Министерство образования и науки Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

**технология разработки качественного программного обеспечения**

**Курсовой проект: фитнес-трекер**

|  |  |
| --- | --- |
| Выполнил  студент гр.в3530904/70321 | А. М. Ильинский |
| Преподаватель | Н.Г.Смирнов |

Санкт-Петербург

2020

Содержание

[Обзор предметной области 3](#_Toc39307584)

[**Что такое спортивный трекер?** 3](#_Toc39307585)

[**Для чего предназначены фитнес-трекеры?** 3](#_Toc39307586)

[**Как правильно использовать устройство или дизайн приложения** 4](#_Toc39307587)

[Аналоги приложения. 6](#_Toc39307588)

[Fitbit Charge HR 6](#_Toc39307589)

[Xiaomi Mi Band 2 6](#_Toc39307590)

[Misfit Shine 7](#_Toc39307591)

[Google Fit 7](#_Toc39307592)

[Apple Health 9](#_Toc39307593)

[**Описание архитектуры** 11](#_Toc39307594)

[**В общем плане** 11](#_Toc39307595)

[**Весь код приложения с комментариями** 12](#_Toc39307596)

[Тесты или улучшение приложения для удобства его использования 20](#_Toc39307597)

[**Персистенция данных** 23](#_Toc39307598)

[**Вывод** 29](#_Toc39307599)

[**Список литературы** 30](#_Toc39307600)

# Обзор предметной области

В последние годы выросла тенденция на здоровый образ жизни. Сбалансированное питание, водный баланс, физическая активность – вот в чем заинтересованы люди двадцать первого века.

Физические упражнения помогают оставаться здоровыми и держать себя в тонусе, а регулярные тренировки и вовсе способствуют красивому стройному и рельефному телу.

В последнее время становятся все более популярными фитнес-трекеры, которые помогают людям следить за их физической активностью. В свою очередь маркетологи и производители предлагают широкий выбор различных моделей трекеров – от самых простых до многофункциональных.

**Что такое спортивный трекер?**

Трекер представляет собой современное приложение, совмещающий в себе пульсометр, шагомер, подсчет калорий определенного вида упражнения и другие функции. Это многофункциональное приспособление, которое уже завоевало сердца десятков тысяч людей своей практичностью, универсальностью и привлекательным дизайном.

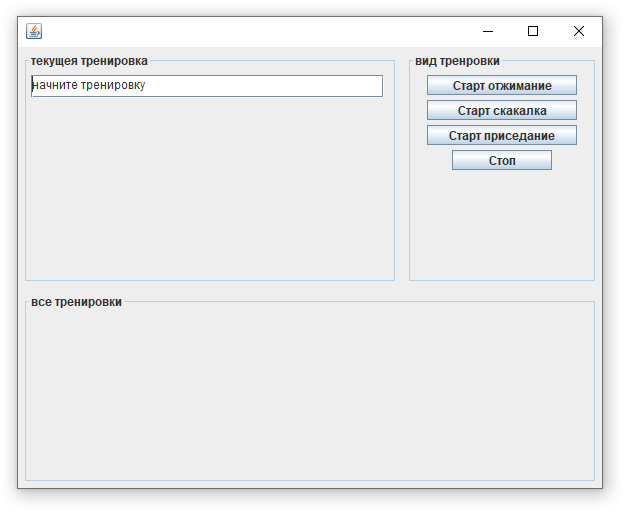
**Для чего предназначены фитнес-трекеры?**

Такие аксессуары имеют множество функций, начиная от шагомера и заканчивая фазами сна. Устройства помогают отслеживать активность пользователя и строить графики, благодаря чему можно с легкостью следить за своими успехами, оценивать их и делать выводы. Пользователи взаимодействуют с устройством через приложение для смартфонов. Имеющийся функционал делает фитнес-браслет очень практичной и удобной вещью.

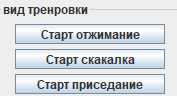
В нашем трекере заложен простенький функционал, а именно подсчет калорий одного из трех физических упражнений: отжимание скакалка, приседание. Можно выбрать одно из них и в процессе тренировки увидеть текущее время, потраченное на тренировку, и сколько затрачено калорий на эту тренировку. В процессе тренировки можно менять упражнения и видеть соответственно уже время и калории за это тренировку. После серии упражнений пользователь может остановить работу приложения, отдохнуть и начать тренировку заново (соответственно подсчет данных начнется заново), после окончание серий пользователь видит помимо текущей тренировки еще и все тренировки, общий затрат времени и общий затрат калорий за отдельное упражнение, а также общий затрат времени и калорий за все три упражнения. Подсчет калорий осуществляется следующим образом: у нас три вида упражнения, за каждое упражнение определенно количество затрата калорий в час (отжимание – 30, скакалка -100, приседание -200), во время тренировки течет секундомер (часы: минуты: секунды), секундомер насчитает считать время с секунд, поэтому калорий за час делим на 3600 ( секунд в часе) и умножаем на уже пройдённое время секундомера, то есть время/3600 \* калории определенного упражнения. Подсчет общего затрата трех упражнения вычисляется суммой трех упражнений (сумма времени трех секундомеров, и суммой калорий трех упражнений)

**Как правильно использовать устройство или дизайн приложения**

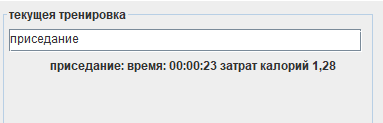
Запускаем приложение, видим три панели: текущее тренировка, вид тренировки и все тренировки



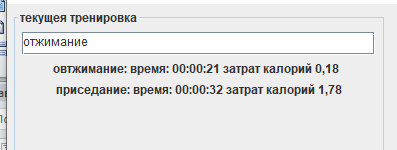
Чтобы начать тренировки выберем одно из трех упражнений, для этого есть кнопки справа



Когда мы выберем одно из упражнений, слева появится наш вид тренировки, подсчет времени и подсчет калорий



Поменяем вид тренировки, кнопкой справа

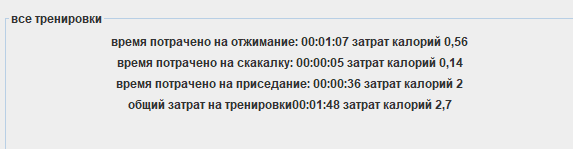


Так можно проделать много раз, остановим наш трекер кнопкой стоп



Можем немного отдохнуть и запустить тренировку заново аналогичным способом, время и калории в левом окне обнулятся.

После каждой остановки трекера в нижнем окне появляется общий затрат на времени и калорий на определённое упражнение, а также сумму всех тренировок: времени и калорий.



# Аналоги приложения.

На сегодняшний день есть куча аналогичных приложений, мы рассмотрим только некоторые из них. Эти приложения можно разделить на два вида: фитнес браслеты, фитнес трекеры – приложения для телефонов на android и на ios.

1. Фитнес браслеты- эти трекеры в виде наручных часов, вот так они выглядят:

## Fitbit Charge HR



Функционал:

счетчик калорий, анализ сна, его качества, шагомер, наличие виброзвонка

## Xiaomi Mi Band 2



Функционал:

измерение пульса, шагомер, счетчик калорий, анализ сна, будильник, оповещение о звонках и сообщениях.

## Misfit Shine



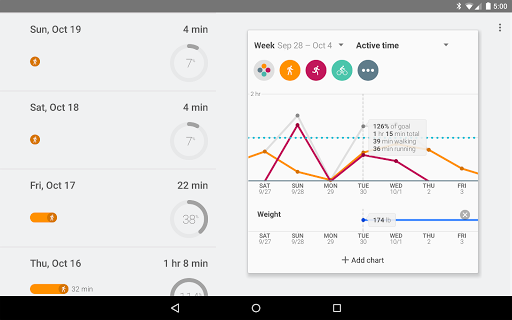
Функционал:

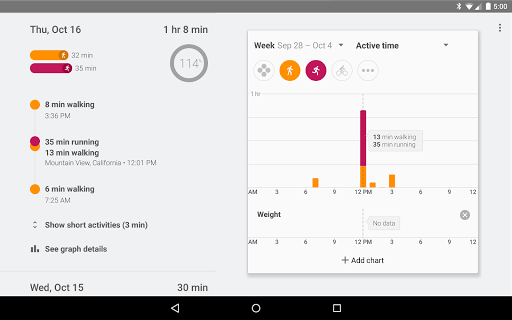
шагомер, счетчик калорий, анализ качества, продолжительности сна, подсчет дистанции.

1. Приложения для android и ios

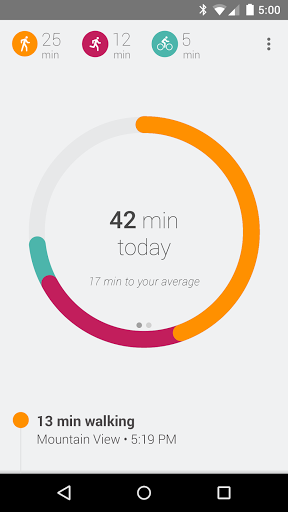
## Google Fit

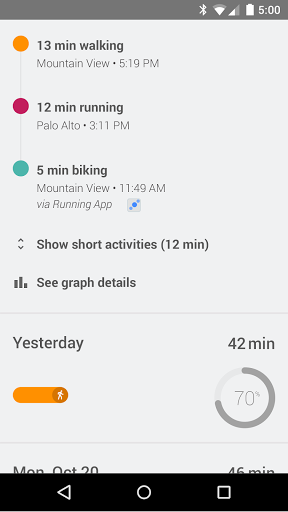
Это приложение для смартфона, которое может взаимодействовать с Android, но способно работать и без сторонних устройств. Вам достаточно один раз запустить программу и забыть про неё. Google Fit будет собирать статистику о ваших передвижениях, активности, пройденных шагах и затем все эти данные будут отображены в виде простых графиков.



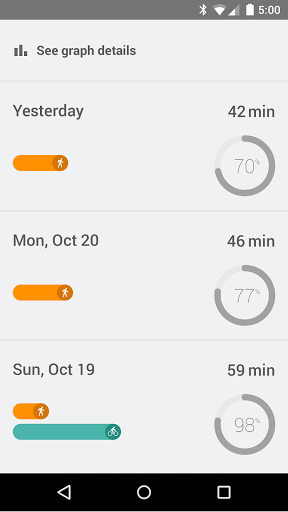


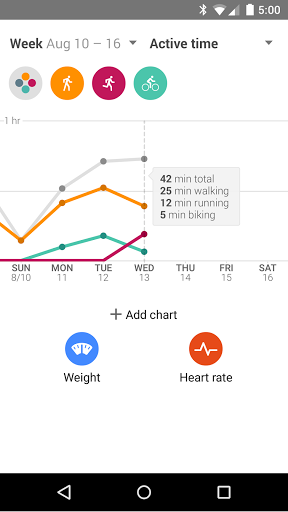
  Умеет эта программа определять, какой именно активностью вы занимаетесь, что очень важно, чтобы данные учитывались верно. Статистика в программе подробная, но выглядит она примитивно, так что разобраться будет несложно.





  В Google Fit могут собираться данные из других приложений, например, Nike+ или Runtastic. Таким образом, даже если вы пользуетесь другой программой для тренировок, то все результаты легко объединяются в Google Fit.





## Apple Health

Приложение Здоровье (Health) поставляется вместе с iOS. Так же, как и Google Fit, это не только приложение для измерения активности, но и целая система по сбору данных.



Apple Здоровье умеет определять активность. Там сенсор отслеживает количество пройденных вами шагов, дистанцию.

|  |  |
| --- | --- |
| https://www.ferra.ru/thumb/570x0/filters:quality(75):no_upscale()/imgs/2018/11/26/00/2485272/509be4f1a842d07d7fac71fa09ca6177adda7c40.jpg | https://www.ferra.ru/thumb/570x0/filters:quality(75):no_upscale()/imgs/2018/11/26/00/2485274/54a237ba01dfb083c20f27c3f894be51158d6d06.jpg |

  Программа Здоровье показывает статистику за день, неделю, месяц или год.

**Описание архитектуры**

**В общем плане**

За основу был взят класс import javax.swing.Timer;

Этот класс у нас служит секундомером по нажатие кнопки он начинает работать. Секундомер, после нажатие на кнопку стоп остановится.

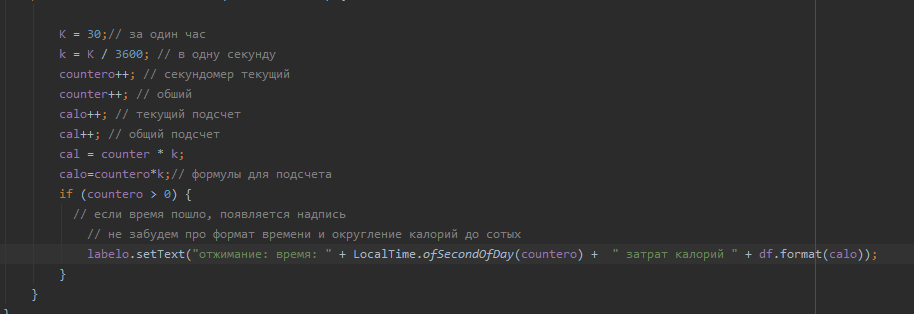
У нас три упражнения, значит и три режима секундомера ( три класса)

В каждом классе есть два общих одинаковых секундомера (для текущей тренировки и для всех тренировок) по умолчание время они измеряет в микросекундах, но мы можем задать значение 1000мс для подсчета времени в секундах.



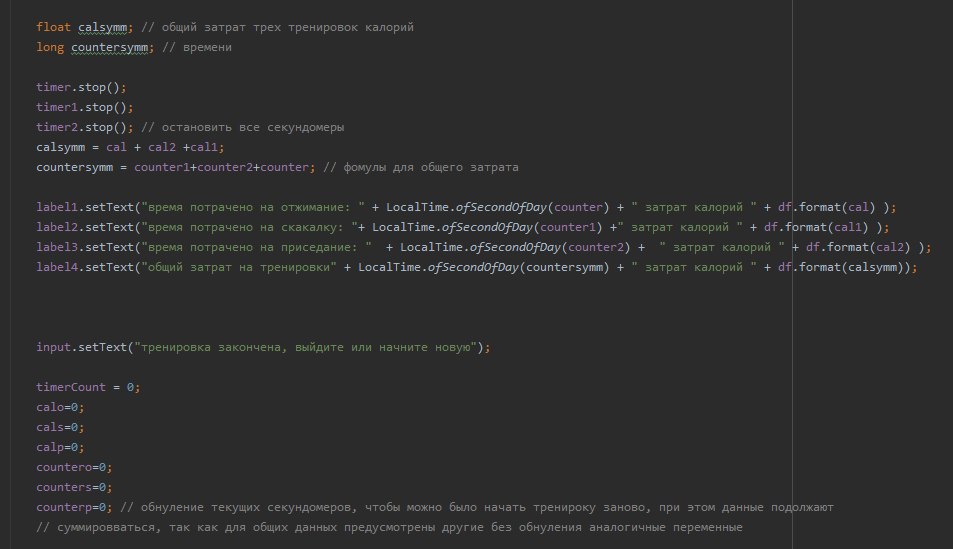
Подсчет калорий осуществляется с помощью того же самого секундомера.

Берем наш первый секундомер и умножаем его на количество калорий, таким образом, мы получаем такой же секундомер, только с другими единицами времени.



Запуск и стоп его осуществляется командами: timer.start, timer.stop.

По нажатие на кнопку стоп происходит подсчет секундомера времени и калорий за все тренировки, секундомеры для текущих тренировок приравняется к нулю, для обнуления.



**Весь код приложения с комментариями**

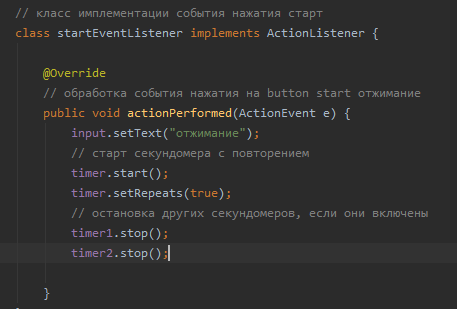
package com.company;  
  
import java.awt.BorderLayout;  
import java.awt.Container;  
import java.awt.Dimension;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.text.DecimalFormat;  
import java.time.LocalTime;  
import javax.swing.JButton;  
import javax.swing.JFrame;  
import javax.swing.JLabel;  
import javax.swing.JPanel;  
import javax.swing.JScrollPane;  
import javax.swing.JTextArea;  
import javax.swing.Timer;// у нас он будет секундомером  
import javax.swing.border.CompoundBorder;  
import javax.swing.border.EmptyBorder;  
import javax.swing.border.TitledBorder;  
  
  
class Main extends JFrame {  
  
 private int counter, countero, counter1, counters, counter2, counterp; // время на разные виды трененеровок  
 // счетчик срабатываний секундомера  
 private int timerCount;  
 // период срабатывания секундомером для всех упражнений  
 private int timerDel = 0;  
 private int timerDel1 = 0;  
 private int timerDel2 = 0;  
 private float K;//калорие в один час  
 private float cal, calo, cal1, cals, cal2, calp;//подсчет калорий  
 private float k; // в одну секунду  
//----------------------------------------------------------------------  
// создание объектов  
  
 private JTextArea input = new JTextArea("начните тренировку");  
 private JScrollPane scrollPaneInput = new JScrollPane(input);  
 private JPanel panel1 = new JPanel();  
 private JPanel panel2 = new JPanel();  
 private JPanel panelob = new JPanel();  
 private JButton start = new JButton("Старт отжимание");  
 private JButton start1 = new JButton("Старт скакалка");  
 private JButton start2 = new JButton("Старт приседание");  
 private JButton stop = new JButton("Стоп");  
 private JLabel labelo = new JLabel("");  
 private JLabel labels = new JLabel("");  
 private JLabel labelp = new JLabel("");  
 private JLabel label1 = new JLabel("");  
 private JLabel label2 = new JLabel("");  
 private JLabel label3 = new JLabel("");  
 private JLabel label4 = new JLabel("");  
 private DecimalFormat df = new DecimalFormat("#####.##"); // округление калорий до сотых  
  
 private Timer timer;  
 private Timer timer1;// секундомеры для трех кнопок  
 private Timer timer2;  
  
  
 // конструктор  
 public Main() {  
  
 // инициализация компонентов  
 initComponents();  
 // объект класса счетчика  
 TimeClass tc = new TimeClass(timerDel);  
 // объект секундомера  
 // период секундомера 1000мс = 1 сек  
 int timerStep = 1000;  
 timer = new Timer(timerStep, tc);  
  
 TimeClass1 tc1 = new TimeClass1(timerDel1);  
 // объект секундомера  
 timer1 = new Timer(timerStep, tc1);  
  
 TimeClass2 tc2 = new TimeClass2(timerDel2);  
 // объект секундомера  
  
 timer2 = new Timer(timerStep, tc2);  
 }  
  
 // метод инициализации компонентов формы  
 private void initComponents() {  
  
 // положение на экране  
 setBounds(15, 30, 500, 250);  
 // размер формы  
 setSize(600, 480);  
 // контейнер для размещения компонентов формы  
 Container container = getContentPane();  
  
 // окно вывода текстовой информации  
 input.setPreferredSize(new Dimension(350, 20));  
 input.setSize(20, 20);  
 // кнопка старт  
 start.setPreferredSize(new Dimension(150, 20));  
 start1.setPreferredSize(new Dimension(150, 20));  
 start2.setPreferredSize(new Dimension(150, 20));  
 // зарегистрировать экземпляр класса обработчика события start  
 start.addActionListener(new startEventListener());  
 start1.addActionListener(new startEventListener1());  
 start2.addActionListener(new startEventListener2());  
  
 // кнопка стоп  
 stop.setPreferredSize(new Dimension(100, 20));  
 // зарегистрировать экземпляр класса обработчика события stop  
 stop.addActionListener(new stopEventListener());  
  
 // панель 1  
 panel1.add(BorderLayout.*CENTER*, scrollPaneInput);  
  
 panel1.setBorder(new CompoundBorder(new EmptyBorder(5, 5, 5, 5), new TitledBorder("текущея тренировка")));  
 panel1.setPreferredSize(new Dimension(200, 200));  
 panel1.add(labelo);  
 panel1.add(labels);  
 panel1.add(labelp);  
  
  
  
 // панель 2  
 panel2.setBorder(new CompoundBorder(new EmptyBorder(5, 5, 5, 5), new TitledBorder("вид тренровки")));  
 panel2.setPreferredSize(new Dimension(200, 200));  
 panel2.add(start);  
 panel2.add(start1);  
 panel2.add(start2);  
 panel2.add(stop);  
  
 // панель 3  
  
 panelob.setBorder(new CompoundBorder(new EmptyBorder(5, 5, 5, 5), new TitledBorder("все тренировки")));  
 panelob.setPreferredSize(new Dimension(200, 200));  
 panelob.add(label1);  
 panelob.add(label2);  
 panelob.add(label3);  
 panelob.add(label4);  
  
 // размещение панелей  
 container.add(BorderLayout.*CENTER*, panel1);  
 container.add(BorderLayout.*EAST*, panel2);  
 container.add(BorderLayout.*SOUTH*, panelob);  
  
 }  
//----------------------------------------------------------------------  
  
 // класс имплементации события нажатия старт  
 class startEventListener implements ActionListener {  
  
 @Override  
 // обработка события нажатия на button start отжимание   
 public void actionPerformed(ActionEvent e) {  
 input.setText("отжимание");  
 // старт секундомера с повторением  
 timer.start();  
 timer.setRepeats(true);  
 // остановка других секундомеров, если они включены  
 timer1.stop();  
 timer2.stop();  
  
 }  
 }  
 class startEventListener1 implements ActionListener {  
  
 @Override  
 // обработка события нажатия на button start скакалка  
 public void actionPerformed(ActionEvent e) {  
 input.setText("скакалка");  
 // старт секундомера с повторением  
 timer1.start();  
 timer1.setRepeats(true);  
 // остановка других секундомеров, если они включены  
 timer.stop();  
 timer2.stop();  
  
 }  
 }  
  
 class startEventListener2 implements ActionListener {  
  
 @Override  
 // обработка события нажатия на button start приседание  
 public void actionPerformed(ActionEvent e) {  
 input.setText("приседание");  
 // старт секундомера с повторением  
 timer2.start();  
 timer2.setRepeats(true);  
 // остановка других секундомеров, если они включены  
 timer1.stop();  
 timer.stop();  
  
 }  
 }  
  
 // класс перезагружаемого счетчика секундомера  
 public class TimeClass implements ActionListener {  
  
 // конструктор  
 public TimeClass(int count) {  
 counter = count;  
 }  
  
 // время пошло, отжимание  
 @Override  
 public void actionPerformed(ActionEvent ts) {  
  
 K = 30;// за один час  
 k = K / 3600; // в одну секунду  
 countero++; // секундомер текущий  
 counter++; // обший  
 calo++; // текущий подсчет  
 cal++; // общий подсчет  
 cal = counter \* k;  
 calo=countero\*k;// формулы для подсчета  
 if (countero > 0) {  
 // если время пошло, появляется надпись  
 // не забудем про формат времени и округление калорий до сотых  
 labelo.setText("отжимание: время: " + LocalTime.*ofSecondOfDay*(countero) + " затрат калорий " + df.format(calo));  
 }  
 }  
 }  
//----------------------------------------------------------------------  
// остальные два класса построенны аналогичным образом  
 public class TimeClass1 implements ActionListener {  
 // время пошло , скакалка  
 public TimeClass1(int count) {  
 counter1 = count;  
 }  
 @Override  
 public void actionPerformed(ActionEvent ts1) {  
  
 K = 100;  
 k = K / 3600;  
 counters++;  
 counter1++;  
 cal1++;  
 cals++;  
 cal1 = counter1 \* k;  
 cals=counters\*k;  
 if (counters > 0) {  
 labels.setText("скакалка: время: " + LocalTime.*ofSecondOfDay*(counters) + " затрат калорий " + df.format(cals));  
 }  
 }  
  
 }  
  
 // время пошло, приседание  
 public class TimeClass2 implements ActionListener {  
  
 // конструктор  
 public TimeClass2(int count) {  
 counter2 = count;  
 }  
  
 // время пошло приседание  
 @Override  
 public void actionPerformed(ActionEvent ts) {  
  
 K = 200;  
 k = K / 3600;  
 counter2++;  
 counterp++;  
 cal2++;  
 calp++;  
 cal2 = counter2 \* k;  
 calp = counterp \* k;  
 if (counterp >= -1) {  
  
 labelp.setText("приседание: время: "+ LocalTime.*ofSecondOfDay*(counterp) + " затрат калорий " + df.format(calp));  
 }  
 }  
  
 }  
  
 //

класс имплементации события нажатия стоп  
 class stopEventListener implements ActionListener {  
  
 @Override  
 // обработка события нажатия на button start  
 public void actionPerformed(ActionEvent e) {  
  
 float calsymm; // общий затрат трех тренировок калорий  
 long countersymm; // времени  
  
 timer.stop();  
 timer1.stop();  
 timer2.stop(); // остановить все секундомеры  
 calsymm = cal + cal2 +cal1;  
 countersymm = counter1+counter2+counter; // формулы для общего затрата  
  
 label1.setText("время потрачено на отжимание: " + LocalTime.*ofSecondOfDay*(counter) + " затрат калорий " + df.format(cal) );  
 label2.setText("время потрачено на скакалку: "+ LocalTime.*ofSecondOfDay*(counter1) +" затрат калорий " + df.format(cal1) );  
 label3.setText("время потрачено на приседание: " + LocalTime.*ofSecondOfDay*(counter2) + " затрат калорий " + df.format(cal2) );  
 label4.setText("общий затрат на тренировки" + LocalTime.*ofSecondOfDay*(countersymm) + " затрат калорий " + df.format(calsymm));  
  
  
  
 input.setText("тренировка закончена, выйдите или начните новую");  
  
 timerCount = 0;  
 calo=0;  
 cals=0;  
 calp=0;  
 countero=0;  
 counters=0;  
 counterp=0; // обнуление текущих секундомеров, чтобы можно было начать тренировку заново, при этом данные продолжают  
 // суммироваться, так как для общих данных предусмотрены другие без обнуления аналогичные переменные  
  
 }  
 }  
 public static void main(String[] args) {  
 // объект графической формы  
 Main graphic = new Main();  
 // закрытие формы  
 graphic.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 // визуализация формы  
 graphic.setVisible(true);  
 }  
}

# Тесты или улучшение приложения для удобства его использования

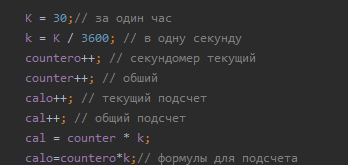
Чтобы привести приложения его к итоговом удобному финальному варианту были проделаны тесты

1. Первая версия приложения была без остановок тренировок, при переходе на следующею, остановка всех тренировок секундомеров осуществлялось только кнопкой стоп. Это не очень удобно, так как пользователь может захотеть сделать серию упражнений без остановок. Было решено добавить команды остановки секундомеров. После каждого нажатие вида тренировки

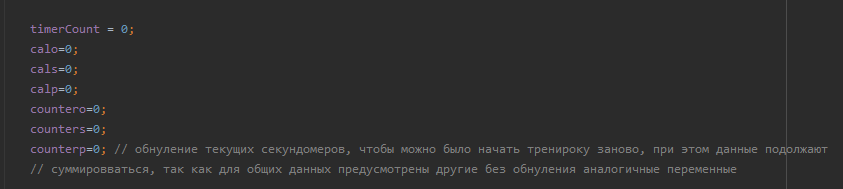


1. После нажатие на кнопку стоп были показаны общие затраты и при нажатии на старт пользователь не мог видеть текущие затраты, так как данные не обнулялись. Было решено сделать следующее оставить общие затраты и добавить переменные и засунуть аналогичный функционал под формулы для этих переменных, но при этом обнулять эти переменные после каждого нажатие кнопок стоп

Функционал



Обнуление

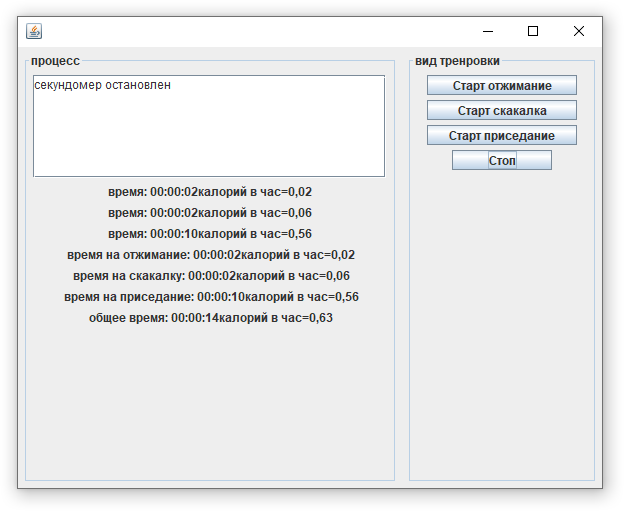


1. Данные времени были в секундах (20, 60, 120, 1238 секунд) и калории показывались с большим количеством знаков после запятой(45,56356721 калорий). Это неудобно для обычного пользователя. Было решено добавить первичную форму для времени командой LocalTime.ofSecondOfDay(…). Для калорий была добавлена новая переменная, в которой форма записи числа

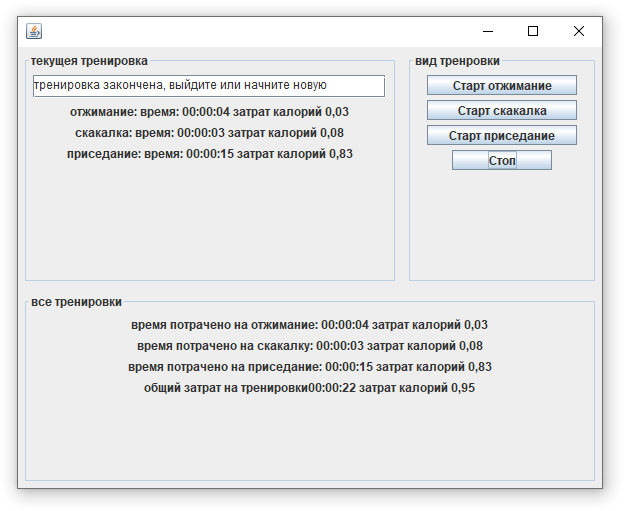
private DecimalFormat df = new DecimalFormat("#####.##"); // округление калорий до сотых

а команда для ее осуществления следующая

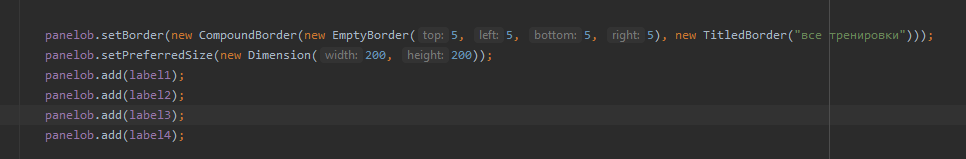
df.format(…)

1. После всех преведущих обновлений получился не очень удобный дизайн приложения, что было 

В одной панели было очень много надписей и при просмотре данных глаза разбегались было решено добавить еще одну панель для общих затрат на тренировки, что стало



Код для добавления панели



**Персистенция данных**

В нашем фитнес-трекере процесс идет только во время работы приложения, и по закрытию все данные обнуляются. В этом большой минус приложения, так общие данные пользователь может смотреть только во время работы приложения, после закрытия они удалятся, и когда он снова откроет приложение, вновь увидит нулевые результаты. Чтобы этого не происходило, создадим персистенцию данных.

Персистенция данных - это процесс сохранения данных в файл, если было произведено сохранение, при закрытие приложения никуда данные не пропадут, и когда вновь откроется приложение, пользователь, к примеру, по нажатию кнопки может вновь увидеть свои данные.

**Алгоритм для осуществления кода персистенции данных**

Итак, нам нужно создать файл и записать туда данные, а по нажатию кнопки прочесть эти данные и вывести в приложение и не забыть для обновления уже новых данных сделать их запись в файл.

Воспроизведем алгоритм:

1. Предположим у нас есть переменная (с) в процессе каких-либо действий она равняется, к примеру, 30 (с=30).
2. Если нет файла его нужно создать, туда засунем переменную равную нулю (с2=0). Выведем нашу с=30. При закрытие приложения вновь с=0, даные потеряны. Поэтому во время процесса работы приложения сохраним данные. Для с2 = с (теперь с2=30) и запишем с2 в файл.
3. Закрыли приложение, вновь его открыли с=0, после процесса работы к примеру с=100, но пользователь должен увидеть данные за все сеансы работы приложения.
4. Поэтому делаем следующие: иначе если уже файл создан, читаем из него данные, переменную с2=30. Сложим с2 с новым результатом (с) получим с2 =130 (те самые данные за все сеансы работы приложения).
5. Для того чтобы сохранить обновленные данные вновь запишем с2 в файл.
6. Таким образом, при всех следующих сеансах работы приложения, будет проделываться вновь и вновь 4 и 5 пункт.

**Код**

Сделаем персистенцию данных при помощи ObjectOutputStream.

Нам понадобится пакет import java.io.\*;

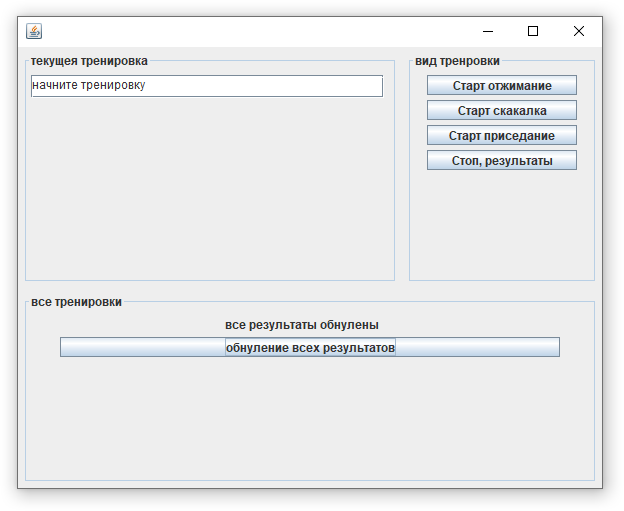
Воспроизведем код:

int counteroper = 0;// и так сделаем переменные для персистенции  
 int countersper = 0;  
 int counterpper = 0;  
 float caloper = 0;  
 float calsper = 0;  
 float calpper = 0;  
  
 if (!(new File(fileName)).exists()) {  
 File file = new File("file.txt");  
 try {  
 file.createNewFile();//если нет, создать файл  
 } catch (IOException ioException) {  
 ioException.printStackTrace();  
 }  
  
 FileOutputStream fout = null;  
 try {  
 fout = new FileOutputStream("file.txt");  
 } catch (FileNotFoundException fileNotFoundException) {  
 fileNotFoundException.printStackTrace();  
 }  
 ObjectOutputStream oot = null;  
 try {  
 oot = new ObjectOutputStream(fout);  
 oot.writeInt(counteroper);//запись всех переменных  
 oot.writeInt(countersper);  
 oot.writeInt(counterpper);  
 oot.writeFloat(caloper);  
 oot.writeFloat(calsper);  
 oot.writeFloat(calpper);  
  
 calsymm= calo+calp+cals;// реализуем сумму  
 countersymm=countero+counterp+counters;  
  
 label1.setText("время потрачено на отжимание: " + LocalTime.*ofSecondOfDay*(countero) + " затрат калорий " + df.format(calo) );  
  
 label2.setText("время потрачено на скакалку: "+ LocalTime.*ofSecondOfDay*(counters) +" затрат калорий " + df.format(cals) );  
  
 label3.setText("время потрачено на приседание: " + LocalTime.*ofSecondOfDay*(counterp) + " затрат калорий " + df.format(calp) );  
  
 label4.setText( "всего потраченного времени " + LocalTime.*ofSecondOfDay*(countersymm) + " потрачено калорий "  
 + df.format(calsymm) );// выведем резульаты первой тренировки  
 counteroper=countero;  
 countersper=counters;  
 counterpper=counterp;  
 caloper=calo;  
 calsper=cals;  
 calpper=calp;//прираняем наши переменнные для запси в файл после if else  
  
 // закрываем поток  
 oot.close();  
 } catch (IOException ioException) {  
 ioException.printStackTrace();  
 }  
 }else {  
  
 FileInputStream fin = null;  
 try {  
 fin = new FileInputStream("file.txt");  
 } catch (FileNotFoundException fileNotFoundException) {  
 fileNotFoundException.printStackTrace();  
 }  
 ObjectInputStream oit = null;  
 try {  
 oit = new ObjectInputStream(fin);  
 counteroper = oit.readInt();// иначе вытащем переменные для персистенции  
 countersper = oit.readInt();  
 counterpper = oit.readInt();  
 caloper = oit.readFloat();  
 calsper = oit.readFloat();  
 calpper = oit.readFloat();  
  
 counteroper = counteroper + countero;// сложем их с дааными за одну серию  
 countersper = countersper + counters;  
 counterpper = counterpper + counterp;  
 caloper = caloper + calo;  
 calsper = calsper + cals;  
 calpper = calpper + calp;  
  
 countersymm=counteroper+countersper+counterpper;  
 calsymm=caloper+calsper+calpper;//незабываем про сумму  
  
 label1.setText("время потрачено на отжимание: " + LocalTime.*ofSecondOfDay*(counteroper) + " затрат калорий " + df.format(caloper));  
 label2.setText("время потрачено на скакалку: " + LocalTime.*ofSecondOfDay*(countersper) + " затрат калорий " + df.format(calsper));  
 label3.setText("время потрачено на приседание: " + LocalTime.*ofSecondOfDay*(counterpper) + " затрат калорий " + df.format(calpper));  
 label4.setText("всего потраченного времени " + LocalTime.*ofSecondOfDay*(countersymm) + " потрачено калорий "  
 + df.format(calsymm) );//вывод результатов  
  
 oit.close();  
 } catch (IOException ioException) {  
 ioException.printStackTrace();  
 }  
 }//выд=йдем из if else и совершим запись, так мы запишем нужные данные вне зависимости от того  
 // какое из двух действий было совершенно программой  
  
 FileOutputStream fout = null;  
 try {  
 fout = new FileOutputStream("file.txt");  
 } catch (FileNotFoundException fileNotFoundException) {  
 fileNotFoundException.printStackTrace();  
 }  
 ObjectOutputStream oot = null;  
 try {  
 oot = new ObjectOutputStream(fout);  
 oot.writeInt(counteroper);  
 oot.writeInt(countersper);  
 oot.writeInt(counterpper);  
 oot.writeFloat(caloper);  
 oot.writeFloat(calsper);  
 oot.writeFloat(calpper);  
  
  
 // закрываем поток  
 oot.close();  
 } catch (IOException ioException) {  
 ioException.printStackTrace();  
 }  
//  
  
  
 input.setText("тренировка закончена, выйдите или начните новую");  
  
 // timerCount = 0;  
 calo=0;  
 cals=0;  
 calp=0;  
 countero=0;  
 counters=0;  
 counterp=0; // обнуление секундомеров за сессию  
  
 }  
 }

Персистенция была реализована по кнопке стоп, заметим, что нам стали не нужны секундомеры для общих значений, так как сохранение происходит при каждой кнопке стоп, и при следующем нажатие на эту кнопку читаем данные из файла суммируем с данными за процесс и вновь все сохраняем.

Добавим также кнопку обнуление данных, при нажатии на нее наш файл с данными будет удаляться и данные снова будут нулевыми, таким образом, пользователю не придется лазить в папку приложения и искать файл, чтобы удалить все свои результаты.

Обновленный дизайн приложения



Код реализации кнопки обнуления

class nullrEventListener implements ActionListener{  
 @Override  
 // обработка события нажатия на кнопку обнуление  
 public void actionPerformed(ActionEvent tsn) {  
  
 if ((new File(fileName)).exists()) {  
 File file = new File("file.txt");  
 file.delete();//если есть удалить  
 }  
 label1.setText("все результаты обнулены");  
 label2.setText("");  
 label3.setText("");  
 label4.setText("");  
 }  
  
}

**Вывод**

В процессе построения архитектуры были изучены следующие java покеты:

import java.awt.BorderLayout;// для размещения местоположения панелей  
import java.awt.Container;// контейнер для размещения компонентов формы

import java.awt.Dimension;// размер панелей  
import java.awt.event.ActionEvent;// для включение секундомера и его функционала  
import java.awt.event.ActionListener;// отработка нажатие кнопок  
import java.text.DecimalFormat;//округление чисел(калорий)  
import java.time.LocalTime;// для формата времени  
import javax.swing.JButton;// для размещение кнопок  
import javax.swing.JFrame;// для появление окна  
import javax.swing.JLabel;// для размещение текста (время, калории)  
import javax.swing.JPanel;// для размещений панелей  
import javax.swing.JScrollPane;// для размещения окна с информацией, о состояние приложения в нужной панелей  
import javax.swing.JTextArea;// для размещения окна с информацией, о состояние приложения  
import javax.swing.Timer;// таймер у на это секундомер  
import javax.swing.border.CompoundBorder;// для размещения рамки вокруг панели  
import javax.swing.border.EmptyBorder;// расположение названия панели  
import javax.swing.border.TitledBorder;//надписи панели

Была построена архитектура и приведены тесты, описаны ранее.

В итоге на выходе пользователь получает фитнес-трекер, приобретя это приложение, может регулярно выполнять тренировки, которые в приложение предложены, может не следить за часами, за стрелками, а смотреть на секундомер, и узнавать, сколько на каждое упражнение было затрачено, также может смотреть, сколько ему удалось сбросить калорий за определенную тренировку и за все тренировки.

**Список литературы**

<https://www.ferra.ru/review/apps/7-prilozheniy-kotorye-zamenyayut-fitnes-treker.htm>

https://zen.yandex.ru/media/id/5bf716cfe3445200a9d2f161/chto-takoe-fitnesbraslet-dlia-zaniatiia-sportom-5c0ae1d055af0e00aaf21369