MEDTRACK: A Medical Appointment and Records Management System

Pedro Ramirez

Assignment 1

4/29/2025C

For this assignment, you will describe and implement release 1 of your term project. You will appropriately incorporate *an abstract class*, *inheritance*, *upcasting or downcasting*, and *polymorphism*. Choose a demanding project that interests you—preferably for the whole term, but you can introduce an additional project later if necessary. The instructor and your facilitator will be happy to help with a topic. You are expected to build a challenging application because you will be leveraging AI to the maximum and we are interested in what you do with it. We expect you to select a project with much more scope than you can accomplish in the course. We will not require you to complete every aspect of it. What we do expect is that you specify and implement an additional set of demanding but do-able requirements each week.

For this assignment only, the application is not required to read input from a file: you can build all data into the code if you wish.

Leverage an AI generator such as ChatGPT as much as you can to create a real-world application. As described in the evaluation criteria below, your work will be assessed in terms of *your value added* (not simply on AI-generated material). Your value added consists of your choice of prompts together with your edits and additions to AI-generated material that result in capable and high quality code. Show your value added in red font and by means of explanations. For figures, insert comments (in red) that describe clearly your value added.

Please provide all code in text format, not in screenshots, so you can highlight in red your value added. If you performed significant prompt work, please note this in the relevant sections with added explanations. Accompany code and diagrams with explanations.

Leveraging AI at the start of a project is easy. Continuing to leverage it as the project grows, however, requires discipline and well-structured code—a skill we will instruct you on to gain your mastery.

Submit this completed Word document. Insert your material as indicated. Please observe and retain the gray text. Your materials—in black 12-point Times New Roman—should not exceed 5 pages excluding the gray instructions, references, figures, and appendices. Use the Appendix sections for additional material if you need to and refer to them in the document body. These will be read only on an as-needed basis.

Please develop in Eclipse—preferably—or else IntelliJ (talk to your facilitator about exceptions). As you code, use JUnit tests whenever possible—package-by-package, class-by-class, and method-by-method, except for trivial methods and those requiring I/O. Use testing classes for testing the latter. Keep the evaluation criteria in mind, listed at the end.

Housekeeping:

1. Include a ReadMe file that contains necessary execution notes and describing where to run the application from. All JUnit tests will be assumed runnable.
2. After you have completed the questions, make sure you have saved the file.
3. Please save this completed document with the file name: METCS622_Assignment1_FirstnameLastname.
4. To upload the completed Draft Assignment 1, click the "Browse My Computer" to upload your Word file, and then click "Submit".
5. Export your project from your IDE using its export feature and provide it as a second attachment.

1. SUMMARY DESCRIPTION

Give a one- or two-paragraph overall description of your proposed term project—half-page (12-point Times New Roman) limit. By the end, term projects will incorporate most of the techniques discussed in the course. To do this, you may need to alter the

direction of your project or introduce an additional project in future, but a wide scope has better potential for incorporating techniques. You will probably find it useful to use a project acronym.

MEDTRACK is a Java-based application for managing medical appointments, records, and user roles such as patient, doctor, and administrator. This project employs object-oriented principles, including inheritance and polymorphism, to simulate real-world interactions in a clinic or hospital setting. In this initial release, the system focuses on creating and identifying users (patients and doctors), viewing available doctors, and booking appointments. The abstract class User provides a common interface for patients and doctors, while polymorphism enables dynamic method behavior based on the role. The long-term scope includes file I/O, patient history management, and GUI integration. The application will be created with IntelliJ IDEA and will include JUnit tests.

2. I/O EXAMPLE FROM PROJECTED COMPLETED PROJECT

Provide an example of projected *concrete* output of the anticipated complete application—for example input. You will not be held to fulfilling exactly this—it is just explanatory at this point, to indicate where your project is going. We recognize that project direction and details will change as the term progresses. This section refers to the project as a whole, not just to what you will produce this week.

Welcome to MEDTRACK

Please select your role:

1. Patient

2. Doctor

3. Administrator

4. Register New User

> 1

Enter your Patient ID:

> P1001

Welcome back, John Doe.

Main Menu:

1. Book an Appointment

2. View Appointment History

3. Exit

> 1

Available Doctors:

[2001] Dr. Emily Smith – Family Medicine (no available)

[2002] Dr. Daniel Lee – Pediatrics (available)

Enter the doctor ID:

> 2002

Enter preferred appointment date (YYYY-MM-DD):

> 2025-05-10

Enter preferred time (HH:MM, 24-hour format):

> 14:00

Appointment confirmed:

John Doe with Dr. Daniel Lee on 2025-05-10 at 14:00.

Confirmation Code: APT-P1001-20250510-1400

Supply the functional requirements that you accomplished for this assignment, i.e., functionality that the application provides for the user. Please state requirement in declarative form, as illustrated in the examples, because we want to know the functionality intended (*what*, not *how*). For example, the following is *not* a proper functional requirement (it is a software design statement): *TicTac will have a class for O's and a class for X's.* It is common to mistake design elements like this for functional requirements. To get started, state what the application will accept as input.

See here for how to state requirements and how to specify functions (and methods). These formats are required.

Keep in mind that the implementation of your requirements will incorporate *an abstract class*, *inheritance*, *upcasting or downcasting*, and *polymorphism*; that will probably influence the requirements you choose to implement in this assignment.

3.1 **Requirement Title**: Book an Appointment with a Doctor

The system enables patients to book an appointment by selecting a doctor from a list, providing a preferred date and time, and confirming the booking. The program will validate the input, check for availability, and return a confirmation message with a unique appointment code.

3.2 Your second requirement *title* replaces this

Your response should replace this.

4 ILLUSTRATIVE OUTPUT FROM (ACTUAL) IMPLEMENTATION

Provide illustrative output from your implemented application (so far) showing that the requirements have been met. State what *class.method*(s) implement each requirement.

Your response should replace this.

5 YOUR DIRECTORY

Show a screenshot of your directory. This should include a parallel directory of JUnit tests where possible—package-by-package, class-by-class, and method-by-method, except for trivial ones.

Your response should replace this.

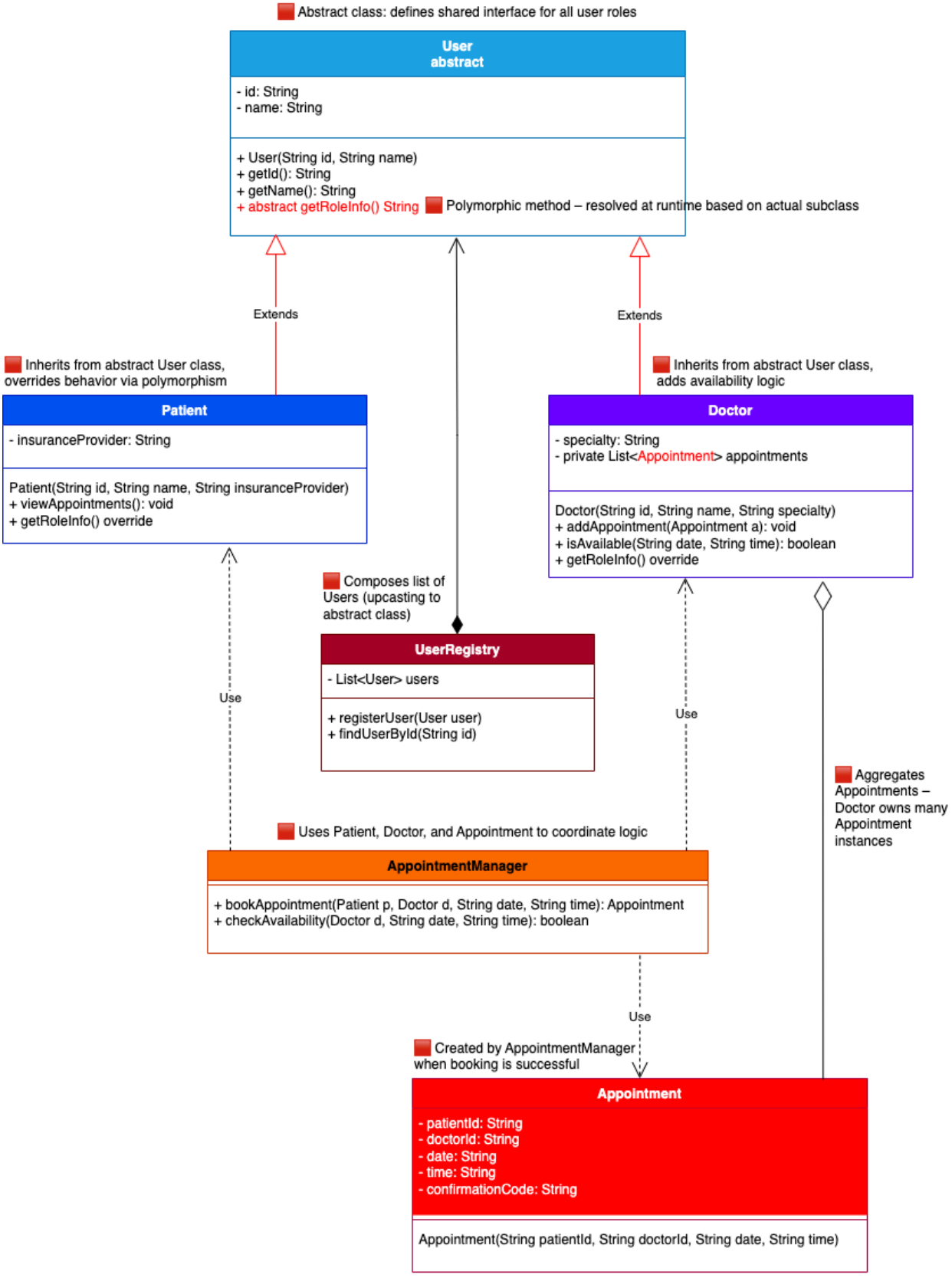
6 TECHNIQUES IMPLEMENTED

Your implementation should include *inheritance*, *polymorphism*, and *either an abstract class or interface* at least once, and in a manner that is useful to your application. Explain where and how you applied these, using the format below.

6.1 Class model and Sequence Diagram

Identify where you included *inheritance*, *polymorphism*, and *abstract classes* or *interfaces* in your class model. Make classes and members *static* or not as per their intended usage. To do this use tools (e.g., Visio, Lucidchart, or draw.io), PowerPoint, or a combined model as in this example (which you are free to cut and paste from). Insert indications in red to show where the three features below apply.

See Class Diagram and Sequence Diagram: Provided on following page with value-added labels in red.


Abstract class: defines shared interface for all user roles
User
abstract
- id: String
- name: String
+ User(String id, String name)
+ getId(): String
+ getName(): String
+ abstract getRoleInfo() String
Polymorphic method – resolved at runtime based on actual subclass
Extends
Extends
Inherits from abstract User class, overrides behavior via polymorphism
Inherits from abstract User class, adds availability logic
Patient
- insuranceProvider: String
Patient(String id, String name, String insuranceProvider)
+ viewAppointments(): void
+ getRoleInfo() override
Doctor
- specialty: String
- private List<Appointment> appointments
Doctor(String id, String name, String specialty)
+ addAppointment(Appointment a): void
+ isAvailable(String date, String time): boolean
+ getRoleInfo() override
Composes list of Users (upcasting to abstract class)
UserRegistry
- List<User> users
+ registerUser(User user)
+ findUserById(String id)
Use
Use
Aggregates Appointments – Doctor owns many Appointment instances
Uses Patient, Doctor, and Appointment to coordinate logic
AppointmentManager
+ bookAppointment(Patient p, Doctor d, String date, String time): Appointment
+ checkAvailability(Doctor d, String date, String time): boolean
Use
Created by AppointmentManager when booking is successful
Appointment
- patientId: String
- doctorId: String
- date: String
- time: String
- confirmationCode: String
Appointment(String patientId, String doctorId, String date, String time)

6.2 Code showing an abstract class or interface

Show the relevant code (only) implementing this and explain why an abstract class or interface is appropriate here. It should be clear where the code is located (class and method).

Your response should replace this.

6.3 Code showing polymorphism

Show the relevant code (only) and explain why *polymorphism* is appropriate here. Recall that polymorphism is implemented in one of two ways – overriding methods in subclasses or overloading methods in the same class where the method signatures are different – and allowing the language runtime to dynamically invoke the correct method. It should be clear where the code is located (class and method).

Your response should replace this.

6.4 Code showing upcasting or downcasting

Show the relevant code (only) and explain why upcasting or downcasting is appropriate here. It should be clear where the code is located (class and method).

Your response should replace this.

7 EVALUATION OF ASSIGNMENT

Criterion (based on your value added)	**D**	**C**	**B**	**A**	**Letter Grade**	**%**
Functionality and technical correctness of your value added	**Little** technical justification.	Correct technically. **Satisfactory** technical explanation.	Correct technically. **Good**. Did what was required. Very capable application	Correct technically. Went significantly **beyond** what's required. Extremely capable application.		0.0
Clarity of your value added	**Unclear**	**Somewhat** clear	**Clear**	Everything **competely** clear		0.0
Overall understanding as evidenced by your value added	**Minor** understanding evidenced	**Satisfactory** understanding evidenced	Evidence of **good** understanding throughout	Evidence throughout of **thorough** understanding		0.0
				Assignment Grade:		0.0
The resulting grade is the average of these, using A+=100 (outstanding--rare), A=95 (excellent in all ways), A-=90 (excellent),						
B+=87 (excellent / very good), B=85 (very good), B-=80 (good) etc.						
To obtain an A grade for the course, your weighted average should be >93. A-:>=90. B+:>=87. B:>83. B-:>=80 etc.						

Appendix 1 (if needed; should be referenced above, and will be read as-needed only)

Appendix 2 (if needed; should be referenced above, and will be read as-needed only)