Санкт-Петербургский Политехнический Университет

Институт компьютерных наук и технологий

«Высшая школа программной инженерии»

**Курсовая работа**

по дисциплине: «Системы управления базами данных»

Выполнил:

студент гр. В3530904/80322 Быков А.Ю.

Руководитель: Смирнов Н.Г.

Санкт-Петербург

2021

**Содержание**

[**Введение** 3](#_Toc71827501)

[**Цель задания** 4](#_Toc71827502)

[**Ход работы** 5](#_Toc71827503)

[**1.** **Создание формы** 5](#_Toc71827504)

[**2.** **Создание алгоритма таймера** 6](#_Toc71827505)

[**3.** **Создание алгоритма сохранение результатов** 7](#_Toc71827506)

[**4.** **Создание метода загрузки прошлых результатов** 7](#_Toc71827507)

[**Итоговый вид работы** 8](#_Toc71827508)

[**Вывод** 9](#_Toc71827509)

[**Приложение** 10](#_Toc71827510)

# **Введение**

Для выполнения работы используется IntelliJ IDEA Community Edition. IntelliJ IDEA — интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains.

# **Цель задания**

Создать фитнес-трекер. Приложение позволяет указать вид тренировки: отжимание, скакалка, приседания. Доступны команды: начать тренировку (запускается таймер тренировки), закончить тренировку (таймер останавливается). За отработанное время высчитывается количество потраченных калорий по формуле K\*t=cal, где K - количество калорий в час, затрачиваемое на определенный вид тренировки, t - время, засеченное трекером. От запуска к запуску программы данные должны сохраняться и общее количество калорий - суммироваться.

**Доп задание:**

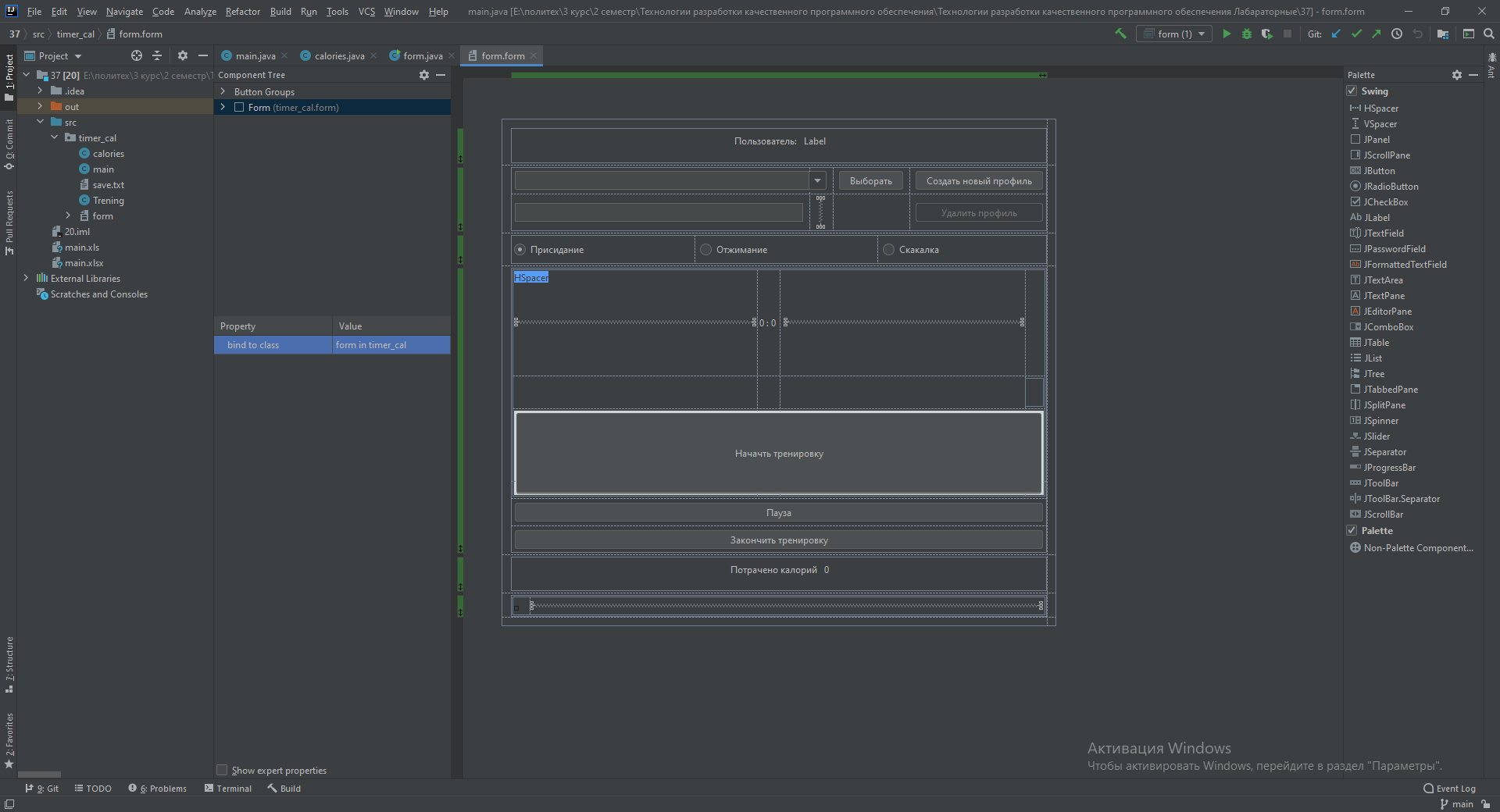
* Персистенция данных приложения с помощью ObjectOutputStream +5 баллов или JAXB + 10 баллов
* Поддержка нескольких профилей пользователей приложением + 5 баллов"

# **Ход работы**

## **Создание формы**

Для создания визуального интерфейса используется встроенный конструктор форм IntelliJ IDEA.

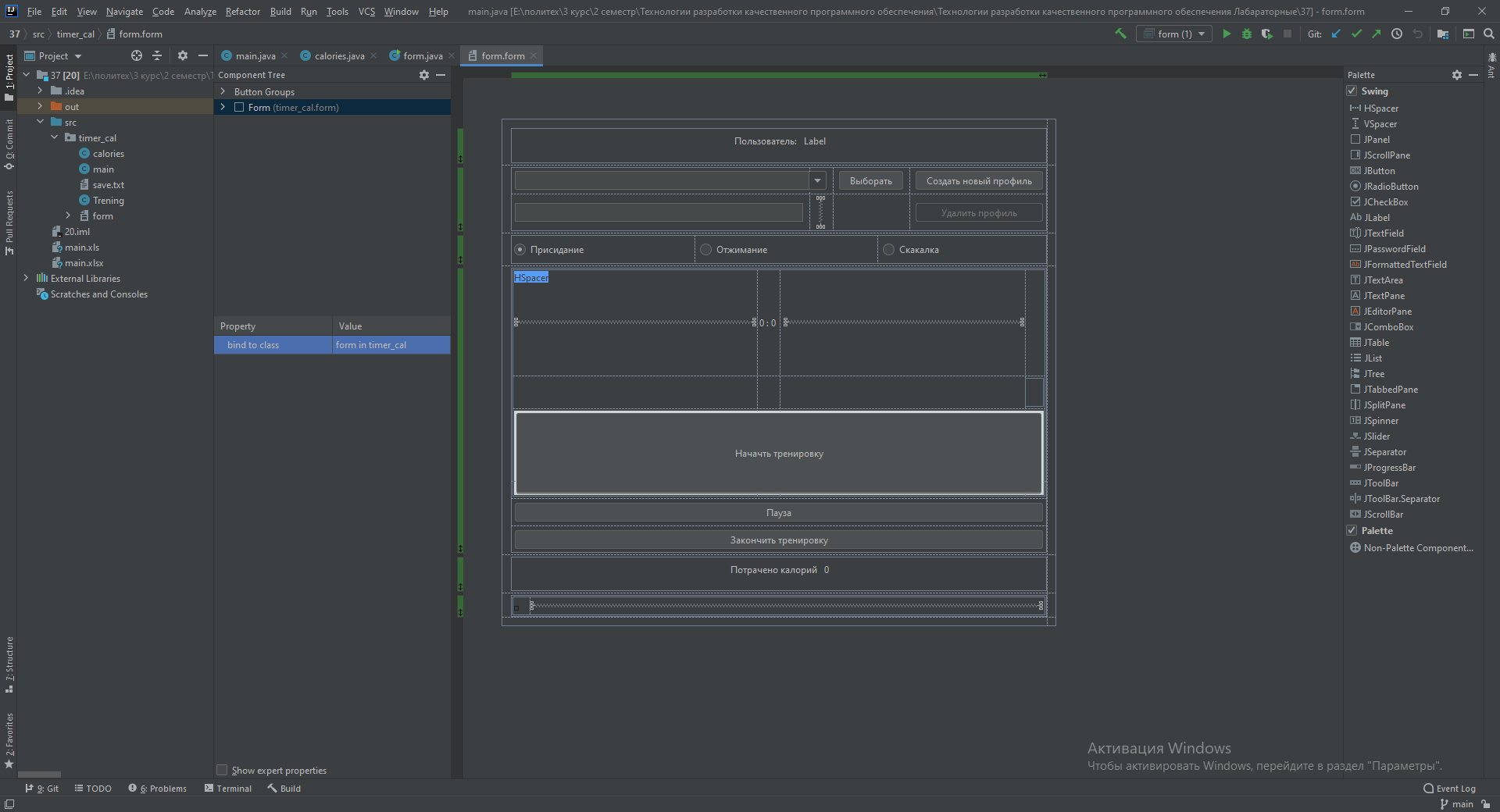
Добавление элементов с помощью бокового окна



Для трекера требуется:

* Кнопки для управления трекиром и его профилями;
* Поле для ввода названия профиля;
* СomboBox для выбора профиля;
* Поля для вывода таймера и количества потраченных калорий;
* RadioButton для выбора тренировки.

Итоговый вид формы:



## **Создание алгоритма таймера**

Runnable run\_timer = new Runnable() {  
 @Override  
 public void run() {  
 while (true){  
 try{  
 Thread.*sleep*(1000);  
 if (time\_stop == true) {  
 sec = sec + 1;  
 sec\_all = sec\_all + 1;  
 if (sec == 60) {  
 minut++;  
 sec = 0;  
 }  
 labelTime.setText(minut + " : " + sec);   
 }  
 }  
 catch (InterruptedException er){  
 er.printStackTrace();  
 }  
 }  
 }  
 };

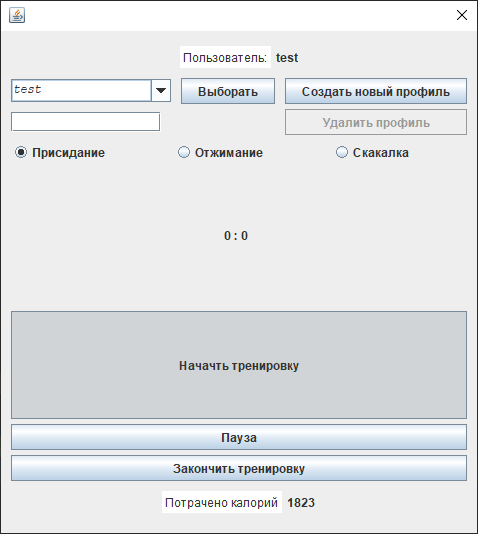
## **Создание алгоритма сохранение результатов**

PrintWriter writer = null;  
try {  
 writer = new PrintWriter(file, "UTF-8");  
} catch (FileNotFoundException e) {  
 e.printStackTrace();  
} catch (UnsupportedEncodingException e) {  
 e.printStackTrace();  
}  
for (int i=0; i<stringArr\_form.length; i++){  
 writer.println(stringArr\_form[i]);  
  
}  
writer.close();

## **Создание метода загрузки прошлых результатов**

public static String[] open\_fail() throws IOException {  
 BufferedReader reader = new BufferedReader(new FileReader("E:\\политех\\3 курс\\2 семестр\\Технологии разработки качественного программного обеспечения\\Технологии разработки качественного программного обеспечения Лабараторные\\37\\src\\timer\_cal\\save.txt"));  
 String str;  
  
 ArrayList<String> list = new ArrayList<String>();  
 while((str = reader.readLine()) != null ){  
 if(!str.isEmpty()){  
 list.add(str);  
 System.*out*.println(str);  
 }  
 }  
 String[] stringArr = list.toArray(new String[0]);  
 //System.out.println(stringArr[0]);  
 return stringArr;  
  
}

# **Итоговый вид работы**



# **Вывод**

При выполнение данной работы были получены навыки работы на языке Java, а также работы по созданию визуального интерфейса. Создавая таймер для приложения не были использованы стандартные библиотеки Java, а писался самописный код для него.

# **Приложение**

Исходный код:

1. Form.java

package timer\_cal;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.\*;  
import java.io.\*;  
import java.nio.charset.StandardCharsets;  
import java.nio.file.Files;  
import java.io.File;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.nio.file.StandardOpenOption;  
import java.util.ArrayList;  
import java.util.Scanner;  
import java.util.Timer;  
import java.util.TimerTask;  
import com.google.common.base.Stopwatch;  
import java.util.concurrent.TimeUnit;  
  
  
public class form extends JDialog {  
 private JPanel contentPane;  
 private JRadioButton RadioButton1;  
 private JRadioButton RadioButton3;  
 private JRadioButton RadioButton2;  
 private JButton ButtonStart;  
 private JComboBox comboBox1;  
 private JButton ButtonCreateProf;  
 private JButton ButtonEnd;  
 private JButton ButtonDelete;  
 private JLabel labelTime;  
 private JTextField textField1;  
 private JTextPane CalTextPane;  
 private JLabel LabelCal;  
 private JPanel PanelCal;  
 private JButton ButtonPaus;  
 private JLabel label\_test;  
 private JTextPane UserTextPane;  
 private JButton button\_users;  
 private JLabel users\_label;  
  
 private Timer myTimer;  
 private TimerTask mMyTimerTask;  
 public String[] stringArr\_form;  
 public String[] stringArr\_form\_new;  
 private File file;  
  
  
  
  
  
  
 public form() {  
 setContentPane(contentPane);  
 setModal(true);  
  
 file = new File("E:\\политех\\3 курс\\2 семестр\\Технологии разработки качественного программного обеспечения\\Технологии разработки качественного программного обеспечения Лабараторные\\37\\src\\timer\_cal\\save.txt");  
  
  
  
 ButtonStart.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent actionEvent) {  
  
 labelTime.setText(textField1.getText());  
 Thread thread = new Thread(run\_timer);  
 thread.start();  
  
 }  
 });  
 ButtonCreateProf.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent actionEvent) {  
 if(textField1.getText()!= " ") {  
 comboBox1.addItem(textField1.getText());  
 }  
  
  
 stringArr\_form\_new = new String[stringArr\_form.length+2];  
 for (int i=0; i<stringArr\_form.length; i++){  
 stringArr\_form\_new[i]=stringArr\_form[i];  
 }  
 System.*out*.println(stringArr\_form\_new.length);  
  
 stringArr\_form\_new[stringArr\_form\_new.length-2]=textField1.getText();  
 stringArr\_form\_new[stringArr\_form\_new.length-1]="0";  
  
 PrintWriter writer = null;  
 try {  
 writer = new PrintWriter(file, "UTF-8");  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 } catch (UnsupportedEncodingException e) {  
 e.printStackTrace();  
 }  
 for (int i=0; i<stringArr\_form\_new.length; i++){  
 writer.println(stringArr\_form\_new[i]);  
  
 }  
 writer.close();  
  
 stringArr\_form = new String[stringArr\_form\_new.length];  
 for (int i=0; i<stringArr\_form.length; i++){  
 stringArr\_form[i]=stringArr\_form\_new[i];  
 }  
  
 }  
 });  
 ButtonDelete.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent actionEvent) {  
 //comboBox1.removeItem();  
 }  
 });  
  
  
  
  
 ButtonStart.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent actionEvent) {  
 labelTime.setText(minut + " : " + sec);  
 time\_stop=true;  
 if(RadioButton1.isSelected()==true){  
 view\_tren = RadioButton1.getText();  
 }  
 if(RadioButton2.isSelected()==true){  
 view\_tren = RadioButton2.getText();  
 }  
 if(RadioButton3.isSelected()==true) {  
 view\_tren = RadioButton3.getText();  
 }  
 RadioButton1.setEnabled(false);  
 RadioButton2.setEnabled(false);  
 RadioButton3.setEnabled(false);  
 ButtonStart.setEnabled(false);  
  
 }  
 });  
  
  
  
 ButtonEnd.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent actionEvent) {  
  
 time\_stop=false;  
 labelTime.setText(minut + " : " + sec);  
 cal = calories.*view\_trening*(view\_tren);  
 spen\_cal = calories.*spent\_calories*(cal, sec\_all);  
 for (int i=0; i<stringArr\_form.length; i++){  
 if(stringArr\_form[i]==users\_label.getText()){  
 spen\_cal=Integer.*parseInt*(stringArr\_form[i+1])+spen\_cal;  
 stringArr\_form[i+1]=Integer.*toString*(spen\_cal);  
 }  
 }  
 LabelCal.setText(" "+spen\_cal+" ");  
 PrintWriter writer = null;  
 try {  
 writer = new PrintWriter(file, "UTF-8");  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 } catch (UnsupportedEncodingException e) {  
 e.printStackTrace();  
 }  
 for (int i=0; i<stringArr\_form.length; i++){  
 writer.println(stringArr\_form[i]);  
  
 }  
 writer.close();  
  
 sec = 0;  
 minut = 0;  
 sec\_all = 0;  
 spen\_cal = 0;  
 cal = 0;  
 ButtonStart.setEnabled(true);  
 RadioButton1.setEnabled(true);  
 RadioButton2.setEnabled(true);  
 RadioButton3.setEnabled(true);  
  
 }  
 });  
 ButtonPaus.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent actionEvent) {  
 time\_stop=false;  
 labelTime.setText(minut + " : " + sec);  
 ButtonStart.setEnabled(true);  
 }  
 });  
 button\_users.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent actionEvent) {  
  
 users\_label.setText(comboBox1.getSelectedItem().toString());  
  
  
 for (int i=0; i<stringArr\_form.length; i++){  
 if(stringArr\_form[i]==users\_label.getText()){  
 LabelCal.setText(stringArr\_form[i+1]);  
 }  
 }  
  
 System.*out*.println(comboBox1.getSelectedItem());  
 System.*out*.println();  
  
  
 }  
 });  
 }  
  
 int sec = 0;  
 int minut = 0;  
 int sec\_all = 0;  
 int cal = 0;  
 int spen\_cal = 0;  
 String view\_tren = "";  
 boolean time\_stop = false;  
  
 Runnable run\_timer = new Runnable() {  
 @Override  
 public void run() {  
 while (true){  
 try{  
 Thread.*sleep*(1000);  
 if (time\_stop == true) {  
 sec = sec + 1;  
 sec\_all = sec\_all + 1;  
 if (sec == 60) {  
 minut++;  
 sec = 0;  
 }  
 labelTime.setText(minut + " : " + sec);  
 }  
 }  
 catch (InterruptedException er){  
 er.printStackTrace();  
 }  
 }  
 }  
 };  
  
 public static void main(String[] args) throws IOException {  
  
 String[] stringArr;  
 stringArr= main.*open\_fail*();  
 form dialog = new form();  
 for(int i=0; i<stringArr.length;i++){  
 if(i==0 | i%2==0){  
 dialog.comboBox1.addItem(stringArr[i]);  
 }  
 }  
 dialog.stringArr\_form=stringArr;  
  
 dialog.users\_label.setText(stringArr[0]);  
 dialog.LabelCal.setText(stringArr[1]);  
 dialog.pack();  
 dialog.setVisible(true);  
 System.*exit*(0);  
  
 }  
  
  
}

1. Main.java

package timer\_cal;  
  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
import java.util.ArrayList;  
  
  
public class main {  
 public static String[] open\_fail() throws IOException {  
 BufferedReader reader = new BufferedReader(new FileReader("E:\\политех\\3 курс\\2 семестр\\Технологии разработки качественного программного обеспечения\\Технологии разработки качественного программного обеспечения Лабараторные\\37\\src\\timer\_cal\\save.txt"));  
 String str;  
  
 ArrayList<String> list = new ArrayList<String>();  
 while((str = reader.readLine()) != null ){  
 if(!str.isEmpty()){  
 list.add(str);  
 System.*out*.println(str);  
 }  
 }  
 String[] stringArr = list.toArray(new String[0]);  
 //System.out.println(stringArr[0]);  
 return stringArr;  
  
 }  
}

1. Calories.java

package timer\_cal;  
  
public class calories {  
  
 public static int view\_trening(String view\_tren){  
 int spen\_cal = 0;  
 switch (view\_tren){  
 case "Присидание":  
 spen\_cal = 150;  
 break;  
 case "Отжимание":  
 spen\_cal = 200;  
 break;  
 case "Скакалка":  
 spen\_cal = 250;  
 break;  
 }  
 return spen\_cal;  
 }  
  
  
 public static int spent\_calories(int cal, int t){  
 int spent\_cal = 0;  
 spent\_cal = cal\*t;  
 return spent\_cal;  
 }  
}