МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики»

Факультет информационных технологий и программирования

Кафедра информационных систем

Лабораторная работа №1

INI файл

Выполнил студент группы М3201:

Дымчикова Аюна

САНКТ-ПЕТЕРБУРГ

2020

Задание

Создать инструмент для обработки конфигурационного INI файла.

Описать и реализовать необходимые классы, которые позволят производить обработку конфигурационного файла, который представляет собой текстовый файл, разделенный на СЕКЦИИ, которые содержат пары ИМЯ, ЗНАЧЕНИЕ.

Пример файла:

```
[COMMON]
StatisterTimeMs = 5000
LogNCMD = 1 ; Logging ncmd proto
LogXML = 0 ; Logging XML proto
DiskCachePath = /sata/panorama ; Path for file cache
OpenMPThreadsCount = 2

[ADC_DEV]
BufferLenSeconds = 0.65 ; Buffer length for ADC data in GPU memory, seconds.
SampleRate = 120000000.0 ; Sample rate of ADC.
Driver = libusb ; cypress / libusb / random / fileIQS

[NCMD]
EnableChannelControl = 1 ; Use or not CHG / CHGEXT commands
SampleRate = 900000.0 ; ANOTHER Sample Rate.
TidPacketVersionForTidControlCommand = 2

; TidPacket versions
; 0 - no packets
; 1 - header: data size, tid
; 2 - header: data size, tid, timestamp

[LEGACY_XML]
ListenTcpPort = 1976

[DEBUG]
PlentySockMaxQSize = 126
```

Все имена параметров и секций – это строки без пробелов, состоящие из символов латинского алфавита, цифр и знаков нижнего подчеркивания. Имена секций заключены в квадратные скобки, без пробелов. Значения параметров отделены от имен параметров знаком = (равенство)

Значения параметров могут быть одним из типов:

- целочисленным;
- вещественным;
- строковым: без пробелов, но в отличие от имени параметра может содержать также символ «точка».

Файл может содержать комментарии. Комментарием считается всё, что находится после знака «точка с запятой». Комментарии, как и сам знак «точка с запятой» должны быть проигнорированы.

Должны быть реализованы методы «получить значение определенного типа с таким-то именем из такой-то секции» (например, получить целое ListenTcpPort из секции LEGACY_XML)

Должны быть обработаны ошибки:

- Ошибка файловой подсистемы (например, если файл не найден);
- Ошибка формата файла (если файл имеет неверный формат);
- Неверный тип параметра (ошибка при приведении типа);

• Заданной пары СЕКЦИЯ ПАРАМЕТР нет в конфигурационном файле;
• и другие, при необходимости.

Ход рассуждений

В ходе проведенной работы была создана система классов, которые можно описать, начиная от меньшего к большему: класс параметра, класс секции, класс INI файла (Data), а также класс для парсера, осуществляющиего соединение этих классов.

При обработке файла класс парсера с помощью встроенного класса Scanner считывает строку одну за другой. Парсер разбивает строку на слова, которые поочередно проверяет на выполнение условий формата названия секции или параметра с использованием регулярных выражений. Считанные секции записываются в экземпляр класса Data, а параметры - в соответствующую секцию. При любом несоответствии выбрасывается исключение. Комментарии игнорируются и не записываются в экземпляр класса Data.

В результате работы функции parse класса Parser возвращается экземпляр класса Data со всеми данными файла, структурированными описанным выше способом.

Листинг

Файл Data.java

```
import java.util.HashMap;
import java.util.Map;

public class Data {
    Map<String, INISection> sections = new HashMap<>();

    public Data() {};

    public void put_section(String _name) {
        sections.put(_name, new INISection(_name));
    }

    public void put_parameter(String _section, String _name, String _value) {
        if(sections.containsKey(_section)) {
            sections.get(_section).put_parameter(_name, _value);
        }
    }



    public double tryGetDouble(String section, String parameter) throws  Exception{
        try {
            if(sections.containsKey(section)) {
                try {
                    return Double.parseDouble(sections.get(section).get_parameter(parameter).get_value());
                } catch (Exception e) {
                    throw new Exception(e.getMessage() + ", section: " + section);
                }
            } else {
                throw new Exception("No such section: " + section);
            }
        } catch (NumberFormatException e1) {
            throw new Exception("Type of " + parameter + " is not double");
        } catch (Exception e) {
            throw new Exception(e.getMessage());
        }
    }

    public int tryGetInt(String section, String parameter) throws  Exception{
        try {
            if(sections.containsKey(section)) {
                try {
                    return Integer.parseInt(sections.get(section).get_parameter(parameter).get_value());
                } catch (Exception e) {
                    throw new Exception(e.getMessage() + ", section: " + section);
                }
            } else {
                throw new Exception("No such section: " + section);
```

```
            }
        } catch (NumberFormatException e1) {
            throw new Exception("Type of " + parameter + " is not int");
        } catch (Exception e) {
            throw new Exception(e.getMessage());
        }
    }

    public String tryGetString(String section, String parameter) throws  Exception {
        if (sections.containsKey(section)) {
            try {
                return sections.get(section).get_parameter(parameter).get_value();
            } catch (Exception e) {
                throw new Exception(e.getMessage() + ", section: " + section);
            }
        } else {
            throw new Exception("There is no such section: " + section);
        }
    }
}
```

Файл INIParameter.java

```
public class INIParameter {
    private String name = "";
    private String value = "";

    public INIParameter() {};

    public INIParameter(String _name, String _value) {
        name = _name;
        value = _value;
    }

    public String get_name() {
        return name;
    }

    public String get_value() {
        return value;
    }
}
```

Файл INISection.java

```
import java.util.HashMap;
import java.util.Map;

public class INISection {
    private String name = "";
    private Map<String, INIParameter> parameters = new HashMap<>();

    public INISection() {};

    public INISection(String _name) {
        name = _name;
    }

    public INISection(String _name, INIParameter _parameter) {
        name = _name;
        parameters.put(_parameter.get_name(), _parameter);
    }

    public String get_name() {
        return name;
    }

    public INIParameter get_parameter(String _name) throws Exception {
        if(parameters.containsKey(_name)) {
            return parameters.get(_name);
        } else {
            throw new Exception("There is no such parameter: " + _name);
        }
    }

    public void put_parameter(INIParameter _parameter) {
        parameters.put(_parameter.get_name(), _parameter);
    }

    public void put_parameter(String _name, String _value) {
        parameters.put(_name, new INIParameter(_name, _value));
    }
}
```

Файл Main.java

```
public class Main {
    public static void main(String[] args) {

        Parser parser = new Parser();
        try {
            Data data = parser.parse("input.txt");
            System.out.println(data.tryGetDouble("Common", "StatisterTimeMs"));
            //System.out.println(data.tryGetInt("ADC_DEV", "BufferLenSeconds"));
            System.out.println(data.tryGetString("ADC_DEV", "Driver"));
            System.out.println(data.tryGetInt("LEGACY_XML", "ListenTcpPort"));
            System.out.println(data.tryGetDouble("LEGACY_XML", "ListenTcpPort"));
            System.out.println(data.tryGetString("DEBUG", "PlentySockMaxQSize"));
            //System.out.println(data.tryGetString("SMTH", "IDK"));
            System.out.println(data.tryGetString("LEGACY_XML", "Port"));
            System.out.println(data.tryGetString("NEW0", "Smth"));

        } catch (Exception e) {
            System.out.println("Exception: " + e.getMessage());
        }
    }
}
```

Файл Parser.java

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class Parser {

    public Parser() {};

    public Data parse(String name) throws Exception {
        File file = new File(name);
        Scanner scanner = null;
        Data data = new Data();
        try {
            scanner = new Scanner(file);
            String lastSection = "";
            while (scanner.hasNextLine()) {
                String[] line = scanner.nextLine().strip().split(" ");
                String last_param = "";
                boolean flag = false;
                for(String word : line) {
                    if(word.length() == 0 || word.charAt(0) == ';') {
                        break;
                    } else if(word.charAt(0) == '[') {
                        if(check_name(word.substring(1, word.length() - 1)) && word.charAt(word.length() - 1) == ']') {
                            lastSection = word.substring(1, word.length() - 1);
                            data.put_section(lastSection);
                        } else {
                            throw new Exception("Wrong name: " + word);
                        }
                    } else if(!flag && check_name(word)) {
                        last_param = word;
                        flag = true;
                    } else if (flag && word.contentEquals("=")) {
                        continue;
                    } else if(flag && check_value(word) && !last_param.isEmpty()) {
                        data.put_parameter(lastSection, last_param, word);
                        flag = false;
                    } else {
                        throw new Exception("Wrong parameter: " + word);
                    }

                }
            }
        } catch (FileNotFoundException e) {
            throw new Exception("File was not found");
        } catch (Exception e1) {
            throw new Exception(e1.getMessage());
        } finally {
            scanner.close();
```

```
        }
        return data;
    }
    private boolean check_name(String _name) {
        if(!_name.matches("(\\d|\\w)+")) {
            return false;
        }
        return true;
    }
    private boolean check_value(String _value) {
        if(!_value.matches("(\\d|\\w|\\.)+")) {
            return false;
        }
        return true;
    }
}
```