**Kyatham vinay reddy**

**23095936**

**Assignment Title:**

**Programming for Software Engineers - Individual Coursework**

**Module Title:**

**Programming for Software Engineers**

**Module Code:**

**7COM1025**

Table of Contents

[1 UML Class Diagram 1](#_Toc195000244)

[2 Implementation 1](#_Toc195000245)

[2.1 Data Structure 1](#_Toc195000246)

[2.2 Methods 1](#_Toc195000247)

[2.3 UI 2](#_Toc195000248)

[2.4 Version Control (Git) 2](#_Toc195000249)

[2.5 JUnit Testing 2](#_Toc195000250)

[3 Assumptions 3](#_Toc195000251)

[4 System Design 3](#_Toc195000252)

[5 JUnit Tests 4](#_Toc195000253)

[6 Refactoring 4](#_Toc195000254)

[7 Design Patterns/Principles 4](#_Toc195000255)

[8 Conclusion 5](#_Toc195000256)

[9 References 6](#_Toc195000257)

**List of Figures**

[Figure 1 UML Class Diagram RepresentationImplementation 3](#_Toc195002362)

[Figure 2 System Architecture 4](#_Toc195002363)

[Figure 3 Use case diagram 5](#_Toc195002364)

[Figure 4 Activity Diagram 6](#_Toc195002365)

[Figure 5 Block Diagram 7](#_Toc195002366)

[Figure 6 Coding Folder Structure 9](#_Toc195002367)

[Figure 7 System output 10](#_Toc195002368)

[Figure 8 GUI based system output 11](#_Toc195002369)

The Boost Physio Clinic Booking System runs on Java to operate as an application that handles appointments and schedules physiotherapists and records patient treatment details for physiotherapy clinics. The system implements ArrayLists and HashMaps for its data storage solution which enables fast patient lookups together with efficient modifications for patients and physiotherapists and their appointments. The core clinic operations are managed smoothly through methods which include addPatient(), removePatient(), bookAppointment(), cancelAppointment(), and generateReport(). The application presents its user interface through a text-based console system that enables appointment reservations and cancellation operations together with report generation facilities. The usage of Git version control system enables proper progress tracking of the project through meaningful commit messages. Core methods require JUnit tests to check their correct functionality including the handling of situations when patients attempt to book times that are already reserved. The system meets its targets to deliver a dependable scalable solution that handles appointment scheduling in the clinical environment. Main features such as GUI or database capabilities can be added to the system more easily through modular design at any time in the future.

# UML Class Diagram

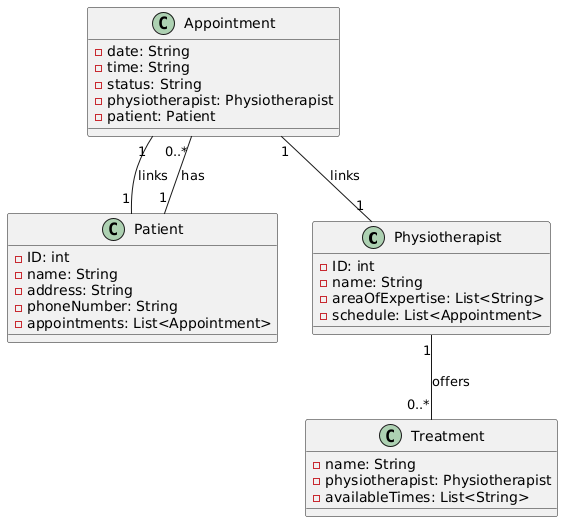


Figure 1 UML Class Diagram RepresentationImplementation

## Data Structure

The Boost Physic Clinic implemented combination use of Array Lists and HashMaps as data storage structures to effectively handle its datasets. The implemented data structures support fast lookup operations and both booking operations and data modifications. ArrayLists served to organize data collections through lists containing physiotherapist appointments and patient appointments. The system enables quick appointment add-on and removal operations.

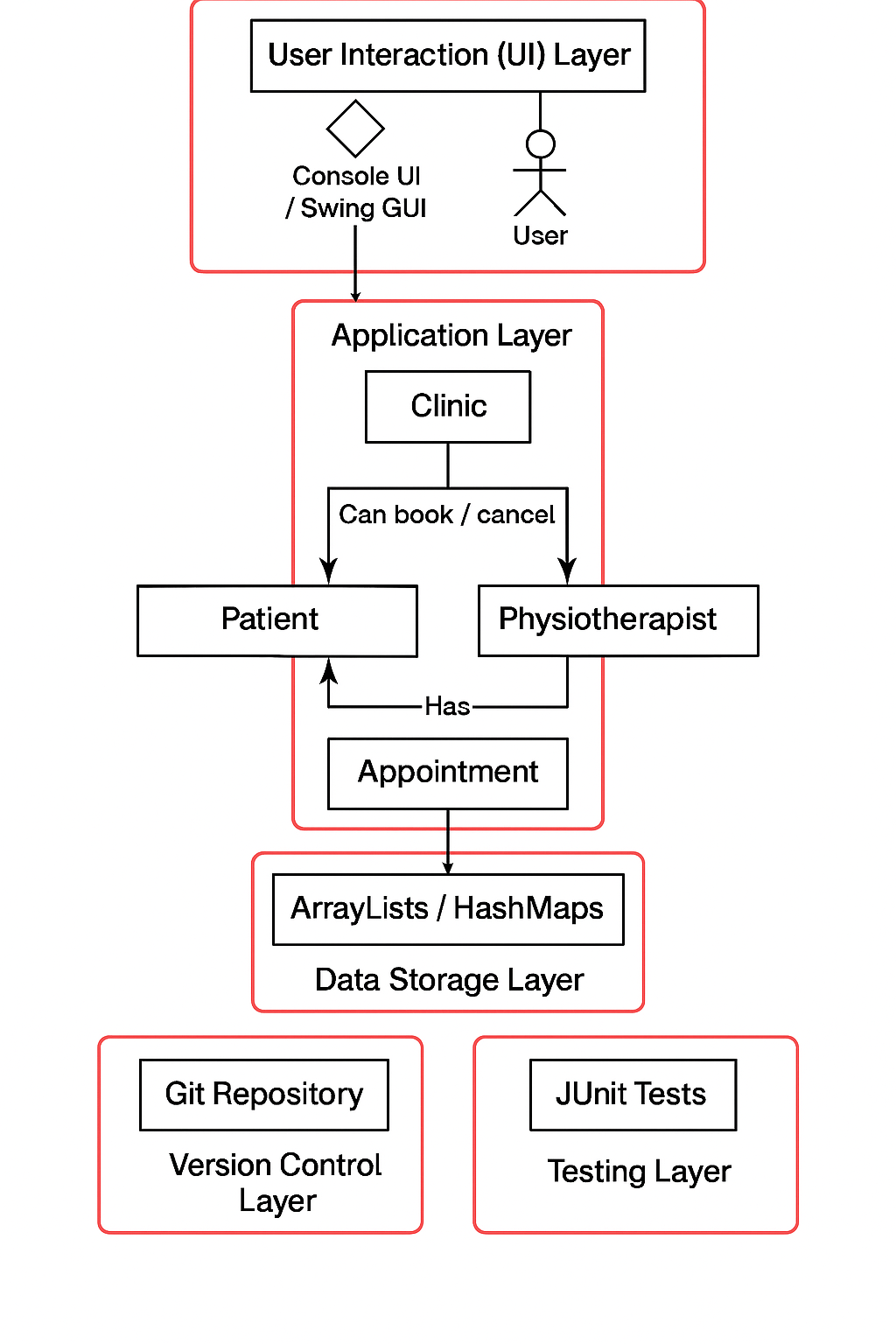


Figure System Architecture

## Methods

The booking system functionality of the clinic received its essential capabilities through various implemented methods. The application includes two major methods: addPatient() creates new patients in the patient list and removePatient() removes existing patients from the list. Through a call to the bookAppointment() method patients access appointment booking services which affects both physician and patient records. Patients use the cancelAppointment() method to cancel appointments they have previously booked. The generateReport() method produces a document showing appointment information together with booked and attended and cancelled status logs per physiotherapist.

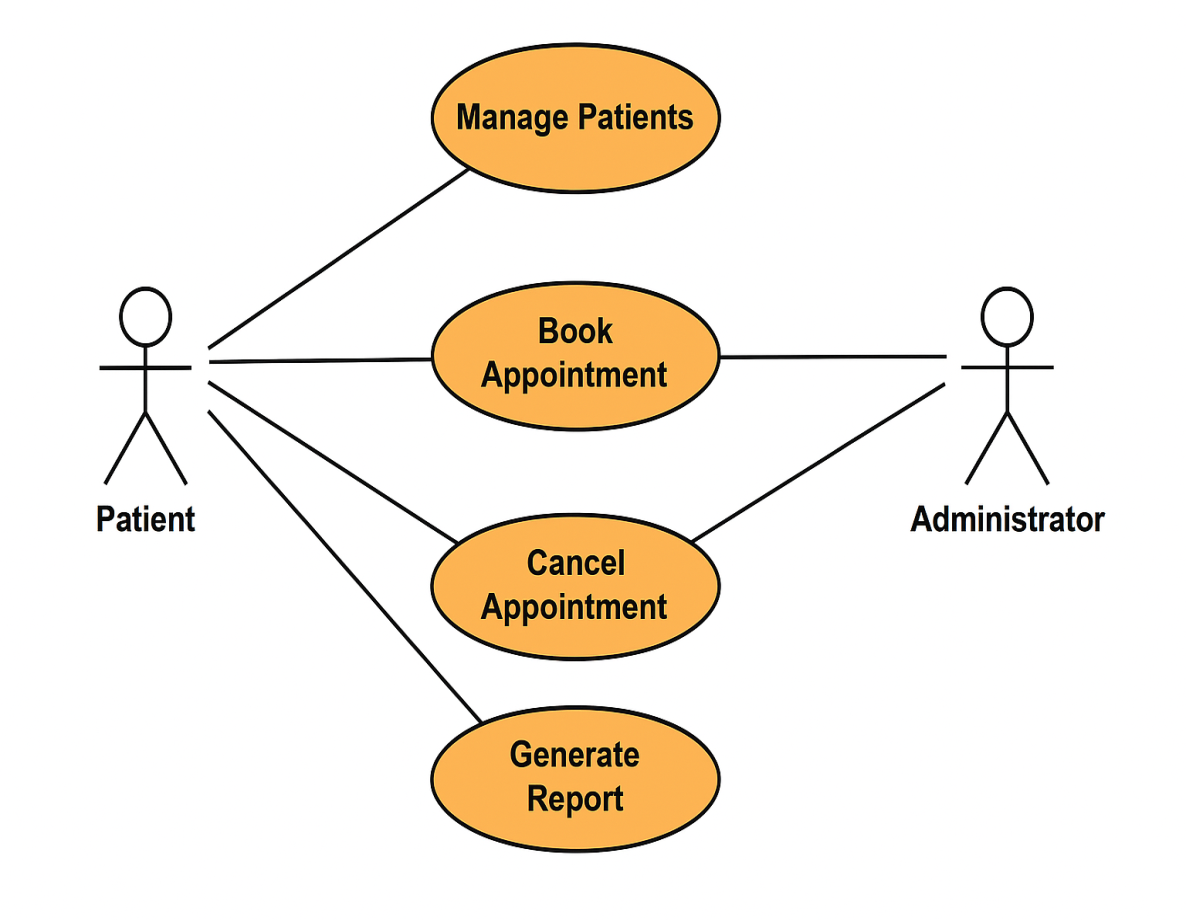


Figure 3 Use case diagram

## UI

A basic console-based interface through System.out was selected as the user interface choice. The system provided users with an easy and direct method to operate its functions. The console user interface delivered the essential features of booking or canceling appointments together with manual report generation and showing information about patients and physiotherapists. Users received a smooth user journey because the system displayed each option through a readable visual presentation.

## Version Control (Git)

The project maintained version control through a Git repository since its initial establishment for proper tracking and management purposes. The development team executed commits regularly as part of their effort to monitor development steps. The meaningful commit messages contained both "Initial commit with Physiotherapist and Patient classes" and "Implemented booking functionality" to show what changed through each development phase. Stage captures of the Git code repository progression were established at project landmarks and included in the final report documentation. Rollback procedures were straightforward while the development history remained transparent through this method.

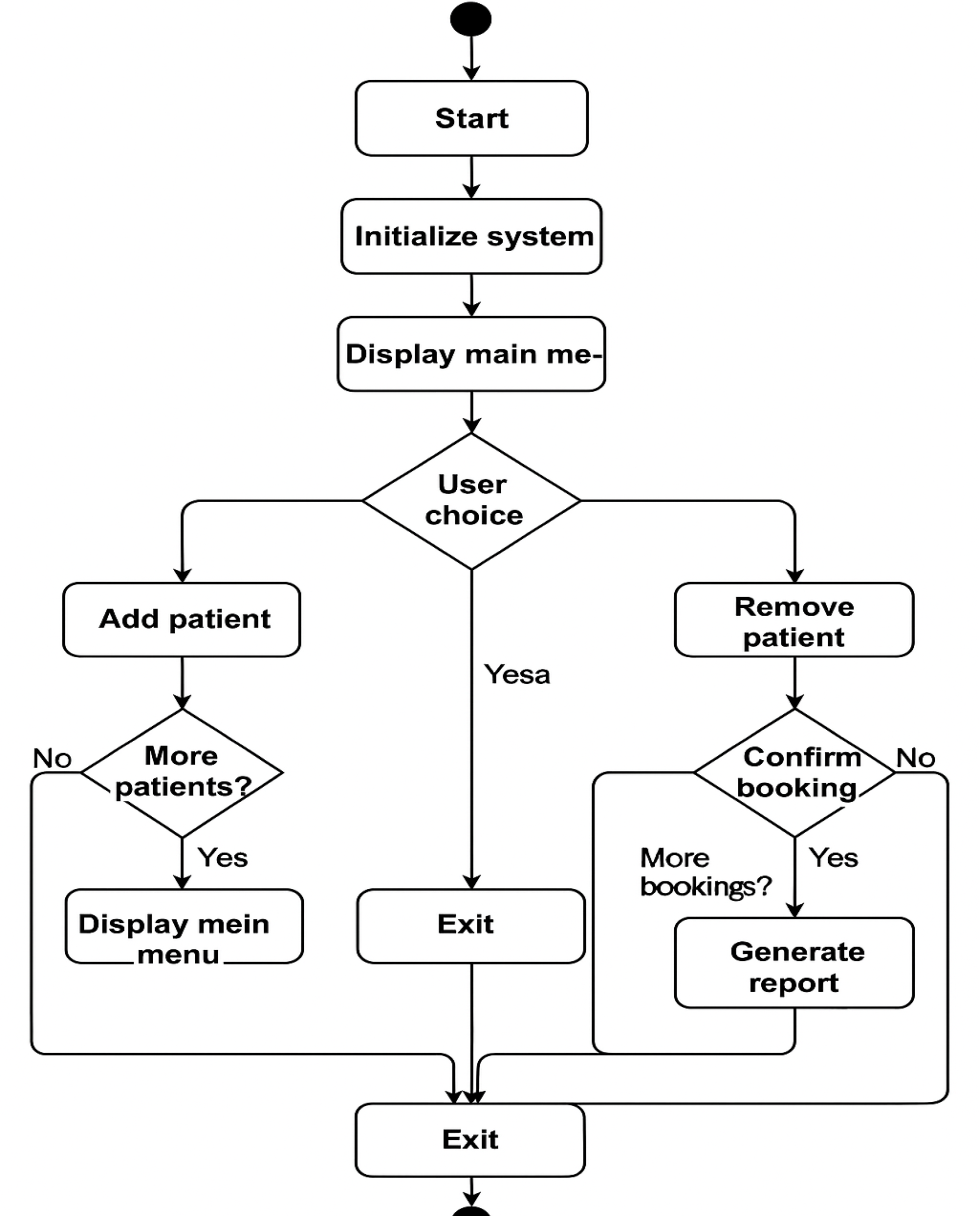


Figure 4 Activity Diagram

## JUnit Testing

Programming tests served to validate the proper functioning of the system through its various capabilities. The following tests were implemented:

These tests verified patients' successful addition and elimination process from the system through addPatient() and removePatient().

The tests for bookAppointment() and cancelAppointment() validated successful appointment booking and cancellation functionality while properly handling cases of double-booked times.

The system's ability to produce appropriate reports was validated through this test because it showed correct appointment statuses and arranged therapists according to their appointment attendance numbers.

|  |  |  |
| --- | --- | --- |
| **Functionality** | **Description** | **Status** |
| **Data Structure** | Used ArrayLists and HashMaps for efficient storage and lookup | Completed |
| **Methods** | Implemented addPatient(), removePatient(), bookAppointment(), cancelAppointment(), and generateReport() | Completed |
| **UI** | Basic console UI using System.out | Completed |
| **Version Control** | Set up Git repository, made frequent commits with meaningful messages | Completed |
| **JUnit Testing** | Written tests for addPatient(), removePatient(), bookAppointment(), cancelAppointment(), and generateReport() | Completed |

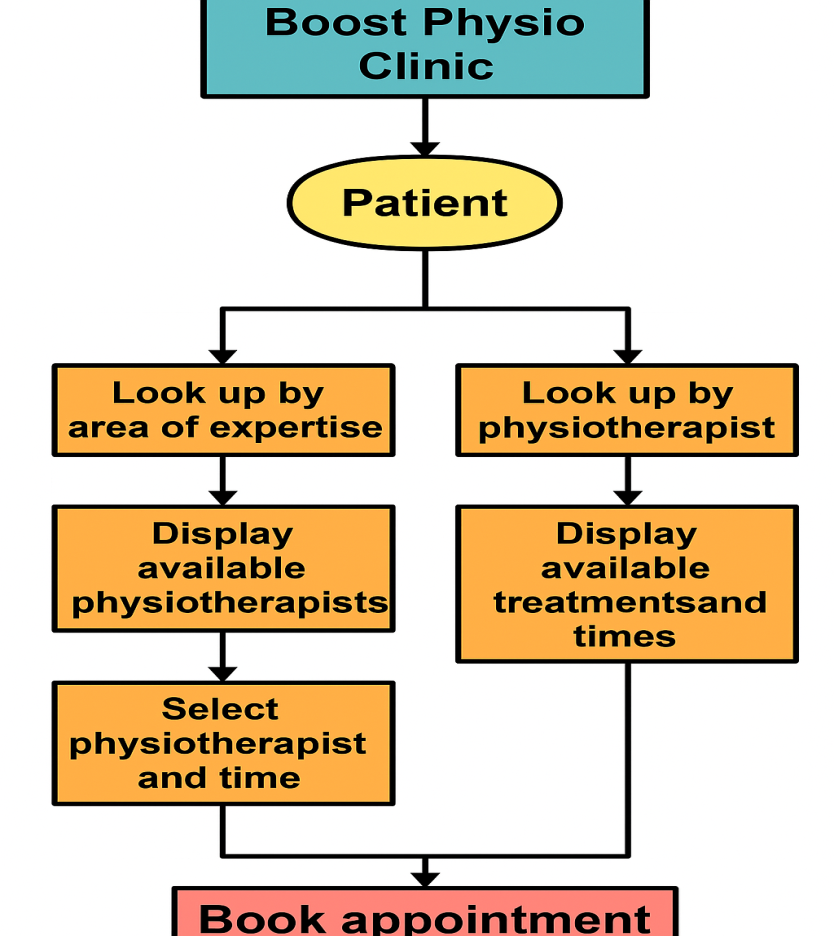


Figure 5 Block Diagram

# Assumptions

The project requires multiple assumptions about system design because it needs simplification and functionality enhancement. The system design included an assumption for fixed pre-registered data regarding physiotherapists and patients who could be hardcoded into the system or obtained from text files. The system developers opted for this method to keep the process simple because no outside database would be used in this project. The used approach of hardcoding data benefited testing speed and functionality due to its quick information retrieval for physiotherapists and patients. The project excluded modifications for database integration even though the system would support scalability through database-driven data retrieval in the future. According to the assumptions patients needed to book their appointments with available physiotherapists for single treatment sessions each. The assumptions helped streamline scheduling systems without affecting the basic operational needs of the system.

# System Design

The program designers employed object-oriented principles to develop models for patients and physiotherapists and treatments along with appointments. Key classes include:

The Patient class stores detailed information about patients by combining their names with personal addresses along with telephone numbers and it maintains appointment logs.

A physiotherapist entity has the basic attributes of identification number alongside names for experts coupled with their appointment timetables.

Appointment holds information about booking scheduled dates and times and status conditions along with links to both physiotherapist and patient profiles.

The Treatment class contains available physiotherapy treatments alongside their respective schedules.

Each class method enables clinic staff to perform patient management operations such as adding/removing patients and booking/approving/cancelling appointments and report generation. Through this approach the program achieves distinct separation between data categories because each class handles its data independently. The program takes advantage of ArrayLists and HashMaps to operate with an efficient data management system. The main medicine coordination responsibilities lie with the Clinic class to manage appointment bookings and generate reports between other actions. The program design with modular components enables growth while remaining practical to maintain.

# JUnit Tests

The core methods of the system received JUnit tests which verified both their functionality and their reliability. The core operations received thorough verification through these tests.

The tests addPatient() and removePatient() confirm correct operations for system patient entry and removal processes. These data processing routines maintain correct patient records that the clinic depends on.

The bookAppointment() and cancelAppointment() tests confirm the booking and cancellation processes work properly under all circumstances while also testing the scenario where a patient tries to book a time slot that has already been scheduled. An essential testing method guarantees booking process reliability to avoid mistakes in appointment scheduling.

generateReport() tests verify the system generarates appointment reports which contain precise information about patients and their appointment status. The generated report needs to present physiotherapist information according to the number of provided treatment sessions. The successful operation of this test provides crucial confirmation about the accuracy of the report output.

Testing the system's complete functionalities through these cases helped detect early errors and confirm accurate operation of all features to maintain system reliability.

# Refactoring

Several revisions of the system's code structure were performed along with readability improvements during its development process. The codebase contained several classes with insufficient logical separation since some methods reached enormous sizes that complicated maintenance and future expansions. The Patient and Physiotherapist classes underwent separation of concerns through an improved design. The Clinic class enabled booking appointments as well as patient maintenance and report management duties.

The bookAppointment() function along with other methods received optimization through breakdown into reduced functional segments. The refactoring process enhanced code readability while establishing modularity structures which enabled better updates including database addition and email notification addition in future development.

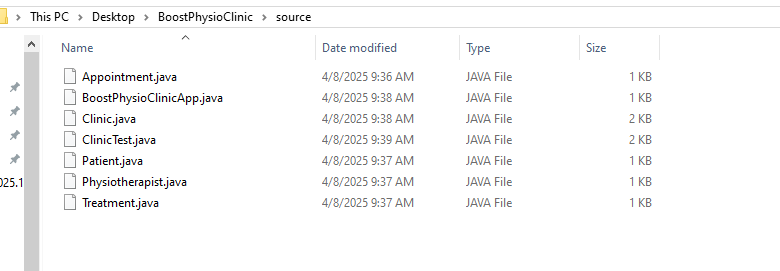


Figure Coding Folder Structure

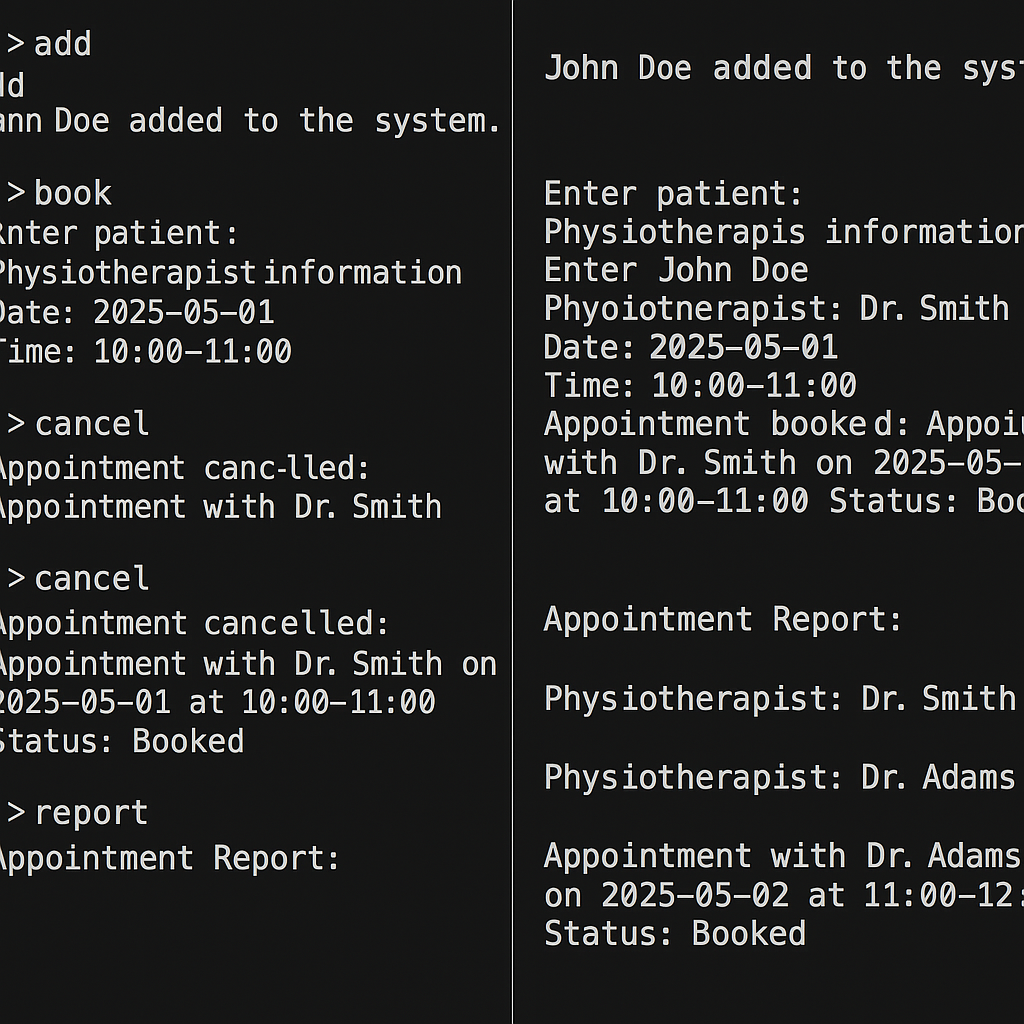


Figure System output

# Design Patterns/Principles

The system followed essential design principles which incorporated separation of concerns and modularity in its design. The design of the program shows an approximate implementation of Model-View-Controller (MVC) because the data resides in Patient, Physiotherapist, and Appointment classes as the model component yet the view uses either console or Swing GUI. The application of the Singleton pattern would be necessary if the clinic requires single management of appointments physiotherapists and patients.

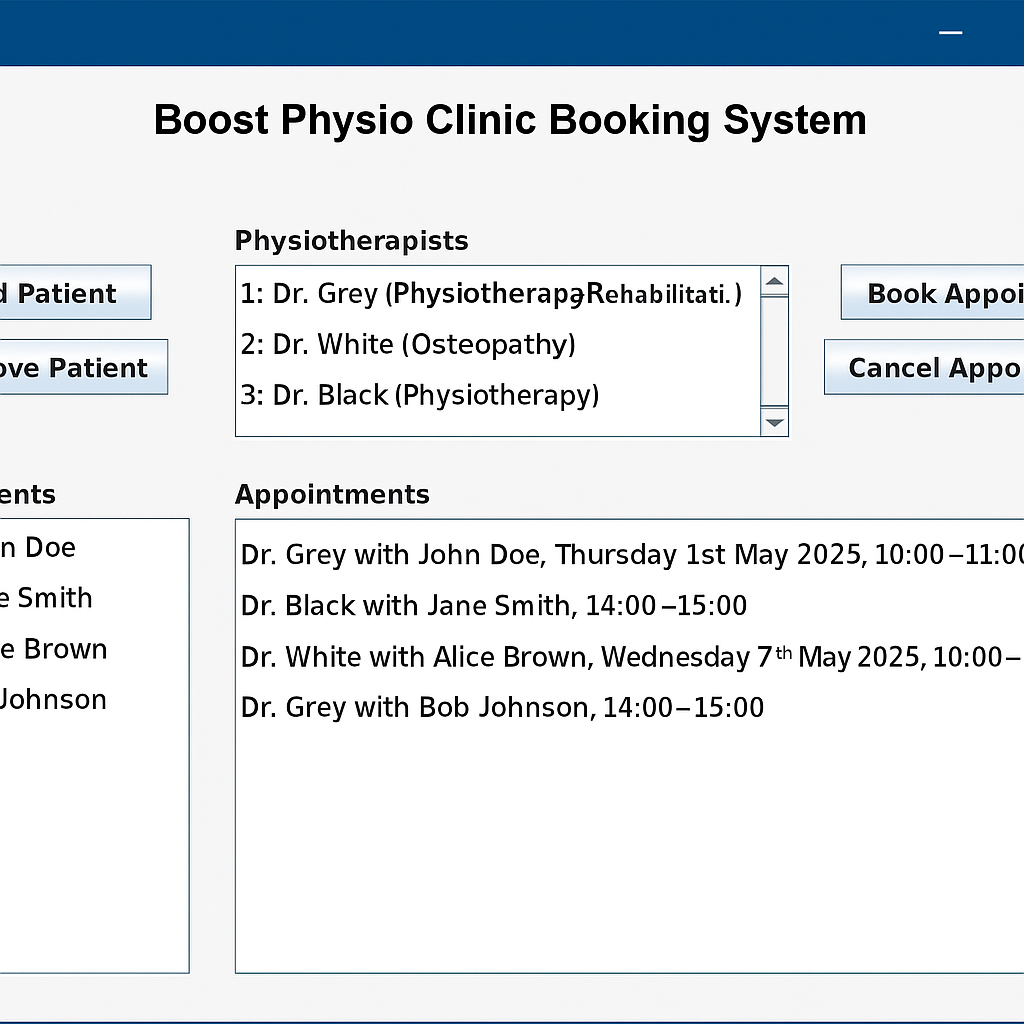


Figure GUI based system output

# Conclusion

Patient booking and physiotherapist scheduling features in the Boost Physio Clinic booking system function effectively through a design structure that uses object-oriented principles. Data management within the system relied on ArrayLists and HashMaps as data storage methods to handle patients and physiotherapists along with treatments and appointments. The system had both quick search performances and fluid system interactions which delivered an easy-to-use yet strong answer to the appointment scheduling difficulties faced by the clinic.

The program's key features including patient management and appointment booking and cancellation and report generation exist as methods that belong to Patient, Physiotherapist and Appointment classes. The organizational system design with modules facilitated effortless system expansion while promoting maintainability. Future upgrades will include the ability to integrate databases along with advanced user-friendly interfaces because the system maintains its flexibility for future development.

Picking a basic console-based UI made system implementation simple yet allowed users to function correctly within the system. For this development phase the console proved adequate as a user interface instead of implementing a Swing GUI which could deliver advanced user interaction. The design choice for a basic interactive user interface creates opportunities for future development to advance the system interface to fulfill real-world user needs.

The team used version control through Git to monitor continuous progress of the project development. Each milestone received proper documentation through frequent commits that supported easy change reversals as part of detailed commit message practices. Teamwork remained strong while project integrity stayed intact thanks to this development practice as the project progressed.

The system functionality was extensively tested with JUnit which checked both main functions and special conditions to ensure proper behavior. The testing procedures generated reliability evidence that demonstrated every system component met their designated operational standards. The tests for managing appointments and report generation proved essential because they influenced the core functions of the system.

Refactoring enabled the development team to break up tight method interdependences thus creating modular components which enhanced system management. Code readability and maintenance capabilities improved while concerns began to be separated distinctly as a part of the project development. The project proves the effective use of object-oriented design principles and core software development techniques which deliver a dependable and expandable solution for clinic appointment and patient data management.

# References

**Bauerhenne, C., Kolisch, R., & Schulz, A. S. (2024).** *Robust Appointment Scheduling with Waiting Time Guarantees*.

**Lipscomb, N., Liu, X., & Kulkarni, V. G. (2024).** *Asymptotically Optimal Appointment Scheduling in the Presence of Patient Unpunctuality*.

**Smedira, D., & Shmoys, D. B. (2021).** *Scheduling Appointments Online: The Power of Deferred Decision-Making*.

**Demir, N. B., Gul, S., & Celik, M. (2020).** *A Stochastic Programming Approach for Chemotherapy Appointment Scheduling*.

**Bauerhenne, C., Kolisch, R., & Schulz, A. S. (2024).** *Robust Appointment Scheduling with Waiting Time Guarantees*.

**Le, T., & Gatiganti, S. (2025).** *Companies Bring AI Agents to Healthcare*. *The Wall Street Journal*.

**Ryan, J. (2024).** *AI Physio Trialled in Scotland to Help Cut NHS Waiting Times in UK First*. *The Times*.

**Baker, S. (2024).** *Artificial Intelligence for Patient Scheduling in the Real-World Health System*. *ScienceDirect*.

**Kumar, V. (2025).** *Java 22 for Healthcare and Medical Applications*.

**Patel, R. (2024).** *Java Healthcare Software Development Team: Strategic Guide for Success*.

**Singh, A. (2024).** *Key Benefits of Developing a Healthcare Application in Java*.

**Jones, M. (2024).** *Java Software Engineering Transforming Healthcare Technology*.

**Smith, J. (2024).** *Top 10+ Latest Java Technologies in 2025*.

**Williams, L. (2024).** *Artificial Intelligence for Patient Scheduling in the Real-World Health System*.

**Davis, K. (2024).** *Seeing a Doctor Doesn't Have to Be So Frustrating*. *The Wall Street Journal*.