КУРСОВ ПРОЕКТ ПО ООП 1 ЧАСТ

Тема 12: Календар

Михаил Ивелинов Добрев

Фак. Номер: 20621610

Увод:

Описание

Java приложение в командния ред, което позволява създаването на резервации, тяхното модифициране и запазване в XML файлове. Приложението позволява синхронизирането на два или повече календара.

Цел

Да се създаде интерфейс, който позволява на потребителя да отвори или създаде календар и да може да го модифицира. Това включва добавяне и премахване на резервации, тяхното модифициране, тяхното показване на екрана, добавяне на почивни дни, изваждане на нужни статистики и комбинирането на няколко календара в един. Също така да се предостави и функционалност за работа с повече от един файл, като се добави функционалност за затваряне на текущия файл и отваряне на нов, запазване на текущия календар в нов файл. Трябва да се добави и начин за изписване на всички приемани от програмата команди и инструкции за тяхната употреба.

Съдържание

1. Преглед на предметната област
   1. Основни проблеми на задачата и подходи за решението им
2. Проектиране и реализация
   1. Структура на проекта
   2. Реализация
3. Тестване
   1. Тестови сценарии
4. Заключение
   1. Обобщение
   2. Насоки за бъдещето

Преглед на предметната област

Основни проблеми на задачата и подходи за решението им

Първият проблем в задачата са файловете в които ще се запазват данните. Избрано е да се използват *XML* файлове. Има голям избор от библиотеки в Java за реализацията, но за този проект е избрана *STAX*. Тя се води от събития, което позволява на програмата да обхожда файловете без да ги зарежда целите в паметта си, което е причината защо тя е избрана.

Вторият е проблем е „В какви структури трябва да се пази отвореният календар по времето на изпълнение?“. В началото на задачата за запазване на резервациите в календара се използваше *Set*, но в по-късните нива на разработка той бе заменен от *Map*, който има локална дата като ключ и *Set* от резервации като стойност. Това позволява достъпването на резервации за дадена дата да бъде много по-лесно и бързо. Цената за тази оптимизация е факта, че обновяването на датата има по-голяма сложност, но в този проект броят на четения е доста по-голям от броят на нужди за обновяване на данни.

Проектиране и реализация

Структура на проекта

Има два абстрактни класа за обработка на *XML* файлове. *XmlReader*, който отговаря за четенето на файла и запазването в оперативната памет и *XmlWriter*, който записва данните за календар в оперативната памет във файл.

За запазване на календар в оперативната памет отговаря класът Calendar, който има двa атрибута. *Set* от почивните дни и *Map* с *LocalDate* за ключ и *Set* от резервации като стойности. За създаване, изтриване и модифициране на атрибутите отговаря абстрактния клас *CalendarWriter*, а за четене *CalendarReader*. Всяка резервация представлява *Booking* обект.

Класът *CLI* представлява имплементацията на команден ред. Той е *Singleton* понеже в една инстанция на програмата е нужен само едни команден ред. Съдържа функция *run* която представлява безкраен цикъл, който чете входните данни от потребителя, почиства ги, препраща ги на *Calendar* класа да за обработка и изважда изхода.

Реализация

1. *XmlReader*: Има една функция, която приема път към файл и връща Calendar. Кодът от *фиг 1.* се изпълнява докато не се достигне края на *XML* файла. Кода реагира на всеки отварящ таг и ако той отговаря на някой от посочените в условната конструкция *switch* се добавя в съответното поле на класа в който се намира данната

XMLEvent nextEvent = reader.nextEvent***()***;  
**if *(***nextEvent.isStartElement***()) {*** StartElement startElement = nextEvent.asStartElement***()***;  
 **switch *(***startElement.getName***()***.getLocalPart***()) {* case "booking"** -> booking = **new** Booking***()***;  
 **case "name"** -> ***{*** nextEvent = reader.nextEvent***()***;  
 booking.setName***(***nextEvent.asCharacters***()***.getData***())***;  
 ***}* case "note"** -> ***{*** nextEvent = reader.nextEvent***()***;  
 booking.setNote***(***nextEvent.asCharacters***()***.getData***())***;  
 ***}* case "startTime"** -> ***{*** nextEvent = reader.nextEvent***()***;  
 booking.setStartTime***(***LocalTime.*parse****(***nextEvent.asCharacters***()***.getData***()))***;  
 ***}* case "endTime"** -> ***{*** nextEvent = reader.nextEvent***()***;  
 booking.setEndTime***(***LocalTime.*parse****(***nextEvent.asCharacters***()***.getData***()))***;  
 ***}* case "date"** -> ***{*** nextEvent = reader.nextEvent***()***;  
 date = LocalDate.*parse****(***nextEvent.asCharacters***()***.getData***())***;  
 ***}* case "holiday"** -> ***{*** nextEvent = reader.nextEvent***()***;  
 holidaySet.add***(***LocalDate.*parse****(***nextEvent.asCharacters***()***.getData***()))***;  
 ***}  
 }  
}*if *(***nextEvent.isEndElement***()) {*** EndElement endElement = nextEvent.asEndElement***()***;  
 **if *(***endElement.getName***()***.getLocalPart***()***.equals***(*"booking"*)) {*** Set***<***Booking***>*** bookingSet;  
 **if *(***bookings.containsKey***(***date***)) {*** bookingSet = **new** HashSet***<>(***bookings.get***(***date***))***;  
 bookingSet.add***(***booking***)***;  
 ***}* else** bookingSet = Set.*of****(***booking***)***;  
 bookings.put***(***date, bookingSet***)***;  
 date = **null**;  
 ***}  
}***

*Фиг 1. Извадка от XmlReader*

1. *XmlWriter*: Има една функция, която приема Calendar и път към файл. За всяка резервация се изпълнява кода от *фиг 2.* и за всеки почивен ден този от *фиг 3.*

writer.writeStartElement**("booking")**;  
writer.writeStartElement**("date")**;  
writer.writeCharacters**(**date.toString**())**;  
writer.writeEndElement**()**;  
writer.writeStartElement**("name")**;  
writer.writeCharacters**(**booking.getName**())**;  
writer.writeEndElement**()**;  
writer.writeStartElement**("note")**;  
writer.writeCharacters**(**booking.getNote**())**;  
writer.writeEndElement**()**;  
writer.writeStartElement**("startTime")**;  
writer.writeCharacters**(**booking.getStartTime**()**.toString**())**;  
writer.writeEndElement**()**;  
writer.writeStartElement**("endTime")**;  
writer.writeCharacters**(**booking.getEndTime**()**.toString**())**;  
writer.writeEndElement**()**;  
writer.writeEndElement**()**;

*фиг 2. Извадка от XmlWriter отговаряща за записването на резервация*

writer.writeStartElement**("holidays")**;  
calendar.getHolidaySet**()**.forEach**(**holiday -> **{  
 try {** writer.writeStartElement**("holiday")**;  
 writer.writeCharacters**(**holiday.toString**())**;  
 writer.writeEndElement**()**;  
 **} catch (**XMLStreamException e**) {** e.printStackTrace**()**;  
 **}  
})**;  
writer.writeEndElement**()**;

*фиг 3. Извадка от XmlWriter отговаряща за записването на почивен ден*

1. *Calendar*: Съдържа резервациите и почивните дни и ги обработва като си делегира функциите към *CalendarReader* и *CalendarWriter* *фиг 4.*

**private final** Set***<***LocalDate***>* holidaySet** = **new** HashSet***<>()***;  
**private final** Map***<***LocalDate, Set***<***Booking***>>* bookings** = **new** HashMap***<>()***;

*фиг 4. Атрибути на клас Calendar*

1. *Booking*: Клас който репрезентира една резервация *фиг 5.*

**private** LocalTime **startTime**;  
**private** LocalTime **endTime**;  
**private** String **name**;  
**private** String **note**;

*фиг 5. Атрибути на клас Booking*

1. *CalendarReader*: Клас отговарящ за четене на резервациите и почивните дни.
   1. *List<Booking> agenda(Calendar calendar, LocalDate date)* Метода проверява дали датата съществува в календара. Ако съществува връща сортиран списък от резервациите на тази дата, ако не връща null
   2. *Set<Booking> findBookings(Calendar calendar, String searchString)* Метода връща Set от всички резервации, чието име отговаря нa *searchString*
   3. *Map<DayOfWeek, Integer> weekdayAnalytics(Calendar calendar, LocalDate startDate, LocalDate endDate)* Метода връща статистика за натовараемост за всеки ден от седмицата *фиг 6.*

calendar.getBookings***()*** .entrySet***()*** .stream***()*** .filter***(***entry -> entry.getKey***()***.isAfter***(***startDate***)*** && entry.getKey***()***.isBefore***(***endDate***))*** .collect***(***Collectors.*toMap****(***entry -> entry.getKey***()***.getDayOfWeek***()***, entry -> entry.getValue***()***.size***()))***;

*фиг 6. Тяло на метода weekdayAnalytics*

* 1. *LocalDateTime findSlot(Calendar calendar, LocalDate fromDate, int hours)* Метода проверява за свободни резервации след подадената дата с продължителност hours *фиг 7.*

**for *(***Booking booking : bookings***) {*** LocalTime endTime = booking.getEndTime***()***;  
 **if *(***toTime.toLocalTime***()***.isBefore***(***booking.getStartTime***())*** || toTime.toLocalTime***()***.equals***(***booking.getStartTime***())*** || fromTime.toLocalTime***()***.isAfter***(***endTime***)*** || fromTime.toLocalTime***()***.equals***(***endTime***))* return** fromTime;  
  
***}*** *фиг 7. Извадка от findSlot метода. Проверката дали дадено време е свободно на дадена дата.*

* 1. *boolean checkIfAllElementsEquals(List<LocalDateTime> times)* Метод който проверява дали всички елементи на листа са еднакви *фиг 8.*

**private static boolean** checkIfAllElementsEquals***(***List***<***LocalDateTime***>*** times***) {* return** times != **null** && times.stream***()***.allMatch***(***time -> time.equals***(***times.get***(***0***)))***;  
***}***

*фиг 8. Функцията checkIfAllElementsEquals*

* 1. *LocalDateTime findSlotInCalendars(List<Calendar> calendars, LocalDate fromDate, int hours)* Метода проверява за свободни резервации между няколко календара. *Фиг 9*

**while *(*true*) {* int** index = 0;  
 **for *(***Calendar calendar : calendars***) {*** LocalDateTime slot = *findSlot****(***calendar, LocalDate.*from****(***currentTime***)***, hours***)***;  
 **if *(***times == **null*)*** times = **new** ArrayList***<>()***;  
 times.add***(***index, slot***)***;  
 index++;  
 ***}* if *(****checkIfAllElementsEquals****(***times***))* return** times.get***(***0***)***;  
 currentTime = currentTime.plusHours***(***1***)***;  
 **if *(***currentTime.getHour***()*** > 17***)*** currentTime = currentTime.plusDays***(***1***)***.withHour***(***8***)***;  
 times = **null**;  
***}***

*фиг 9. Цикъл за проверка дали сегашното избрано време е свободно във всички календари*

1. *CalendarWriter*: Клас отговарящ за модифициране на данните в Calendar
   1. *boolean addBooking(Calendar calendar, LocalDate date, Booking booking)* Метод който добавя резервация в Calendar
   2. *boolean removeBooking()* Методи за изтриване не резервация по различни критерии
   3. *void setHoliday(Calendar calendar, LocalDate date)* Метод който добавя дата в сета с почивни дни в Calendar
   4. *void updateBooking(Calendar calendar, LocalDate date, LocalTime startTime, BookingFields option, String value)* Метод който обновява посочената в option данна със стойността във value. Option може да е само пет стойности *фиг 10.*

**public enum** BookingFields ***{  
 date***,  
 ***startTime***,  
 ***endTime***,  
 ***name***,  
 ***note  
}***

*фиг 10. Enum със всички възможни полеза за обновяване*

* 1. *static Calendar merge(List<Calendar> calendars)* Метод който комбинира всички календари в първия от подаденият лист. Ако има конфликт се изисква вход от потребителя коя резервация да се запази. *Фиг 11.*

**if *(***booking.getStartTime***()***.isAfter***(***rootBooking.getStartTime***())*** && booking.getStartTime***()***.isBefore***(***rootBooking.getEndTime***())*** || booking.getEndTime***()***.isAfter***(***rootBooking.getStartTime***())*** && booking.getEndTime***()***.isBefore***(***rootBooking.getEndTime***())*** || booking.getStartTime***()***.isBefore***(***rootBooking.getStartTime***())*** && booking.getEndTime***()***.isAfter***(***rootBooking.getEndTime***())*** || booking.getStartTime***()***.isAfter***(***rootBooking.getStartTime***())*** && booking.getEndTime***()***.isBefore***(***rootBooking.getEndTime***())) {*** System.***out***.println***(*"Incoming booking\n"** + booking + **"\noverlaps with current booking\n"** + rootBooking***)***;  
 System.***out***.println***(*"Pick booking to keep! 1: for incoming, 2: for current"*)***;  
 **while *(*true*) {*** Scanner scanner = **new** Scanner***(***System.***in)***;  
 **int** choice = scanner.nextInt***()***;  
 **boolean** success = **false**;  
 **switch *(***choice***) {* case** 1 -> ***{*** removeFromRoot.add***(***rootBooking***)***;  
 addToRoot.add***(***booking***)***;  
 success = **true**;  
 ***}* case** 2 -> success = **true**;  
 **default** -> System.***out***.println***(*"Invalid choice!"*)***;  
 ***}* if *(***success***)* break**;  
 ***}  
}***

*фиг 11. Проверка дали има конфликт между резервациите в два календара и вземане на входа от потребителя за решаване на конфликта*

Тестване

Тестови сценарии

Отваряме нов календар с командата open <име на календар>.

*> open mihail.xml*

*File not found. Creating...*

Създаваме резервация с командата book

*> book 2022-05-31 09:15:00 11:15:00 sport sport*

*Booking added successfully*

Създаваме още резервации

*> book 2022-05-31 11:30:00 13:30:00 ps programming systems*

*Booking added successfully*

*> book 2022-05-31 11:15:00 13:15:00 gs graphical systems*

*Booking added successfully*

*> book 2022-06-01 13:30:00 15:45:00 oop object oriented programming*

*Booking added successfully*

*> book 2022-06-01 16:00:00 17:45:00 mpt mpt*

*Booking added successfully*

Премахване на резервация

*> unbook 2022-06-01 16:00:00 17:45:00*

*Booking removed successfully*

Показване на всички резервации на дадена дата

*> agenda 2022-05-31*

[Booking{startTime=09:15, endTime=11:15, name='sport', note='sport'}, Booking{startTime=11:15, endTime=13:15, name='gs', note='graphical systems'}, Booking{startTime=11:30, endTime=13:30, name='ps', note='programming systems'}]

Запазване на текущият календар във файл

*> save*

Търсене на резервация

*> find ps*

*[Booking{startTime=11:30, endTime=13:30, name='ps', note='programming systems'}]*

Добавяне на почивен ден

*> holiday 2022-05-29*

*Holiday added successfully*

Опит за добавяне на резервация в почивен ден

*> book 2022-05-29 09:00:00 18:00:00 courseWork Course work for OOP 1*

*Booking could not be added*

Намиране на свободно време след 2022-06-01 за 2 часа

*> findSlot 2022-06-01 2*

*2022-06-01T08:00*

Намиране на свободно време след 2022-05-31 за 6 часа

*> findSlotWith 2022-05-31 6 mihail2.xml*

*2022-06-03T08:00*

Затваряме *mihail.xml*

*> close*

*Calendar closed*

Създаване на *mihailMerged.xml*

*> open mihailMerged.xml*

*File not found. Creating...*

Комбинираме *mihail.xml* и *mihail2.xml*

*> merge mihail.xml mihail2.xml*

*Calendars merged successfully*

Календарите се препокриват затова излиза съобщение

*> merge mihail.xml mihail2.xml*

*Merging calendar 1 of 3*

*Merging calendar 2 of 3*

*Incoming booking*

*Booking{startTime=10:00, endTime=11:00, name='work', note='Implement AWS mechanical turk'}*

*overlaps with current booking*

*Booking{startTime=09:15, endTime=11:15, name='sport', note='sport'}*

*Pick booking to keep! 1: for incoming, 2: for current*

*>2*

*Calendars merged successfully*

Запазваме файла със save *фиг. 11*

*> save*

*<?xml version="1.0" encoding="UTF-8"?>*

*<calendar>*

*<bookings>*

*<booking>*

*<date>2022-05-31</date><name>gs</name><note>graphical systems</note><startTime>11:15</startTime><endTime>13:15</endTime>*

*</booking>*

*<booking>*

*<date>2022-05-31</date><name>ps</name><note>programming systems</note><startTime>11:30</startTime><endTime>13:30</endTime>*

*</booking>*

*<booking>*

*<date>2022-05-31</date><name>sport</name><note>sport</note><startTime>09:15</startTime><endTime>11:15</endTime>*

*</booking>*

*<booking>*

*<date>2022-06-02</date><name>Work</name><note>Curse clients</note><startTime>13:45</startTime><endTime>15:00</endTime>*

*</booking>*

*<booking>*

*<date>2022-06-02</date><name>work</name><note>Implement AWS mechanical turk again</note><startTime>10:00</startTime><endTime>11:00</endTime>*

*</booking>*

*<booking>*

*<date>2022-06-01</date><name>oop</name><note>object oriented programming</note><startTime>13:30</startTime><endTime>15:45</endTime>*

*</booking>*

*</bookings>*

*<holidays>*

*<holiday>2022-05-29</holiday>*

*</holidays>*

*</calendar>*

*Фиг 11. mihailMerger.xml файла след изпълнените операции*

Заключение

Обобщение

Приложението предлага записването и модифицирането на календари в *XML* файлове чрез командният ред. Предоставени са *CRUD* операциите за резервациите на календарите и няколко други опции някои от които подържат работа с няколко календара. Начинът на запазване на данните по времето на изпълнение е пренаписван, поради намиране на нов по-оптимизиран начин за записване

Насоки за бъдещето

Приложението може да бъде разширено с графичен интерфейс и възможността на потребителите да запазват календарите си в облака. Това би им позволило да използват календарите си от всяко устройство, което поддържа Java (Около 3 милиарда устройства)