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

группы в3530904/70321 Земцов А.О.

Преподаватель Смирнов Н.Г.

Санкт-Петербург

2020

Оглавление

[Обзор предметной области 3](#_Toc40115557)

[Техническое задание. 4](#_Toc40115558)

[Основной функционал. 8](#_Toc40115559)

[Краткая инструкция пользователя 9](#_Toc40115560)

[Описание структуры 12](#_Toc40115561)

[Диаграммы. 27](#_Toc40115562)

## Обзор предметной области

Что такое фитнес трекер?

В современном мире для занятий спортом придумано уже множество приложений. Некоторые из них действительно полезны, а какие-то могут оказаться довольно посредственным помощником во время тренировки. Мы рассмотрим, одно из самых популярных сейчас – фитнес трекер.

Оно может быть установлено на большинство устройств, обладающих хотя бы минимальным графическим интерфейсом. (Наибольшим спросом пользуются фитнес-трекеры на смартфонах и специально созданных под них спортивных-браслетах). Такие приложения могут следить за состоянием нашего здоровья и вести множество интересных подсчетов, исходя из наших тренировок.

Рассмотрим, что же умеет фитнес трекер.

— Начнем с самого элементарного – это фиксация пройденного расстояния, [подсчет сожженных калорий](https://teleprogramma.pro/hb/fitness/22626/), а также показатели вашего пульса.

.

— Некоторые спортивные браслеты могут работать и в воде. Фитнес трекер будет считать расстояние, которое вы проплыли, а также может показать даже общее количество гребков, которое вы совершили.

— Фитнес трекер позволит вам детально следить за своим [питанием](https://teleprogramma.pro/lenta/18730/). Вы можете заполнять информацию о еде, которую вы употребляли в течение дня, и следить за общим количеством калорий.

— Полезно для офисных работников. Сидя весь рабочий день за компьютером наша двигательная активность сводится к минимуму. С помощью спортивного браслета вы можете задать длительность, которую вы считаете допустимой для такого сидячего состояния, а фитнес трекер будет напоминать вам в течение этих промежутков, что пора и подвигаться.

— Если вы любите соревновательные моменты, то большинство фитнес трекеров предлагают функцию создания внутренней социальной сети. Это даст вам возможность сравнивать свои результаты с друзьями и коллегами, которые также пользуются подобными приложениями

. — И множество других полезных функций для поддержания здорового образа жизни.

### Техническое задание.

Техническим заданием предусмотрено: Создание приложения. Позволяющего указать вид тренировки: отжимание, скакалка, приседания. С доступными командами: 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.

Персистенция данных приложения с помощью ObjectOutputStream +5 баллов или JAXB + 10 баллов.  
Исходя из этого. Будем опираться на создание консольного приложения, предусматривающего самые основные функции программ. Которые уже пришлись по вкусу миллионам пользователей.

Таких как (представление основных аналогов):

*Xiaomi Mi Band 3 - Браслет для фитнеса*

Именно китайский бренд Xiaomi первым запустил свою серию фитнес трекеров Mi Band. Недавно на рынке появилась уже третья модель браслета отличающаяся своим непревзойденным стилем и функционалом. Так, трекер обладает монохромным OLED-дисплеем 128 x 80 пикселей благодаря которому осуществляется управление. Шагомер, датчик сердцебиения, поиск телефона и мониторинг сна — отнюдь не полный список сенсоров установленных в гаджете.

*Samsung Gear Fit 2 Pro L - Фитнес-браслет с GPS*



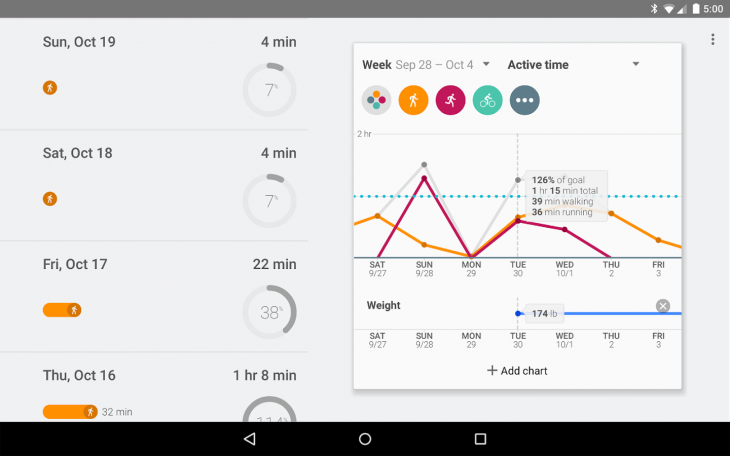
Устройство от известного корейского производителя Samsung. Данный гаджет является гармоничным сочетанием стильного дизайна и широкого функционала. Так, устройство имеет инновационный изогнутый сенсорный экран Super AMOLED, которым чрезвычайно удобно пользоваться даже в солнечную погоду или в воде. В отличие от упомянутых конкурентов, в гаджете имеется встроенный GPS-навигатор, благодаря которому можно точно отследить пройденное расстояние и свое местоположения при отсутствии смартфона.

*Google Fit*



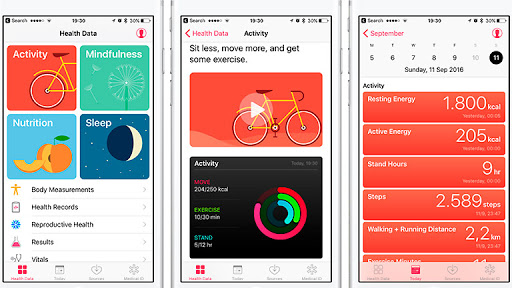
Google Fit — это не просто приложение, а целая платформа, облачный сервис, ведущий наблюдение за активностью человека и некоторыми характеристиками его организма. Приложение совместимо с рядом планшетов и смартфонов, но лучше всего работать с Google Fit при помощи [фитнес-трекеров, умных часов](http://medgadgets.ru/shop/katalog/braslety.html) и других специализированных устройств.

Приложение может использоваться для отслеживания активности человека при занятиях различными видами спорта. При этом Google Fit автоматически определяет бег, ходьбу, езду на велосипеде.



Во время тренировки приложение ведет учет пройденного расстояния, количества сделанных шагов, сожженных калорий и другую информацию. Дополнительные возможности можно получить, подключив специализированные фитнес-гаджеты (официально заявлено о поддержке устройств от Nike, HTC, LG, RunKeeper, Withings, Motorola, Noom, Runtastic и Polar).  
  
У пользователя есть возможность добавлять цели (например, пройти не менее 2 километров в день), после чего приложение будет следить за выполнением плана. Для того, чтобы результаты можно было оценить, пользователю предоставляется графическая визуализация полученной Google Fit информации.

*Apple Health*



Apple Health («Здоровье») — это аггрегатор данных о состоянии организма в iPhone. Приложение собирает данные от различных фитнес-приложений, установленных на смартфоне, а также с Apple Watch, анализирует их и оформляет всю информацию в удобном виде, чтобы дать комплексное представление о состоянии здоровья.

Apple Health: категории медданных

В приложении «Здоровье» есть четыре различных категории данных. Они позволяют быстро получить представление о питании, ритме сна, физических нагрузках и других важных вещах.

* Активность: в этой категории собираются данные о физических нагрузках и занятиях спортом. Здесь вы узнаете, сколько шагов сделали за день, как долго сидели неподвижно и сколько калорий потратили.
* Сон: для хорошего сна очень важен отрегулированный биоритм. В этой категории можно найти информацию о том, сколько длится ваш сон, и во сколько вы ложитесь спать.
* Осознанность: в этой категории вы найдете данные о ритме вашего дыхания. Кроме того, вы сможете получить представление о том, насколько регулярно в течение дня делали перерывы, которые помогают избежать развития стресса.

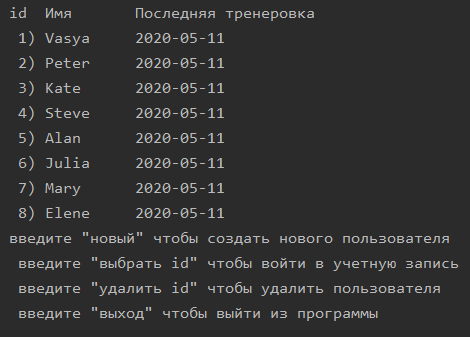
Кроме того, в приложении есть и дополнительные категории: питание, сердце, измерения тела, репродуктивное здоровье, анализы, основные показатели.

## Основной функционал.

В трекере заложен простенький функционал, а именно подсчет калорий одного из трех физических упражнений: отжимание скакалка, приседание. Можно выбрать одно из них и по окончанию тренировки увидеть затраченное время, и количество калорий затраченных на эту тренировку. После выполнения упражнения пользователь может остановить работу приложения, отдохнуть и начать заново. После завершения тренировки пользователь может зайти в собственный профиль и посмотреть количество затраченных за сегодня колорий. А так же результаты работы за дни до этого. Для составления верной динамики, и приведения тренировок к наиболее оптимальному варианту. Подсчет калорий осуществляется следующим образом: у нас три вида упражнения, за каждое упражнение определенно количество затрата калорий в час.

## Краткая инструкция пользователя

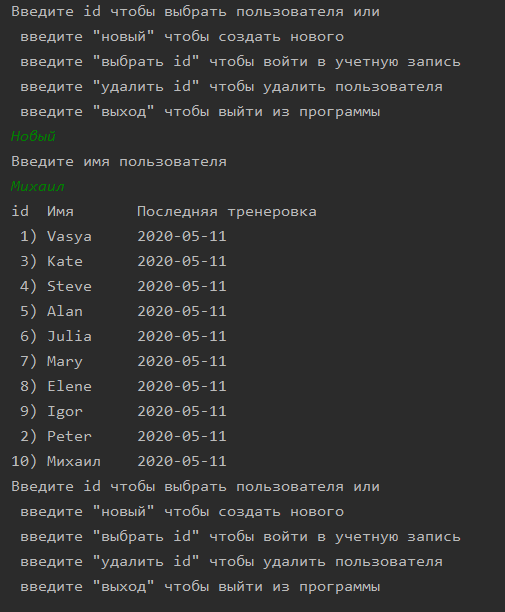
Шаг 1.  
Запустить приложение. В открывшемся меню. Видим список пользователей и основные команды.



Перечень основных команд:

**«Новый»** Предназначена для регистрации новых пользователей. Введите в консоли “новый” и нажмите ввод. После чего введите имя нового пользователя. И нажмите клавишу ввода. Пользователю будет присвоен уникальный IP. И он будет внесён в список пользователей.

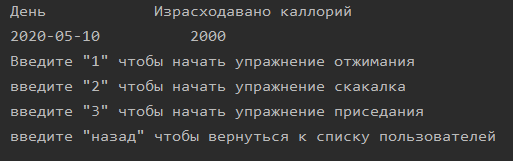
Пример:



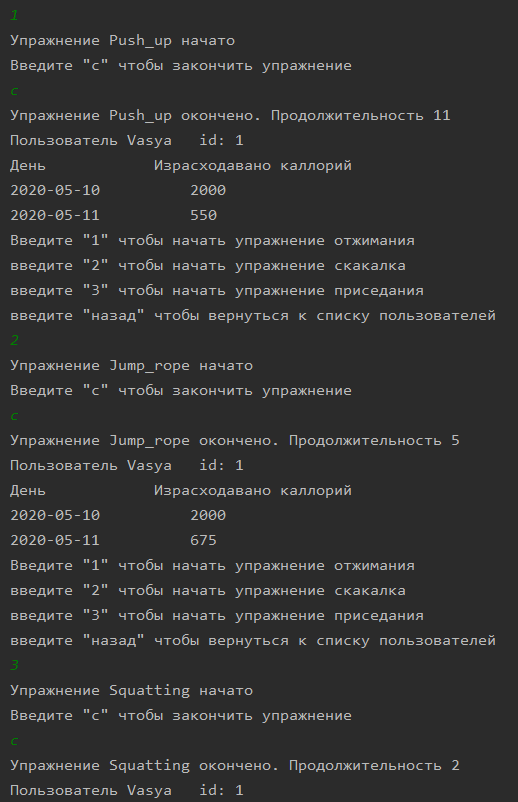
**«Выбрать id»** Для того, чтобы начать тренировку пользователю необходимо всего лишь набрать: «выбрать», добавить уникальный номер собственной учётной записи и нажать клавишу ввода.   
Пример для пользователя Василий:



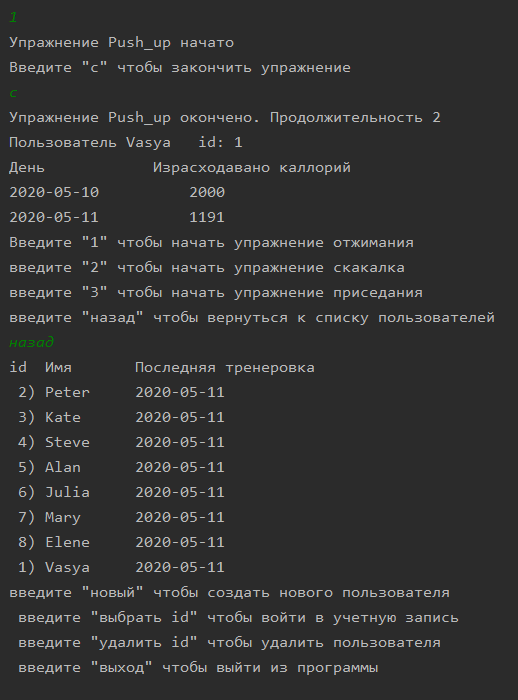
После этого откроется меню тренировки.



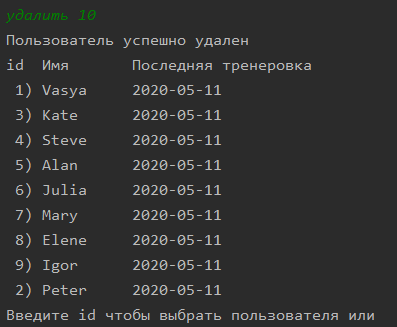
Как мы видим. Здесь указаны дата прошедших занятий. И общая сумма калорий, затраченных пользователем в этот день.  
А также список команд для выполнения того или иного упражнения.  
Введя «1» и нажав кнопку ввода - запускается таймер. Он отсчитывает время, затраченное на выполнения отжиманий. После того, как упражнение выполнено. Наберите «с» и нажмите клавишу ввода. Вы увидите сообщение о завершении выполнения упражнения и общую сумму затраченных калорий за сегодняшний день.  
Аналогично. Для выполнения упражнений со скакалкой наберите «2» и нажмите клавишу ввода. Для выполнения приседаний наберите «3» и нажмите клавишу ввода.   
Упражнения могут быть выполняться в произвольном порядке.  
Пример:



По завершении тренировки наберите «назад» и нажмите клавишу ввода. Тем самым вы вернётесь в основное меню.



**«удалить id»** Предназначено для удаления пользователей из общего списка. Введите “удалить”, затем номер пользователя, которого хотите удалить. И нажмите кнопку ввода.   
Пример:



**«Выход»** Для завершения работы. Приложения введите «Выход» и нажмите клавишу ввода.

## Описание структуры

Класс Day.   
Хранит информацию о определённом дне. Обо всех тренировках и сожжённых за день калориях.

package Entities;  
import sample.LocalDateAdapter;  
import javax.xml.bind.annotation.XmlElement;  
import javax.xml.bind.annotation.XmlElementWrapper;  
import javax.xml.bind.annotation.XmlRootElement;  
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;  
import java.io.Serializable;  
import java.time.LocalDate;  
import java.util.ArrayList;  
import java.util.Date;  
  
@XmlRootElement()  
public class Day {  
 private LocalDate date;  
 private ArrayList<Exercise> exercises;  
 private int total\_Calories;  
  
 public Day(LocalDate date) {  
 this.date = date;  
 exercises = new ArrayList<Exercise>();  
 }  
  
 public Day(LocalDate date, int total\_Calories) {  
 this.date = date;  
 this.total\_Calories = total\_Calories;  
 exercises = new ArrayList<Exercise>();  
 }  
  
 @XmlJavaTypeAdapter(value = LocalDateAdapter.class)  
 public LocalDate getDate() {  
 return date;  
 }  
  
  
 public ArrayList<Exercise> getExercises() {  
 return exercises;  
 }  
  
 @XmlElement  
 public void setDate(LocalDate date) {  
 this.date = date;  
 }  
  
 public Day() {  
 }  
  
 @XmlElementWrapper  
 public void setExercises(ArrayList<Exercise> exercises) {  
 this.exercises = exercises;  
 }  
  
 @XmlElement  
 public void setTotal\_Calories(int total\_Calories) {  
 this.total\_Calories = total\_Calories;  
 }  
  
 public int getTotal\_Calories() {  
 return total\_Calories;  
 }  
  
  
 public void addExercise(Exercise exercise) {  
 exercises.add(exercise);  
 total\_Calories = 0;  
 for (Exercise ex : exercises)  
 total\_Calories += ex.getExercise\_consumption();  
 }  
}

Класс Exercise. Хранит и обрабатывает информацию о выбранном упражнении.

package Entities;  
  
import javax.xml.bind.annotation.XmlElement;  
import javax.xml.bind.annotation.XmlRootElement;  
import java.io.Serializable;  
import java.time.Duration;  
import java.time.LocalDate;  
import java.util.Date;  
  
  
@XmlRootElement()  
public class Exercise{  
 private Exercise\_Type type;  
 private long duration;  
 private int exercise\_consumption;  
  
 public Exercise(Exercise\_Type type) {  
 this.type = type;  
 }  
  
 public Exercise\_Type getType() {  
 return type;  
 }  
  
  
 public int getExercise\_consumption() {  
 return exercise\_consumption;  
 }  
  
  
 public long getDuration() {  
 return duration;  
 }  
  
  
 public Exercise() {  
 }  
  
  
 public void setType(Exercise\_Type type) {  
 this.type = type;  
 }  
  
  
  
 public void setDuration(long duration) {  
 this.duration = duration;  
 if(type != null)  
 this.exercise\_consumption = (int)(type.getConsumption()/3600 \* duration);  
 }  
  
  
 public void setExercise\_consumption(int exercise\_consumption) {  
 this.exercise\_consumption = exercise\_consumption;  
 }  
}

Фиксированный список Exercise\_Type. Все классы объектов, хранящие информацию.

package Entities;  
  
import java.io.Serializable;  
  
public enum Exercise\_Type implements Serializable {  
 *Push\_up*("отжимания", 3000\*60), // 100\*30 за одно \* 30 в минуту  
 *Jump\_rope*("скакалка", 1500\*60), // 100 прыжков за 1 минуту  
 *Squatting*("приседания", 5000\*60); // минуту  
  
 private final String name;  
 private final int consumption;  
  
 Exercise\_Type(String name, int consumption) {  
 this.name = name;  
 this.consumption = consumption;  
 }  
  
 public int getConsumption() {  
 return consumption;  
 }  
}

Класс Memory. Память приложения. Здесь хранятся данные о текущем состоянии памяти. Во время работы приложения. Вызывает XmlWriter для обновления. Изменения данных хранящихся в xml документе.

package Entities;  
  
import sample.XmlWriter;  
  
import javax.xml.bind.annotation.XmlElementWrapper;  
import javax.xml.bind.annotation.XmlRootElement;  
import java.io.\*;  
import java.util.ArrayList;  
  
@XmlRootElement()  
public class Memory implements Serializable {  
 private static ArrayList<User> *users* = new ArrayList<User>();;  
 private static Memory *memory*;  
 static final long *serialVersionUID* = -6039216354863297212L;  
 private static String *fileName* = "src/memory.xml";  
  
 public static long getSerialVersionUID() {  
 return *serialVersionUID*;  
 }  
  
 public static void setMemory(Memory memory) {  
 Memory.*memory* = memory;  
 }  
  
 public static synchronized Memory getMemory(){  
 File file = new File(*fileName*);  
 if(file.exists() && !file.isDirectory())  
 *memory* = (Memory)XmlWriter.*getFromXml*(*fileName*);  
  
 if(*memory* == null)  
 return *memory* = new Memory();  
 else  
 return *memory*;  
 }  
  
 /\* public Memory(){  
  
 }\*/  
  
 @XmlElementWrapper  
 public ArrayList<User> getUsers() {  
 return *users*;  
 }  
  
 public static void setUsers(ArrayList<User> users) {  
 Memory.*users* = users;  
 }  
  
 public static void addUser(User user) {  
 *users*.add(user);  
 }  
  
 public static User getById(int id){  
 boolean success = false;  
 User userToFind = null;  
 for(User user : *users*)  
 if(user.getId() == id) {  
 userToFind = user;  
 }  
 if(userToFind != null)  
 return userToFind;  
 return null;  
 }  
  
 public static void refreshUsersData(User user){  
 *deleteById*(user.getId());  
 *addUser*(user);  
 }  
  
 public static boolean deleteById(int id){  
 boolean success = false;  
 User userToFind = null;  
 for(User user : *users*)  
 if(user.getId() == id) {  
 userToFind = user;  
 }  
 if(userToFind != null)  
 success = *users*.remove(userToFind);  
 return success;  
 }  
  
 public static void writeToDisk(){  
 XmlWriter.*saveToXml*(*memory*, *fileName*);  
 }  
  
  
 @Override  
 public String toString() {  
 return "Memory{}" +  
 "\n "+*users*.toString();  
 }  
}

Класс User. Класс хранит информацию о пользователях. Дни тренировок, а также сумма затраченных калорий за определённый день тренировки.

package Entities;  
  
import sample.LocalDateAdapter;  
  
import javax.xml.bind.annotation.XmlElement;  
import javax.xml.bind.annotation.XmlRootElement;  
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;  
import javax.xml.bind.annotation.XmlType;  
import java.io.Serializable;  
import java.time.LocalDate;  
import java.util.ArrayList;  
  
  
  
@XmlRootElement  
@XmlType(propOrder = {"id","name","daysArrayList", "start\_of\_trainings", "last\_training"})  
public class User {  
 private int id;  
 private String name;  
 private ArrayList<Day> daysArrayList = new ArrayList<Day>();  
 private LocalDate start\_of\_trainings;  
 private LocalDate last\_training;  
  
 public User() {  
 }  
  
 public User(int id, String name) {  
 this.id = id;  
 this.name = name;  
 this.start\_of\_trainings = LocalDate.*now*();  
 this.last\_training = start\_of\_trainings;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public ArrayList<Day> getDaysList() {  
 return daysArrayList;  
 }  
  
 public void addDay(Day day) {  
 this.daysArrayList.add(day);  
 this.last\_training = day.getDate();  
 }  
  
 public LocalDate getStart\_of\_trainings() {  
 return start\_of\_trainings;  
 }  
  
 public LocalDate getLast\_training() {  
 return last\_training;  
 }  
  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public ArrayList<Day> getDaysArrayList() {  
 return daysArrayList;  
 }  
  
  
 public void setDaysArrayList(ArrayList<Day> daysArrayList) {  
 this.daysArrayList = daysArrayList;  
 }  
  
 @XmlJavaTypeAdapter(value = LocalDateAdapter.class)  
 public void setStart\_of\_trainings(LocalDate start\_of\_trainings) {  
 this.start\_of\_trainings = start\_of\_trainings;  
 }  
  
 @XmlJavaTypeAdapter(value = LocalDateAdapter.class)  
 public void setLast\_training(LocalDate last\_training) {  
 this.last\_training = last\_training;  
 }  
  
 @Override  
 public String toString() {  
 return name + '\'' +  
 ", last training=" + last\_training +  
 '}';  
 }  
}

Main класс Fitness\_Tracker. Основной класс программы. Точка входа в программу. Содержит меню и отображает данные класса Memory. Также участвует в отсчёте выполнения тренировки, создании новых объектов. (Можно создавать\удалять пользователей, заносить в память новые упражнения.)

package sample;  
import Entities.\*;  
import java.time.Duration;  
import java.time.LocalDate;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.HashSet;  
import java.util.Scanner;  
import static Entities.Exercise\_Type.\*;  
  
  
public class Fitness\_Tracker {  
 static Memory *memory* = Memory.*getMemory*();  
 static User *currentUser*;  
  
 public static void main(String[] args) {  
 // fillMemory(); // метод заполнения тестовыми данными  
 *mainMenu*();  
 }  
  
 /\* главное меню  
 выбор аккаунта  
 \*/  
 public static void mainMenu(){  
 *currentUser* = null;  
 *showUserList*();  
 System.*out*.println("Введите id чтобы выбрать пользователя или " +  
 "\n введите \"новый\" чтобы создать нового" +  
 "\n введите \"выбрать id\" чтобы войти в учетную запись " +  
 "\n введите \"удалить id\" чтобы удалить пользователя" +  
 "\n введите \"выход\" чтобы выйти из программы");  
 String input = new Scanner(System.*in*).nextLine();  
 String[] inputString = input.split(" "); // добавляет ввод в массив строк. Новый элемент через пробел  
  
 if (inputString[0].equalsIgnoreCase("новый")) {  
 *createUser*();  
 } else if((inputString.length == 2)&&(inputString[0].startsWith("выбрать"))&&(inputString[1].matches("[-+]?\\d+"))){  
 int pickId = Integer.*parseInt*(inputString[1]);  
 *currentUser* = *memory*.*getById*(pickId);  
 if(*currentUser* != null)  
 *goIntoAccount*();  
 else  
 System.*out*.println("Пользователь не найден");  
 } else if((inputString.length == 2)&&(inputString[0].startsWith("удалить"))&&(inputString[1].matches("[-+]?\\d+"))) { //  
 int deleteId = Integer.*parseInt*(inputString[1]);  
 if(*memory*.*deleteById*(deleteId))  
 System.*out*.println("Пользователь успешно удален");  
 else  
 System.*out*.println("Пользователь не найден");  
  
 } else if ((inputString.length > 0)&&(inputString[0].equalsIgnoreCase("выход"))) {  
 *safeAndExit*();  
 } else {  
 System.*out*.println("Команда не распознана");  
 *mainMenu*();  
 }  
  
 }  
  
 public static void showUserList(){  
 ArrayList<User> userArrayList = *memory*.getUsers();  
 System.*out*.format("%2s%-2s%-10s%-10s%n", "id", " ", "Имя","Последняя тренеровка");  
 for (User user : userArrayList)  
 System.*out*.format("%2d%-2s%-10s%-10s%n", user.getId(), ") ",user.getName(), user.getLast\_training());  
 }  
  
 public static void createUser(){  
 ArrayList<User> userArrayList = *memory*.getUsers();  
 System.*out*.println("Введите имя пользователя" );  
 int newId=1;  
 String newName = new Scanner(System.*in*).nextLine();  
  
 // сохраняем все id, и перебираем их в цикле чтобы найти не занятый  
 HashSet<Integer> idSet = new HashSet<>();  
 for (User user : userArrayList)  
 idSet.add(user.getId());  
 while (idSet.contains(new Integer(newId)))  
 newId++;  
 userArrayList.add(new User(newId, newName));  
 }  
  
 /\* личное меню для пользователя\*/  
 public static void goIntoAccount(){  
 *showUsersData*();  
 System.*out*.println("Введите \"1\" чтобы начать упражнение отжимания" +  
 "\nвведите \"2\" чтобы начать упражнение скакалка" +  
 "\nвведите \"3\" чтобы начать упражнение приседания" +  
 "\nвведите \"назад\" чтобы вернуться к списку пользователей");  
 String input = new Scanner(System.*in*).nextLine();  
  
 if (input.equalsIgnoreCase("1")) {  
 *startExercise*(*Push\_up*);  
 } else if(input.equalsIgnoreCase("2")){  
 *startExercise*(*Jump\_rope*);  
 } else if(input.equalsIgnoreCase("3")){ //  
 *startExercise*(*Squatting*);  
 } else if (input.equalsIgnoreCase("назад")) {  
 *memory*.*refreshUsersData*(*currentUser*);  
 *mainMenu*();  
 } else {  
 System.*out*.println("Команда не распознана");  
 *goIntoAccount*();  
 }  
 }  
  
 /\* вывод последних тренеровок пользователя \*/  
 public static void showUsersData(){  
 ArrayList<Day> daysArrayList = *currentUser*.getDaysList();  
 System.*out*.print("Пользователь "+*currentUser*.getName()+" id: " +*currentUser*.getId()+  
 "\nДень\t\t\tИзрасходавано каллорий\n");  
 if (daysArrayList.size() == 0)  
 System.*out*.print(" Тренеровки еще не начаты! \nВыберите упражнение для старта\n");  
  
 // определяет сколько последних дней выводить на экран  
 final int showDays = 5;  
 int startFrom = (daysArrayList.size()-1) - showDays;  
 if(startFrom<0)  
 startFrom = 0;  
 for (int i=startFrom; i<=(daysArrayList.size()-1); i++ )  
 System.*out*.print(daysArrayList.get(i).getDate().toString()+"\t\t\t"+ daysArrayList.get(i).getTotal\_Calories()+"\n");  
 }  
  
 /\* метод выполнения упражнения\*/  
 public static void startExercise(Exercise\_Type exercise\_type) {  
 System.*out*.println("Упражнение " + exercise\_type + " начато");  
 Exercise currentExercise = new Exercise(exercise\_type);  
 long start = (long) (System.*currentTimeMillis*() / 1000);  
 long durationSeconds = 0;  
  
 System.*out*.println("Введите \"с\" чтобы закончить упражнение");  
 String input = "";  
 do {  
 input = new Scanner(System.*in*).nextLine();  
 } while (input.equalsIgnoreCase("c"));  
 long duration = (long) (System.*currentTimeMillis*() / 1000) - start;  
 currentExercise.setDuration(duration);  
 System.*out*.println("Упражнение " + exercise\_type + " окончено. Продолжительность " + duration);  
  
 Day currentDay;  
 Day lastDay;  
 // если есть сохраненные дни у пользователя  
 if (*currentUser*.getDaysArrayList().size() > 0){  
 lastDay = *currentUser*.getDaysArrayList().get(*currentUser*.getDaysArrayList().size() - 1);  
 // если последний сохраненный день - сегодня  
 if ((lastDay.getDate().getDayOfYear() == LocalDate.*now*().getDayOfYear())  
 && (lastDay.getDate().getYear() == LocalDate.*now*().getYear()))  
 {  
 currentDay = lastDay;  
 currentDay.addExercise(currentExercise);  
 } else {  
 currentDay = new Day(LocalDate.*now*());  
 currentDay.addExercise(currentExercise);  
 *currentUser*.addDay(currentDay);  
 }  
 } else {  
 currentDay = new Day(LocalDate.*now*());  
 currentDay.addExercise(currentExercise);  
 *currentUser*.addDay(currentDay);  
 }  
  
 *goIntoAccount*();  
 }  
  
 /\* выход из программы с сохранением данных\*/  
 public static void safeAndExit(){  
 *memory*.*writeToDisk*();  
 System.*exit*(0);  
 }  
  
  
 /\* метод заполнения тестовыми данными\*/  
 public static void fillMemory(){  
 // объект Exercise с какими-то данными  
 Exercise exercise = new Exercise();  
 exercise.setDuration(Duration.*ofHours*(1).getSeconds());  
 exercise.setExercise\_consumption(1500);  
 exercise.setType(*Push\_up*);  
  
 // список Exercises  
 ArrayList<Exercise> exercisesArrayList = new ArrayList<>();  
 exercisesArrayList.add(exercise);  
  
 // день  
 Day day = new Day();  
 day.setDate(LocalDate.*now*().minusDays(1));  
 day.setExercises(exercisesArrayList);  
 day.setTotal\_Calories(2000);  
  
 ArrayList<Day> daysArrayList = new ArrayList<>();  
 daysArrayList.add(day);  
  
 User user = new User(1,"Vasya");  
 user.setStart\_of\_trainings(LocalDate.*now*().minusDays(5));  
 user.setLast\_training(LocalDate.*now*());  
 user.setDaysArrayList(daysArrayList);  
  
  
 *memory*.*addUser*(user);  
 *memory*.*addUser*(new User(2,"Peter"));  
 User user3 = new User(3,"Kate");  
 user3.getDaysList().add(new Day(LocalDate.*now*().minusDays(4), 1400));  
 user3.getDaysList().add(new Day(LocalDate.*now*().minusDays(3), 560));  
 user3.getDaysList().add(new Day(LocalDate.*now*().minusDays(2), 870));  
 user3.getDaysList().add(new Day(LocalDate.*now*().minusDays(1), 320));  
 *memory*.*addUser*(user3);  
 *memory*.*addUser*(new User(4,"Steve"));  
 *memory*.*addUser*(new User(5,"Alan"));  
 *memory*.*addUser*(new User(6,"Julia"));  
 *memory*.*addUser*(new User(7,"Mary"));  
 *memory*.*addUser*(new User(8,"Elene"));  
 }  
}

Класс LocalDateAdapter. Необходим для сохранения сущностей объектов LocalDate.

package sample;  
  
import javax.xml.bind.annotation.adapters.XmlAdapter;  
import java.time.LocalDate;  
  
public class LocalDateAdapter extends XmlAdapter<String, LocalDate> {  
 public LocalDate unmarshal(String v) throws Exception {  
 return LocalDate.*parse*(v);  
 }  
  
 public String marshal(LocalDate v) throws Exception {  
 return v.toString();  
 }  
}

Класс XmlWriter. Выполняет запись, изменение xml файла.

package sample;  
  
import Entities.Day;  
import Entities.Exercise;  
import Entities.Memory;  
import Entities.User;  
  
import javax.xml.bind.JAXBContext;  
import javax.xml.bind.JAXBException;  
import javax.xml.bind.Marshaller;  
import javax.xml.bind.Unmarshaller;  
import javax.xml.bind.annotation.XmlRootElement;  
import java.time.Duration;  
import java.time.LocalDate;  
import java.io.File;  
import java.util.ArrayList;  
  
import static Entities.Exercise\_Type.*Push\_up*;  
  
  
public class XmlWriter {  
  
 public static void main(String[] args) {  
 String fileName = "src/memory.xml";  
  
  
  
  
 // сохраняем  
 /\*saveToXml(memory, fileName);  
  
 // восстанавливаем объект из XML файла  
 Memory unmarshUser = getFromXml(fileName);  
 if (unmarshUser != null) {  
 System.out.println(unmarshUser.toString());  
 }\*/  
  
 }  
  
  
  
 // сохраняем объект в XML файл  
 public static void saveToXml(Memory user, String filePath) {  
 try {  
 JAXBContext context = JAXBContext.*newInstance*(Memory.class);  
 Marshaller marshaller = context.createMarshaller();  
  
 // устанавливаем флаг для читабельного вывода XML в JAXB  
 marshaller.setProperty(Marshaller.*JAXB\_FORMATTED\_OUTPUT*, Boolean.*TRUE*);  
  
 // маршаллинг объекта в файл  
 marshaller.marshal(user, new File(filePath));  
 } catch (JAXBException e) {  
 e.printStackTrace();  
 }  
 }  
  
 // восстанавливаем объект из XML файла  
 public static Memory getFromXml(String filePath) {  
 try {  
 // создаем объект JAXBContext - точку входа для JAXB  
 JAXBContext jaxbContext = JAXBContext.*newInstance*(Memory.class);  
 Unmarshaller un = jaxbContext.createUnmarshaller();  
  
 return (Memory) un.unmarshal(new File(filePath));  
 } catch (JAXBException e) {  
 e.printStackTrace();  
 }  
 return null;  
 }  
  
  
}

Memory.xml Файл хранящий информацию о пользователях.

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>  
<memory>  
 <users>  
 <users>  
 <id>1</id>  
 <name>Vasya</name>  
 <daysArrayList>  
 <date>2020-05-09</date>  
 <exercises>  
 <exercises>  
 <duration>3600</duration>  
 <exercise\_consumption>1500</exercise\_consumption>  
 <type>Push\_up</type>  
 </exercises>  
 </exercises>  
 <total\_Calories>2000</total\_Calories>  
 </daysArrayList>  
 <start\_of\_trainings>2020-05-05</start\_of\_trainings>  
 <last\_training>2020-05-10</last\_training>  
 </users>  
 <users>  
 <id>3</id>  
 <name>Kate</name>  
 <daysArrayList>  
 <date>2020-05-09</date>  
 <exercises/>  
 <total\_Calories>1400</total\_Calories>  
 </daysArrayList>  
 <daysArrayList>  
 <date>2020-05-08</date>  
 <exercises/>  
 <total\_Calories>560</total\_Calories>  
 </daysArrayList>  
 <daysArrayList>  
 <date>2020-05-07</date>  
 <exercises/>  
 <total\_Calories>870</total\_Calories>  
 </daysArrayList>  
 <daysArrayList>  
 <date>2020-05-06</date>  
 <exercises/>  
 <total\_Calories>320</total\_Calories>  
 </daysArrayList>  
 <start\_of\_trainings>2020-05-10</start\_of\_trainings>  
 <last\_training>2020-05-10</last\_training>  
 </users>  
 <users>  
 <id>4</id>  
 <name>Steve</name>  
 <start\_of\_trainings>2020-05-10</start\_of\_trainings>  
 <last\_training>2020-05-10</last\_training>  
 </users>  
 <users>  
 <id>5</id>  
 <name>Alan</name>  
 <start\_of\_trainings>2020-05-10</start\_of\_trainings>  
 <last\_training>2020-05-10</last\_training>  
 </users>  
 <users>  
 <id>6</id>  
 <name>Julia</name>  
 <start\_of\_trainings>2020-05-10</start\_of\_trainings>  
 <last\_training>2020-05-10</last\_training>  
 </users>  
 <users>  
 <id>7</id>  
 <name>Mary</name>  
 <start\_of\_trainings>2020-05-10</start\_of\_trainings>  
 <last\_training>2020-05-10</last\_training>  
 </users>  
 <users>  
 <id>8</id>  
 <name>Elene</name>  
 <start\_of\_trainings>2020-05-10</start\_of\_trainings>  
 <last\_training>2020-05-10</last\_training>  
 </users>  
 <users>  
 <id>2</id>  
 <name>Peter</name>  
 <daysArrayList>  
 <date>2020-05-10</date>  
 <exercises>  
 <exercises>  
 <duration>7</duration>  
 <exercise\_consumption>350</exercise\_consumption>  
 <type>Push\_up</type>  
 </exercises>  
 </exercises>  
 <total\_Calories>350</total\_Calories>  
 </daysArrayList>  
 <start\_of\_trainings>2020-05-10</start\_of\_trainings>  
 <last\_training>2020-05-10</last\_training>  
 </users>  
 <users>  
 <id>1</id>  
 <name>Vasya</name>  
 <daysArrayList>  
 <date>2020-05-09</date>  
 <exercises>  
 <exercises>  
 <duration>3600</duration>  
 <exercise\_consumption>1500</exercise\_consumption>  
 <type>Push\_up</type>  
 </exercises>  
 </exercises>  
 <total\_Calories>2000</total\_Calories>  
 </daysArrayList>  
 <start\_of\_trainings>2020-05-05</start\_of\_trainings>  
 <last\_training>2020-05-10</last\_training>  
 </users>  
 <users>  
 <id>2</id>  
 <name>Peter</name>  
 <start\_of\_trainings>2020-05-10</start\_of\_trainings>  
 <last\_training>2020-05-10</last\_training>  
 </users>  
 <users>  
 <id>3</id>  
 <name>Kate</name>  
 <daysArrayList>  
 <date>2020-05-06</date>  
 <exercises/>  
 <total\_Calories>1400</total\_Calories>  
 </daysArrayList>  
 <daysArrayList>  
 <date>2020-05-07</date>  
 <exercises/>  
 <total\_Calories>560</total\_Calories>  
 </daysArrayList>  
 <daysArrayList>  
 <date>2020-05-08</date>  
 <exercises/>  
 <total\_Calories>870</total\_Calories>  
 </daysArrayList>  
 <daysArrayList>  
 <date>2020-05-09</date>  
 <exercises/>  
 <total\_Calories>320</total\_Calories>  
 </daysArrayList>  
 <start\_of\_trainings>2020-05-10</start\_of\_trainings>  
 <last\_training>2020-05-10</last\_training>  
 </users>  
 <users>  
 <id>4</id>  
 <name>Steve</name>  
 <start\_of\_trainings>2020-05-10</start\_of\_trainings>  
 <last\_training>2020-05-10</last\_training>  
 </users>  
 <users>  
 <id>5</id>  
 <name>Alan</name>  
 <start\_of\_trainings>2020-05-10</start\_of\_trainings>  
 <last\_training>2020-05-10</last\_training>  
 </users>  
 <users>  
 <id>6</id>  
 <name>Julia</name>  
 <start\_of\_trainings>2020-05-10</start\_of\_trainings>  
 <last\_training>2020-05-10</last\_training>  
 </users>  
 <users>  
 <id>7</id>  
 <name>Mary</name>  
 <start\_of\_trainings>2020-05-10</start\_of\_trainings>  
 <last\_training>2020-05-10</last\_training>  
 </users>  
 <users>  
 <id>8</id>  
 <name>Elene</name>  
 <start\_of\_trainings>2020-05-10</start\_of\_trainings>  
 <last\_training>2020-05-10</last\_training>  
 </users>  
 </users>  
</memory>

## Диаграммы.

