Разбор вариант №1 для 5 лабораторной работы

Данные документы рассматривает выполнение 5 лабораторной работы для варианта 1, текст задания для данного варианта приведен.

Содержание

Текст задания

Реализовать консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме. В коллекции необходимо хранить объекты класса MusicBand, описание которого приведено ниже.

- Класс, коллекцией экземпляров которого управляет программа, должен реализовывать сортировку по умолчанию.
- Все требования к полям класса (указанные в виде комментариев) должны быть выполнены.
- Для хранения необходимо использовать коллекцию типа java.util.LinkedHashMap
- При запуске приложения коллекция должна автоматически заполняться значениями из файла.
- Имя файла должно передаваться программе с помощью: аргумент командной строки.
- Данные должны храниться в файле в формате json
  - Разбор и формирование файла заданного формата должны быть реализованы при помощи сторонних библиотек/пакетов
- Чтение данных из файла необходимо реализовать с помощью класса по вашему выбору
- Запись данных в файл необходимо реализовать с помощью класса по вашему выбору
- Все классы в программе должны быть задокументированы в формате javadoc и представлены на домашнем каталоге на сервере helios
- Программа должна корректно работать с неправильными данными (ошибки пользовательского ввода, отсутствие прав доступа к файлу и т.п.).

В интерактивном режиме программа должна поддерживать выполнение следующих команд:

- help : вывести справку по доступным командам
- info : вывести в стандартный поток вывода информацию о коллекции (тип, дата инициализации, количество элементов и т.д.)
- show : вывести в стандартный поток вывода все элементы коллекции в строковом представлении
- insert null {element} : добавить новый элемент с заданным ключом
- update id {element} : обновить значение элемента коллекции, id которого равен заданному
- remove_key null : удалить элемент из коллекции по его ключу
- clear : очистить коллекцию
- save : сохранить коллекцию в файл
- execute_script file_name : считать и исполнить скрипт из указанного файла. В скрипте содержатся команды в таком же виде, в котором их вводит пользователь в интерактивном режиме.
- exit : завершить программу (без сохранения в файл)
- remove_greater {element} : удалить из коллекции все элементы, превышающие заданный
- replace_if_lowe null {element} : заменить значение по ключу, если новое значение меньше старого
- remove_greater_key null : удалить из коллекции все элементы, ключ которых превышает заданный
- count_greater_than_description description : вывести количество элементов, значение поля description которых больше заданного
- filter_less_than_genre genre : вывести элементы, значение поля genre которых меньше заданного
- print_field_descending_number_of_participants : вывести значения поля numberOfParticipants всех элементов в порядке убывания

Формат ввода команд:

- Все аргументы команды, являющиеся стандартными типами данных (примитивные типы, классы-оболочки, String, классы для хранения дат), должны вводиться в той же строке, что и имя команды.
- Все составные типы данных (объекты классов, хранящиеся в коллекции) должны вводиться по одному полю в строку.
- При вводе составных типов данных пользователю должно показываться приглашение к вводу, содержащее имя поля (например, "Введите дату рождения:")
- Если поле является enum'ом, то вводится имя одной из его констант (при этом список констант должен быть предварительно выведен).
- При некорректном пользовательском вводе (введена строка, не являющаяся именем константы в enum'е; введена строка вместо числа; введённое число не входит в указанные границы и т.п.) должно быть показано сообщение об ошибке и предложено повторить ввод поля.
- Для ввода значений null использовать пустую строку.
- Поля с комментарием "Значение этого поля должно генерироваться автоматически" не должны вводиться пользователем вручную при добавлении.

Описание хранимых в коллекции классов:

```
public class MusicBand {
    private Integer id; //Поле не может быть null, Значение поля
должно быть больше 0, Значение этого поля должно быть уникальным,
Значение этого поля должно генерироваться автоматически
    private String name; //Поле не может быть null, Строка не может
быть пустой
    private Coordinates coordinates; //Поле не может быть null
    private java.time.ZonedDateTime creationDate; //Поле не может быть
null, Значение этого поля должно генерироваться автоматически
    private Integer numberOfParticipants; //Поле не может быть null,
Значение поля должно быть больше 0
    private Integer albumsCount; //Поле может быть null, Значение поля
должно быть больше 0
    private String description; //Поле может быть null
    private MusicGenre genre; //Поле может быть null
    private Album bestAlbum; //Поле не может быть null
}
public class Coordinates {
    private float x;
    private Integer y; //Поле не может быть null
}
public class Album {
    private String name; //Поле не может быть null, Строка не может
быть пустой
    private long length; //Значение поля должно быть больше 0
}
public enum MusicGenre {
    PROGRESSIVE_ROCK,
    HIP_HOP,
    PSYCHEDELIC_CLOUD_RAP,
    SOUL,
    POST_PUNK;
}
```

Комментарии по командам

Команда	Описание	Формат	Комментарии
help	вывести справку по доступным командам	команда используется без аргументов	вывести справку по доступным командам и формату их использования
info	вывести в стандартный поток вывода информацию о коллекции (тип, дата инициализации, количество элементов и т.д.)	команда используется без аргументов	
show	вывести в стандартный поток вывода все элементы коллекции в строковом представлении	команда используется без аргументов	
insert null {element}	добавить новый элемент с заданным ключом	имя_команды ключ_id, далее осуществляется запрос на ввод данных для нового элемента	если происходит добавление элемента, то все элементы, следующие за ним, должны изменить свой номер на 1, потому что номер уникальный. 2 одинаковых номеров быть не должно
update id {element}	обновить значение элемента коллекции, id которого равен заданному	имя_команды ключ_id, далее осуществляется запрос на ввод данных для нового элемента	
remove_key null	удалить элемент из коллекции по его ключу	имя_команды ключ_id	при удалении элемента должно произойти изменение номеров (id) других элементов
clear	очистить коллекцию	команда используется без аргументов	
save	сохранить коллекцию в файл	команда используется без аргументов	сохранение в тот файл, из которого коллекция была открыта
execute_script file_name	считать и исполнить скрипт из указанного файла. В скрипте содержатся команды в таком же виде, в котором их вводит пользователь в интерактивном режиме.	имя_команды имя_скрипта	
exit	завершить программу (без	команда используется	завершение програм-

Команда	Описание	Формат	Комментарии
	сохранения в файл)	без аргументов	мы
remove_greater {element}	удалить из коллекции все элементы, превышающие заданный	имя_команды, далее осуществляется запрос на ввод данных для нового элемента	при удалении можно руководствоваться значением/длиной любого другого поля элемента, например, числового
replace_if_lowe null {element}	заменить значение по ключу, если новое значение меньше старого	имя_команды ключ_id, далее осуществляется запрос на ввод данных для нового элемента	при замене можно руководствоваться значением/длиной любого другого поля элемента, например, числового
remove_greater_ key null	удалить из коллекции все элементы, ключ которых превышает заданный	имя_команды ключ_id	
count_greater_th an_description description	вывести количество элементов, значение поля description которых больше заданного	имя_команды значение_поля_descripti on	Поле description — это строка, можно сравнивать по длине строки, то есть надо вывести количество элементов, у которых длина строки поля description того, что указано в качестве аргумента
filter_less_than_g enre genre	вывести элементы, значение поля genre которых меньше заданного	имя_команды одно_из_значений-Enum	можно руководствоваться длиной значения или номером ENUM
print_field_desce nding_number_of _participants	вывести значения поля numberOfParticipants всех элементов в порядке убывания	команда используется без аргументов	

Как должно выглядеть ваше приложение

Запуск приложения
[helia@helios ~/java/lab1]$ java lab5 имя_файла

если файл не найден или недостаточно прав для его чтения или записи, необходимо вывести строку сообщения об произошедшей ошибке, далее либо выйти из приложения, либо предложить ввести новое имя файл. Во втором случае необходимо предусмотреть способ выхода из программы

введите команду > insert 1
введите имя музыкального коллектива >
введите координату X >
введите координату Y >
введите дату создания музыкального коллектива >
введите количество участников музыкального коллектива >
введите количество выпущенных альбомов музыкального коллектива >
введите описание музыкального коллектива >
выберите жанр музыкального коллектива 1 — PROGRESSIVE_ROCK, 2 — HIP_HOP, 3 — PSYCHEDELIC_CLOUD_RAP, 4 — SOUL, 5 — POST_PUNK, указав цифру или название жанра >
введите название лучшего альбома >
введите длительность звучания в минутах лучшего альбома >
Новый элемент был добавлен в коллекцию.

Рассмотрим вариант, если данные были введены некорректно.

введите команду > insert 1
введите имя музыкального коллектива >
введите координату X > ха-ха
Ошибка! Координата X – число с плавающей точкой
введите координату X >

ВАЖНО: подсказки могут быть любыми другими, но они должны понятны и очевидны для пользователя

Работа с файлы в Unix.

Данный раздел — небольшое отступление от основной темы, но данные знания вам будут необходимы при выполнении задания.

У каждого файла есть путь к нему и определенные для него права доступа. Путь может быть абсолютным и относительным.
Абсолютный (иначе полный) путь начинается с корня файловой системы:

для windows системы	Для Unix-систем (наш helios)
C:\Users\helia\Documents\java\Laba5.java	/home/teachers/helia/java/Laba5.java
Путь всегда начинается с имени диска	Путь всегда начинается с корня, иначе root, иначе /

Относительный указывает пусть относительно текущего месторасположения,

для windows системы	Для Unix-систем
(текущее расположение — C:\Users\helia\Documents)	(текущее расположение — /home/teachers/helia/)
java\Laba5.java	java/Laba5.java

У каждого файла есть свои права доступа к нему, которые определяют, что какие операции тот или иной пользователь может делать с файлом: читать, изменять, выполнять. Это права обозначаются буквами r – читать, w – изменять, модифицировать, записывать, x – исполнять. Группы пользователей разделены на 3 категории: владелец файла **u**ser, группа, к которой принадлежит владелец **g**roups и все остальные **o**ther, все они вместе называются **a**ll.
С файлами можно совершать следующие действия: открыть, прочитать, записать, закрыть, создать, удалить, поменять права доступа.
Теперь посмотрим, где и как мы можем увидеть права доступа к файлам.

Базовые команды интерпретатора

Права доступа

Начнем с самой известной команды

```
ls - посмотреть содержимого каталога
ls [ключи] [путь_к_каталогу]
```

Рассмотрим некоторые ключи для команды ls
a — выводит все файлы и каталоги, в том числе включая скрытые файлы (.)
l – обеспечивает расширенный вывод информации о содержимом каталога

```
ls -la
```

если для команды ls не указывается путь_к_каталогу, то выводится содержимое текущего каталога. Пример вывода команды ls

```
drwxr-xr-x   3 helia    teachers      11 окт. 28 15:43 .
drwxr-xr-x  22 helia    staff         35 нояб. 29 10:15 ..
-rw-r--r--   1 helia    teachers     694 нояб.  3 12:45 lab2.class
-rw-r--r--   1 helia    teachers     941 нояб.  3 12:52 Lab2.jar
-rw-r--r--   1 helia    teachers     275 окт. 27 13:52 lab2.java
```

Что мы видим в этом выводе, разберем по колонкам

1. права доступа к файлу/каталогу, вернемся к этому чуть позже
2. кол-во жестких ссылок на файл/каталог
3. имя владельца файла/каталога
4. имя группы владельца файла/каталога
5. размер файла

6. дата и время последнего изменения
7. имя файла
   ВАЖНО: файлов с именами . и .. Они означают следующее: . - указывает на текущий каталог, .. - на каталог уровня выше.

Права доступа к файлу выглядят следующим образом -[1]rw-[2]r--[3]r--[4] и d[1]rwx[2]r-x[3]r-x[4]

1	2	3	4
-	rw-	r--	r--
Указатель на тип файла d — каталог - — файл l – символическая ссылка	Права доступа для владельца файла, owner (u)	Права доступа для группы владельца файла group (g)	Права доступа для всех остальных other (o)
d	rwx	r-x	r-x
Это каталог	Владелец может читать, записывать, заходить в этот каталог	Пользователи группа, к которой принадлежит владелец может читать и заходить в этот каталог	Все остальные могут читать и заходить в этот каталог

Каждая из тетрад доступа кодируется 0 и 1, если право устанавливается, то устанавливается 1, в противном случае 0. Полные права, например, для владельца будет выглядеть так 111 или 7 в 8(16)-ричной системе исчисления.
Для изменения прав доступа к файлу используется команда chmod (change modification)
Формат использования команды

1. chmod цифровой_код имя_ файла
   chmod 666 lab2.java
2. chmod тип_пользователя-/+/=тип_права имя_файлы
   chmod u-w lab2.java
   chmod o+w lab2.java
   chmod u=rwx lab2.java
3. есть еще кое-что, но нам это пока не надо

Еще несколько полезных команд

```
mkdir - создать каталог.
mkdir путь_к_новому_каталогу

cd - сменить каталог.
cd [путь_к_каталогу]*[1]

cat – вывести содержимого файла на экран
cat путь_к_файлу
```

Если файл большой, и его вывод не помещается на 1 экран, то он все равно будет выведен полностью. Для вывода файла по страницам и по строкам можно использовать команды more и less

```
more путь_к_файлу
less путь_к_файлу
```

1 Если в формате команды используются квадратные скобки, это означает, что данный аргумент не является обязательным

Отредактировать файл

```
vim путь_к_файлу
vi путь_к_файлу
```

Скопировать файл

```
cp [ключи] откуда куда
```

Если копируем файлы, тот все просто, если необходимо скопировать каталог или даже дерево каталогов, то необходимо воспользоваться ключами, парочка полезных

- ключ R позволит скопировать директорию
- ключ f позволит при копировании отключить интерактивный режим, вам не будут задаваться вопросы типа, файл уже существует, перезаписать? Если файл существует, он будет удален и создан новый.

Удалить файл / каталог/ дерево каталогов

```
rm [ключи] путь к файлу/каталогу
```

Ключей, как водится, много, некоторые аналогичны указанным в cp.

Переместить (переименовать файл)

```
mv [ключи] откуда куда
```

Переменные окружения

Переменная среды или переменная окружения — текстовая переменная операционной системы, хранящая какую-либо информацию, например, данные о настройках системы или настоек пользователя.
Данные переменные используются как в ОС Unix, так и Windows, в часть из них совпадают и имеют одинаковое значение.
Например, переменная PATH указывает путь поиска исполняемых файлов, когда вы набираете какую-либо команду, то осуществляется последовательный поиск по указанным в переменной путям.

Как посмотреть значение конкретной переменной

```
[helia@helios ~/java/lab34]$ echo $PATH
/export/labs/eTester:/export/labs/lcheck:/usr/jdk/jdk1.8.0/bin:/opt/
glassfish3/bin:/usr/postgres/9.6-pgdg/bin/64:/usr/bin:/sbin:/usr/
sbin:/usr/ccs/bin:/usr/ucb:/usr/openwin/bin:/usr/dt/bin:/usr/sfw/
bin:/opt/sfw/bin:/usr/local/bin:/opt/csw/bin:/opt/SUNWut/bin:/opt/
oracle/product/11.2.0/dbhome_1/bin:/opt/oracle/product/11.2.0/
dbhome_1/sqldeveloper/sqldeveloper/bin
```

Обратим внимание на следующее

- названия переменных пишутся заглавными буквами
- доступ к переменной осуществляется через знак $
- переменная PATH содержит перечисление множества путей, разделенных знаком :

набирая в командной строке, например,

```
javac laba5.java
```

интерпретатор начинает последовательно искать `javac` в перечисленных в переменной PATH путях. Если указанная вами команда не найдена, то

```
[helia@helios ~/java/lab34]$ javak
-bash: javak: команда не найдена
```

Как посмотреть значения всех установленных переменных

```
[helia@helios ~/java/lab34]$ export
declare -x BLOCKSIZE="K"
declare -x HOME="/home/teachers/helia"
declare -x LANG="ru_RU.UTF-8"
declare -x LOGNAME="helia"
declare -x MAIL="/var/mail/helia"
declare -x MM_CHARSET="UTF-8"
declare -x OLDPWD="/home/teachers/helia/java"
declare -x
PATH="/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin:/
home/teachers/helia/bin"
declare -x PWD="/home/teachers/helia/java/lab34"
declare -x SHELL="/usr/bin/bash"
declare -x SHLVL="1"
declare -x SSH_CLIENT="192.168.64.32 64420 2222"
declare -x SSH_CONNECTION="192.168.64.32 64420 192.168.10.80 2222"
declare -x SSH_TTY="/dev/pts/66"
declare -x TERM="xterm"
declare -x USER="helia"
```

Как установить значение переменной

Добавим новое путь для переменной PATH

```
[helia@helios ~/java/lab34]$ PATH=$PATH:./
```
Сохраняем предыдущее значение переменной и добавляем новое значение

```
[helia@helios ~/java/lab34]$ export PATH
```
Делаем новое значение переменной активным

```
[helia@helios ~/java/lab34]$ export PATH=$PATH:/home/teachers/helia
```
Или сразу так

Инициализируем новую переменную

```
[helia@helios ~/java/lab34]$ CF=/home/teachers/helia/java/laba5/coll-csv
```
Создали новую переменную и присвоили ей значение

```
[helia@helios ~/java/lab34]$ export CF
```
Делаем новую переменную активной

```
[helia@helios ~/java/lab34]$ export CF=/home/teachers/helia/java/laba5/coll-csv
```
Или сразу так

ВАЖНО: эта переменная будет активна только на время вашей сессии, если есть необходимость постоянно использовать ее, то необходимо внести информацию о ней в конфигурационные файлы, например, .profile или .bash_profile. Конфигурационные файлы находятся в вашем домашнем каталоге /home/students/s123456/

```
export CF=/home/teachers/helia/java/laba5/coll-csv
```

Как прочитать значение переменной окружение из Java

Здесь нам на помощь приходит хорошо известный класс System, а именно его метод getenv (String name).

```
public static String getenv(String name)
```

который в качестве параметра принимает имя переменной, а возвращает строку с ее значением.

Как подготовить документацию

Документацию вашего проекта необходимо разместить на сервере helios. У каждого пользователя может быть своя страница по адресу

https://se.ifmo.ru/~helia/

где после знака ~ пишется имя пользователя, в вашем случае s123456.
Страница может быть, но по умолчанию ее нет, исправим это

1. в домашнем каталоге, для вас это /home/students/s*, создадим каталог
   `mkdir public_html`
2. когда идет обращение к https://se.ifmo.ru/~s* ищется и в дальнейшем отображается файл с именем index.html, создадим и его со следующим содержимым
   ```
   <HTML>
   <HEAD>
   <TITLE>Elena N. Blokhina</TITLE>
   </HEAD>
   <BODY>
   <H1>Hello world</H1><P>
   </BODY>
   </HTML>
   ```
3. Теперь при обращении к вашей странице вы увидите надпись
   `Hello world`

Теперь нам надо разместить там не просто абстрактную надпись, а нашу документацию. Рассмотрим на примере самой первой нашей лабораторной работы.
Создайте отдельный каталог, куда поместите 3 файла (Lab1.java, Hello.java, Morning.java). Это страница 15 в файле java_lab1.pdf

```
// файл Lab1.java
/** Основной класс программы */
public class Lab1 {
     public static void main(String[] args) {
          Morning h = new Morning();
          for (String s : args) {
               h.setName(s);
               h.say();
          }
     }
}
// файл Hello.java
/** Класс для приветствия типа «Привет» */
public class Hello {
     protected String name;
/** Если имя было не задано,
    то будет использоваться слово «мир». */
     public Hello() {
          name = "мир";
     }
     public void setName(String s) {
          name = s;
     }
/** вывод приветствия */
     public void say() {
          System.out.println("Привет, " + name + "!");
     }
}
// файл Morning.java
/** Класс для приветствия типа «Доброе утро!» */
public class Morning extends Hello {
     public void say() {
          System.out.println("Доброе утро, " + name + "!");
     }
}
```

Обратите внимание на формат комментариев.
Теперь воспользуемся уже знакомой утилитой javadoc, но немного откорректируем формат ее использования по сравнению с более ранним

```
javadoc -d ~/public_html/ *java
```

где ключ -d указывает на директорию, куда будет отправлена сгенеренная документация. Главным (первым) файлом документации будет index.html, поэтому при обращении к странице https://se.ifmo.ru/~s* вы увидите именно ее.
Обратите внимание на warnings, которые вернул javadoc, они сообщают о том, что не для всех методов есть описания, а значит документация неполная.

Общий алгоритм функционирования приложения

Схематично порядок выполнения нашего приложения можно представить следующим образом

1	ЗАПУСК ПРИЛОЖЕНИЯ	
1.1	работа с файлом	
2	ФОРМИРОВАНИЕ КОЛЛЕКЦИИ	
2.1	чтение из файла	
3	ИНТЕРПРЕТАТОР	И так много-много раз
3.1	ожидание ввод команды	
3.2	проверка валидности команды	
3.3	выполнение команды	
3.4	проверка валидности введенных данных	
3.5	возвращение результата и/или ошибки	
3.6	ПОЛУЧЕНИЕ РЕЗУЛЬТАТА	
4	ЗАВЕРШЕНИЕ РАБОТЫ ПРОГРАММЫ	
4.1	Формирование и запись файла JSON, csv, XML в зависимости от задания	

Рассмотрим по шагам выполнение программы, что нам надо знать и уметь для получения результата.

Как прочитать файлы разного формата

В перечне заданий фигурирует 3 разных варианта формата файлов, также отдельно оговорено, что разбор файлов, а также их запись должна осуществляться с использованием сторонних библиотек. Рассмотрим предлагаемые форматы файлов, их структуру, а также пакеты, которые можно использовать для их разбора.

про CSV, JSON и XML

Начнем с общего: это все текстовые файлы, в которых в том или ином виде хранятся данные.

CSV

Файл формата csv, пожалуй, самый простой из них, каждая строка которого — это отдельная строка таблицы, а столбцы отделены один от другого символами-разделителями. Это могут быть любые символы. Как правило используется запятая, но можно использовать и любой другой служебный символ.

Пример строки для нашей коллекции

```
Мираж,1,2,,8,5,советская и российская музыкальная группа в стиле
евродиско,PROGRESSIVE_ROCK,Звезды нас ждут,9
Браво,1,2,,8,13,советская и российская поп-рок-группа из
Москвы,PROGRESSIVE_ROCK,Московский бит,10
Beatles,2,4,,13,English rock band,PROGRESSIVE_ROCK,Help!,14
```

В качестве сторонней библиотеки можно использовать OpenCSV. Рассмотрим простейший вариант программы

```
import com.opencsv.*;
FileReader filereader = new
FileReader("skill.csv");
CSVReader csvReader = new
CSVReader(filereader);
String[] nextRecord;

while ((nextRecord =
csvReader.readNext()) != null) {
for (int i=0; i<9; i++) {

System.out.println(nextRecord[i]); }
```

Импортируем библиотеку
Здесь для простоты сразу указываем имя нашего csv файла.
И отдаем его на чтение

Объявили массив строк, в который будем читать наш файл по строкам
Читать будем до тех пор, пока не закончатся строки
Знаем, что в нашей строке 10 позиций, организуем цикл по выводу каждого элемента.

На выходе получаем следующее (показан вывод разбора только первой строки)

```
Мираж
1
2

8
5
советская и российская музыкальная группа в стиле евродиско
PROGRESSIVE_ROCK
Звезды нас ждут
9
```

Полезные ссылки
https://javadoc.io/doc/com.opencsv/opencsv/latest/index.html — почитать документацию
https://opencsv.sourceforge.net/ - скачать

XML

XML (eXtensible Markup Language) — расширяемый язык разметки. Используется для хранения и передачи данных.
В XML каждый элемент должен быть заключен в теги. Тег — это некий текст, обернутый в угловые скобки: <tag>
В любом XML-документе есть корневой элемент. Это тег, с которого документ начинается, и которым заканчивается. Значение элемента хранится между открывающим и закрывающим тегами. Это может быть число, строка, или даже вложенные теги! У элемента могут быть атрибуты — один или несколько. Их мы указываем внутри отрывающегося тега после названия тега через пробел в виде?
Правильно сформированный XML такой:

- Есть корневой элемент
- У каждого элемента есть закрывающийся тег
- Теги регистрозависимы
- Правильная вложенность элементов
- Атрибуты оформлены в кавычках

Пример файла XML на базе нашей предметной области

```
<?xml version="1.1" encoding="UTF-8" ?>
<MusicBandCatalog>
<MusicBand>
<name>Мираж</name>
<coordinates>
      <x></x>
      <y>5</y>
</coordinates>
<creationDate> </creationDate>
<numberOfParticipants>8</numberOfParticipants>
<albumsCount>5</albumsCount>
<description>советская и российская музыкальная группа в стиле
евродиско.</description>
<genre>POST_PUNK</genre>
<bestAlbum>
      <name>Звезды нас ждут</name>
      <length>9</length>
</bestAlbum>
</MusicBand>
</MusicBandCatalog>
```

Рассмотрим простейший вариант разбора такого файла.

```
import javax.xml.parsers.*;
import org.xml.sax.*;
import org.w3c.dom.*;
DocumentBuilder documentBuilder =
DocumentBuilderFactory.newInstance().
newDocumentBuilder();
Document document =
documentBuilder.parse("MusicBand.xml");
Node root = document.getDocumentElement();
NodeList musicbands = root.getChildNodes();
for (int i = 0; i < musicbands.getLength(); i+
+) {
Node music = musicbands.item(i);
```

Нам нужны будут сторонние библиотеки, подключаем их

Создаем построитель документов

Создаем дерево из файла

Находим корневой элемент
Смотрим все подэлементы
Организуем цикл по количеству подэлементов
Создаем ноду. Для нас нода — это музыкальный коллектив

```
if (music.getNodeType() != Node.TEXT_NODE) {
NodeList musicProps = music.getChildNodes();

for (int j = 0; j < musicProps.getLength(); j+
+) {
Node musicProp = musicProps.item(j);

if (musicProp.getNodeType() != Node.TEXT_NODE)
{
System.out.println(musicProp.getChildNodes().
item(0).getTextContent()); }
}
}
```

И проверяем, а какая она. Если она не текст, то разбираем

Создаем новый цикл количеству элементов внутри нашей ноды (музыкального коллектива)

Если нода нужного нам типа, то выводим значение

Полученный вывод

```
Мираж



8
5
советская и российская музыкальная группа в стиле евродиско
POST_PUNK
```

Однако, если внимательно посмотреть, то можно заметить, что у нас обработались не все элементы. Какие были проигнорированы?

Полезные ссылки

https://docs.oracle.com/javase/8/docs/api/ - документацию по нужным вам библиотекам, есть в стандартной документации

JSON

JSON (JavaScript Object Notation) – это текстовый формат представления данных в нотации объекта JavaScript.
JSON основан на двух структурах данных:

- Коллекция пар ключ/значение. В разных языках, эта концепция реализована как объект, запись, структура, словарь, хэш, именованный список или ассоциативный массив.
- Упорядоченный список значений. В большинстве языков это реализовано как массив, вектор, список или последовательность.

Пример для нашей коллекции

```
{
"name": "Мираж",
"coordinates": {
"x":1,
"y":2},
"creationDate":"",
"numberOfParticipants": 8,
"albumsCount": 5,
"description": "советская и российская музыкальная группа в стиле
евродиско",
"genre": PROGRESSIVE_ROCK,
"bestAlbum": {
"name": "Звезды нас ждут",
"lenght": 9
}
}
```

Как прочитать/разобрать и записать

- при помощи стандартной библиотеки Java
  - javax.json.* (Java EE) – это немного другая Java Enterprise Edition, мы используем для своих задач Standart Edition.
- или по помощи сторонних
  - GSON (ссылка на документацию ниже)
  - Jackson

Полезные ссылки

https://docs.oracle.com/javaee/7/api/javax/json/package-summary.html — документация
https://download.oracle.com/otndocs/jcp/json_p-1_1-final-spec/index.html — скачать
https://jsonformatter.curiousconcept.com/# - проверить, правильный ли у вас файл JSON

Что делать после того, как прочитали и разобрали файл

Обратите внимание, что после разбора файла любого формата вам возвращается тип String, из которого вам необходимо вернуть требуемый формат. Давайте, посмотрим, как это можно сделать.
Например, при разборе файла мы поместили полученные значения в строковый массив.

```
String[] nextRecord;
```

Рассмотрим на примере этого класса

```
public class Coordinates {
    private float x;
    private Integer y; //Поле не может быть null
}
```

Если с float вы уже точно знакомы, то вот Integer может вызвать некоторые вопросы. Это класс-оболочка, ноги которой растут из класса Number. Подробнее про класс-оболочки будет на лекции.
Итак, у нас есть переменная типа String. Как нам получить из него переменную другого типа

```
float c1 = Float.parseFloat(nextRecord [i]);
```

Разберем по частям

```
float c1 =        Float.         parseFloat          (nextRecord [i])
      1                  2                  3                  4
```

Объявляем переменную типа float (1) и сразу присваиваем ей значение, используя статический метод класса Float (2), в качестве параметра (4) указываем нашу строку. Метод вернет нам переменную нужного нам примитивного типа.
Кроме примитивного типа нам надо еще использовать и объекты другого типа. Сконвертируем строку в Integer

```
Integer c2 = Integer.parseInt(nextRecord [i]);
```

Здесь тоже аналогично примеру выше, используем статический метод класса Integer, возвращая нужным нам объект.

Вопрос на размышление

А что может случиться непредвиденного при разборе и попытке конвертирование из строки?
Подсказка
NullPointerException
NumberFormatException

Класс Scanner

А теперь сделаем огромный шаг назад и вспомним самое первое занятие, где осваивая азы мы здоровались со всеми. Немного модифицируем нашу программу и будем передавать параметры чуть иначе

```
import java.util.*;
public class ScReader {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.println("Как вас зовут?");


String  name = sc.next();


System.out.println("Hello " + name);
   }
}

[helia@helios ~/java/lab34]$ java ScReader
Как вас зовут?
Елена
Hello Елена
[helia@helios ~/java/lab34]$
```

Тут все знакомо и понятно

Создаем объект класса Scanner, указывая в конструкторе, что будем ожидать ввода данных от пользователя

Мы ожидаем, что пользователь введет свое имя, читаем первую строку до пробела

Выводим приветствие

При помощи класса Scanner можно не только прочитать введенные данные, но и разобрать их на блоки, разделенные пробелом.
В качестве источника данных можно использовать файл, поток ввода-вывода (Stream)

Автоматизация сборки

В первом семестре наши программы были достаточно просты, и нам в целом хватало командной строки, что собрать проект. Сейчас же уже очевидно, что сборка будет не так проста, появятся сторонние библиотеки, а может и не одна. Наш проект будет также содержать в себе несколько пакетов. А требование по демонстрации на сервере helios остается прежним. К тому же большинство из вас все равно будет использовать Idea для разработки и отладки. Как же легко и безболезненно перенести наш проект со своего ноутбука на сервер и заново его там собрать.

Здесь нам на помощь приходят инструменты для автоматизации сборки проектов. Сейчас их на рынке представлено несколько.

Для решения наших задач предлагается использовать Maven, который уже установлен на helios, и который есть в Idea.

Как его найти

```
which mvn
```

Вывод

```
/usr/local/bin/mvn
```

Это значит, что он уже прописан в вашей переменно окружения PATH

Получить информацию об установленном Maven

```
mvn -v
```

Вывод следующий

```
Apache Maven 3.8.4 (9b656c72d54e5bacbed989b64718c159fe39b537)
Maven home: /usr/local/share/java/maven
Java version: 17.0.2, vendor: OpenJDK BSD Porting Team, runtime:
/usr/local/openjdk17
Default locale: ru_RU, platform encoding: UTF-8
OS name: "freebsd", version: "13.0-stable", arch: "amd64", family:
"unix"
```

По команде mvn, функционал которой широк, есть подсказка

```
mvn -h
```

Порядок действий примерно следующий

создание проекта (все пишется в одну строку, для простоты восприятия разобьем ее на части

`mvn`	Имя команды
`archetype:generate`	Создаем проект из архетипа, далее указывая его параметры
`-D`	Указывает ключи
`-DgroupId=`**`Elena`**	Указываем разработчика, здесь буду я
`-DartifactId=`**`Lab5`**	Мой проект называется Lab5
`-DarchetypeArtifactId=maven-archetype-`**`quickstart`**	В качестве шаблона используем простой
`-DarchetypeVersion=`**`1.4`**	Указываем версию
`-DinteractiveMode=`**`false`**	Выключаем интерактивный режим

```
mvn archetype:generate -DgroupId=Elena -DartifactId=Lab5  -
DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.4
-DinteractiveMode=false
```

В результате создается иерархия каталогов, в котором

```
Lab5/src/main/java/Elena
```

будут находиться исходники.

Но нас в первую очередь интересует файл pom.xml, содержащий описание проекта, а также зависимости (сторонние библиотеки и пакеты). Фактически pom.xml является ядром проекта, прописав единожды свойства проекта и его зависимости компиляция, сборка проекта проходят легко и просто.

Дополнительное задание

Разобраться и рассказать на следующем занятии всей группе, как собрать проект при использовании Maven на основе 2 лабораторной работы (Покемоны) на сервере helios.

О чем-то еще рассказать?

Coming soon