Картина, която съдържа текст, графична колекция

Описанието е генерирано автоматичноТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА

Факултет по изчислителна техника и автоматизация

Катедра „Компютърни науки и технологии“

**КУРСОВ ПРОЕКТ**

по дисциплината „Обектно-ориентирано програмиране”

на тема: „Хотел”

Вариант №11

|  |  |
| --- | --- |
| Изготвил: Ясен Росенов Александров | Проверил: ас. инж. Стефка Попова |
| Специалност: „СИТ“ |  |
| Група: 2ра А |  |
| Факултетен номер: 23621759 |  |
|  |  |

Съдържание

[I. Увод 3](#_Toc197121343)

[1. Описание и идея на проекта 3](#_Toc197121344)

[2. Цел и задачи на разработката: 3](#_Toc197121345)

[II. Преглед на предметната област 5](#_Toc197121346)

[1. Основни концепции и алгоритми, които ще бъдат използвани 5](#_Toc197121347)

[2. Подход, методи за решаване на поставените задачи 5](#_Toc197121348)

[3. Функционални и нефункционални изисквания 6](#_Toc197121349)

[III. Проектиране 7](#_Toc197121350)

[1. Обща структура на проекта 7](#_Toc197121351)

[2. Диаграми / Блок-схеми 8](#_Toc197121352)

[IV. Реализация 16](#_Toc197121353)

[1. Реализация на класове 16](#_Toc197121354)

[2. Оптимизации 17](#_Toc197121355)

[V. Тестване 18](#_Toc197121356)

[1. Планиране, описание и създаване на тестови сценарии (създаване на примери за всички възможни входове на командите) 18](#_Toc197121357)

[VI. Заключение 25](#_Toc197121358)

[1. Обобщение на изпълнението на началните цели 25](#_Toc197121359)

[2. Използвана литература 25](#_Toc197121360)

1. Увод
   1. Описание и идея на проекта

Да се напише програма, реализираща информационна система, обслужваща хотел. Програмата съхранява и обработва данните за стаите в хотела във файл. 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):

* 1. Цел и задачи на разработката:
     1. checkin <room> <from> <to> <note> [<guests>]

Регистриране в стая с номер <room> от дата <from> до дата <to> и се добавя бележка <note>. Незадължителният параметър <guests> задава броя на гостите, настанени в стаята. Ако той не е указан, се счита, че броят на настанените гости е равен на броя на леглата в стаята.   
Пример: checkin 229 2020-03-23 2020-03-31 The Simpsons

* + 1. availability [<date>]

Извежда списък на свободните стаи на дата <date>, ако не е зададена, се използва текущата дата.

* + 1. checkout <room>

Освобождаване на заета стая с номер <room>

* + 1. report <from> <to>

Извежда справка за използването на стаи в периода от дата <from> до <to>. Извежда се списък, в който за всяка стая, използвана в дадения период, се извежда и броя на дните, в които е била използвана.

* + 1. find <beds> <from> <to>

Намиране на подходяща свободна стая с поне <beds> на брой легла в периода от <from> до <to>. При наличие на повече свободни стаи се предпочитат такива с по-малко на брой легла.

* + 1. find! <beds> <from> <to>

Да се реализира алгоритъм, който предлага спешно намиране на стая за важен гост в случай на липса на свободни стаи за даден период. Алгоритъмът да предлага разместване на настанените от най-много две стаи.

* + 1. unavailable <room> <from> <to> <note>

Обявява стаята с номер <room> от дата <from> до дата <to> за временно недостъпна и се добавя бележка <note>. В стаята няма регистриран гост, но никой не може да бъде настанен в нея.  
Пример:unavailable 200 2018-06-01 2019-03-01 Under construction

* + 1. Гостите на хотела да могат да се записват по стая за различни дейности от даден списък при настаняване

a. да се извежда списък на програмата на една стая

b. да се извежда списък на всички записали се за дадена дейност

1. Преглед на предметната област
   1. Основни концепции и алгоритми, които ще бъдат използвани
      1. Обектно-ориентирано моделиране

Всеки елемент от реалния свят (като стая, гост, резервация, дейност и хотел) е представен като обектс с определени свойства и поведение.

* + 1. Капсулация

Всички данни за стаи, гости, резервации, дейности и хотел са капсулирани в съответни класове.

* + 1. Файлова обработка

За отваряне и съхранение на данни от/във файл (с команди open, save, save as, close).

* + 1. Обработка на дати

Използват се алгоритми за проверка на заетостта в даден период, пресичане на дати и генериране на отчети.

* + 1. Алгоритъм за намиране на стаи

При командa **find** се търси подходяща свободна стая с определен брой легла за даден период. При командата **find!** ако не е намерена подходяща стая, се опитва да размести гости от най-много две стаи.

* + 1. Филтриране

При справки (например **availability, report**) се прилагат алгоритми за извеждане на сортирани/филтрирани резултати.

* 1. Подход, методи за решаване на поставените задачи
     1. Файлово базирана архитектура

Данните се съхраняват в структуриран текстов файл, който се зарежда и записва при необходимост.

* + 1. Използване на стандартни Java библиотеки

Например **java.time.LocalDate** за работа с дати

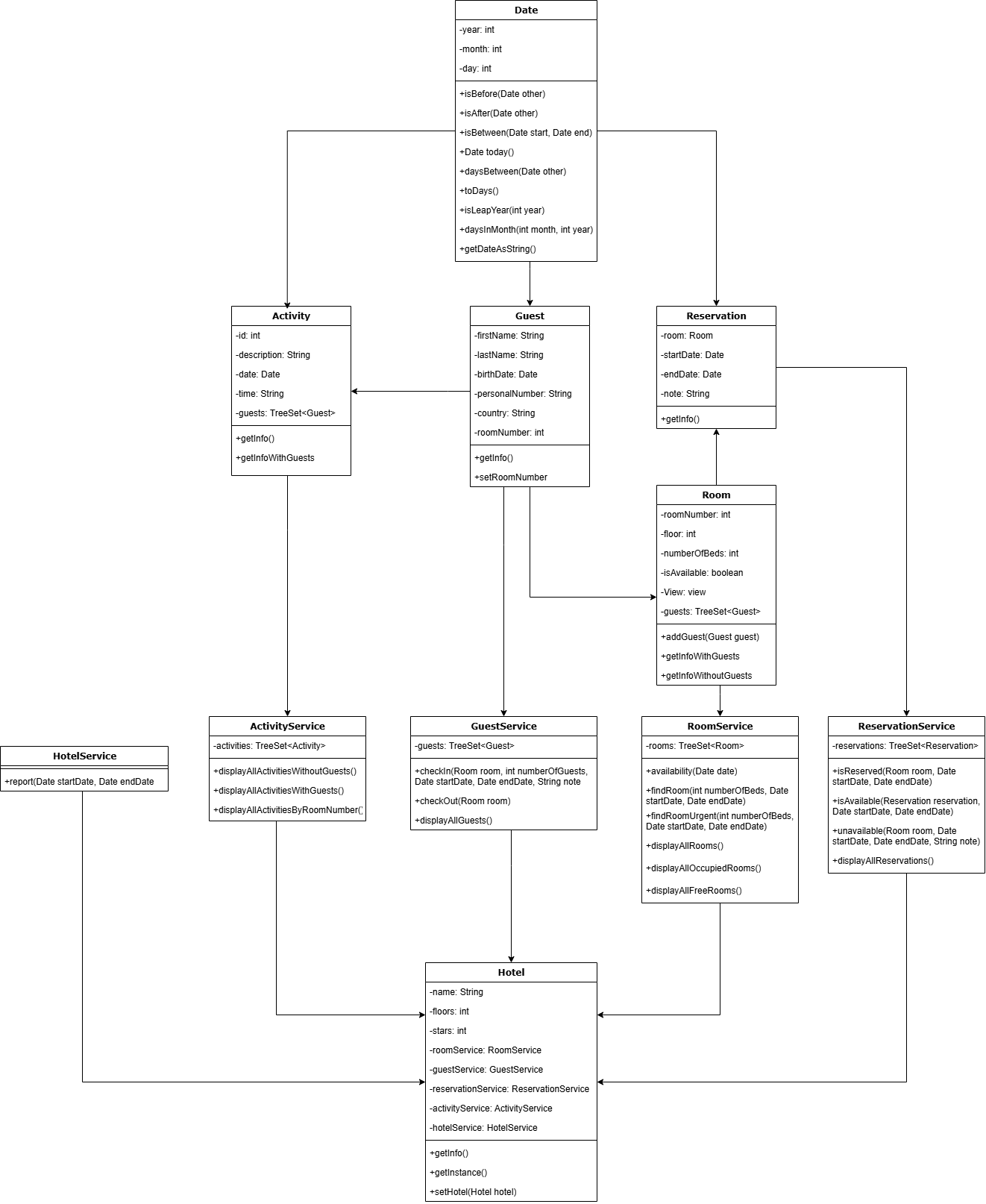
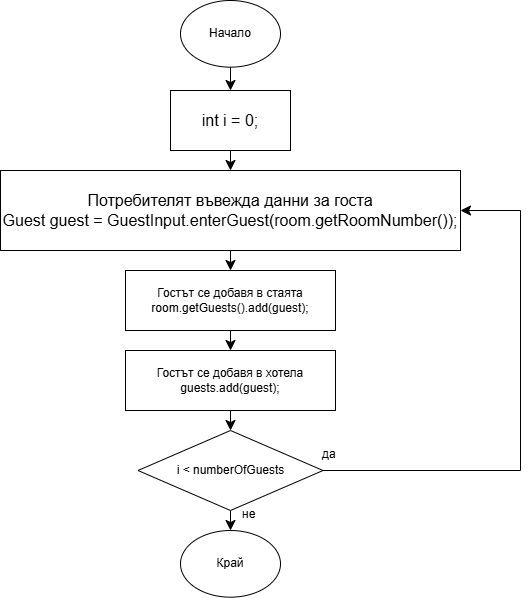
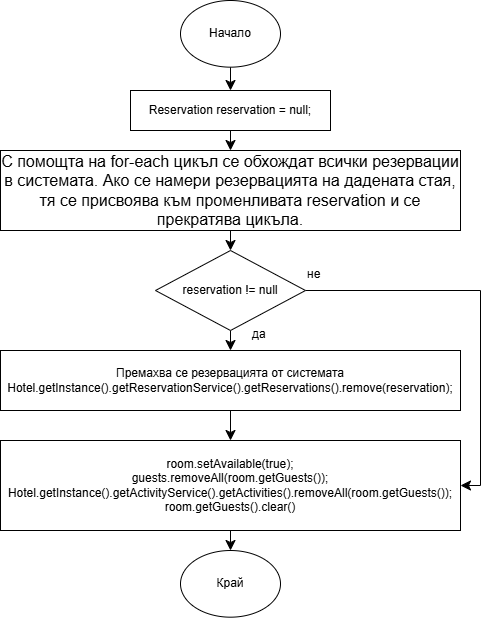
* + 1. Команден интерфейс

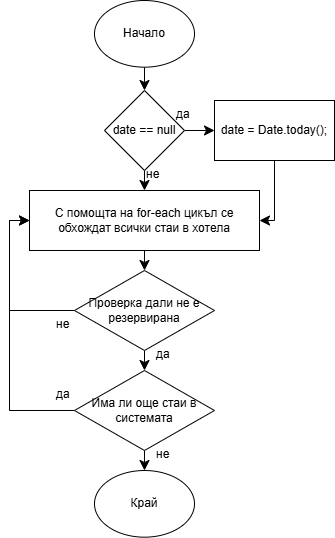
Потребителят въвежда команди, които се интерпретират и обработват от програмата

* 1. Функционални и нефункционални изисквания
     1. Функционални
* Системата трябва да поддържа следните команди за работа с файлове: **open, save, saveas, help, close.**
* Регистрация на гости **(checkin)**
* Освобождаване на стаи **(checkout)**
* Проверка на свободни стаи **(availability)**
* Генериране на справки **(report)**
* Обявяване на стаи за недостъпни **(unavailable)**
* Спешно търсене на стая с разместване **(find!)**
* Дейности по интереси и програма на стая
  + 1. Роли и права
* Оператор: има право да въвежда, променя и запазва информация
* Система: валидира дати, проверява наличност и конфликти
  + 1. Нефункционални изисквания
* Устойчивост при грешки (въвеждане на невалидни дати, файлови грешки)
* Лесна разширяемост – нови команди да могат да се добавят без сериозни промени
* Добра четимост и поддъжка на кода
* Системата трябва да бъде лесна за използване(конзолен интерфейс с ясни съобщения).

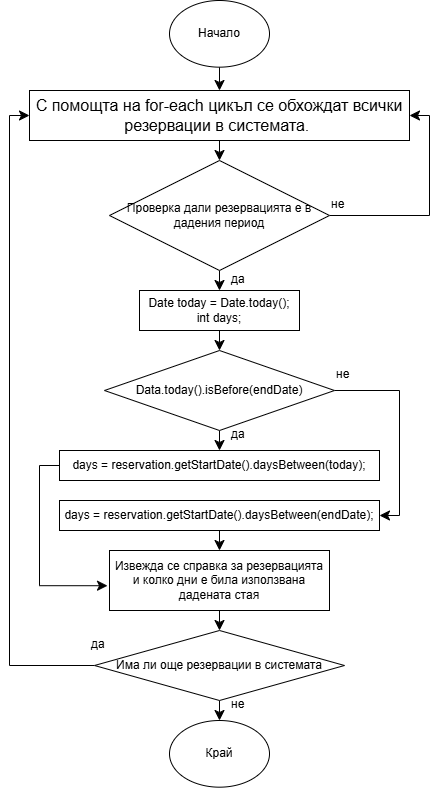
1. Проектиране
   1. Обща структура на проекта

Проектът е организиран по слоев принцип, което улеснява поддръжката, разширяването и разделянето на отговорностите. Основните пакети са:

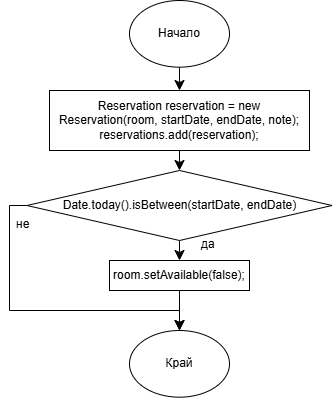
* **app** – съдържа основните класове за стартиране и управление на приложението:
* Application – главен клас за стартиране на приложението.
* MenuClass – реализира потребителско меню.
* **exceptions** – предназначен за обработка на изключения.
* ErrorByReading – грешка при четене на файл.
* **fileManage** – съдържа логика за работа с файлове:
* FileManager, FileRead, FileWrite – управление на четене/запис на файлове.
* **input** – събира всички класове, свързани с въвеждане на данни:
* ActivityInput, FileInput, GuestInput, HotelInput, ReservationInput, RoomInput, ValidateInput, InputHelper.
* **Interfaces** – дефинира интерфейси за услугите (services).
* **GuestServiceInterface, HotelServiceInterface, RoomServiceInterface, ReservationServiceInterface.**
* **model** – съдържа моделите на данни, които представят обекти от реалния свят в системата.
* **Guest, Hotel, Room, Reservation, Activity, Date, View.**
* **service** – съдържа класовете, нужни за съхранение и обработка на данните в системата.
* **GuestService, HotelService, RoomService, ReservationService, ActivityService.**
  1. Диаграми / Блок-схеми
     1. Диаграма на проекта
     2. Блок-схема на **checkin** метода
     3. Блок-схема на **checkout** метода
     4. Блок-схема на **availability** метода



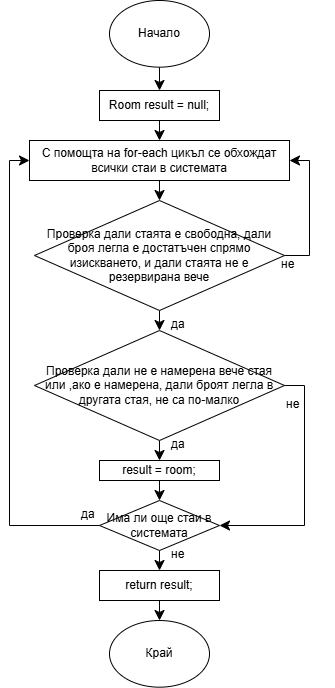
* + 1. Блок-схема на **report** метода

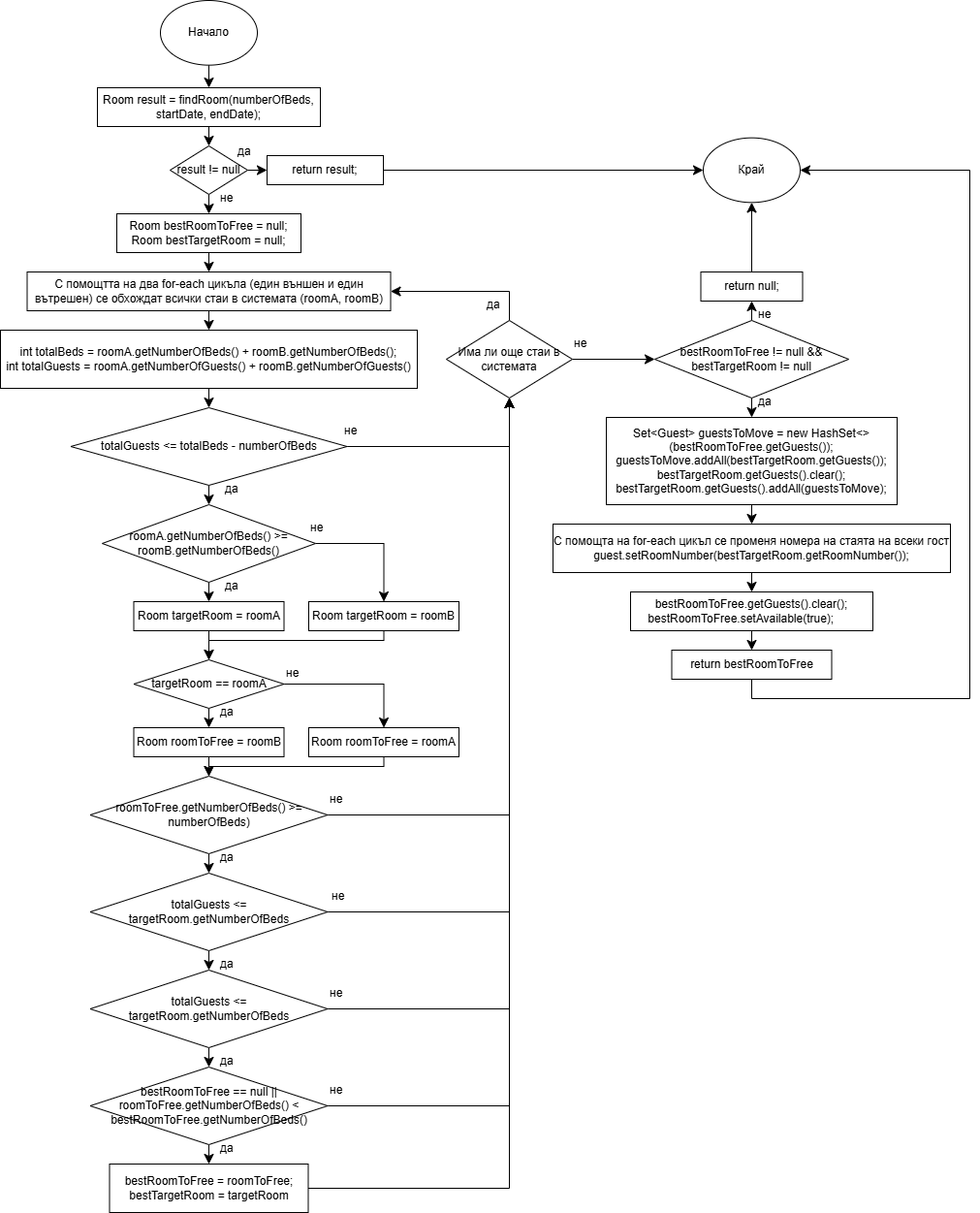


* + 1. Блок-схема на **unavailable** метода



* + 1. Блок-схема на **find** метода



* + 1. Блок-схема на **find!** Метода

1. Реализация
   1. Реализация на класове
      1. Класове за съхранение и обработка на данните в системата

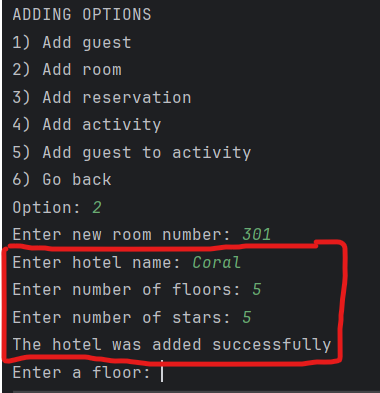
* **RoomService** – отговаря за управление на стаи, с възможност за търсене на свободна стая, преместване на гости при необходимост, с цел освобождаване на стая, проверка за свободни стаи.
* **GuestService** – отговаря за настаняването и изпращането на гости.
* **ReservationService** – отговаря за управление на резервациите, с възможност за задаване на дадена стая като невалидна, проверка дали дадена резервация е валидна за даден период и проверка дали дадена стая е резервирана.
* **ActivityService** – отговаря за управление на дейности
* **HotelService** – отговаря за отчет на всички резервации, които са налични в даден период
  + 1. Класове за обработка на файлове
* **FileRead** – съдържа методи за четене на данни от файлове в системата - за четене на данни на хотел, стаи, гости, резервации и дейности.
* **FileWrite** – съдържа методи за запис на данни от системата във файлове – запис на данни на хотел, стаи, гости, резервации и дейности.
* **FileManager** – отговаря за управлението на четене и запис на данни от/във файлове.
  + 1. Класове за обработка на входни данни
* **ActivityInput** – отговаря за входни данни за дейности, с методи за добавяне на дейност, извеждане на дейности, в които участват гости от конкретна стая, и извеждане на информация за конкретна дейност.
* **FileInput** – предоставя методи за управление на файлов вход и изход.
* **GuestInput** – съдържа методи за добавяне на гости в системата и в дадена дейност.
* **HotelInput** – отговаря за обработка на входни данни, свързани с хотела – създаване на хотел, настаняване, освобождаване на стаи, справки.
* **InputHelper** – предоставя помощни методи за въвеждане и валидиране на различни типове входни данни от потребителя, включително дати, цели числа, времеви часове и егн.
* **ReservationInput** – предоставя методи за добавяне на нова резервация и задаване на стая като невалидна за даден период.
* **RoomInput** – съдържа методи за търсене на стая, добавяне на стая и търсене на свободна стая за дадени период и брой легла.
* **ValidateInput** – предоставя помощни методи за валидиране на входните данни, въведени от потребителя.
  + 1. Класове свързани с работата на приложението
* **MenuClass** – управлява главното меню на приложението и всички негови подменюта.
* **Application** – представлява начална точка на изпълнение на приложението.
  1. Оптимизации
* Използване на **TreeSet** вместо **ArrayList** за по-бързо търсене, обхождане и за уникалност на елементите.
* Извеждане на логика в помощни класове (**InputHelper, ValidateInput**) за повтарящи се проверки.
* Разпределение на отговорности по класове и пакети.

1. Тестване
   1. Планиране, описание и създаване на тестови сценарии (създаване на примери за всички възможни входове на командите)

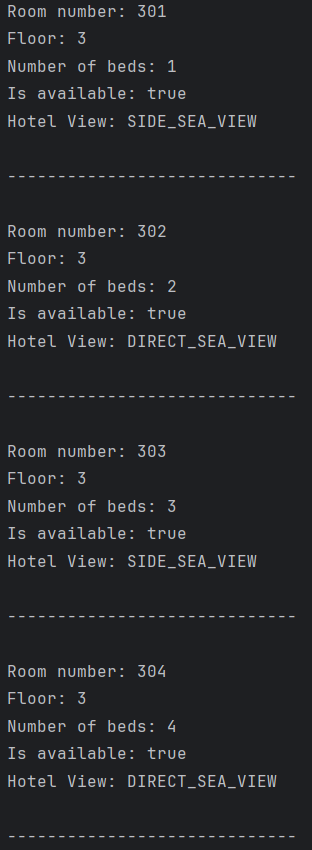
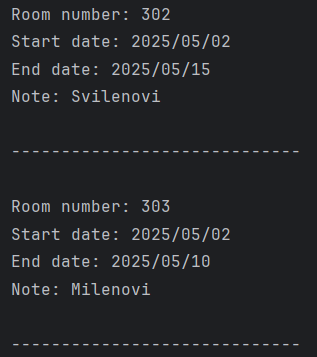
За да се провери дали системата работи коректно, бяха създадени различни тестови сценарии, които обхващат основните ѝ функционалности. Всеки тест показва примерен вход от потребителя през конзолата, както и резултата от него.

Целта на тестването е да се провери как се държи приложението в различни ситуации, включително:

* При правилно въведени данни (валиден вход)
* При грешки или липсваща информация (невалиден вход)
* При гранични случаи
* Зависимости между данните
  + 1. При липса на инстанция на обект от тип Hotel, програмата автоматично задължава потребителя да въведе входни данни за хотела



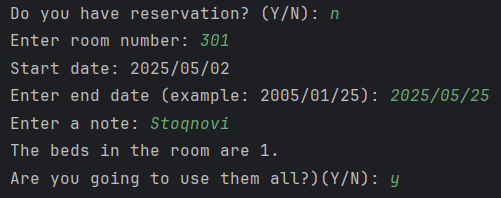
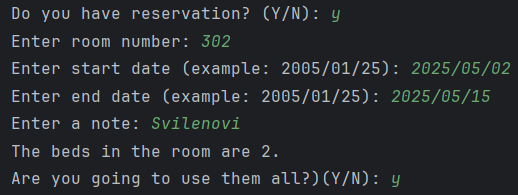
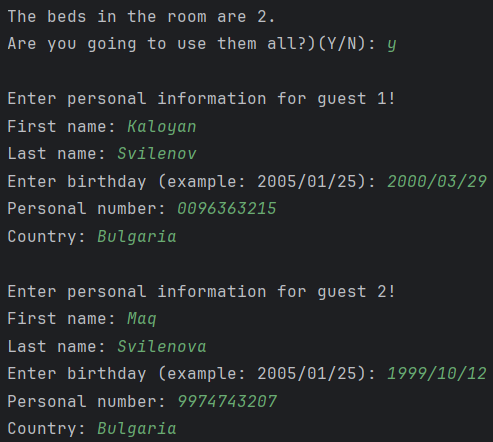
* + 1. Това са въведените тестови данни, за да може да се тества програмата

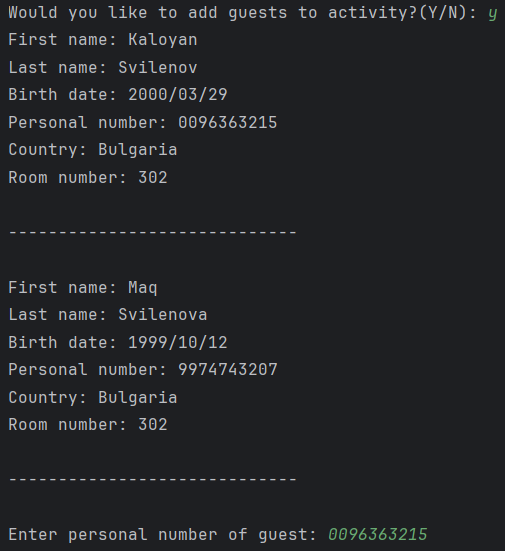
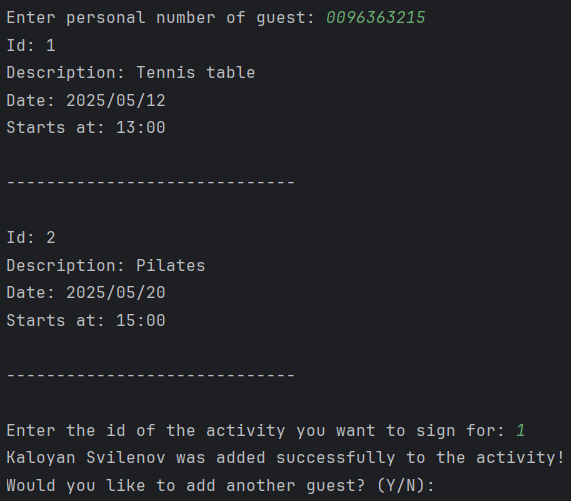


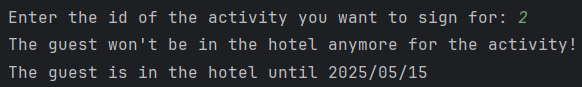
Дейности

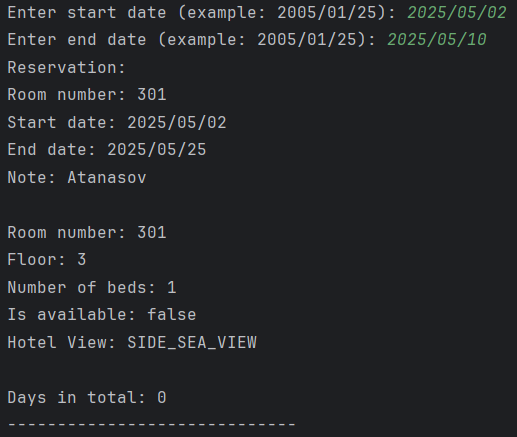
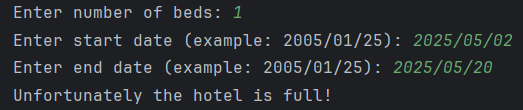
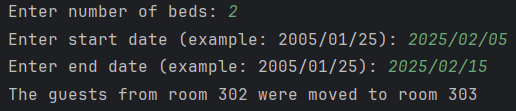
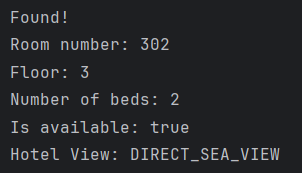
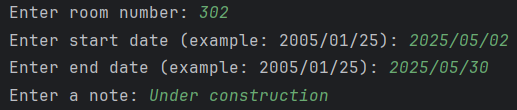
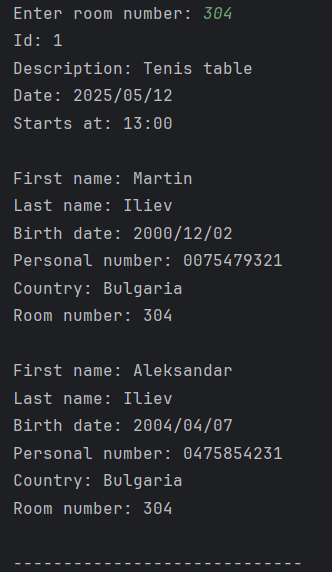
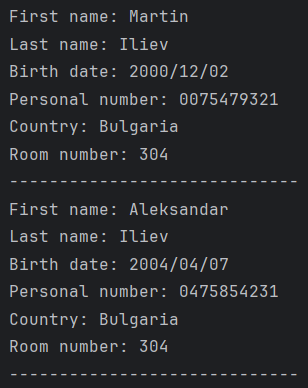
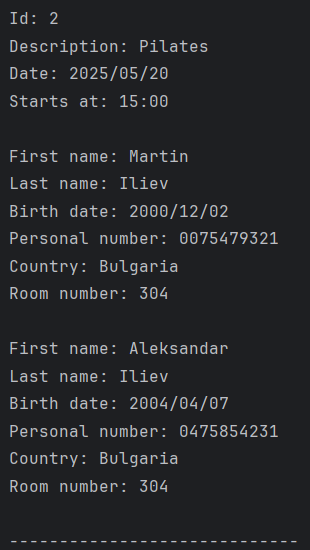
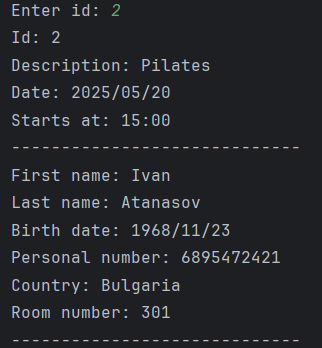
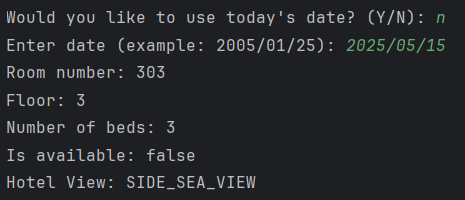
Стаи

Резервации

* + 1. При настаняване, гостът може да бъде настанен с резервация, ако има такава, ако не, тя се създава намомента с данни въведени от потребителя.
    2.  След валидирането или създаването на резервацията се въвеждат данните на гостите, като потребителя оказва колко души ще бъдат в стаята в зависимост от броя легла.
    3. При настаняването на гостите, гостите могат да се запишат за различни дейности. Записването става с въвеждане на ЕГН. Ако резервацията на госта изтича преди провеждането на дейността, системата не го записва.





* + 1. Проверка за резервираните стаи за даден период, като се извежда справка и колко дни са били използвани спрямо днешната дата.
    2. Намиране на свободна стая за даден период
    3. При липса на такава е възможно разместване на гости от най много две стаи.
    4. Задаване на стая за невалидна
    5. Справка за всички гости от дадена стая, записали се на различни дейности
    6. Справка на всички гости, записали се за дадена дейност
    7. Справка за свободни стаи за даден дата
    8. Изпращане на гости (премахват се от системата на хотела, стаята и ако са били записани на някаква дейност.

1. Заключение
   1. Обобщение на изпълнението на началните цели

В проекта беше реализирана софтуерна система за управление на хотел, която отговаря на предварително зададените функционални изисквания. Системата предоставя възможности за управление на стаи, гости, резервации и дейности, както и извършване на справки и отчети за определени периоди.

Реализацията обхваща ясно структурирани класове, разделени според тяхното предназначение – за обработка на данни, вход, файлове и управление на работния процес. Създадени са помощни методи за валидиране на входни данни.

Системата беше тествана чрез поредица от тестови сценарии, обхващащи валидни входове, гранични случаи и логически зависимости между данните. Това помогна за откриване и коригиране на грешки в логиката на някои методи, което подобри стабилността на приложението.

* 1. Използвана литература
     1. Уеб ресурс (W3Schools) за работа с дати - https://www.w3schools.com/java/java\_date.asp