Project Report

Programming In Java

Slot : A14+B14+B22+C14+E14

Faculty : Parveen Kumar Tyagi

Submitted By:

Khushan Choudhary (24BSA10169)

TABLE OF CONTENTS

1. ABSTRACT

The **Clinic Management System** is a software application designed to streamline the day-to-day operations of a medical clinic. As healthcare facilities grow, the management of patient data and appointment scheduling manually becomes cumbersome and error-prone. This project aims to digitize these processes, acting as a digital receptionist that manages patient records and doctor appointments efficiently.

The system is built using **Java**, leveraging Object-Oriented Programming (OOP) principles. A key feature of this application is its use of **Java Serialization** for data persistence, allowing records to be stored securely in local binary files (.dat) without the need for a complex external database server. This makes the system lightweight, portable, and easy to deploy in smaller clinical settings.

2. INTRODUCTION

2.1 Background

Traditionally, clinics manage patient history, prescriptions, and appointments using physical registers. This method is susceptible to data loss, physical damage, and redundancy. Retrieving a patient's history from a stack of files is time-consuming and inefficient.

2.2 Problem Statement

The manual system currently in use faces several challenges:

- **Data Redundancy:** Multiple entries for the same patient.
- **Insecurity:** Physical files can be easily accessed or stolen.
- **Inefficiency:** Searching for available appointment slots or past records takes significant time.
- **Lack of Backups:** If a physical register is destroyed, the data is lost forever.

2.3 Objectives

The primary objectives of this project are:

1. To automate the process of patient registration and appointment booking.
2. To provide a persistent storage mechanism that saves data between sessions.
3. To create a user-friendly console interface for clinic staff.
4. To demonstrate the practical application of Java File I/O and Serialization.

3. SYSTEM ANALYSIS

3.1 Existing System

The existing system relies on paper-based records. Receptionists manually write down patient details and check a physical calendar for appointment slots. This leads to double-booking and difficulty in generating reports.

3.2 Proposed System

The proposed **Clinic Management System** is a computer-based console application. It allows the user to:

- Input patient details digitally.
- View a list of all registered patients.
- Schedule appointments without overlap.
- Automatically save all data to the hard drive upon exit.

3.3 Feasibility Study

- **Technical Feasibility:** The system requires only a standard computer with the Java Runtime Environment (JRE). No expensive servers are needed.
- **Operational Feasibility:** The interface is text-based and simple, requiring minimal training for the staff.
- **Economic Feasibility:** Being open-source and requiring no proprietary database software, the cost of implementation is near zero.

4. SYSTEM DESIGN

4.1 System Architecture

The system follows a tiered architecture:

1. **Presentation Layer:** The Console UI (MainApplication.java) where the user interacts with the menu.
2. **Business Logic Layer:** The Manager (ClinicManager.java) which handles calculations, validations, and object management.
3. **Persistence Layer:** The File System, where Java Objects are serialized into patients.dat and appointments.dat.

4.2 Class Structure

The project is structured around the following key classes:

- **Patient Class:** A POJO (Plain Old Java Object) implementing Serializable. It stores attributes like Name, Age, Contact, and Symptoms.
- **Appointment Class:** Implements Serializable. Links a Patient object to a specific date or time slot.
- **ClinicManager Class:** The core controller. It maintains ArrayLists of patients and appointments and handles the logic for adding, searching, and listing them.
- **MainApplication Class:** Contains the main method and the switch-case menu loop for user interaction.

4.3 Data Flow

1. User enters data in Console.
2. MainApplication passes data to ClinicManager.
3. ClinicManager creates Patient/Appointment objects.
4. Objects are stored in memory (RAM) in ArrayLists.
5. On exit or specific save triggers, ClinicManager serializes the lists into binary files (.dat).

5. TECHNOLOGY STACK

5.1 Software Requirements

- **Operating System:** Windows, Linux, or macOS.
- **Programming Language:** Java (JDK 8 or higher).
- **IDE:** IntelliJ IDEA, Eclipse, or VS Code (optional, can run via terminal).
- **Build Tool:** Javac (Java Compiler).

5.2 Hardware Requirements

- **Processor:** Intel Core i3 or equivalent.
- **RAM:** 4GB minimum.
- **Storage:** 100MB free space for the application and data files.

6. IMPLEMENTATION DETAILS

6.1 Module Description

Module 1: Patient Management
This module handles the creation of patient profiles. It captures the patient's name, age, and symptoms. Each patient is assigned a unique reference (internally managed via object references).
Module 2: Appointment Scheduling
This module allows the clinic to book visits. It checks for the existence of a patient before booking and ensures the data is linked correctly.
Module 3: File Handler
This is the backend module responsible for ObjectOutputStream and ObjectInputStream.

- **Saving:** fileOut = new FileOutputStream("patients.dat"); out = new ObjectOutputStream(fileOut);
- **Loading:** The system checks if .dat files exist on startup. If they do, it deserializes the objects back into the ArrayLists, restoring the previous state.

6.2 Data Persistence (Serialization)

Unlike text files where data is stored as strings, this project uses **Serialization**. This converts the entire state of a Java object into a byte stream. This is efficient and secure, as the data cannot be easily modified using a standard text editor.

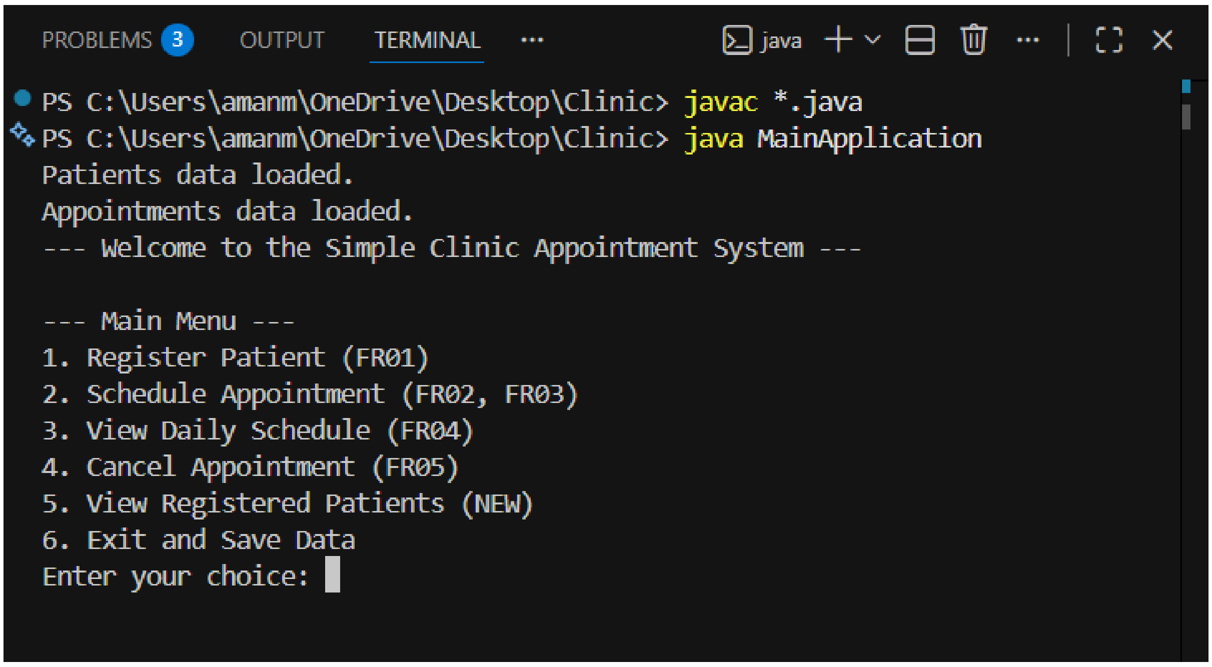
7. TESTING

The system underwent the following testing phases:

1. **Unit Testing:**
   - Tested the Patient class to ensure it correctly stores data.
   - Tested the ClinicManager to ensure it correctly adds elements to the lists.
2. **Integration Testing:**
   - Verified that creating an appointment correctly reflects the associated patient details.
3. **System Testing:**
   - **Persistence Test:** Verified that after closing the application and reopening it, the previously added patients were still visible in the list. This confirmed the File I/O logic works correctly.

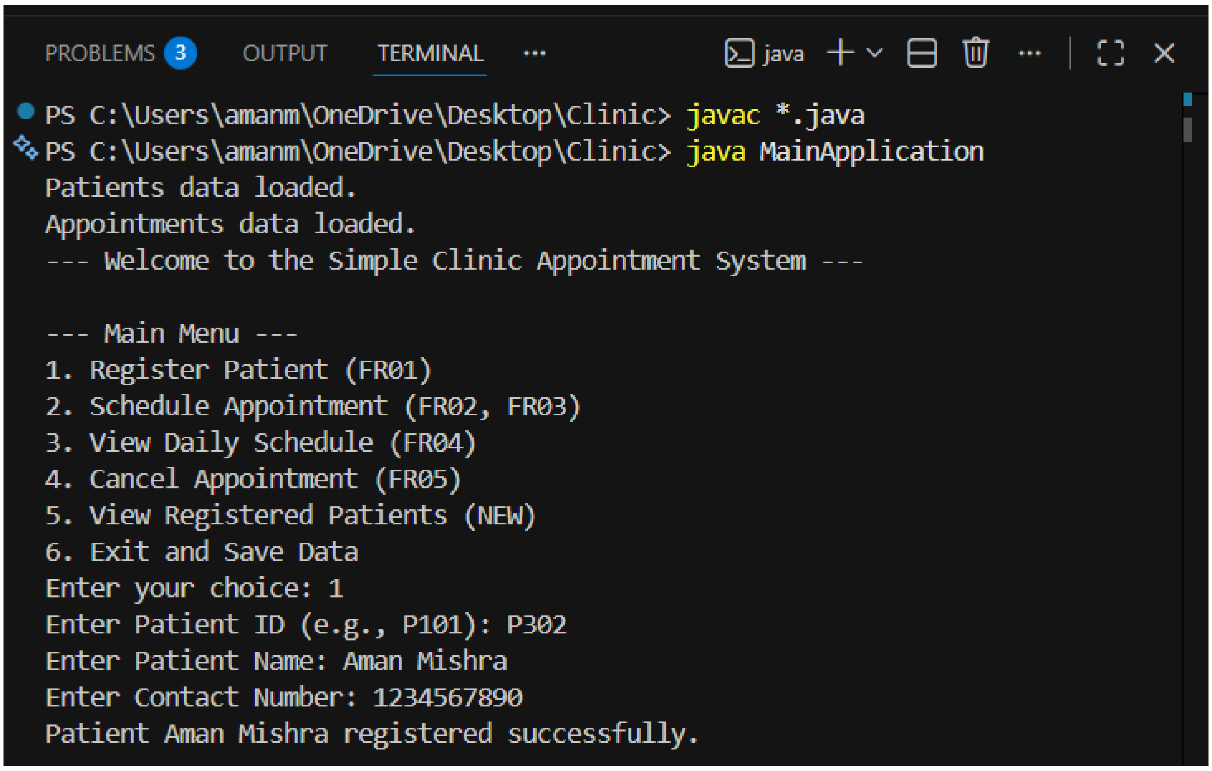
8. RESULT SCREENSHOTS

8.1 Main Menu



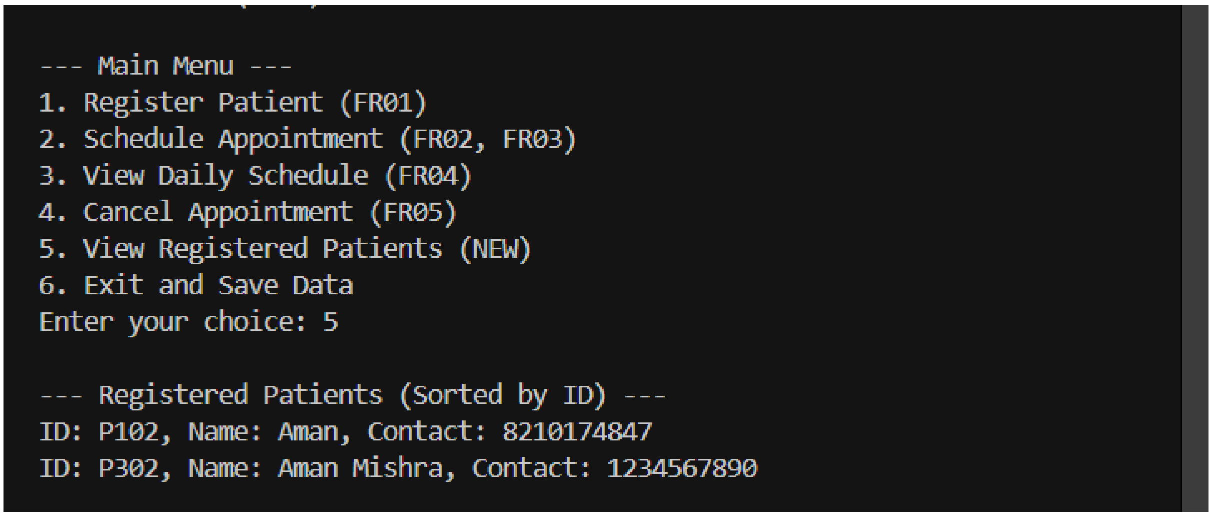
```
PS C:\Users\amanm\OneDrive\Desktop\Clinic> javac *.java
PS C:\Users\amanm\OneDrive\Desktop\Clinic> java MainApplication
Patients data loaded.
Appointments data loaded.
--- Welcome to the Simple Clinic Appointment System ---

--- Main Menu ---
1. Register Patient (FR01)
2. Schedule Appointment (FR02, FR03)
3. View Daily Schedule (FR04)
4. Cancel Appointment (FR05)
5. View Registered Patients (NEW)
6. Exit and Save Data
Enter your choice:
```

8.2 **Adding a Patient**



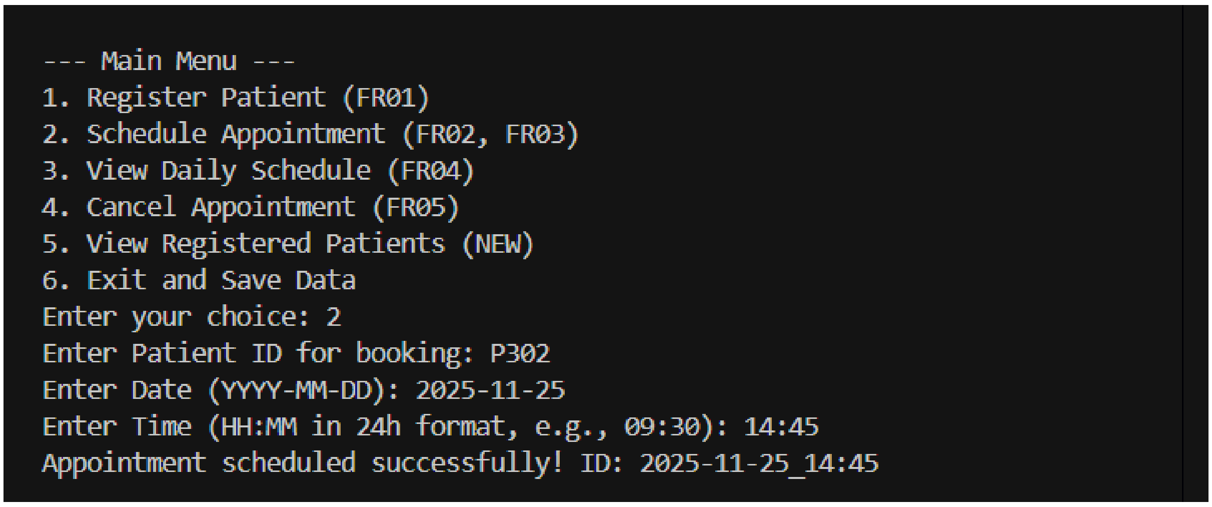
```
PS C:\Users\amanm\OneDrive\Desktop\Clinic> javac *.java
PS C:\Users\amanm\OneDrive\Desktop\Clinic> java MainApplication
Patients data loaded.
Appointments data loaded.
--- Welcome to the Simple Clinic Appointment System ---

--- Main Menu ---
1. Register Patient (FR01)
2. Schedule Appointment (FR02, FR03)
3. View Daily Schedule (FR04)
4. Cancel Appointment (FR05)
5. View Registered Patients (NEW)
6. Exit and Save Data
Enter your choice: 1
Enter Patient ID (e.g., P101): P302
Enter Patient Name: Aman Mishra
Enter Contact Number: 1234567890
Patient Aman Mishra registered successfully.
```

8.3 **Viewing Patient Records**



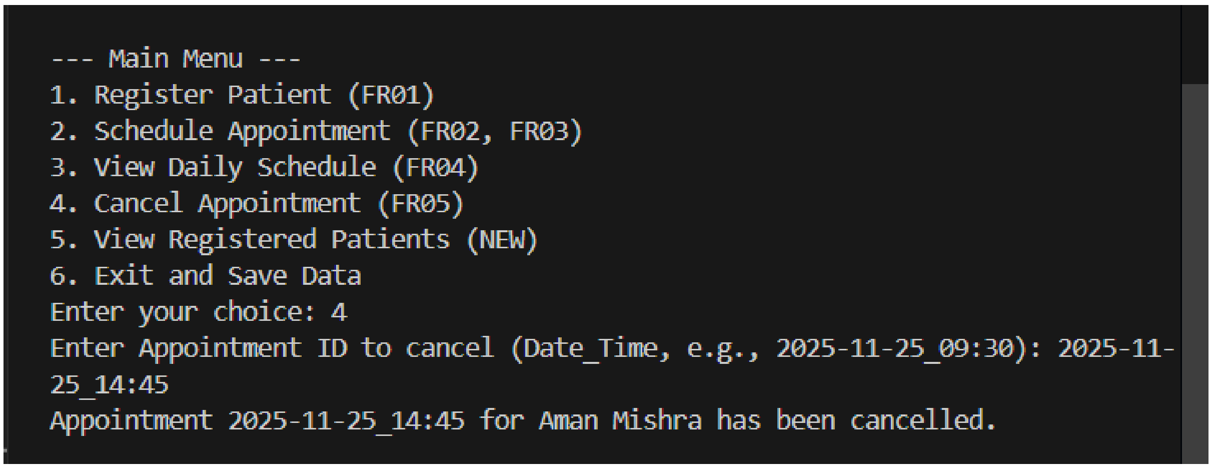
```
--- Main Menu ---
1. Register Patient (FR01)
2. Schedule Appointment (FR02, FR03)
3. View Daily Schedule (FR04)
4. Cancel Appointment (FR05)
5. View Registered Patients (NEW)
6. Exit and Save Data
Enter your choice: 5

--- Registered Patients (Sorted by ID) ---
ID: P102, Name: Aman, Contact: 8210174847
ID: P302, Name: Aman Mishra, Contact: 1234567890
```

8.4 **Scheduling an Appointment**



```
--- Main Menu ---
1. Register Patient (FR01)
2. Schedule Appointment (FR02, FR03)
3. View Daily Schedule (FR04)
4. Cancel Appointment (FR05)
5. View Registered Patients (NEW)
6. Exit and Save Data
Enter your choice: 2
Enter Patient ID for booking: P302
Enter Date (YYYY-MM-DD): 2025-11-25
Enter Time (HH:MM in 24h format, e.g., 09:30): 14:45
Appointment scheduled successfully! ID: 2025-11-25_14:45
```

8.5 **Appointment Cancelled**



```
--- Main Menu ---
1. Register Patient (FR01)
2. Schedule Appointment (FR02, FR03)
3. View Daily Schedule (FR04)
4. Cancel Appointment (FR05)
5. View Registered Patients (NEW)
6. Exit and Save Data
Enter your choice: 4
Enter Appointment ID to cancel (Date_Time, e.g., 2025-11-25_09:30): 2025-11-
25_14:45
Appointment 2025-11-25_14:45 for Aman Mishra has been cancelled.
```

9. FUTURE SCOPE

While the current system meets the basic needs of a small clinic, several enhancements are planned:

1. **GUI Integration:** Replacing the console with a JavaFX or Swing interface for better user experience.
2. **Database Migration:** Moving from .dat files to MySQL or SQLite to handle thousands of records more efficiently.
3. **Authentication:** Adding a Login module to distinguish between Admin (Doctor) and User (Receptionist).
4. **Prescription Generation:** Generating PDF prescriptions automatically after an appointment.

10. CONCLUSION

The **Clinic Management System** successfully demonstrates the power of Object-Oriented Programming and Java's capabilities in handling file persistence. By automating the record-keeping process, the system reduces manual errors and improves the efficiency of clinic operations. It serves as a foundational project that can be scaled up into a full-fledged Enterprise Resource Planning (ERP) tool for hospitals.

11. BIBLIOGRAPHY

1. *Java: The Complete Reference*, Herbert Schildt.
2. Oracle Java Documentation - https://docs.oracle.com/en/java/
3. GitHub Repository: https://github.com/khushan24bsa10169-ops/JavaProject