Министерство образования и науки Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого

—

Институт компьютерных наук и технологий

**Кафедра «Программная инженерия»**

**Отчёт по зачётной работе**

по дисциплине «Технологии разработки

качественного программного обеспечения»

Выполнила

Студент группы в3530904/90322

Аверкина Е.К.

Преподаватель

Смирнов Н. Г.

Санкт-Петербург, 2022

Содержание

[Цель работы 3](#_Toc101436651)

[Описание архитектуры 4](#_Toc101436652)

[Ход работы приложения 6](#_Toc101436653)

[Выводы 9](#_Toc101436654)

[ПРИЛОЖЕНИЕ 10](#_Toc101436655)

[Код основного класса Main 10](#_Toc101436656)

[Код класса SaveFile 13](#_Toc101436657)

[Код класса MyTimer 15](#_Toc101436658)

[Код класса PersonList 16](#_Toc101436659)

[Код класса Person 17](#_Toc101436660)

[Код класса MainTest 18](#_Toc101436661)

# Цель работы

Необходимо разработать приложение, которое будет представлять из себя фитнес-трекер.

Приложение должно указывать 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.

Также необходимо реализованные сохранение данных о калориях нескольких пользователей.

# Описание архитектуры

Приложение состоит из следующих классов:

* класс Main – основной класс, из которого запускается программа;
* класс PersonList – класс для работы со списком пользователей системы;
* класс Person – класс для работы с профилем системы;
* класс MyTimer – класс для работы с секундомером;
* класс SaveFile – класс для сохранения и чтения файлов.

Соответствующая диаграмма классов приведена на рисунке 1.

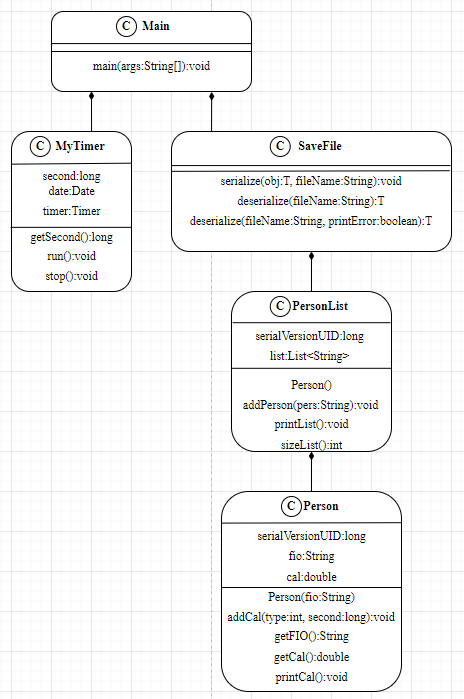


Рисунок 1. Диаграмма классов

Описание последовательности взаимодействий представлено на рисунке 2.

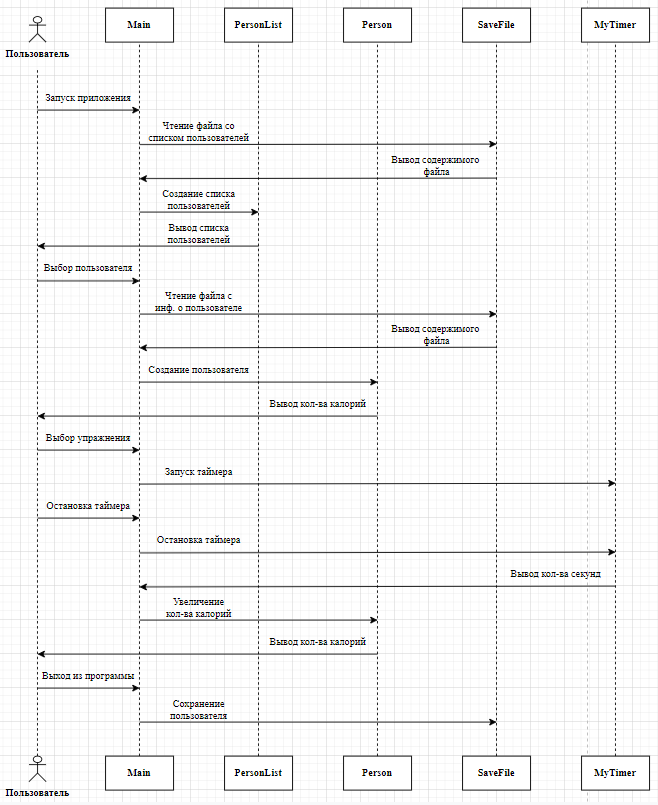


Рисунок 2. Диаграмма последовательности

# Ход работы приложения

При запуске приложения пользователь выбирает профиль и создаёт новый. Происходит проверка на корректный ввод пункта меню.

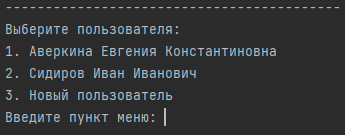


Рисунок 3. Выбор пользователя

Для добавления нового пользователя необходимо ввести ФИО.

После создания/выбора пользователя выводятся количество сожжённым им калорий и меню выбора упражнения.

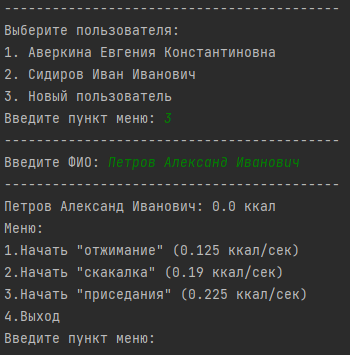


Рисунок 4. Создание пользователя

После выбора упражнения запускается секундомер.

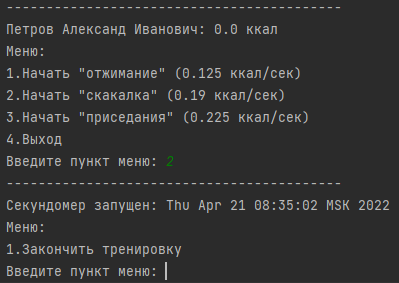


Рисунок 5. Запуск секундомера

Для окончания упражнения необходимо ввести «1». После остановки секундомера происходит расчёт калорий.

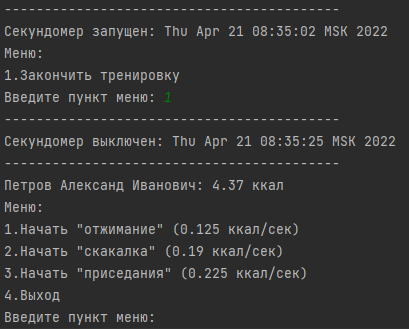


Рисунок 6. Вывод калорий

Для выхода из приложения необходимо ввести «4». После этого данные о пользователи будут сохранены и приложение завершиться.

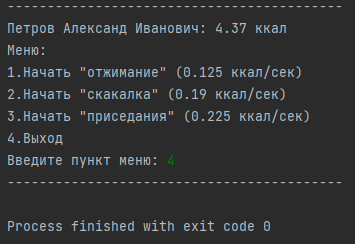


Рисунок 7. Выход из приложения

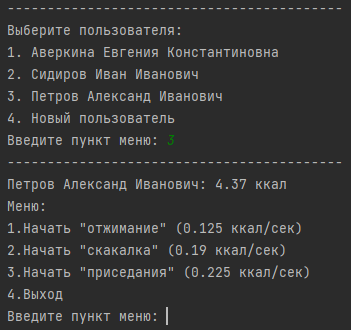


Рисунок 8. Вывод сохранённых данных

# Оценка покрытия кода модульными тестами

Были разработаны 3 теста:

* «testCal» - тестирование расчета калорий;
* «testSaveFile» - тестирование запись и чтения файлов;
* «testTimer» - тестирование работы таймера.

Созданные тесты покрывают 70% кода.

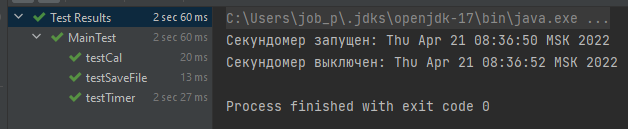


Рисунок 9. Результат работы тестов

# Выводы

В результате выполнения работы было реализовано приложение согласно поставленной задачи. Были написаны unit тесты, покрывающие в общей сложности примерно 70% кода. Для приложения были использованы библиотеки:

* FileOutputStream – предназначен для записи байтов в файл.
* ObjectOutputStream – это подкласс класса OutputStream, который управляет объектом OutputStream и предоставляет методы для записи примитивных данных (primitive data) или объектов в OutputStream, которым он управляет.
* ObjectInputStream – используется, чтобы восстановить те объекты, ранее сериализированные. Другое использование включает объекты передачи между узлами, используя поток сокета или для маршалинга и неупорядочивая параметры и параметры в удаленной системе связи.
* TimerTask – является абстрактным классом, который реализует интерфейс Runnable. Мы должны унаследоваться от этого класса, создавая наш собственный TimerTask. Класс Timer является потокобезопасным, поэтому несколько потоков могут совместно использовать один объект Timer без необходимости внешней синхронизации.

# ПРИЛОЖЕНИЕ

## Код основного класса Main

package com.averkina;

import java.io.\*;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

SaveFile sf = new SaveFile();

Scanner in = new Scanner(System.in);

MyTimer timer = new MyTimer();

int menu = 0;

int submenu = 0;

Person pers = new Person("");

PersonList persList = (PersonList) sf.deserialize("archive/list.data", false);

if(persList == null){

persList = new PersonList();

}

int numberUser = 0;

while (numberUser == 0) {

System.out.println("------------------------------------------");

System.out.println("Выберите пользователя:");

persList.printList();

System.out.println((persList.sizeList() + 1) + ". Новый пользователь");

System.out.print("Введите пункт меню: ");

try {

numberUser = in.nextInt();

if (numberUser < 0 || numberUser > (persList.sizeList() + 1)) {

throw new Exception();

}

in.nextLine();

} catch (Exception ex) {

System.out.printf("Ошибка! Некорректный ввод пункта меню!\n");

numberUser = 0;

in.nextLine();

}

}

if(numberUser == (persList.sizeList() + 1)) {

String fio = "";

System.out.println("------------------------------------------");

try {

System.out.print("Введите ФИО: ");

fio = in.nextLine();

} catch (Exception ex) {

System.out.printf("Ошибка! Некорректный ввод!\n");

numberUser = 0;

in.nextLine();

}

pers = new Person(fio);

persList.addPerson(fio);

sf.serialize(pers, "archive/" + numberUser + ".data");

sf.serialize(persList, "archive/list.data");

}

pers = (Person) sf.deserialize("archive/" + numberUser + ".data");

while (menu < 4) {

menu = 0;

submenu = 0;

System.out.println("------------------------------------------");

pers.printСal();

System.out.print("Меню:" +

"\n1.Начать \"отжимание\" (0.125 ккал/сек)" +

"\n2.Начать \"скакалка\" (0.19 ккал/сек)" +

"\n3.Начать \"приседания\" (0.225 ккал/сек)" +

"\n4.Выход" +

"\nВведите пункт меню: ");

try {

menu = in.nextInt();

if (menu < 1 || menu > 4) {

throw new Exception();

}

in.nextLine();

} catch (Exception ex) {

System.out.printf("Ошибка! Некорректный ввод пункта меню!\n");

menu = 0;

in.nextLine();

}

System.out.println("------------------------------------------");

if(menu >= 1 && menu <= 3) {

timer.run();

while(submenu != 1) {

System.out.print("Меню:" +

"\n1.Закончить тренировку" +

"\nВведите пункт меню: ");

try {

submenu = in.nextInt();

if (submenu != 1) {

throw new Exception();

}

in.nextLine();

} catch (Exception ex) {

System.out.printf("Ошибка! Некорректный ввод пункта меню!\n");

submenu = 0;

in.nextLine();

}

System.out.println("------------------------------------------");

}

timer.stop();

pers.addCal(menu, timer.getSecond());

}

}

sf.serialize(pers, "archive/" + numberUser + ".data");

}

}

## Код класса SaveFile

package com.averkina;

import java.io.\*;

import java.util.Iterator;

public class SaveFile<T> implements Serializable {

public void serialize(T obj, String fileName) {

try {

FileOutputStream fos = new FileOutputStream(fileName, false);

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(obj);

oos.close();

fos.close();

} catch (IOException e) {

e.printStackTrace();

}

}

public T deserialize(String fileName) {

T obj = null;

try {

FileInputStream fis = new FileInputStream(fileName);

ObjectInputStream ois = new ObjectInputStream(fis);

obj = (T) ois.readObject();

ois.close();

fis.close();

} catch (IOException i) {

i.printStackTrace();

return null;

} catch (ClassNotFoundException c) {

System.out.println("Class not found");

c.printStackTrace();

return null;

}

return obj;

}

public T deserialize(String fileName, boolean printError) {

T obj = null;

try {

FileInputStream fis = new FileInputStream(fileName);

ObjectInputStream ois = new ObjectInputStream(fis);

obj = (T) ois.readObject();

ois.close();

fis.close();

} catch (IOException i) {

if (printError) {

i.printStackTrace();

}

return null;

} catch (ClassNotFoundException c) {

if (printError) {

System.out.println("Class not found");

c.printStackTrace();

}

return null;

}

return obj;

}

}

## Код класса MyTimer

package com.averkina;

import java.util.Date;

import java.util.Timer;

import java.util.TimerTask;

public class MyTimer{

private long second;

private Date date;

private Timer timer;

public long getSecond() {

return second;

}

public void run() {

timer = new Timer();

System.out.println("Секундомер запущен: " + new Date());

TimerTask task = new TimerTask() {

int second = 1;

public void run() {

second++;

}

};

date = new Date();

timer.schedule(task, date, 1000);

}

public void stop() {

if(timer != null) {

timer.cancel();

Date date\_temp = new Date();

second = (date\_temp.getTime() - date.getTime()) / 1000 % 60;

System.out.println("Секундомер выключен: " + new Date());

}

else {

System.out.println("Секундомер уже выключен!");

}

}

}

## Код класса PersonList

package com.averkina;

import java.io.Serializable;

import java.util.\*;

public class PersonList implements Serializable {

private static final long serialVersionUID = 813248781814176996L;

List<String> list;

public PersonList() {

list = new ArrayList<>();

}

public void addPerson(String pers) {

list.add((sizeList() + 1) + ". " + pers);

}

public void printList() {

list.stream().forEach(System.out::println);

}

public int sizeList() {

return list.size();

}

}

## Код класса Person

package com.averkina;

import java.io.Serializable;

public class Person implements Serializable {

private static final long serialVersionUID = -3427837268287956311L;

private String fio;

private double cal;

public Person(String fio) {

this.fio = fio;

cal = 0;

}

public void addCal(int type, long second) {

switch (type) {

case 1 -> cal += (0.125 \* second);

case 2 -> cal += (0.19 \* second);

case 3 -> cal += (0.225 \* second);

};

}

public String getFIO() {

return fio;

}

public double getCal() {

return cal;

}

public void printСal() {

System.out.println(fio + ": " + cal + " ккал");

}

}

## Код класса MainTest

package com.averkina;

import org.junit.jupiter.api.DisplayName;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

class MainTest {

@Test

@DisplayName("testSaveFile")

void testSaveFile() {

assertAll(

() -> assertEquals("cat", MainTest.testSaveFile("cat")),

() -> assertEquals("", MainTest.testSaveFile(""))

);

}

@Test

@DisplayName("testTimer")

void testTimer() {

assertAll(() -> assertEquals(2, MainTest.testTimer(2)));

}

@Test

@DisplayName("testCal")

void testCal() {

assertAll(

() -> assertEquals(0.25, MainTest.testCal(1,2)),

() -> assertEquals(0.95, MainTest.testCal(2,5)),

() -> assertEquals(0.675, MainTest.testCal(3,3))

);

}

private static String testSaveFile(String name) {

SaveFile sf = new SaveFile();

Person pers = new Person(name);

sf.serialize(pers, "archive/test.data");

return ((Person) sf.deserialize("archive/test.data")).getFIO();

}

private static long testTimer(long second) {

MyTimer timer = new MyTimer();

timer.run();

try {

Thread.sleep(second \* 1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

timer.stop();

return timer.getSecond();

}

private static double testCal(int type, long second) {

Person pr = new Person("Test");

pr.addCal(type, second);

return pr.getCal();

}

}