintOK)
         panel.wyswietl();
      else if (zrodlo==plikOK)
         panel.zapisz_do_pliku();
      repaint();          //wywołanie funkcji odświeżającej zawartość tabeli i pokazanie jej na ekranie
    }
}
//*************************************************************************************************************
public class DaneOsob
 {  Vector<Dane1> dane;
```

1) Brak pliku *dane.dat* przy starcie programu

```
public void odczytaj_z_pliku()
    {    try
          {  FileInputStream p = new FileInputStream ("dane.dat");
             ObjectInputStream plik = new ObjectInputStream (p);
             dane=(Vector<Dane1>)plik.readObject();
             plik.close();
          } catch(Exception e)
            {      System.out.println("Nie udalo sie odczytac danych z pliku - utworzono pusty pojemnik");
                   dane = new Vector <Dane1> ();    }
    }

 static  public void main(String arg[])
      {  DaneOsob baza = new  DaneOsob();
         baza.odczytaj_z_pliku();
        Okno okno = new Okno(baza.dane);
      }
}
```

5) Obecność pliku d*ane.dat* przy starcie programu

```
//*************************************************************************************************************
import java.util.*;
import java.io.*;

public class Dane1 implements Comparable <Dane1>,  Serializable
 { String nazwisko;
   float srednia;
   String uwagi;
   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 static Dane1 Wstaw(String nazwisko_, String uwagi_, String srednia_)
    {  Dane1 d=new Dane1();
       d.Nadaj_nazwisko(nazwisko_);
       d.Nadaj_srednia(Float.parseFloat(srednia_));
       d.Nadaj_uwagi(uwagi_);
       return d;    }
public String toString()
   { String napis="";
     napis+=" Nazwisko: "+nazwisko;
     napis+=" Srednia: "+srednia;
     napis+=" Uwagi: "+uwagi+"\n";
     return napis;  }
 public boolean equals(Object o)                //metoda umożliwiająca przetwarzanie danych w pojemnikach typu Hash
   {  Dane1 d=(Dane1)o;
      return  nazwisko.equals(d.nazwisko)  && srednia==d.srednia  && uwagi.equals(d.uwagi);  }
 public int hashCode()                          //metoda umożliwiająca przetwarzanie danych w pojemnikach typu Hash
  {return nazwisko.hashCode()+(int)srednia+uwagi.hashCode();   }
 //metoda umożliwiająca sortowanie, wyszukiwanie w pojemnikach List oraz przetwarzanie w pojemnikach typu TreeSet i TreeMap
 public int compareTo(Dane1 o)
   { return nazwisko.compareTo(o.nazwisko);   } }
```