НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Інститут прикладного системного аналізу

Факультет другої та післядипломної освіти

**ЛАБОРАТОРНА РОБОТА**

з дисципліни "Технології розподілених систем

і паралельних обчислень"

тема лабораторної роботи:

«Використання технології Swing та багатопоточності

при розробці додатків на Java»

**Виконала:**

Студентка групи ІСзп-31

Варіант №2

Садовець А.М.

**Перевірив:**

Мурга М.О.

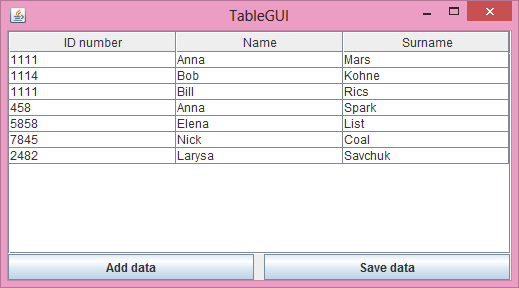
КИЇВ – 2014

**Опис завдання по лабораторному практикуму:**

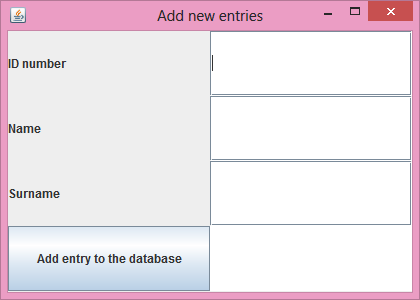
Варіант 2. Розробити два програмних додатка. Необхідно, щоб в табличний елемент першого програмного додатку можна було додавати записи та зберігати існуючі в ньому записи на диск. Другий програмний додаток повинен мати поля для створення нового запису та керуючі кнопки для передачі запису та зберігання результату на перший програмний додаток. Для реалізації системи використати технологію Java-RMI та засоби синхронізації.

**Скріншоти GUI програмного додатку**

**Програмний додаток 1**

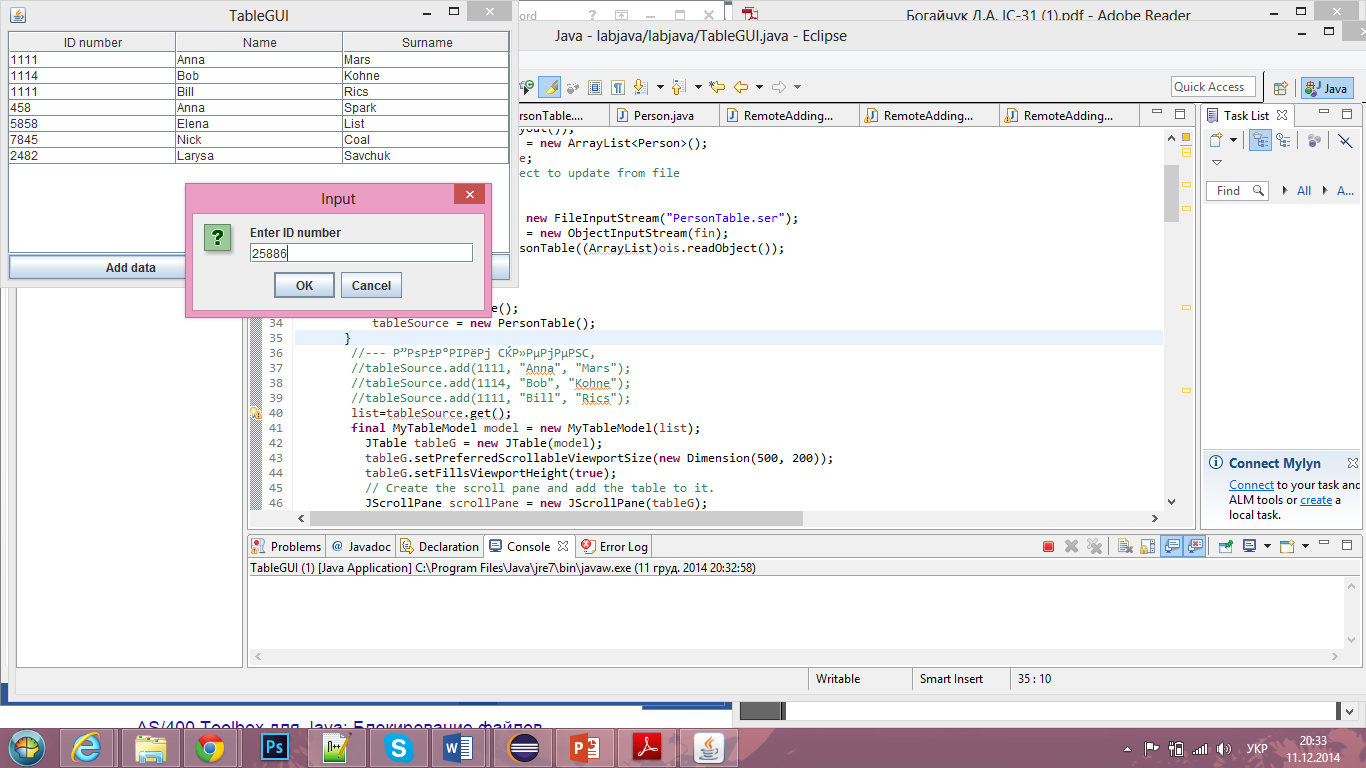


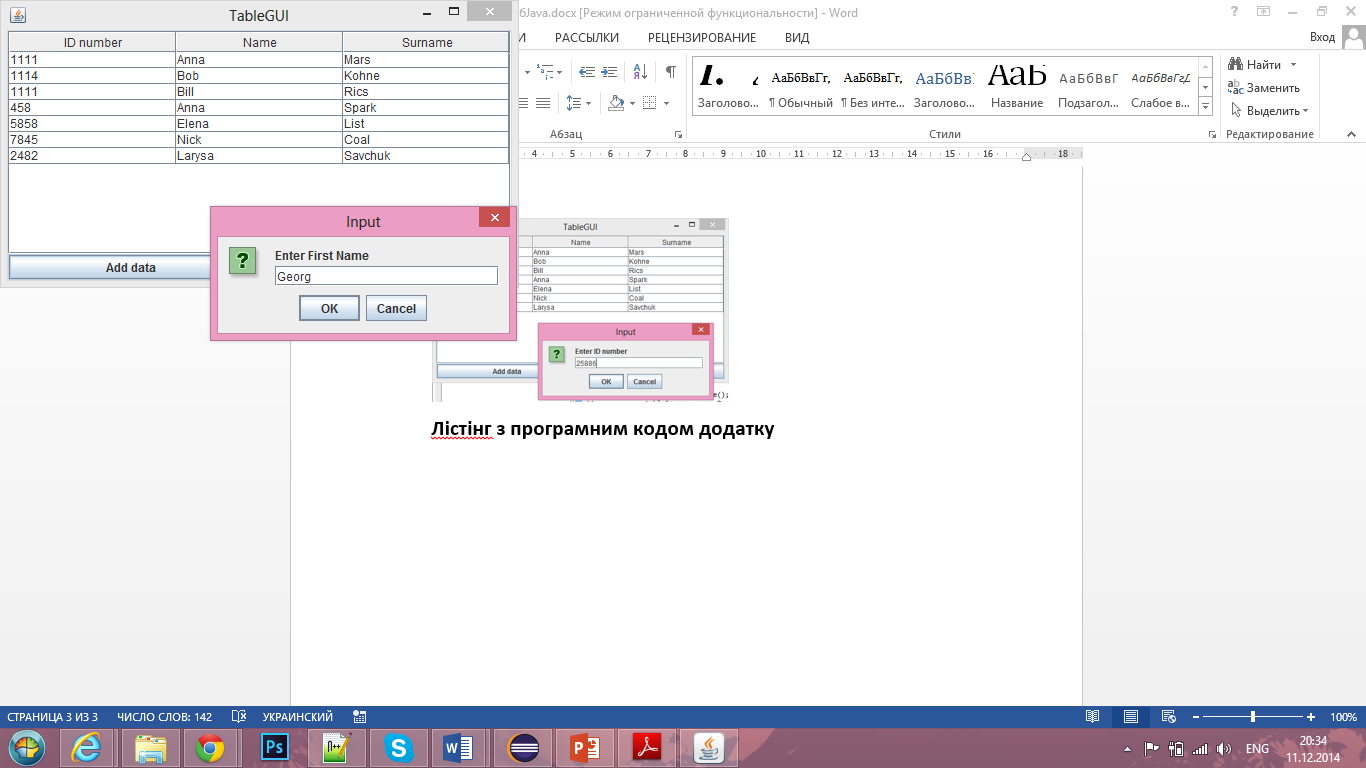
**Програмний додаток 2**

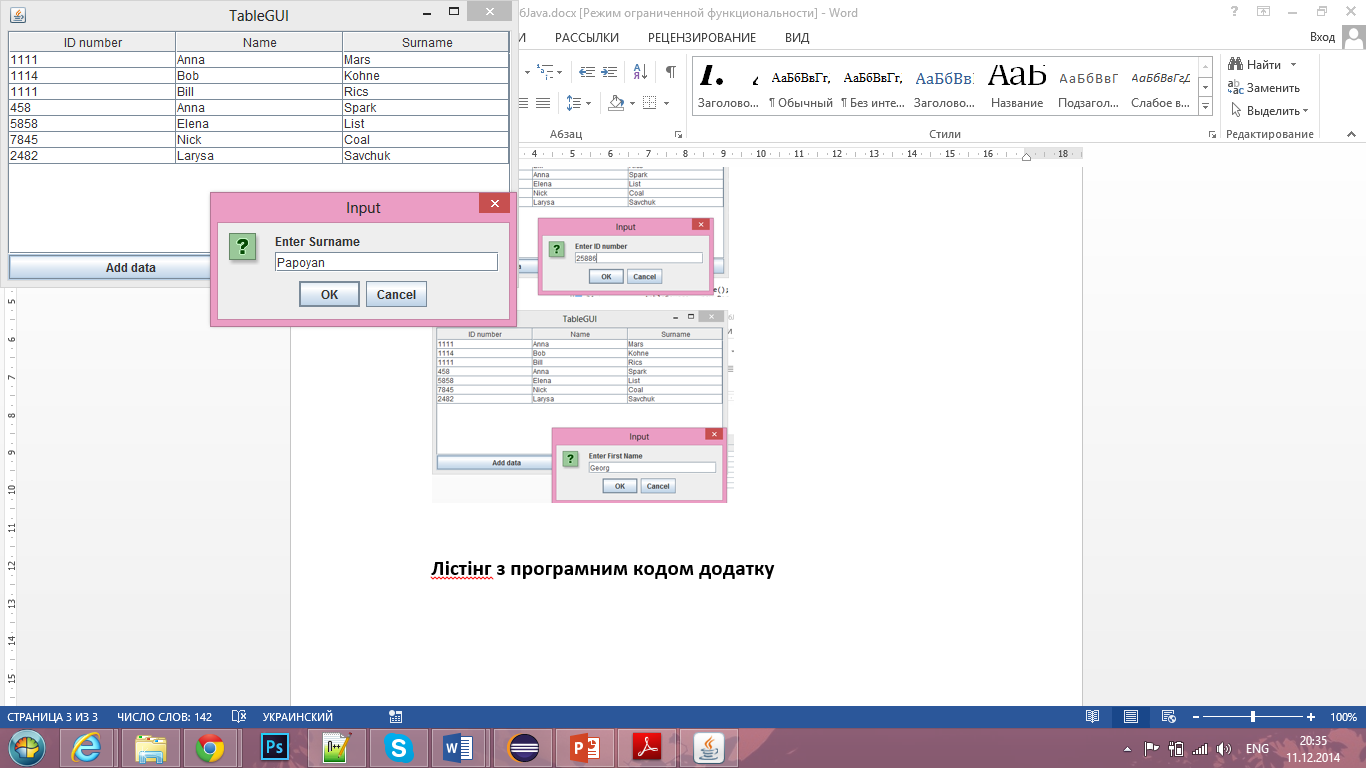
****

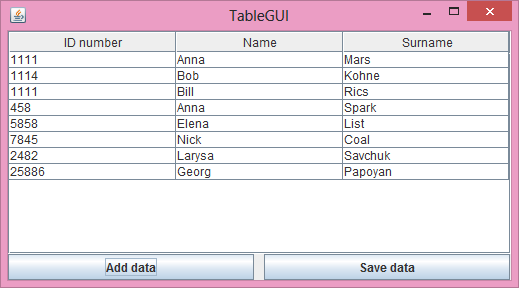
**Тестові приклади, що демонструють роботу додатку**

Додавання нового запису в першому додатку

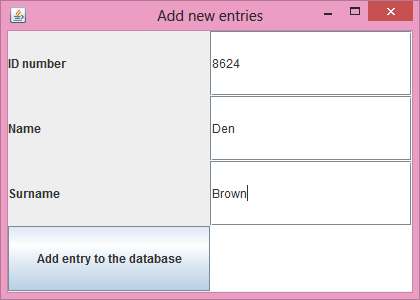
****

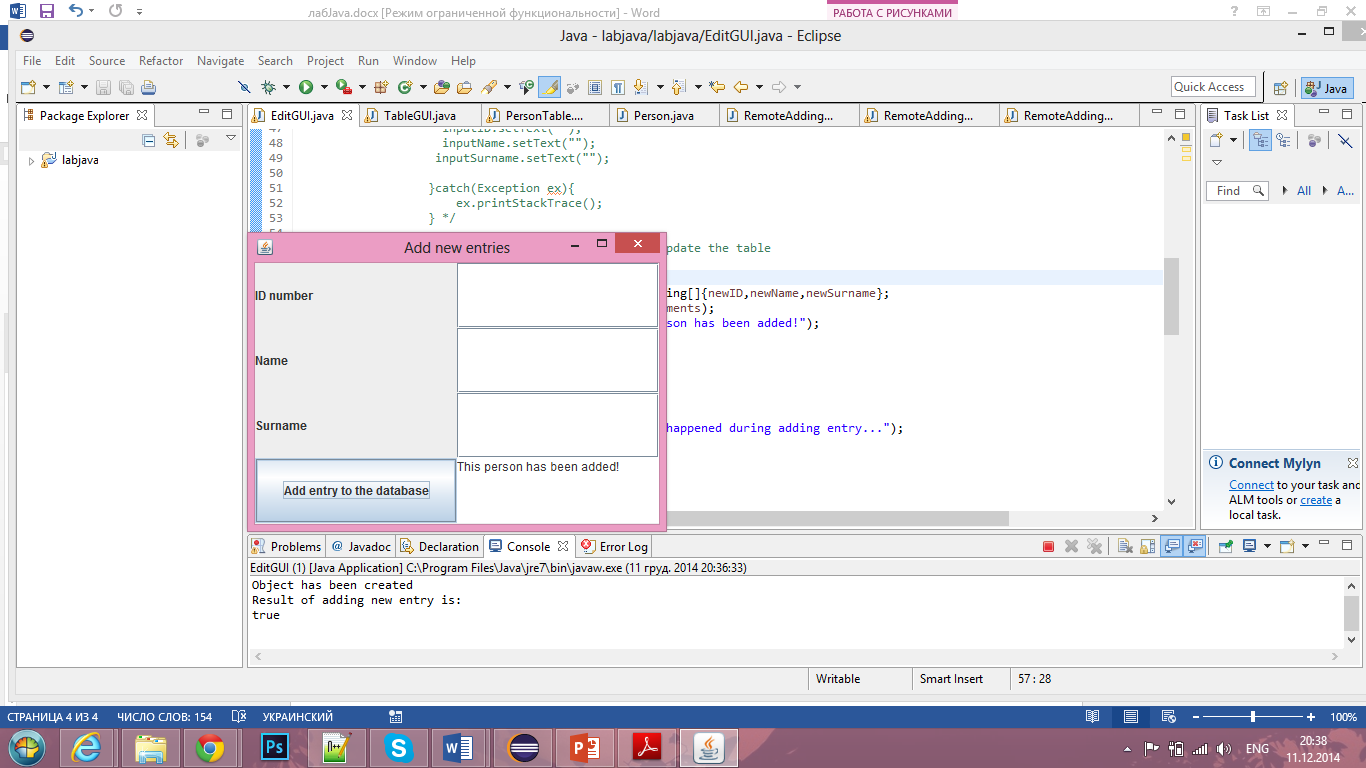
****

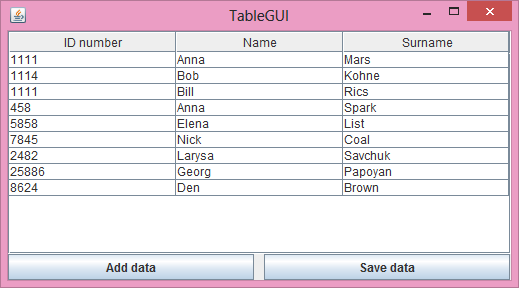
****

****

Додавання нового запису з другого додатку в таблицю з використанням технології RMI

****

****

****

**Лістінг з програмним кодом додатку**

**package** labjava;

**import** java.rmi.\*;

**public** **interface** **RemoteAdding** **extends** Remote

{

**public** **boolean** add(String id,String name, String surname) **throws** RemoteException;

}

**package** labjava;

**import** java.rmi.\*;

**import** java.rmi.registry.LocateRegistry;

**import** java.rmi.registry.Registry;

**public** **class** **RemoteAddingUser**

{

**public** **static** **void** main(String[] args) **throws** Exception

{

System.***out***.println("Trying to connect...");

**try**{

Registry myReg = LocateRegistry.*getRegistry*("127.0.0.1",1099);//Get reference to the remote object Registry on the specified host and port

RemoteAdding aObj = (RemoteAdding) Naming.*lookup*("newAdd");//Get reference, a stub, for the remote object associated with the specified name.

System.***out***.println("Object has been created");

System.***out***.println("Result of adding new entry is:");

System.***out***.println(aObj.add(args[0],args[1],args[2]));

}

**catch**(RemoteException re)

{

System.***out***.println("A problem happened during adding entry...");

}

}

}

**package** labjava;

**import** java.util.ArrayList;

**import** java.awt.\*;

**import** java.awt.event.ActionListener;

**import** java.awt.event.ActionEvent;

**import** javax.swing.\*;

**import** java.io.FileInputStream;

**import** java.io.ObjectInputStream;

**import** labjava.TableGUI.MyTableModel;

**public** **class** **EditGUI** **extends** JFrame

{ // Instantiate textfields for input and a textarea for output.

**public** JLabel idLabel = **new** JLabel("ID number");

**public** JLabel nameLabel = **new** JLabel("Name");

**public** JLabel surnameLabel = **new** JLabel("Surname");

**public** JButton addDataButton = **new** JButton("Add entry to the database");

**public** JTextArea output = **new** JTextArea(3,20);

**private** JTextField inputID = **new** JTextField(8);

**private** JTextField inputName = **new** JTextField(20);

**private** JTextField inputSurname = **new** JTextField(20);

**public** EditGUI()

{ // Register a listener with the button

**class** MyListener **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent e){

String newID = inputID.getText();

String newName = inputName.getText();

String newSurname = inputSurname.getText();

//try connecting through rmi to update the table

**try**{

String[] arguments=**new** String[]{newID,newName,newSurname};

RemoteAddingUser.*main*(arguments);

output.setText("This person has been added!");

inputID.setText("");

inputName.setText("");

inputSurname.setText("");

}**catch**(Exception ex){

ex.printStackTrace();

output.setText("A problem happened during adding entry...");

}

}

}

addDataButton.addActionListener(**new** MyListener());

// Don't let the user change the output.

output.setEditable(**false**);

// Add all the components to the frame

**this**.getContentPane().add(idLabel);

**this**.getContentPane().add(inputID);

**this**.getContentPane().add(nameLabel);

**this**.getContentPane().add(inputName);

**this**.getContentPane().add(surnameLabel);

**this**.getContentPane().add(inputSurname);

**this**.getContentPane().add(output);

**this**.getContentPane().add(addDataButton);

**this**.getContentPane().add(output);

inputID.requestFocus(); // start with focus on this field

}

/\*\*

\* Create the GUI and show it. For thread safety, this method should be

\* invoked from the event-dispatching thread.

\*/

**private** **static** **void** createAndShowGUI() {

// Create and set up the window.

EditGUI frame = **new** EditGUI();

frame.setTitle("Add new entries");

frame.setLocation(20,100);

frame.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

frame.setLayout(**new** GridLayout(4,2));

frame.setPreferredSize(**new** Dimension(420, 300));

frame.pack();//sets the minimum size needed to display GUI

frame.setVisible(**true**); // Display the window.

}

**public** **static** **void** main(String[] args) {

// Schedule a job for the event-dispatching thread: creating and showing this application's GUI.

javax.swing.SwingUtilities.*invokeLater*(**new** Runnable() {

**public** **void** run() {

*createAndShowGUI*();

}

});

}

}

**package** labjava;

**import** java.rmi.\*;

**import** java.util.\*;

**import** java.rmi.registry.LocateRegistry;

**import** java.rmi.registry.Registry;

**import** java.rmi.server.\*;

**import** java.lang.Throwable;

**public** **class** **RemoteAddingImpl** **extends** UnicastRemoteObject **implements** RemoteAdding{

RemoteAddingImpl() **throws** RemoteException{

**super**();

}

**public** **synchronized** **boolean** add(String id,String name, String surname) **throws** RemoteException //Only one thread can execute inside a synchronized instance method

{

**try**{

PersonTable table=**new** PersonTable();

table.loadData();

table.add(id, name, surname);

table.saveData();

**return** **true**;

}

**catch** (Throwable t){**return** **false**;}

}

**public** **synchronized** **static** **void** main(String args[]) **throws** Exception

{

**try**{

Registry reg = LocateRegistry.*createRegistry*(1099); //Creates Registry instance on the local host that accepts requests on the specified port.

RemoteAdding aObj = **new** RemoteAddingImpl();

Naming.*rebind*("newAdd", aObj);//Rebind the specified name to a new remote object

System.***out***.println("Server is ready...");

}

**catch**(Exception ex){

ex.printStackTrace();

}

}

}

**package** labjava;

**import** java.awt.Dimension;

**import** java.awt.GridLayout;

**import** java.awt.BorderLayout;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.util.ArrayList;

**import** javax.swing.\*;

**import** javax.swing.table.AbstractTableModel;

**import** java.io.FileInputStream;

**import** java.io.ObjectInputStream;

**import** java.io.Serializable;

**public** **class** **TableGUI** **extends** JPanel {

**public** TableGUI() {

**super**();

setLayout(**new** BorderLayout());

ArrayList<Person> list = **new** ArrayList<Person>();

PersonTable tableSource;

//read table object to update from file

**try**{

FileInputStream fin = **new** FileInputStream("PersonTable.ser");

ObjectInputStream ois = **new** ObjectInputStream(fin);

tableSource = **new** PersonTable((ArrayList)ois.readObject());

ois.close();

}**catch**(Exception ex){

ex.printStackTrace();

tableSource = **new** PersonTable();

}

list=tableSource.get();

**final** MyTableModel model = **new** MyTableModel(list);

JTable tableG = **new** JTable(model);

tableG.setPreferredScrollableViewportSize(**new** Dimension(500, 200));

tableG.setFillsViewportHeight(**true**);

JScrollPane scrollPane = **new** JScrollPane(tableG);// Create the scroll pane and add the table to it.

add(scrollPane,BorderLayout.***CENTER***); // Add the scroll pane to this panel.

**final** PersonTable temp=tableSource;

**class** MyListener **implements** ActionListener{

**int** action;

MyListener(**int** action){

**this**.action=action;

}

**public** **void** actionPerformed(ActionEvent e){

**switch**(action){

**case** 1:

{

String newID = JOptionPane.*showInputDialog*("Enter ID number");

String newName = JOptionPane.*showInputDialog*("Enter First Name");

String newSurname = JOptionPane.*showInputDialog*("Enter Surname");

Person person=**new** Person(newID, newName, newSurname);

((MyTableModel)model).add(person);};

**case** 2:

{ temp.saveData();} ;

}

}

}

JPanel inner = **new** JPanel();

inner.setLayout(**new** GridLayout(1, 2, 10, 0));

add(inner,BorderLayout.***SOUTH***);

JButton addDataButton = **new** JButton("Add data");

inner.add(addDataButton);

addDataButton.addActionListener(**new** MyListener(1));

JButton saveDataButton = **new** JButton("Save data");

inner.add(saveDataButton);

saveDataButton.addActionListener(**new** MyListener(2));

}

**class** **MyTableModel** **extends** AbstractTableModel {

**private** String[] columnNames = { "ID number", "Name", "Surname" };

ArrayList<Person> list = **null**;

MyTableModel(ArrayList<Person> list) {

**this**.list = list;

}

**public** **int** getColumnCount() {

**return** columnNames.length;

}

**public** **int** getRowCount() {

**return** list.size();

}

**public** String getColumnName(**int** col) {

**return** columnNames[col];

}

**public** Object getValueAt(**int** row, **int** col) {

Person object = list.get(row);

**switch** (col) {

**case** 0:

**return** object.getID();

**case** 1:

**return** object.getName();

**case** 2:

**return** object.getSurname();

**default**:

**return** "unknown";

}

}

**public** Class getColumnClass(**int** c) {

**return** getValueAt(0, c).getClass();

}

**public** **void** add(Person person) {

**int** size = list.size();

list.add(person);

fireTableRowsInserted(size, size);

}

}

//Create the GUI and show it, should be invoked from the event-dispatching thread.

**private** **static** **void** createAndShowGUI() {

// Create and set up the window.

JFrame frame = **new** JFrame("TableGUI");

frame.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

TableGUI newContentPane = **new** TableGUI();// Create and set up the content pane.

newContentPane.setOpaque(**true**); // content panes must be opaque

frame.setContentPane(newContentPane);

frame.pack();// Display the window.

frame.setVisible(**true**);

}

**public** **static** **void** main(String[] args) {

// Schedule a job for the event-dispatching thread: creating and showing this application's GUI.

javax.swing.SwingUtilities.*invokeLater*(**new** Runnable() {

**public** **void** run() {

*createAndShowGUI*();

}

});

}

}

**package** labjava;

**import** java.util.\*;

**import** java.io.\*;

**public** **class** **PersonTable** {

**private** ArrayList list;

**public** PersonTable() {

list = **new** ArrayList();

}

**public** PersonTable(ArrayList l) {

list = l;

}

**public** ArrayList get() {

**return** list;

}

**public** **void** add(**int** a,String n,String m) {

Person obj = **new** Person(a,n,m);

list.add(obj);

}

**public** **void** add(String a,String n,String m) {

Person obj = **new** Person(Integer.*valueOf*(a),n,m);

list.add(obj);

}

**public** String toString() {

**return** list.toString();

}

**public** **void** print(String title) {

System.***out***.println(title);

Iterator iter = list.iterator();

**for** ( **int** i = 0; iter.hasNext(); i++ ) {

System.***out***.println("N "+i+"="+iter.next());

}

}

**public** **void** saveData(){//save data to the file

ObjectOutputStream out = **null**;

**try** {

out = **new** ObjectOutputStream(**new** BufferedOutputStream(

**new** FileOutputStream("PersonTable.ser")));

out.writeObject(list);

} **catch** ( IOException ex ) {

ex.printStackTrace();

} **finally** {

**if** ( out != **null** )

**try** {

out.close();

} **catch** ( IOException ex ) {

ex.printStackTrace();

}

}

}

**public** **void** loadData(){//load data from file

ObjectInputStream in = **null**;

**try** {

in = **new** ObjectInputStream(**new** BufferedInputStream(

**new** FileInputStream("PersonTable.ser")));

list = (ArrayList)in.readObject();

} **catch** ( IOException ex ) {

ex.printStackTrace();

} **catch** ( Exception ex ) {

ex.printStackTrace();

} **finally** {

**if** ( in != **null** )

**try** {

in.close();

} **catch** ( IOException ex ) {

ex.printStackTrace();

}

}

}

**public** **static** **void** main(String args[]) {//optional initialization of new table, will rewrite existing one!!!

PersonTable table1=**new** PersonTable();

table1.add(1111, "Anna", "Mars");

table1.add(1114, "Bob", "Kohne");

table1.add(1111, "Bill", "Rics");

table1.saveData();

table1.print("Initial List");

table1.loadData();

table1.print("Recreated List");

}

}

**package** labjava;

**import** java.io.\*;

**public** **class** **Person** **implements** Serializable {

**private** String idnumber;

**private** String name;

**private** String surname;

**public** Person(**int** numb,String n,String s) {

idnumber = String.*valueOf*(numb);

name=n;

surname=s;

}

**public** Person(String numb,String n,String s) {

idnumber = numb;

name=n;

surname=s;

}

**public** Person() {

idnumber = **null**;

name="";

surname="";

}

**public** String getID() {**return** **this**.idnumber;}

**public** String getName() {**return** **this**.name;}

**public** String getSurname() {**return** **this**.surname;}

**public** String toString() {

**return** " "+": "+idnumber+" "+name+" "+surname;

}

}