**Технически университет – Варна**

Факултет по изчислителна техника и автоматизация

Катедра: ,,Софтуерни и интернет технологии”

Специалност: ,,Софтуерни и интернет технологии”

Тема: ,, Информационна система, която поддържа личен календар”

Изготвил: Стилиан Тодоров Каридов

Факултетен номер: 20621517

**1**

**I.Увод**

**1.1. Основна идея** – Да позволи на потребителя, управление върху личен календар. Информационната система предоставя набор от най-базовите команди ,за един личен календар. Информация за работата на командите е предоставена в следващата подточка.

**1.2. Цел и задачи на разработка –** Целта на проекта е да позволи на потребителя безпрепятствена работа с командния ред, като му предостави меню със следните команди.

* Book – Запазва час за среща на дадена дата с начален и краен час, като позволява и добавяне на име и коментар.
* Unbook – Отменя час за среща на дадена дата с начален и краен час.
* Agenda – Извежда хронологичен списък с всички ангажименти за деня.
* Change – Задава нова стойност на събитие с дата и начален час.
* Find – Търси срещи, в чиито име или бележка се съдържа подаден от потребителя низ.
* Holiday – Подадената дата се прави неработна.
* Busy days – По дадени началната и крайна дата се извежда списък с дните от седмицата, подредени по критерия „брой заети часове “.
* Find slot – По дадена дата и желана продължителност на срещата търси дата, на която е възможно да се запази такава среща.
* Find slot with – По дадена дата и желана продължителност на срещата търси дата, на която е възможно да се запази такава среща, методът е синхронизиран с друг календар.
* Merge – Прехвърля всички събития от календара, записан във файл, в текущия календар.
* Open – Зарежда и прочита съдържанието на даден файл, ако такъв не съществува се създава нов с празно съдържание.
* Save – Записва направените промени обратно в същия файл, от който са били прочетени данните.
* Save as – Записва направените промени във файл, като позволява на потребителя да укаже неговия път.
* Help – Извежда кратка информация за поддържаните от програмата команди.
* Close – Затваря текущо отворения документ. Затварянето изчиства текущо заредената информация и след това програмата не може да изпълнява други команди, освен отваряне на файл (Open).
* Exit – Излиза от програмата.

**2**

**1.3. Структура на документацията :**

1. **Увод –** описват се основната идея, цел и задачи на проекта.
2. **Преглед на предметната област -** описват се основни дефиниции, концепции и алгоритми, както и подходи и методи на решаване на поставените проблеми.
3. **Проектиране –** Общата структура на проекта, както и диаграми или блок-схеми.
4. **Реализация, тестване –** тук се описва реализацията на класовете, използваните алгоритми и оптимизации. Създаване на тестови сценарии (създаване на примери).
5. **Заключение –** Обобщение на изпълнението на началните цели. Насоки за бъдещо развитие и усъвършенстване.

**II.Преглед на предметната област**

Сложността на проекта не е висока, но темата и задачите, които трябва да се проектират са комплексни сами по себе си. В процесът на обработка е обърнато внимание на доста крайни случаи, които могат да се получат при изпълняване на някои от командите, както и на работата с дати и часове. Един от проблемите, който беше срещнат е получаване на информация за датата. Първоначално датата се подаваше с помощта на обект от класът **Date**, но това не беше оптимално за логиката на дадени методи и в процеса на работа параметърът, който се приемаше от всички методи се замени с **String-ова** (низова) имплементация на датата. Денят и месецът се взимаха с помощта на **regex** (регулярен израз), след което се обработваха. Датите се обработваха с помощта и на вграденият в Java клас **LocalDate**, който позволяваше използването на полезни методи, които улесняваха писането на код и четимостта на програмата. Аналогично за работата с часове се използваше вграденият в Java клас **LocalTime**, който позволяваше парсването на низовете към класа **LocalTime**. С негова помощ се имплементираше логика в методите, които изискваха работа с часове. Минус на информационната система, която поддържа личния календар, е че работи единствено с ден и месец от годината, като това позволява единствено и само поддръжката на една година, която се състои от 365 дни. Друг от проблемите е със записването на информацията в **XML (Extensible Markup Language)** файл. Проблемът възникваше, защото данните трябваше да се вземат от **Map**. Това се реши с добавянето на трите класа **Entry, ListOfEntry** и **Adapter**. В глава IV подробно е разгледана работата на всеки метод.

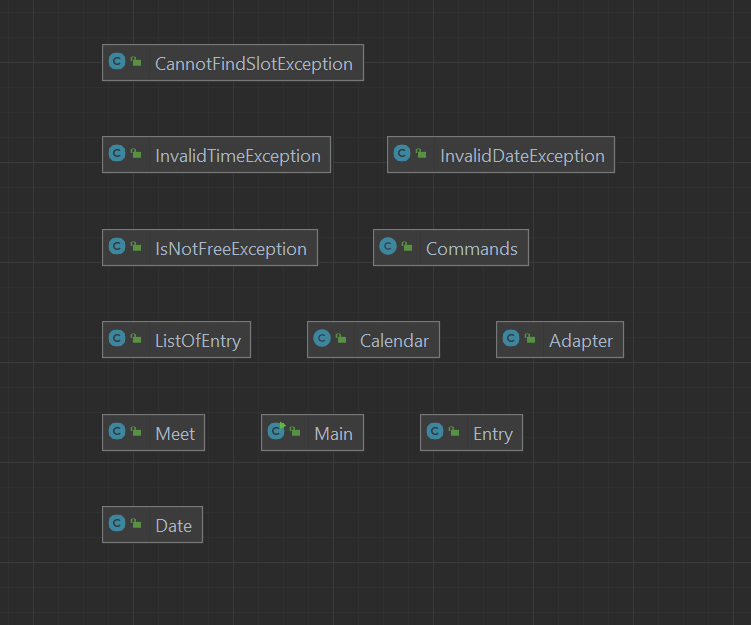
**3**

**III.Проектиране**

Проектът разполага с 3 пакета, от които всички са в главния пакет **src**:

1. Classes – тук се съдържат всички класове, които спомагат за реализирането на проекта.
2. Exceptions – в него са дефинирани 4 класа, които наследяват RuntimeException и позволяват на програмата да работи безпроблемно, при въведени некоректни данни от потребителя.
3. Files – там се създават всички файлове, които потребителя управлява.
4. Main – класът Main за стартиране на програмата.

**Диаграма на класовете в проекта**



1.1 Диаграма на класовете

**4**

**IV.** **Реализация, тестване**

* Entry **–** Това е един от трите класа, които спомагат за решаване на по-горе описания проблем със записването в XML файл. Той се съдържа от два частни атрибута. Първият e с наименование key и e от тип обект на класът Date. Вторият е List от обект на класът Meet и е инициализиран като нов ArrayList. В методът също има „гетъри“ и „сетъри“.
* ListOfEntry – Класът се състои от един частен атрибут, с наименование list и е инициализиран като нов ArrayList.
* Adapter – Последният клас и най-важен от горе посочените, защото в него е имплементирана логиката, за да може да се взима информацията от Map-а и да се записва в XML файла. Той се състои от два метода marshal и unmarshal. Първият е за превръщане на Java обектите в XML данни. Вторият прави обратното на marshal, взима данните от XML файла и ги превръща в Java обекти.
  + **Източникът, от който е взета информацията за по-горните три класа е от сайта stackoverflow, като се прилага линк към темата:** <https://stackoverflow.com/questions/33190939/how-to-marshall-mapstring-listobjects-using-jaxb>
* Date – Съдържа три атрибута, day и month, които са от тип String и isHoliday, който е от тип boolean. Всички атрибути са частни, което означава, че за тях са създадени „гетъри“ и „сетъри“. В конструктора на класа се валидират денят и месецът, с помощта на метода validateDay. Самият метод приема два аргумента, това са денят и месецът и проверява с switch case-ове броя на дните, които може да има всеки един месец, ако подадените данни не преминат успешно валидирането, се хвърля InvalidDateException ексепшън.
* Meet – Състои се от 4 атрибута, всички от които са от тип String. Това са началния и крайния час на срещата, име и бележка. Класът имплементира интерфейсите Comparable и Serializable. Първият съдържа метод compareTo, който се презаписва, за да може да се сравнява началния час на срещите. Класът Meet има default-тен конструктор както и такъв с параметри. Конструкторът без параметри е създаден, за да може да се работи с XML файловете.

**5**

* Calendar - Тук са имплементирани всички методи, които поддържа личния календар. Състои се от един единствен атрибут, LinkedHashMap който има за ключ обект от класа Date, а за стойност е подаден ArrayList от Meet-ове. Анотацията за XML е XMLJavaTypeAdapter, като в нея е класа Adapter. Логиката на методите е описана по-долу:
  + setDates – Връща LinkedHashMap. Използва се за предварително добавяне на всичките 365 дни от годината в колекцията. Методът се вика в конструктора на класа.
  + Book – За запазване на среща. Приема 2 параметъра. Дата от тип String и среща от тип обект от класа Meet. Валидира датата, ако тя е валидна итерира върху колекцията от данни, в противен случай се хвърля ексепшън за невалидна дата. При всяка итерация проверява дали текущата дата е равна на подадената такава, ако е вярно проверява дали датата е отбелязана като почивна. След това обхожда колекцията, която съдържа всички срещи за дадената дата и проверява дали часът е свободен, ако не е се хвърля ексепшън за заетост.
  + Unbook – За отмяна на среща. Тук методът приема 3 параметъра дата, начален и краен час на срещата, всички от които от тип String. Ако датата е валидна се итерира върху колекцията от данни, в противен случай се хвърля ексепшън за невалидна дата. При обхождането се проверява дали подадената дата е равна на текущо обходената, ако това е вярно се обхожда листа със срещи. Следва проверка дали съществува среща с началния и краен час, ако съществува се премахва от колекцията и се принтира подходящото съобщение, ако не съществува се хвърля ексепшън за невалидно подадени часове.
  + Agenda – За извеждане на срещи по зададена дата. Като параметър се приема датата, която е от тип String. Прави се валидация. Обхожда се колекцията и се проверява за еквивалентност на подадената дата и текущо обходената, срещите се сортират по начален час и се извеждат.
  + Change – За промяна на една от стойностите на среща. Валидира се датата, итерира се върху колекцията и се проверява за еквивалентност на подадената дата с текущо обходената. Обхожда се и листа от срещи и се проверява дали съществува среща с подадения начален час. С помощта на switch case-ове по дадената опция, се разклонява към една от стойностите на срещата и се изпълнява логиката.
  + Find – Търси среща, в чието име или бележка се съдържа подадения низ. Колекцията се обхожда и се проверява за съответствие. Ако се намери се извежда подходящото съобщение.
  + Holiday – Подадената дата се валидира. При успешна валидация се итерира върху колекцията и се проверява за еквивалентност с подадената дата. Датата се прави почивен ден, като това не позволява запазване на срещи в деня и изчиства текущите срещи ако има такива.

**6**

* + BusyDays – Важната част тук е използването на класа LocalDate и DateTimeFormatter. За датите се използва StringBuilder, с който подготвяме начина им на представяне. Създава се нов Map, в който са датите между двете подадени. Обхождаме новата колекция, която е с ключ обект от класа Date, а за стойност e wrapper класа Integer. Създава се променлива от тип int, в която се запазват общо заетите минути на всички срещи за всяка дата. След това се използва поток, който сортира новата колекция и извежда информацията на конзолата.
  + FindSlot – Тук отново се използва StringBuilder за подготовката на датите. Прави се проверка, дали продължителността е по-голяма от 9 часа. Ако не е се обхожда колекцията, като нейните срещи се сортират по начален час, за да може по-лесно да се проверява, дали е възможно да се запази среща с тази продължителност. След което се правят множество проверки, за да се провери възможно ли е. В зависимост от това дали е възможно или не, се извежда съобщение на конзолата. Като тип на връщане на метода е зададен обект от класа Date, поради факта че връщания резултат се използва в метода findslotwith.
  + FindSlotWith – Използва се try catch блок, за да се използва обекта Date, който връща по-горния метод findslot. Обекта, който се връща се запазва в променливи. Следва while цикъл, който се изпълнява докато двете данни съвпаднат. В цикъла се сравняват дните и месеците на двете дати, а след това отново се извиква findslot с други аргументи.
* Commands – Съдържа 4 атрибута. Обект от класа Calendar, който е инициализиран. Директория на файла от тип String, файл от тип File и друг от тип boolean.
  + Open методът отваря даден файл за четене, ако няма такъв създава нов с празно съдържание. Ако има такъв файл чете от него с помощта на jaxbcontext и unmarshaller, след като прочете всичко от файла, input stream-а се затваря.
  + Close методът проверява дали има отворен файл, ако има го затваря и зачиства заредената до тук информация. Ако няма извежда подходящото съобщение.
  + Save запазва направените промени в текущо отворения файл, след проверка дали такъв е отворен.
  + Save as е подобен на Save с разликата, че дава възможност за указване на пътя, в който да се запази файла.
  + Help извежда командите, които се поддържат от информационната система.
  + Exit излиза от програмата, като извежда подходящото съобщение.
  + Menu методът взима информацията с помощта на вградения в Java клас Scanner, след което с while цикъл обработва информацията. Използва се switch case, с който се изпълнява подадената команда. Преди изпълнението на всеки метод се прави проверка дали е отворен файл.

**7**

* Main – Прави се инстанция на класа Commands. Извиква се методът menu, който управлява входните данни.

Във всеки клас са използвани подходящите анотации за XML файловете.

**V.Заключение**

Информационната система е проектирана да поддържа основните команди за един личен календар. За бъдещето развитие на системата е добре да се преосмисли цялата структура на класовете както и да се имплементира и опция за избор на година. Тъй като в момента личният календар е ограничен до използването на една календарна година.

Github : https://github.com/StilianKaridov/OOP1-Project

**8**