Json Parser

Изготвил: Йово Неделчев

От: Катедра СИТ

2Курс 1А Група

Студентски номер:23621691

**1.Задача на проекта :**

JSON е популярен текстов формат за описване на данни. Да се напише програма, която работи с файлове в такъв формат.

След като приложението отвори даден файл, то трябва да може да извършва посочените по-долу операции, в допълнение на общите операции (open, close, save, save as, help и exit):

Validate : Да се направи проверка дали отворения файл е валиден спрямо синтаксиса на JSON. Ако има някакъв проблем, програмата трябва да съобщи максимално ясно какво и къде не е наред, така че потребителят да може да го поправи.

Print : Да се изведе съдържанието на обекта в максимално четим вид.

Search <key> : Да се провери дали в обекта се съдържат данни, записани под този ключ и ако да – да изведе списък от всички такива данни.

Set <path> <string> : При подаване на пълен път <path> към даден елемент, да се замени стойността на посочения елемент с нов обект, получен от низа <string> според синтаксиса на JSON стойност, ако такъв елемент съществува и е единствен и ако символният низ е коректен. В противен случай трябва да обяви каква е грешката.

Create <path> <string> : При подаване на пълен път <path> към даден елемент да се добави такъв елемент, получен от низа <string> според синтаксиса на JSON стойност. Ако такъв елемент съществува, това трябва да се обяви като грешка. В случай, че последните елементи от пътя не съществуват, да се създадат. Ако символният низ или пътят не са коректни, да се съобщи с подходящо описание на грешката.

Delete <path> : При подаване на пълен път <path> до елемент, да се изтрие, ако такъв съществува или да съобщи на потребителя при некоректен път.

Move <from> <to> : Всички елементи, намиращи се на път <from> да бъдат преместени на пътя <to>.

Save [<path>] / Saveas <file> [<path>] : Командите save и saves да работят с произволен път, като записват в текущия или в нов файл обекта на дадения път, ако съществува. Ако <path> не е подаден, да се записва целият обект, който в момента е зареден в паметта.

**2.Съдържание в програмата**

2 Папки за Json файлове : JSON\_Files и FilesFolder2

1 Абстрактен клас - BaseWriter и 5 класа , които го наследяват :

Basic Writer , ContextExtendedWriter , FullWriter , TwoFileWriter и LocationExtendedWriter

1 клас за запазване на временна информация за файловете , докато програмата работи – PathReference

1 клас със Main метод и главната логика на програмата – JsonReader

**3.Код**

**3.1. JsonReader клас**

Импортнати пакети :

import java.io.IOException;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.util.\*;

java.io.IOException позволява работата с изключения , java.util.\* позволява работата с колекциите като List<> ,ArrayList<> и Map<>,

а java.nio.file.Files/Path/Paths позволяват работата с файлове .

open функция:

public static void open(PathReference path , String fileName,List<PathReference>list) throws IOException {  
  
 Boolean ExistsInList = false;  
  
 if(path.getLocation() == null) {  
 for(PathReference pr : list){  
 if(pr.getLocation().equals(Paths.*get*("src\\" + fileName))){  
 path.setLocation(pr.getLocation());  
 path.setContent(pr.getContent());  
 ExistsInList = true;  
 System.*out*.println("Файлът е успешно отворен");  
 }  
 }  
 if(!ExistsInList) {  
 if (Files.*exists*(Paths.*get*("src/" + fileName)) && Files.*isReadable*(Paths.*get*("src/" + fileName))) {  
 try {  
 path.setLocation(Paths.*get*("src/" + fileName));  
 path.setContent(Files.*readString*(path.getLocation()));  
 //String content = Files.readString(path.location); // Opens and reads the file  
 System.*out*.println("Файлът е успешно отворен");  
 //System.out.println("Съдържание:\n" + content);  
 } catch (IOException e) {  
 System.*out*.println("Файлът съществува , но не беше успешно отворен");  
 }  
 } else {  
 System.*out*.println("Файлът не съществува или не може да бъде прочетен");  
 return;  
 }  
 }  
 }  
 else{  
 System.*out*.println("Дръг файл е отворен . Затворете отвореният файл , за да може да отворите друг файл.");  
 return;  
 }  
   
}

Функцията взима като аргументи елементи : PathReference path , където ще се съхраняват данните за отворения файл в отделен клас , String fileName , който е името на локацията на файла и List<PathReference> list , в който се съдържат всичките PathReference елементи направени , докато програмата работи.

За начало се създава локална променлива Boolean ExistsInList , със стойност false . След това се проверява дали локацията във path елемента е празна , и ако е , се влиза във for цикъл на всички елементи от list елемента и се проверява дали там има елемент с еднаква локация , и ако има , path елемента взима стойностите си за локация и съдържание от този съществуващ елемент , след което ExistsInList става на true и се изписва съобщение в конзолата „Файлът е успешно отворен“. При случай , че няма елемент с такава локация в list , се проверява дали съществува такъв четим файл в проекта . Ако съществува , то тогава path елемента взима за стойност дадената локация fileName чрез Paths.Get(“src/”+fileName) и взима съдържанието на файла чрез Files.readString(path.getLocation()) , след което се изписва съобщение в конзолата „Файлът е успешно отворен“. Ако файлът не може да бъде успешно отворен се изписва на конзолата съобщенитео „Файлът съществува , но не беше успешно отворен“ , а ако файлът не съществува или не може да бъде прочетен се изписва на конзолата съобщенитео „Файлът не съществува или не може да бъде прочетен“. В случай , че друг файл е вече отворен (което ще стане ясно как работи в main метода) се изписва на конзолата съобщение „Друг файл е вече отворен . Затворете отвореният файл , за да можете да отворите друг файл.“.

close функция :

public static void close(PathReference path , String fileName){  
 if(path.getLocation()!=null) {  
 if (path.getLocation().equals(Paths.*get*("src\\" + fileName))) {  
 path.setLocation(null);  
 path.setContent(null);  
 System.*out*.println("Файлът е успешно затворен");  
 } else {  
 //System.out.println("Given path: " + path.location);  
 //System.out.println("Expected path: "+"src\\JSON\_Files\\" + fileName);  
 System.*out*.println("Няма отворен файл с такова наименование");  
 }  
 }  
 else{  
 System.*out*.println("Няма отворен файл с такова наименование");  
 }  
  
}

Функцията взима като аргументи елементи : PathReference path, който представлява отвореният файл в програмат и String fileName , който представлява дадена име на локация на файл.

Първо се проверява дали локацията във path елемента е празна , ако не е , се проверява дали локацията е същата като тази на даденото име на локация fileName . В слчай , че са еднакви локации , локацията и съдържанието на path елемента се премахват като им се даде стойност null и след това се изписва съобщение на конзолата „Файлът е успешно затворен“. В случай , че не се едакви локациите , тогава се изписва съобщение на конзолата „Няма отворен файл с такова наименование“. В случай , че локацията на дедения елемент path е празна се изписва съобщение на конзолата „Няма отворен файл с такова наименование“.

print функция :

public static void print(PathReference path){  
 if(path.getContent()==null || path.getLocation()==null){  
 System.*out*.println("Няма отворен файл все още");  
 }  
 else {  
 try {  
 //System.out.println(path.content);  
 System.*out*.println(Files.*readString*(path.getLocation()).trim());  
 } catch (IOException e) {  
 System.*out*.println("Грешка при изписването на съдържанието на отвореният файл");  
 }  
 }  
}

Взима като аргумент елемент от тип PathReference . Ако съдържаниете или локацията на елемента са празни (те се попълват , когато се извика open функцията) , тогава се изписва в конзолата съобщението „Няма отворен файл все още“. Ако те съществуват се изписва съдържанието на файла (чете се директно от локацията на файла вместо съдържанието на елемента , защото в комбинация с другите методи в програмата може да има разминавания при променяне и изтриване на файловете). При изключение се изпизва в конзолата съобщението „Грешка при изписването на съдържанието на отвореният файл“.

parseJson функция :

public static Map<String, String> parseJson(String json) {  
 Map<String, String> map = new LinkedHashMap<>();  
  
 json = json.trim().replaceAll("[{}\"]", "");  
 String[] pairs = json.split(",");  
  
  
 for (String pair : pairs) {  
 String[] keyValue = pair.split(":");  
 if (keyValue.length == 2) {  
 map.put(keyValue[0].trim(), keyValue[1].trim());  
 }  
 }  
 return map;  
}

Функцията връща Map<String,String> и приема String като аргумент. Първо се създава локална променлива от тип LinkedHashMap<String,String> map . След това чрез json.trim() се махат всички скоби от JSON файла , който беше подаден като аргумент и се създава масив от тип String[] , в който се пазят данните от JSON файла. След това чрез for цикъл двойките от данни се поставят в map-a след като бъдат преместени във още един масив от тип String[] , който ги разделя на 2те части ключ и стойност. На края се връща map-a с новите стойности.

search функция :

public static void search(PathReference path , String keyName) throws IOException {  
 if(path==null || path.getLocation()==null){  
 System.*out*.println("Няма Отворен файл");  
 return;  
 }  
 Boolean printed = false;  
 String content = new String(Files.*readAllBytes*(path.getLocation()));  
 Map<String, String> jsonMap = *parseJson*(content);  
 for (String key : jsonMap.keySet()) {  
 if(keyName.equals(key))  
 {  
 String value = jsonMap.get(keyName);  
 System.*out*.println(key+ ": "+value);  
 printed = true;  
 }  
 }  
 if(!printed){  
 System.*out*.println("Не бешен намерен ключ с такова име");  
 }  
}

Функцията взима за аргументи 2 елемента от тип PathReference и String и може да хвърли IOException изключение. Тя служи за намиране на двойка данни от JSON файл по името на ключа.Първо се проверява дали path елемента е валиден , и ако не е , се изписва на конзолата съобщение „Няма отворен файл“ и се излиза от функцията. Създават се локални променливи : Boolean printed , String conent и Map<String,String> jsonMap . printed е за определяне дали успешно е намерен ключ с даденото име , при което получава стойност true. content взима съдържанието на файла с локация равна на path елемента. JsonMap извиква функцията parseJson и дава като аргумент content. След това с for цикъл се взимат всички ключове от jsonMap , и ако те за равни на keyName , то тогава се създава локална променлива String value , която взима стойноста от двойката ,като извиква функцията jsonMap.get(keyName) и printed получава стойност true. В случай , че не е намерен ключ с такова наименование ,printed ще си остане със стойност false и ще се изпише на конзолата „Не беше намерен ключ с такова име“.

set функция :

public static void set(Path filePath , String newContent,List<PathReference> list)  
{  
 /\*if(jsonMap.containsKey(keyName)){  
 jsonMap.put(keyName,newValue);  
 }  
 else {  
 System.out.println("Ключ с име : '" + keyName + "' не беше намерен.");  
 }\*/  
 PathReference pathRef = new PathReference(filePath);  
 if (Files.*exists*(filePath)) {  
 /\*try {\*/  
 for(PathReference pr : list) {  
 if (pr.getLocation().equals(filePath)) {  
 pr.setContent(newContent);  
 System.*out*.println("JSON файлът беше пренаписан успешно.");  
 return ;  
 }  
 }  
 //Files.write(filePath, newContent.getBytes());  
 pathRef.setContent(newContent);  
 list.add(pathRef);  
 System.*out*.println("JSON файлът беше пренаписан успешно.");  
 } /\*catch (IOException e) {  
 System.out.println("Грешка при презаписването на JSON файлът.");  
 }\*/  
 //}  
 else{  
 System.*out*.println("JSON файлът с такъв път не е намерен.");  
 }  
}

Функцията приема като аргументи елементи Path filePath , който представлява локацията на файл , String newContent , който представлява новото съдържание , което ще бъде написано във файла и List<PathReference> list , в който се съдържат всичките PathReference елементи направени , докато програмата работи.

Първо се създава нова локална променлива PathReference pathRef , коята взима като стойност за локацията си елемента filePath. След това се проверява дали съществува файл с такава локация в проекта , и ако съществува се влиза във for цикъл , който минава през всички елементи на list , и ако намери елемент с локация равна на filePath , променя съдържанието на този елемент на newContent и се изписва съобщение на конзолата „JSON файлът беше пренаписан успешно.“ и се излиза от функцията. Ако няма елемент с еднаква локация , тогава съдържанието на pathRef локалната променлива се променя на newContent и в list се добавя pathRef елемента и се изписва съобщение на конзолата „JSON файлът беше пренаписан успешно.“. Ако няма файл с дадената локация в проекта изписва съобщение на конзолата „JSON файлът с такъв път не е намерен.“.

save функция :

public static void save(String fileName,List<PathReference> list){  
 Path filePath = null;  
 String fileContent = null;  
 for(PathReference pr : list){  
 if(pr.getLocation().equals(Paths.*get*("src\\" + fileName))){  
 filePath = pr.getLocation();  
 fileContent = pr.getContent();  
 }  
 }  
 if (filePath != null && fileContent != null) {  
 try {  
 Files.*write*(filePath, fileContent.getBytes());  
 System.*out*.println("Промените бяха запазени във файла.");  
 } catch (IOException e) {  
 System.*out*.println("Грешка при запазването на файла.");  
 }  
 } else {  
 System.*out*.println("Няма файл за запазване.");  
 }  
}

Функцията взима като аргументи елементи : String fileName , който е името на локацията на файл и List<PathReference> list , в който се съдържат всичките PathReference елементи направени , докато програмата работи.

Създават се 2 локални променливи Path filePath и String fileContent , и двете със стойност null . Влиза се във for цикъл , който преминава през всички елементи от list , и ако има елемент с локация равна на локацията с име fileName , тогава filePath и fileContent взимат стойностите си от локацията и съдържанието на елемента. След това се проверява дали filePath и fileContent не са null , и ако не са , тогава във файл с локация равна на filePath се записва съдържанието от fileContent чрез метода Files.write(filePath, fileContent.getBytes()) . След това се изписва съобщение в конзолата „Промените бяха запазени във файла.“. При върнато изключение IOException се изписва съобщение в конзолата „Грешка при запазването на файла.“. В случай , че filePath или fileContent са равни на null , се изписва съобщение в конзолата „Няма файл за запазване.“.

saveAs функция :

public static void saveAs(String fileName,String newPathName,List<PathReference> list) throws IOException {  
 Path filePath = Paths.*get*("src\\"+newPathName);  
 Path originalLocation = Paths.*get*("src\\" + fileName);  
 String fileContent = Files.*readString*(originalLocation);  
 for(PathReference pr : list){  
 if(pr.getLocation().equals(originalLocation)){  
 //filePath = pr.location;  
 fileContent = pr.getContent();  
 }  
 }  
 if (filePath != null && fileContent != null) {  
 try {  
 *delete*(originalLocation,list,false);  
 Files.*write*(filePath, fileContent.getBytes());  
 System.*out*.println("Промените бяха запазени във файла.");  
 } catch (IOException e) {  
 System.*out*.println("Грешка при запазването на файла.");  
 }  
 } else {  
 System.*out*.println("Няма файл за запазване.");  
 }  
}

Функцията взима като аргументи елементи : String fileName , който е името на локацията на файл , String newPathName който служи като името на новата локация на файл и List<PathReference> list , в който се съдържат всичките PathReference елементи направени , докато програмата работи.

Създават се 2 локални променливи Path filePath и Path originalLocation , които представляват оригиналната локацията на дадения файл и новата локация на дадения файл , като използват fileName и newPathName .