**Министерство науки и высшего образования российской федерации**

**ФГАОУ ВО «Пермский государственный национальный исследовательский университет»**

Физический факультет

Кафедра радиоэлектроники и защиты информации

Технологии Java

Лабораторная работа №1 на тему

**«****Технологии Java для обработки массивов данных»**

Выполнил:

студент 2 курса физического факультета

группа ФЗ/О ПМИ-2-2021 НБ

направление «01.03.02 Прикладная математика и информатика»

профиль «Инженерия программного обеспечения»

Макарова Полина Фёдоровна

Пермь, 2022 г.

**Оглавление**

[1. Введение 3](#_Toc84930200)

[2. Постановка задачи 4](#_Toc84930201)

[3. Практическая реализация 5](#_Toc84930202)

[3.1. Используемый инструментарий 5](#_Toc84930203)

[3.2. описание входного файла 5](#_Toc84930204)

[3.3. описание основных частей программы в виде листингов с комментариями 5](#_Toc84930205)

[3.4. описание результирующего файла с примером 6](#_Toc84930206)

[3.5. тестирование: 7](#_Toc84930207)

[4. Заключение 8](#_Toc84930208)

[5. Список литературы 9](#_Toc84930209)

[Приложение 10](#_Toc84930210)

# Введение

**Цель работы.** Получение навыков программирования на языке Java; изучение основных классов и методов JavaSE и JavaSDK: коллекций, java.time.\*; изучение библиотеки Apache POI для работы с файлами офисных приложений.

Java – это объектно-ориентированный язык программирования. Понятие «объектно-ориентированный» относится к способу написания структурного кода Java, а именно: разделение кода на так называемые «классы», которые запускаются вместе, чтобы обеспечить согласованное порождение объектов.

Чтобы запустить и использовать Java, вам нужно три вещи:

* JDK – Java Development Kit
* JRE – Java Runtime Environment
* JVM – Java Virtual Machine

Виртуальная машина Java (JVM) гарантирует, что у ваших приложений Java есть доступ к минимальным ресурсам, необходимым для их запуска.

Среда исполнения Java (JRE) предоставляет собой «контейнер» для всех этих элементов и кода для запуска приложения. JDK – это «компилятор», который интерпретирует сам код и выполняет его. В JDK также есть инструменты разработчика, необходимые для написания кода Java (как и следует из названия).[1]

SDK — это базовая среда разработки программ на Java. Новые версии языка и версии SDK являются расширением прежних. Это сделано для того, чтобы не возникала необходимость переписывать существующие программы.[2]

Пакет java.time основной API для дат, времени, мгновений и длительностей.

Определенные здесь классы представляют основные концепции даты и времени, включая моменты времени, длительности, даты, время, часовые пояса и периоды. Они основаны на календарной системе ISO, которая *фактически* является мировым календарем, следующим пролептическим григорианским правилам. Все классы являются неизменяемыми и потокобезопасными.

Каждый экземпляр даты и времени состоит из полей, которые удобно доступны с помощью API. Для доступа к полям более низкого уровня обратитесь к java.time.temporal пакету. Каждый класс включает в себя поддержку печати и синтаксического анализа всевозможных дат и времени. Параметры настройки см. в java.time.format пакете. [5]

Часто для создания отчетов в формате файлов Microsoft Excel требуется программное приложение. Иногда даже ожидается, что приложение получит файлы Excel в качестве входных данных. Например, приложение, разработанное для финансового отдела компании, должно будет генерировать все свои результаты в Excel.

Apache POI – это популярный API, который позволяет программистам создавать, изменять и отображать файлы MS Office с помощью программ Java. Это библиотека с открытым исходным кодом, разработанная и распространяемая Apache Software Foundation для разработки или изменения файлов Microsoft Office с использованием программы Java. Он содержит классы и методы для декодирования вводимых пользователем данных или файла в документы MS Office. [6]

Что такое Collections Framework? Это набор различных интерфейсов для работы с группами объектов при программировании на Java. Коллекции позволяют производить манипуляции с массивами (Array), очередями (Queue, Stack), списками (List), ключами и зна-чениями (Map) и другими типами объектов. Любая коллекция на Java строго структурирована: одни интерфейсы подчиняют другие, последние же — расширяют функционал своих «‎‎старших братьев» по иерархии. [4]

Сам термин берёт своё начало из таких вещей, как "Теория типов" и "Абстрактные типы данных". Но если не смотреть на какие-то высокие материи, то когда у нас несколько вещей, то мы можем назвать их "коллекция вещей". Те, кто собирают предметы. Вообще само слово коллекционировать происходит от лат. collectio «собирание, сбор». То есть коллекция — это сбор чего-то, контейнер для каких-то элементов. [3]

Коллекция (Collection) представляет собой группу объектов, известных как ее элементы. Это объект, который может содержать ссылки на другие объекты. Интерфейс Collection является корнем иерархии коллекций. Это базовый интерфейс для всех коллекций в Java Collection Framework. Он определяет основные методы, которые должны быть реализованы во всех коллекциях, такие как add(), remove() и contains().

Интерфейсы в Java Collection Framework определяют общее поведение и операции, которые могут выполняться с коллекциями. Сюда входят добавление или удаление элементов, повторение элементов в коллекции и многое другое.

* «Collection»: корневой (root) интерфейс в иерархии коллекций, представляющий группу объектов, известных как элементы;
* «List»: упорядоченная коллекция элементов, допускающая дублирование;
* «Set»: коллекция элементов, не допускающих дублирования;
* «Map»: коллекция пар ключ-значение (key-value), где каждый ключ уникален;
* «Queue»: очередь — это структура данных, которая используется для хранения элементов в порядке появления (First-In-First-Out, FIFO);

В этот список вошли далеко не все, а только наиболее используемые интерфейсы в Java Collection Framework. Теперь давайте подробно рассмотрим каждый их них.

«Collection»: коллекция представляет собой группу объектов, известных как ее элементы. Это объект, который может содержать ссылки на другие объекты. Интерфейс Collection является корнем иерархии коллекций. Это базовый интерфейс для всех коллекций в Java Collection Framework. Он определяет основные методы, которые должны быть реализованы во всех коллекциях, такие как add(), remove() и contains().

«List» (список) — это упорядоченный набор объектов, каждый элемент которого занимает определенную позицию в списке. Интерфейс List расширяет интерфейс Collection и добавляет в него несколько методов для работы со списками, таких как методы доступа к элементам по их положению в списке и методы поиска и сортировки списков. List может содержать повторяющиеся элементы, доступ к этим элементам можно получить по их положению в списке.

«Set» — это неупорядоченное множество уникальных элементов, в котором не допускаются повторяющиеся элементы. Интерфейс Set расширяет интерфейс Collection и добавляет в него несколько методов, таких как методы проверки наличия элемента в множестве (set) и методы добавления и удаления элементов из множества.

«Queue» (очередь) — это структура данных, которая используется для хранения элементов в порядке их появления (FIFO). Это означает, что первый элемент, добавленный в очередь, будет первым удаленным.

«Map» используется для сопоставления ключей со значениями. Он позволяет хранить элементы в виде пар ключ-значение и предоставляет методы для доступа, изменения и повторения элементов на карте.

Также рассмотрим класс java.util.ArrayList в коллекциях Java используется для хранения изменяемого размера массива элементов в списке. Это широко используемая реализация интерфейса java.util.List, которая использует массив для хранения элементов и предоставляет эффективные методы для доступа, изменения и повторения элементов в списке. Класс java.util.ArrayList обеспечивает быстрый произвольный доступ к своим элементам, но медленную вставку и удаление в произвольных позициях. [7]

# Постановка задачи

Написать консольное приложение для настольных персональных компьютеров, позволяющее работать с данными электронных таблиц (вариант № 18).

Приложение должно позволять загружать данные из электронной таблицы (формат xlsx).

Приложение должно позволять строить аналитический отчёт по алгоритму «получить данные о еде, времени приёма пищи(завтрак, обед и т.д.) и дате, сформировать список записанных продуктов за определённый день в виде документа docx» (вариант № 18). Подготовка данных для аналитического отчёта должна быть осуществлена средствами коллекций Java на сколько это возможно.

Программа должна использовать инструменты Java, JDK, коллекции, java.time.\*, Apache POI.

Подготовить тестовые данные и произвести тестирование основного функционала программы.

# Практическая реализация

## Используемый инструментарий

Таблица 1

Инструментарий, использованный при решении поставленной задачи

|  |  |
| --- | --- |
| Язык программирования | Java |
| Версия JavaSDK | 18 |
| Среда разработки | Intellij IDEA 2022.2.3. |
| Система автоматизированной сборки | Gradle |
| Инструменты JavaSDK | java.util.Collections  java.time  java.util.UUID |
| Библиотеки | Apache POI |

## Модель данных

Рисунок 1 – Диаграмма классов данной предметной области

Модель данных представляет собой набор классов с такими полями:

1. CTime – UUID id, LocalDate date, String day, List<CFood> foods;

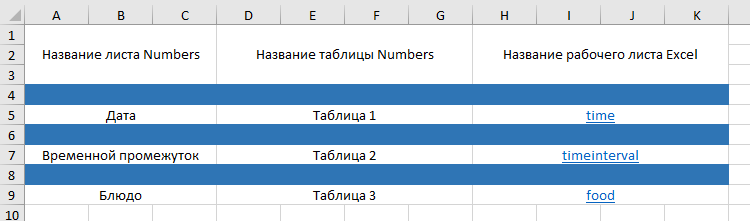
2. CTimeInterval – UUID id, String name, String vr;

3. CFood – UUID id, String name, Double kal, CTime time, CTimeInterval timeinterval

В данной предметной области «Дневник еды» было выявлено три класса: «дата», «временной промежуток», «еда». Класс «дата» отвечает за хранение информации о дне, месяце, годе и дню недели. Класс «временной промежуток» отвечает за хранение информации о временном промежутке (завтрак, обед и т.д.). Класс «еда» отвечает за хранение информации о еде и калориях.

Выбор пал на эти три класса, так как в дневнике еды важно знать, что ты съел, во сколько и когда, чтобы составить зависимость реакции тела на ту или иную еду.

## Описание загружаемого файла

Файл с исходными данными представляет собой электронную таблицу формата timeintervals.xlsx, с рабочими листами дата, временной промежуток и блюдо.

*Рисунок 2 – вид электронной таблицы, лист 1*

Рассмотрим каждую таблицу отдельно:

На листе 2 «Дата» данные разделены по колонкам соответственно:

А – идентификатор даты, UUID;

В – дата, LocalDate.

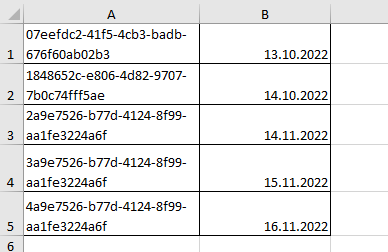


Рисунок 4 –вид электронной таблицы, лист 2

На листе 3 «Временной промежуток» данные разделены по колонкам соответственно:

А – идентификатор временного промежутка, UUID;

В – название, String;

С – временной промежуток, String.

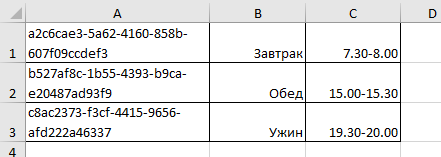


Рисунок 3 – вид электронной таблицы, лист 3

На листе «блюдо» данные разделены по колонкам соответственно:

А – идентификатор блюда, UUID;

B – название, String;

C – количество калорий, Double;

D – идентификатор даты, UUID;

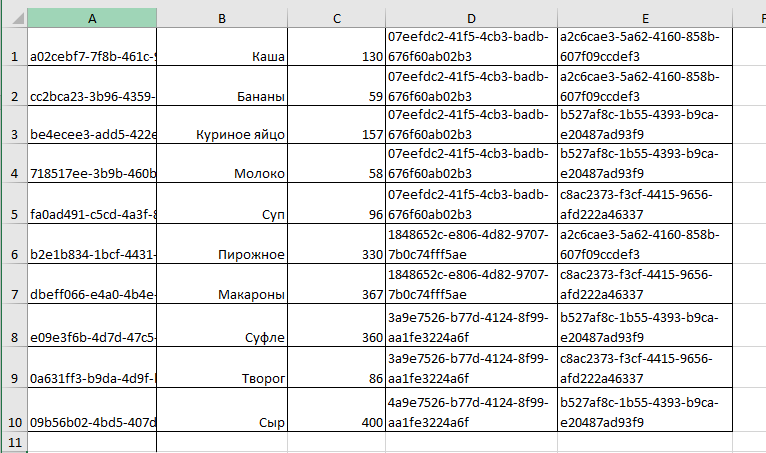
E – идентификатор временного промежутка, UUID.

Рисунок 5 – вид электронной таблицы, лист 4

## Описание алгоритма работы программы

Для чтения файла разработан метод open Excel, в котором инициализируется XSSFWorkbook из библиотеки POI, далее в цикле перебираются листы, строки, колонки, создаются экземпляры классов CTime, CTimeInterval, CFood.

|  |
| --- |
| Листинг 1 – Метод для загрузки данных TimeInterval из электронной таблицы |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Загрузка информации о временных промежутков из электронной таблицы. \**  *\* Результат в карте timeintervals.                              \**  *\** ***@param*** *wb - рабочая книга с данными.                          \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  private static void loadTimeInterval(XSSFWorkbook wb) {     // Берётся 2(считая их от 0) лист(таблица) из файла .xlsx.     Sheet sheet = wb.getSheetAt(2);     Row row;     Cell cell;     int i;     int nRows = sheet.getLastRowNum();     String tiUUID, name, vr;     UUID id;     CTimeInterval timeInterval;     // Перебираются строки 1 таблицы в файле .xlsx.     for (i = 0; i <= nRows; i++) {         row = sheet.getRow(i);         // Просматривается есть ли данные в ряду.         if (row == null)             continue;         // Просматривается все ли заполнены ячейки в ряду.         if (row.getLastCellNum() < 3)             continue;         // Достаётся UUID из 1 ячейки.         cell = row.getCell(0);         tiUUID = cell.getStringCellValue();         if (tiUUID.length() == 0)             continue;         // Создаётся новый объект класса.         timeInterval = new CTimeInterval();         id = UUID.*fromString*(tiUUID);         // Заполняется в объекте UUID из 1 ячейки в сторе.         timeInterval.setId(id);         cell = row.getCell(1);         name = cell.getStringCellValue();         timeInterval.setName(name);         cell = row.getCell(2);         vr = cell.getStringCellValue();         timeInterval.setVr(vr);         // Объект помещается в карточку timeintervals.  *timeintervals*.put(id, timeInterval);     }  } |
| Листинг 2 – Метод для загрузки данных Time из электронной таблицы |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Загрузка информации о дате из электронной таблицы. \**  *\* Результат в карте times.                           \**  *\** ***@param*** *wb - рабочая книга с данными.               \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  private static void loadTime(XSSFWorkbook wb) {     // Берётся 1(считая их от 0) лист(таблица) из файла .xlsx.     Sheet sheet = wb.getSheetAt(1);     Row row;     Cell cell;     int i;     int nRows = sheet.getLastRowNum();     String tUUID;     UUID id;     CTime time;     LocalDate date;     // Перебираются строки 2 таблицы в файле .xlsx.     for (i = 0; i <= nRows; i++) {         row = sheet.getRow(i);         // Просматривается есть ли данные в ряду.         if (row == null)             continue;         // Просматривается все ли заполнены ячейки в ряду.         if (row.getLastCellNum() < 2)             continue;         // Достаётся UUID из 1 ячейки.         cell = row.getCell(0);         tUUID = cell.getStringCellValue();         if (tUUID.length() == 0)             continue;         // Создаётся новый объект класса.         time = new CTime();         id = UUID.*fromString*(tUUID);         // Заполняется в объекте UUID из 1 ячейки в сторе.         time.setId(id);         cell = row.getCell(1);         date = cell.getLocalDateTimeCellValue().toLocalDate();         time.setDate(date);         // Объект помещается в карточку times.  *times*.put(id, time);     }  } |

|  |
| --- |
| Листинг 3 – Метод для загрузки данных Food из электронной таблицы |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Загрузка информации об еде из электронной таблицы.  \**  *\* Результат в карте foods.                            \**  *\** ***@param*** *wb - рабочая книга с данными.                \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *private static void loadFood(XSSFWorkbook wb) {*  *// Берётся 3(считая их от 0) лист(таблица) из файла .xlsx.*  *Sheet sheet = wb.getSheetAt(3);*  *Row row;*  *Cell cell;*  *int i;*  *int nRows = sheet.getLastRowNum();*  *String fUUID, name;*  *double kalors;*  *UUID id;*  *CFood food;*  *// Перебираются строки 3 таблицы в файле .xlsx.*  *for (i = 0; i < nRows; i++) {*  *row = sheet.getRow(i);*  *// Просматривается есть ли данные в ряду.*  *if (row == null)*  *continue;*  *// Просматривается все ли заполнены ячейки в ряду.*  *if (row.getLastCellNum() < 5)*  *continue;*  *// Достаётся UUID из 1 ячейки.*  *cell = row.getCell(0);*  *fUUID = cell.getStringCellValue();*  *if (fUUID.length() == 0)*  *continue;*  *// Создаётся новый объект класса.*  *food = new CFood();*  *id = UUID.fromString(fUUID);*  *// Заполняется в объекте UUID из 1 ячейки в сторе.*  *food.setId(id);*  *// Заполняется в объекте Название из 2 ячейки в строке.*  *cell = row.getCell(1);*  *name = cell.getStringCellValue();*  *food.setName(name);*  *// Заполняется в объекте Калории из 3 ячейки в строке.*  *cell = row.getCell(2);*  *kalors = cell.getNumericCellValue();*  *food.setCal(kalors);*  *// Объект помещается в карточку foods.*  *foods.put(id, food);*  *}*  *}* |

|  |
| --- |
| Листинг 4 – Метод для загрузки данных о связи Food из электронной таблицы |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Загрузка информации о связях еды из электронной таблицы. \**  *\* Результат в объектах карт foods, timeintervals, times.   \**  *\** ***@param*** *wb - рабочая книга с данными.                     \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *private static void loadFoodRelations(XSSFWorkbook wb) {*  *// Берётся 3(считая их он 0) лист(таблица) из файла .xlsx.*  *Sheet sheet = wb.getSheetAt(3);*  *Row row;*  *Cell cell;*  *int i;*  *int nRows = sheet.getLastRowNum();*  *String sId, sTimeIntervalId, sFoodId, sTimeId;*  *UUID id, timeintervalId, foodId, timeId;*  *CFood food;*  *CTimeInterval timeinterval;*  *CTime time;*  *// Перебираются строки 3 таблицы в файле .xlsx.*  *for (i = 0; i < nRows; i++) {*  *row = sheet.getRow(i);*  *// Просматривается есть ли данные в ряду.*  *if (row == null)*  *continue;*  *// Просматривается все ли заполнены ячейки в ряду.*  *if (row.getLastCellNum() < 5)*  *continue;*  *// Достаётся UUID из 1 ячейки.*  *cell = row.getCell(0);*  *sId = cell.getStringCellValue();*  *if (sId.length() == 0)*  *continue;*  *// Достаётся объект класса CFOOD по UUID из карточки foods*  *id = UUID.fromString(sId);*  *food = foods.get(id);*  *// Связываем еду с датой, в дате в список добавляем еду.*  *cell = row.getCell(3); // Берём id даты из таблицы еды.*  *sTimeId = cell.getStringCellValue();*  *timeId = UUID.fromString(sTimeId);*  *time = times.get(timeId);*  *if (time!=null) {*  *food.setTime(time);*  *time.getFoods().add(food);*  *}*  *// Связываем еду с временным интервалом.*  *cell = row.getCell(4); // Берём id интервала из таблицы еды.*  *sTimeIntervalId = cell.getStringCellValue();*  *timeintervalId = UUID.fromString(sTimeIntervalId);*  *timeinterval = timeintervals.get(timeintervalId);*  *if (timeinterval!=null) {*  *food.setTimeinterval(timeinterval);*  *}*  *}*  *}* |

## Описание формируемого файла

В результате выполнения программы формируется файл в формате документа docx, пример которого приведён на рисунке ниже.

|  |
| --- |
| Рисунок 6 – Пример сформированной список записанных продуктов |

На выходе данные таблицы еды: дата, день, интервал, время, еда, калории.

Результирующий файл состоит:

Отцентрированный заголовок «Расписание приема пищи на (день, заданный при формировании файла)» стиль текста Times New Roman, текст жирный, размер 20;

Таблица с 3 колонками:

1. интервал;
2. еда;
3. калории.

стиль текста Times New Roman, текст жирный отцентрированный, размер 14;

в ячейка текст calibri, размер 11.

## Тестирование

При отсутствии файла или нет записей выдаёт строку:

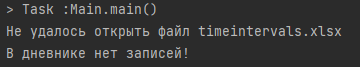


Рисунок 7 – Тест 1

При отсутствии записей в ячейка выдаёт значение null в консоли:



Рисунок 8 – Тест 2

# Заключение

Приложение написано, протестировано, решает поставленную задачу.

Получены навыки программирования на языке Java, а также новые теоретические и практические знания о Системе автоматизированной сборки Gradle, инструментах JDK (java.util.Collections, java.time, java.util.UUID), библиотеке Apache POI.

# **Список литературы**

1. Java: краткое руководство для начинающих. Пишем простое приложение без опыта программирования, 16 сентября 2020 : [Электронный ресурс] – URL: <https://tproger.ru/translations/java-intro-for-beginners/> (дата обращения: 21.12.2022);
2. Конспект лекций по Java. Занятие 1, 1999 — 2010 : [Электронный ресурс] – URL: <http://www.javable.com/tutorials/fesunov/lesson1/> (дата обращения: 21.12.2022);
3. Коротко о главном — Java Collections Framework, 11 июня 2019 : [Электронный ресурс] – URL: <https://javarush.ru/groups/posts/2308-korotko-o-glavnom---java-collections-framework> (дата обращения: 21.12.2022);
4. Коллекционер хорошего кода: что нужно знать о Java Collections Framework, 18 августа 2021 : [Электронный ресурс] – URL: <https://highload.today/blogs/kollektsioner-horoshego-koda-chto-nuzhno-znat-o-java-collections-framework/> (дата обращения: 21.12.2022);
5. Пакет java.time : [Электронный ресурс] – URL: <https://docs.oracle.com/en/java/javase/14/docs/api/java.base/java/time/package-summary.html> (дата обращения: 21.12.2022);
6. Apache POI - Краткое руководство - CoderLessons.com, 3 февраль 2019 г : [Электронный ресурс] – URL: <https://coderlessons.com/tutorials/java-tekhnologii/izuchite-apache-poi/apache-poi-kratkoe-rukovodstvo> (дата обращения: 21.12.2022);
7. Кофе-брейк #185. Подробное руководство по Java Collection Framework, 15 декабря 2022 : [Электронный ресурс] – URL: https://javarush.com/groups/posts/3986-kofe-breyk-185-podrobnoe-rukovodstvo-po-java-collection-framework (дата обращения: 21.12.2022);

Приложение

|  |
| --- |
| Класс Time |
| *package LaBa.One;*  *import java.util.ArrayList;*  *import java.util.List;*  *import java.util.UUID;*  *import java.time.LocalDate;*  *import java.time.\*;*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Класс дата.                            \**  *\** ***@autor*** *Макарова П.Ф. ПМИ-2 21.12.2022. \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *public class CTime {*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Идендификатор.         \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *private UUID id;*  *public UUID getId() {*  *return id;*  *}*  *public void setId(UUID id) {*  *this.id = id;*  *}*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Дата записи.           \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *private LocalDate date;*  *public LocalDate getDate() {*  *return date;*  *}*  *public void setDate (LocalDate date) {*  *this.date = date;*  *}*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* День недели.                   \**  *\* Вычисления дня недели по дате. \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *private String day; //*  *// В интеренете присутствует вывод дня недели английскими буквами и на некоторые другие языки.*  *// Вывод на русский я не нашла, поэтому сделала через цикл и номер дня недели*  *public String getDay() {*  *DayOfWeek dayNumber = date.getDayOfWeek();*  *if (dayNumber.getValue() == 1) {*  *day = "Пн";*  *} else {if (dayNumber.getValue() == 2) {*  *day = "Вт";*  *} else {if (dayNumber.getValue() == 3) {*  *day = "Ср";*  *} else {if (dayNumber.getValue() == 4) {*  *day = "Чт";*  *} else {if (dayNumber.getValue() == 5) {*  *day = "Пт";*  *} else {if (dayNumber.getValue() == 6) {*  *day = "Сб";*  *} else {if (dayNumber.getValue() == 7) {*  *day = "Вс";*  *}}}}}}};*  *return day;*  *}*  *public void setDay(String day) {*  *this.day = day;}*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Вывод записи.          \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *@Override*  *public String toString() {*  *System.out.println((date == null)? "Нет записи" : "Дата: " + date + ";" + " День: "+ getDay() + ";");*  *if (foods.size() != 0){*  *int i;*  *for (i = 0; i < foods.size(); ) {*  *System.out.println(foods.remove(i));*  *}}*  *else{*  *System.out.println("Нет записей");*  *};*  *return "";*  *}*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Список блюд, съеденных в определённый день. \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *private List<CFood> foods;*  *public List<CFood> getFoods() {*  *return foods;*  *}*  *public void setFoods(List<CFood> foods) {*  *this.foods = foods;*  *};*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Конструктор.           \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *public CTime() {*  *foods = new ArrayList<>();*  *}*  *}* |
| Класс CTimeInterval |
| *package LaBa.One;*  *import java.util.UUID;*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Класс временной интервал.              \**  *\** ***@autor*** *Макарова П.Ф. ПМИ-2 21.12.2022. \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *public class CTimeInterval {*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Идендификатор.          \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *private UUID id;*  *public UUID getId() {*  *return id;*  *}*  *public void setId(UUID id) {*  *this.id = id;*  *}*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Название.               \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *private String name;*  *public String getName() {*  *return name;*  *}*  *public void setName(String name) {*  *this.name = name;*  *}*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Временной промежуток.  \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *private String vr;*  *public String getVr() {*  *return vr;*  *}*  *public void setVr(String vr) {*  *this.vr = vr;*  *}*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Вывод записи.          \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *@Override*  *public String toString() {*  *return "Интервал: " + name + ";";*  *}*  *}* |
| Класс CFood |
| *package LaBa.One;*  *import java.util.UUID;*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Класс блюдо(еда).                      \**  *\** ***@autor*** *Макарова П.Ф. ПМИ-2 21.12.2022. \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *public class CFood {*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Идендификатор.          \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *private UUID id;*  *public UUID getId() {*  *return id;*  *}*  *public void setId(UUID id) {*  *this.id = id;*  *}*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Название.               \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *private String name;*  *public String getName() {*  *return name;*  *}*  *public void setName(String name) {*  *this.name = name;*  *}*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Калории.                \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *private double cal;*  *public double getCal() {*  *return cal;*  *}*  *public void setCal(double cal) {*  *this.cal = cal;*  *}*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Вывод записи.           \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *@Override*  *public String toString() {*  *return ((name==null) ? "Нет записи" : timeinterval + " Блюдо - " + name + "; Калории: " + cal);*  *}*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Дата, когда еда записна в дневнике(т.е. съедена). \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *private CTime time;*  *public CTime getTime() {*  *return time;*  *}*  *public void setTime(CTime time) {*  *this.time = time;*  *}*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Интервал времени, когда еда зааписана в дневнике(т.е. съедена). \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  *private CTimeInterval timeinterval;*  *public CTimeInterval getTimeinterval() {*  *return timeinterval;*  *}*  *public void setTimeinterval(CTimeInterval timeinterval) {*  *this.timeinterval = timeinterval;*  *}*  *}* |
| Полный исходный код вашей программы. |
| package LaBa.One;  import org.apache.poi.ss.usermodel.Cell;  import org.apache.poi.ss.usermodel.Row;  import org.apache.poi.ss.usermodel.Sheet;  import org.apache.poi.xssf.usermodel.XSSFWorkbook;  import org.apache.poi.xwpf.usermodel.\*;  import java.io.FileInputStream;  import java.io.FileNotFoundException;  import java.io.FileOutputStream;  import java.io.IOException;  import java.io.\*;  import java.time.LocalDate;  import java.time.format.DateTimeFormatter;  import java.util.\*;  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Дневник еды (данная предметная область). \**  *\* Основной класс программы.                \**  *\** ***@autor*** *Макарова П.Ф. ПМИ-2 21.12.2022.   \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  public class Main {     private static final Map<UUID, CTimeInterval> *timeintervals* = new TreeMap<>();     private static final Map<UUID, CFood> *foods* = new TreeMap<>();     private static final Map<UUID, CTime> *times* = new TreeMap<>();     private static final DateTimeFormatter *formatter* = DateTimeFormatter.*ofPattern*("dd MMMM yyyy");  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Открытие электронной таблицы с входными данными. \**  *\** ***@return*** *- рабочая книга с данными.               \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  private static XSSFWorkbook openExcel() {         XSSFWorkbook wb = null;         try(FileInputStream fis = new FileInputStream("timeintervals.xlsx")) {             wb = new XSSFWorkbook(fis);         }         catch(FileNotFoundException e) {             System.*out*.println("Не удалось открыть файл timeintervals.xlsx");             e.printStackTrace();         }         catch(IOException e) {             System.*out*.println("Не удалось прочитать информацию из файла timeintervals.xlsx");             e.printStackTrace();         }         return wb;     }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Загрузка информации о временных промежутков из электронной таблицы. \**  *\* Результат в карте timeintervals.                              \**  *\** ***@param*** *wb - рабочая книга с данными.                          \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  private static void loadTimeInterval(XSSFWorkbook wb) {         // Берётся 2(считая их от 0) лист(таблица) из файла .xlsx.         Sheet sheet = wb.getSheetAt(2);         Row row;         Cell cell;         int i;         int nRows = sheet.getLastRowNum();         String tiUUID, name, vr;         UUID id;         CTimeInterval timeInterval;         // Перебираются строки 1 таблицы в файле .xlsx.         for (i = 0; i <= nRows; i++) {             row = sheet.getRow(i);             // Просматривается есть ли данные в ряду.             if (row == null)                 continue;             // Просматривается все ли заполнены ячейки в ряду.             if (row.getLastCellNum() < 3)                 continue;             // Достаётся UUID из 1 ячейки.             cell = row.getCell(0);             tiUUID = cell.getStringCellValue();             if (tiUUID.length() == 0)                 continue;             // Создаётся новый объект класса.             timeInterval = new CTimeInterval();             id = UUID.*fromString*(tiUUID);             // Заполняется в объекте UUID из 1 ячейки в сторе.             timeInterval.setId(id);             cell = row.getCell(1);             name = cell.getStringCellValue();             timeInterval.setName(name);             cell = row.getCell(2);             vr = cell.getStringCellValue();             timeInterval.setVr(vr);             // Объект помещается в карточку timeintervals.  *timeintervals*.put(id, timeInterval);         }     }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Загрузка информации о дате из электронной таблицы. \**  *\* Результат в карте times.                           \**  *\** ***@param*** *wb - рабочая книга с данными.               \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  private static void loadTime(XSSFWorkbook wb) {         // Берётся 1(считая их от 0) лист(таблица) из файла .xlsx.         Sheet sheet = wb.getSheetAt(1);         Row row;         Cell cell;         int i;         int nRows = sheet.getLastRowNum();         String tUUID;         UUID id;         CTime time;         LocalDate date;         // Перебираются строки 2 таблицы в файле .xlsx.         for (i = 0; i <= nRows; i++) {             row = sheet.getRow(i);             // Просматривается есть ли данные в ряду.             if (row == null)                 continue;             // Просматривается все ли заполнены ячейки в ряду.             if (row.getLastCellNum() < 2)                 continue;             // Достаётся UUID из 1 ячейки.             cell = row.getCell(0);             tUUID = cell.getStringCellValue();             if (tUUID.length() == 0)                 continue;             // Создаётся новый объект класса.             time = new CTime();             id = UUID.*fromString*(tUUID);             // Заполняется в объекте UUID из 1 ячейки в сторе.             time.setId(id);             cell = row.getCell(1);             date = cell.getLocalDateTimeCellValue().toLocalDate();             time.setDate(date);             // Объект помещается в карточку times.  *times*.put(id, time);         }     }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Загрузка информации об еде из электронной таблицы.  \**  *\* Результат в карте foods.                            \**  *\** ***@param*** *wb - рабочая книга с данными.                \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  private static void loadFood(XSSFWorkbook wb) {         // Берётся 3(считая их от 0) лист(таблица) из файла .xlsx.         Sheet sheet = wb.getSheetAt(3);         Row row;         Cell cell;         int i;         int nRows = sheet.getLastRowNum();         String fUUID, name;         double kalors;         UUID id;         CFood food;         // Перебираются строки 3 таблицы в файле .xlsx.         for (i = 0; i < nRows; i++) {             row = sheet.getRow(i);             // Просматривается есть ли данные в ряду.             if (row == null)                 continue;             // Просматривается все ли заполнены ячейки в ряду.             if (row.getLastCellNum() < 5)                 continue;             // Достаётся UUID из 1 ячейки.             cell = row.getCell(0);             fUUID = cell.getStringCellValue();             if (fUUID.length() == 0)                 continue;             // Создаётся новый объект класса.             food = new CFood();             id = UUID.*fromString*(fUUID);             // Заполняется в объекте UUID из 1 ячейки в сторе.             food.setId(id);             // Заполняется в объекте Название из 2 ячейки в строке.             cell = row.getCell(1);             name = cell.getStringCellValue();             food.setName(name);             // Заполняется в объекте Калории из 3 ячейки в строке.             cell = row.getCell(2);             kalors = cell.getNumericCellValue();             food.setCal(kalors);             // Объект помещается в карточку foods.  *foods*.put(id, food);         }     }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Загрузка информации о связях еды из электронной таблицы. \**  *\* Результат в объектах карт foods, timeintervals, times.   \**  *\** ***@param*** *wb - рабочая книга с данными.                     \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  private static void loadFoodRelations(XSSFWorkbook wb) {         // Берётся 3(считая их он 0) лист(таблица) из файла .xlsx.         Sheet sheet = wb.getSheetAt(3);         Row row;         Cell cell;         int i;         int nRows = sheet.getLastRowNum();         String sId, sTimeIntervalId, sFoodId, sTimeId;         UUID id, timeintervalId, foodId, timeId;         CFood food;         CTimeInterval timeinterval;         CTime time;         // Перебираются строки 3 таблицы в файле .xlsx.         for (i = 0; i < nRows; i++) {             row = sheet.getRow(i);             // Просматривается есть ли данные в ряду.             if (row == null)                 continue;             // Просматривается все ли заполнены ячейки в ряду.             if (row.getLastCellNum() < 5)                 continue;             // Достаётся UUID из 1 ячейки.             cell = row.getCell(0);             sId = cell.getStringCellValue();             if (sId.length() == 0)                 continue;             // Достаётся объект класса CFOOD по UUID из карточки foods             id = UUID.*fromString*(sId);             food = *foods*.get(id);             // Связываем еду с датой, в дате в список добавляем еду.             cell = row.getCell(3); // Берём id даты из таблицы еды.             sTimeId = cell.getStringCellValue();             timeId = UUID.*fromString*(sTimeId);             time = *times*.get(timeId);             if (time!=null) {                 food.setTime(time);                 time.getFoods().add(food);             }             // Связываем еду с временным интервалом.             cell = row.getCell(4); // Берём id интервала из таблицы еды.             sTimeIntervalId = cell.getStringCellValue();             timeintervalId = UUID.*fromString*(sTimeIntervalId);             timeinterval = *timeintervals*.get(timeintervalId);             if (timeinterval!=null) {                 food.setTimeinterval(timeinterval);             }         }     }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Первый этап загрузки данных из электронной таблицы - создание объектов. \**  *\* Результат в картах foods, timeintervals, times.                         \**  *\** ***@param*** *wb - рабочая книга с данными.                                    \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  private static void loadStage1(XSSFWorkbook wb) {  *loadTime*(wb);  *loadFood*(wb);  *loadTimeInterval*(wb);         return;     }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Второй этап загрузки данных из электронной таблицы - создание связей между объектами. \**  *\* Результат в картах foods, timeintervals, times.                                       \**  *\** ***@param*** *wb - рабочая книга с данными.                                                  \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  private static void loadStage2(XSSFWorkbook wb) {  *loadFoodRelations*(wb);         return;     }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Загрузка данных из электронной таблицы.         \**  *\* Результат в картах foods, timeintervals, times. \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  private static void load() {         try (XSSFWorkbook wb = *openExcel*()) {             if (wb==null)                 return;  *loadStage1*(wb);  *loadStage2*(wb);         }         catch(Exception e) {             System.*out*.println("Формат файла не поддерживается!");             e.printStackTrace();             return;         }     }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Вывод данных в консоль.    \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  private static void outData() {         for (Map.Entry<UUID, CTime> entry : *times*.entrySet()) {             System.*out*.println(entry.getValue());         }     }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Создание заголовка для файла-отчёта.                       \**  *\** ***@param*** *document - заготовка файла-отчёта.                  \**  *\** ***@param*** *time - дата, по которой будет создавваться таблица. \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  private static void createTitle(             XWPFDocument document,             CTime time     ) {         // Создание параграфа.         XWPFParagraph par = document.createParagraph();         // Центрирование параграфа.         par.setAlignment(ParagraphAlignment.*CENTER*);         // Создание куска текста.         XWPFRun run = par.createRun();         // Установка содержимого текста.         run.setText("Расписание приема пищи на %s".formatted(time.getDate()));         // Жирность.         run.setBold(true);         // Шрифт.         run.setFontFamily("Times New Roman");         // Размер шрифта.         run.setFontSize(20);         return;     }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Создание верхней ячейки таблицы в файле-отчёте.            \**  *\** ***@param*** *row - положение строки(от 0).                       \**  *\** ***@param*** *pos - положение столбца(от 0).                      \**  *\** ***@param*** *text - текст, который будет находится в ячейке.     \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  private static void createHeaderCell(             XWPFTableRow row,             int pos,             String text) {         XWPFParagraph par;         XWPFRun run;         XWPFTableCell cell;         cell = row.getCell(pos);         par = cell.addParagraph();         par.setAlignment(ParagraphAlignment.*CENTER*);         par.setVerticalAlignment(TextAlignment.*BOTTOM*);         run = par.createRun();         // Установка содержимого текста.         run.setText(text);         // Жирность.         run.setBold(true);         // Шрифт.         run.setFontFamily("Times New Roman");         // Размер шрифта.         run.setFontSize(14);     }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Создание таблицы с едой в файле-отчёте.                    \**  *\** ***@param*** *document - заготовка файла-отчёта.                  \**  *\** ***@param*** *time - дата, по которой будет создавваться таблица. \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  private static void createTable(             XWPFDocument document,             CTime time     ) {         XWPFTable table = document.createTable(1,3);         table.setWidth(5\*1900);         // Создаём верх таблицы, первую строку         XWPFTableRow row = table.getRow(0);  *createHeaderCell*(row, 0, "Интервал");  *createHeaderCell*(row, 1, "Еда");  *createHeaderCell*(row, 2, "Калории");         LocalDate date;         String sDate;         // Создание строк с информацией по дате.         for (CFood food : time.getFoods()) {             row = table.createRow();             row.getCell(0).setText(food.getTimeinterval().getName()); // Добавляем интервал в первый столбец.             row.getCell(1).setText(food.getName()); // Добавляем название еды во второй столбец.             row.getCell(2).setText(String.*valueOf*(food.getCal())); // Добавляем каллории еды в третий столбец.         }         // Прокраска границ таблицы. Необходимость надо проверять в MS Word.         table.setBottomBorder(XWPFTable.XWPFBorderType.*SINGLE*, 8, 0, "000000");         table.setTopBorder(XWPFTable.XWPFBorderType.*SINGLE*, 8, 0, "000000");         table.setLeftBorder(XWPFTable.XWPFBorderType.*SINGLE*, 8, 0, "000000");         table.setRightBorder(XWPFTable.XWPFBorderType.*SINGLE*, 8, 0, "000000");         table.setInsideHBorder(XWPFTable.XWPFBorderType.*SINGLE*, 8, 0, "000000");         table.setInsideVBorder(XWPFTable.XWPFBorderType.*SINGLE*, 8, 0, "000000");     }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Создание дневника еды в формате электронного документа.    \**  *\** ***@param*** *time - дата, по которой будет создавваться таблица. \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  private static void createReport(CTime time) {         try(XWPFDocument document = new XWPFDocument()) {             // Заголовок.  *createTitle*(document, time);             // Таблица с датой.  *createTable*(document, time);             // Сохранение информации в файл.             File report = new File("output.docx");             try(FileOutputStream fos = new FileOutputStream(report)) {                 document.write(fos);             }             catch(IOException e) {                 System.*out*.println("Ошибка при записи файла на диск!");                 e.printStackTrace();             }         }         catch(IOException e)         {             System.*out*.println("Ошибка при сохранении данных в файл!");             e.printStackTrace();         }     }     //  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* Основная функция программы.                          \**  *\** ***@param*** *args - параметры вызова программы из консоли. \**  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  public static void main(String[] args) {         // Загрузка.  *load*();         // Фильтрация данных. Здесь возвращается первый попавшийся.         CTime t;         Iterator<CTime> it = *times*.values().iterator();         if (it.hasNext()) {             t = it.next();             // Построение карточки по выбранной дате.  *createReport*(t);         }         else {             System.*out*.println("В дневнике нет записей!");         }         // Вывод данных в консоль для проверки.  *outData*();     }  } |