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»

Выполнил

Студент группы в3530904/90322

Зидерер А.О.

Преподаватель

Смирнов Н. Г.

Санкт-Петербург, 2022

Оглавление

[ЦЕЛЬ РАБОТЫ 3](#_Toc99813215)

[ОПИСАНИЕ АРХИТЕКТУРЫ 4](#_Toc99813216)

[ХОД РАБОТЫ ПРИЛОЖЕНИЯ 6](#_Toc99813217)

[ВЫВОДЫ 7](#_Toc99813218)

[ПРИЛОЖЕНИЕ 8](#_Toc99813219)

# ЦЕЛЬ РАБОТЫ

Разработать приложение, которое будет представлять из себя фитнес-трекер.

Приложение должно позволять указать вид тренировки. В нём должны быть доступны команды: начать тренировку и закончить тренировку. 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.

# ОПИСАНИЕ АРХИТЕКТУРЫ

Приложение состоит из основного класса Main, из которого запускается программа, класс Person – который является профилем пользователя и класс Training – который отвечает за тренировки, записываемые в профиль. Для тренировок был создан интерфейс и нескольких классов, которые наследуются от интерфейса. Соответствующая диаграмма классов приведена на рисунке 1.

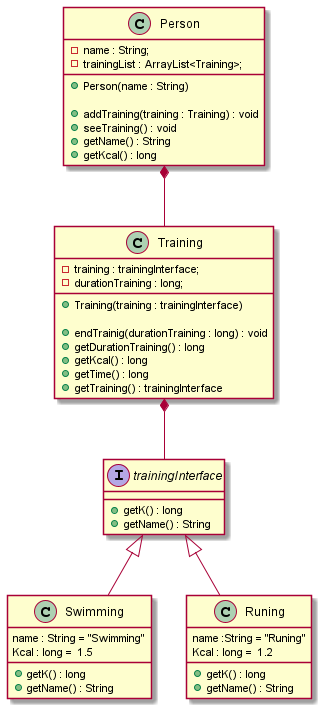
Описание последовательности взаимодействий представлено на рисунке 2.

Рисунок 1. Диаграмма классов.

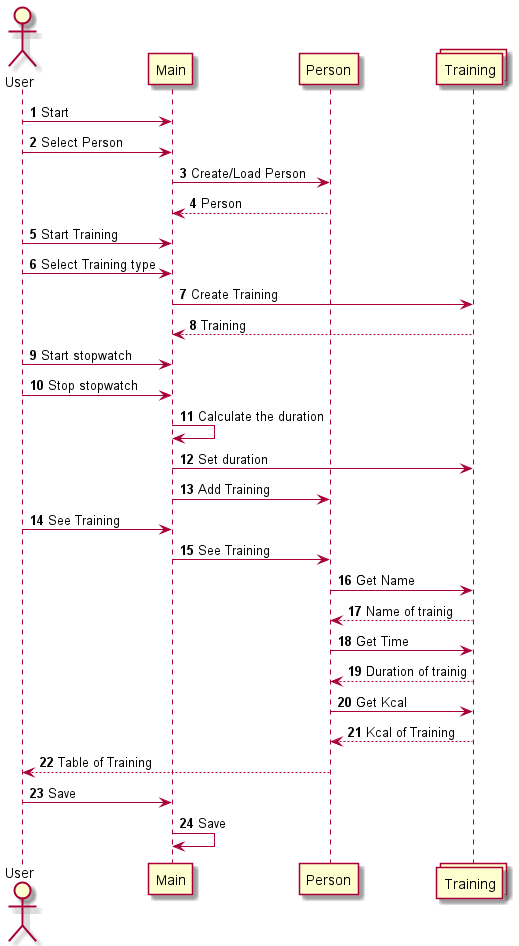


Рисунок 2. Диаграмма последовательности

# ХОД РАБОТЫ ПРИЛОЖЕНИЯ

При запуске приложения у вас попросят либо загрузить профиль, либо создать новый. Стартовый экран изображён на рисунке 3. Меню загрузки изображено на рисунке 4.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 4. Стартовое меню

Рисунок 3. Меню загрузки

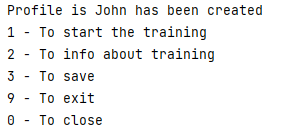
После создание/загрузки профиля появиться меню, изображённое на рисунке 5. Из него можно сохранить результаты, получить информацию о тренировках или стартовать тренировку.

Рисунок 5. Основное меню

Изображение выглядит как текст

Автоматически созданное описаниеМеню выбора тренировки изображено на рисунке 6.

Рисунок 6. Меню выбора тренировок

Изображение выглядит как текст

Автоматически созданное описаниеВыводимая информация о тренировках изображена на рисунке 7.

Рисунок 7. Информация о тренировках

# ВЫВОДЫ

В результате выполнения работы было реализовано приложение согласно поставленной задачи. Были написаны unit тесты, покрывающие в общей сложности примерно 40% кода. Дополнительно при реализации использовался секундомер из пакета Guava.

# ПРИЛОЖЕНИЕ

Код основного класса Main

**package** fitness;  
  
**import** com.google.common.base.Stopwatch;  
**import** fitness.TrainingList.\*;  
**import** java.io.\*;  
**import** java.util.Objects;  
**import** java.util.Scanner;  
**import** java.util.concurrent.TimeUnit;  
  
**public class** Main {  
  
 **static** String *SAVEPATH* = **"src/main/java/fitness/save/"**;  
 **public static void** main(String[] args) **throws** InterruptedException {  
 *//Сделать класс трекира* Scanner in = **new** Scanner(System.***in***);  
 **int** i = 0;  
 Person person = **null**;  
 Training training = **null**;  
 Stopwatch stopwatch;  
  
 **while** (**true**) {  
 **while** ((i != 1 && i != 2)) {  
 System.***out***.println(**"1 - To create new person\n2 - To load person from file\n0 - To close"**);  
 i = in.nextInt();  
 **switch** (i) {  
 **case** 1:  
 System.***out***.println(**"Write profile name:"**);  
 String name = in.next();  
 **if** (*profileNameIsFree*(name)) {  
 person = **new** Person(name);  
 System.***out***.println(**"Profile is "** + person.getName() + **" has been created"**);  
 } **else** {  
 System.***out***.println(**"That profile name's taken"**);  
 i = 0;  
 }  
 **break**;  
 **case** 2:  
 **if** (*canLoad*()) {  
 System.***out***.println(**"Select profile:"**);  
 person = *load*();  
 } **else** {  
 System.***out***.println(**"Save file not found"**);  
 }  
 **break**;  
 **case** 0:  
 **return**;  
 }  
 }  
 **while** (i != 9 && person != **null**) {  
 System.***out***.println(**"1 - To start the training\n2 - To info about training\n3 - To save\n9 - To exit\n0 - To close "**);  
 i = in.nextInt();  
 **switch** (i) {  
 **case** 0:  
 **return**;  
 **case** 9:  
 person = **null**;  
 **break**;  
 **case** 1:  
 System.***out***.println(**"Select the trainig"**);  
 System.***out***.println(**"1 - Runing\n2 - Swimming\n3 - Situps\n4 - SkippingRope\n - Pushups"**);  
 i = 0;  
 **while** (i != 1 && i != 2 && i != 3 && i != 4 && i != 5) { *//можно использовать enum* i = in.nextInt();  
 **switch** (i) {  
 **case** 1 -> training = **new** Training(**new** Runing());  
 **case** 2 -> training = **new** Training(**new** Swimming());  
 **case** 3 -> training = **new** Training(**new** Situps());  
 **case** 4 -> training = **new** Training(**new** SkippingRope());  
 **case** 5 -> training = **new** Training(**new** Pushups());  
 **default** -> System.***out***.println(**"Writhe the right number"**);  
 }  
 }  
 System.***out***.println(**"Press Enter to start the training"**);  
 in.next();  
 stopwatch = Stopwatch.*createStarted*();  
 System.***out***.println(**"Press Enter to end the training"**);  
 in.next();  
 stopwatch.stop();  
 training.endTrainig(stopwatch.elapsed(TimeUnit.***SECONDS***));  
 person.addTraining(training);  
 **break**;  
 **case** 2:  
 System.***out***.println(**"Select the info"**);  
 System.***out***.println(**"1 - Training info\n2 - kCal info"**);  
 i = 0;  
 **while** (i != 1 && i != 2){  
 i = in.nextInt();  
 **switch** (i) {  
 **case** 1 -> person.seeTraining();  
 **case** 2 -> System.***out***.println(**"Total kCal for all time: "** + person.getKcal());  
 **default** -> System.***out***.println(**"Writhe the right number"**);  
 }  
 }  
 **break**;  
 **case** 3:  
 *save*(person);  
 System.***out***.println(**"Writhe the right number"**);  
 **break**;  
 }  
 }  
 }  
 }  
  
 **static** Person load() {  
 Person p = **null**;  
 File folder = **new** File(*SAVEPATH*);  
 Scanner in = **new** Scanner(System.***in***);  
 **int** position = -1;  
 **for** (**int** i = 0; i< Objects.*requireNonNull*(folder.listFiles()).**length**; i++) *//что это?* {  
 System.***out***.println((i+1) + **" - "** + Objects.*requireNonNull*(folder.listFiles())[i].getName());  
 }  
 **while** (position<0||position>folder.listFiles().**length**) {  
 System.***out***.println(**"Write to number"**);  
 position = in.nextInt();  
 }  
 **if**(position == 0){  
 **return null**;  
 }  
 **try** (ObjectInputStream ois = **new** ObjectInputStream(**new** FileInputStream(Objects.*requireNonNull*(folder.listFiles())[position-1].toString()))) {  
 p = (Person) ois.readObject();  
 } **catch** (Exception ex) {  
 System.***out***.println(ex.getMessage());  
 }  
 System.***out***.println(**"Profile "**+p.getName()+**" uploaded"**);  
 **return** p;  
 }  
  
 **static void** save(Person person) {  
 **try**(ObjectOutputStream oos = **new** ObjectOutputStream(**new** FileOutputStream(*SAVEPATH*+person.getName()))){  
 oos.writeObject(person);  
 }  
 **catch**(Exception ex){  
 System.***out***.println(ex.getMessage());  
 }  
 }  
  
 **static boolean** profileNameIsFree(String name){  
 File folder = **new** File(*SAVEPATH*);  
 **for** (**int** i = 0; i< Objects.*requireNonNull*(folder.listFiles()).**length**; i++){  
 **if**(Objects.*requireNonNull*(folder.listFiles())[i].getName().equals(name)){  
 **return false**;  
 }  
 }  
 **return true**;  
 }  
  
 **static boolean** canLoad(){  
 File folder = **new** File(*SAVEPATH*);  
 **return** Objects.*requireNonNull*(folder.listFiles()).**length** > 0;  
 }  
}

Код класса Person

**package** fitness;  
**import** java.io.Serializable;  
**import** java.util.ArrayList;  
  
**public class** Person **implements** Serializable {  
 **private final** String **name**;  
 **private final** ArrayList<Training> **trainingList**;  
  
 **public** Person(String name){  
 **this**.**name** = name;  
 **this**.**trainingList** = **new** ArrayList<>();  
 }  
  
 **public void** addTraining(Training training){**this**.**trainingList**.add(training);}  
  
 **public void** seeTraining(){  
 System.***out***.printf(**"%-20s%-10s%-10s%n"**, **"Training name"** , **"Duration"** , **"kCal"**);  
 **for** (Training training:**this**.**trainingList**) {  
 System.***out***.printf(**"%-20s%-10s%-10s%n"**,training.getTraining().getName(),training.getTime(),training.getKcal());  
 }  
 }  
  
 **public** ArrayList<Training> getTrainingList(){**return this**.**trainingList**;}  
  
 **public long** getKcal(){  
 **long** kcalSum = 0;  
  
 **for** (Training training:**this**.**trainingList**) {  
 kcalSum += training.getTraining().getK()\*training.getDurationTraining();  
 }  
 **return** kcalSum;  
 }  
  
 **public** String getName() {**return name**;}  
}

Код класса Training

**package** fitness;  
  
**import** java.io.Serializable;  
  
**public class** Training **implements** Serializable {  
  
 **private final** trainingInterface **training**;  
 **private long durationTraining**;  
  
 **public** Training(trainingInterface training) {**this**.**training** = training;}  
  
 **public void** endTrainig(**long** durationTraining){**this**.**durationTraining** = durationTraining;}  
  
 **public long** getDurationTraining() {**return durationTraining**;}  
  
 **public long** getKcal(){**return training**.getK()\***durationTraining**;}  
  
 **public long** getTime(){**return durationTraining**;}  
  
 **public** trainingInterface getTraining() {**return training**;}  
}

Код интерфейса trainingInterface

**package** fitness;  
  
**public interface** trainingInterface {  
 **long** getK();  
 String getName();  
}

Код класса Pushups

**package** fitness.TrainingList;  
**import** fitness.trainingInterface;  
**import** java.io.Serializable;  
  
**public class** Pushups **implements** Serializable, trainingInterface {  
 **private static final** String ***name*** = **"Pushups"**;  
 **private static final long *Kcal*** = 1L;  
  
 @Override  
 **public long** getK() {  
 **return *Kcal***;  
 }  
 **public** String getName() {**return *name***;}  
}

Код класса Runing

**package** fitness.TrainingList;  
**import** fitness.trainingInterface;  
**import** java.io.Serializable;  
  
**public class** Runing **implements** trainingInterface, Serializable {  
 **private static final** String ***name*** = **"Runing"**;  
 **private static final long *Kcal*** = 2L;  
  
 @Override  
 **public long** getK() {  
 **return *Kcal***;  
 }  
 **public** String getName() {**return *name***;}  
}

Код класса Situps

**package** fitness.TrainingList;  
**import** fitness.trainingInterface;  
**import** java.io.Serializable;  
  
**public class** Situps **implements** trainingInterface, Serializable {  
 **private static final** String ***name*** = **"Situps"**;  
 **private static final long *Kcal*** = 1L;  
  
 @Override  
 **public long** getK() {  
 **return *Kcal***;  
 }  
 **public** String getName() {**return *name***;}  
}

Код класса SkippingRope

**package** fitness.TrainingList;  
**import** fitness.trainingInterface;  
**import** java.io.Serializable;  
  
**public class** SkippingRope **implements** trainingInterface, Serializable {  
 **private static final** String ***name*** = **"SkippingRope"**;  
 **private static final long *Kcal*** = 1L;  
  
 @Override  
 **public long** getK() {  
 **return *Kcal***;  
 }  
 **public** String getName() {**return *name***;}  
}

Код класса Situps

**package** fitness.TrainingList;  
**import** fitness.trainingInterface;  
**import** java.io.Serializable;  
  
**public class** Swimming **implements** trainingInterface, Serializable {  
 **private static final** String ***name*** = **"Swimming"**;  
 **private static final long *Kcal*** = 3L;  
  
 @Override  
 **public long** getK() {  
 **return *Kcal***;  
 }  
 **public** String getName() {**return *name***;}  
}