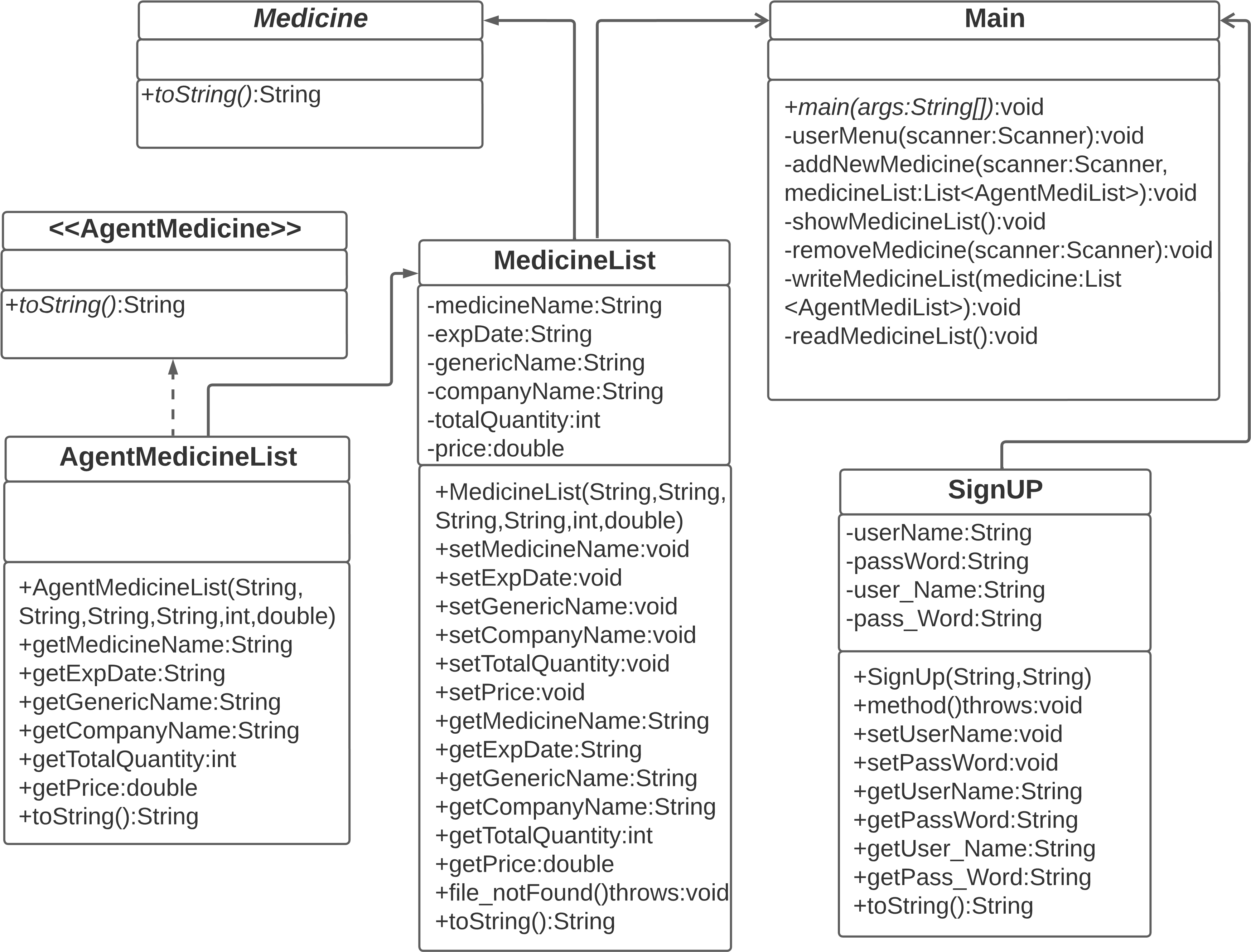
**Pharmacy Store**

Project summary: The pharmacy store project is a Java project written without a database. It utilizes abstract classes, interfaces, scanner, file handling, polymorphism, exception handling, inheritance, and aggregation. The UML diagram for this project is provided.



The project functions as follows: There are two predefined users:

1. User Name: Amir

Password: 1234aa

2. User Name: Amir Hamza

Password: 1234

Upon logging in as User 1, the user menu will be displayed, allowing the user to choose various actions. If User 2 logs in, the medicine details will be shown.

**1. Medicine (Abstract Class):**

**Methods:**

* - toString(): An abstract method.

**2. MedicineList Class:**

**Attributes:**

* - medicineName: Represents the name of the medicine. This attribute has private access modifier and a data type of String.
* - expDate: Represents the expiry date of the medicine. This attribute has private access modifier and a data type of String.
* - genericName: Represents the generic name of the medicine. This attribute has private access modifier and a data type of String.
* - companyName: Represents the name of the company producing the medicine. This attribute has private access modifier and a data type of String.
* - totalQuantity: Represents the total quantity of the medicine. This attribute has private access modifier and a data type of integer.
* - price: Represents the price of the medicine. This attribute has private access modifier and a data type of double.

**Methods:**

* - Constructors: Initializes the Medicine object with the provided values.
* - Getters and setters: Allow access to and modification of the attribute values.
* - file\_notFound(): Throws an "IOException" and prints a "File not found" message.
* - toString(): Responsible for returning a string representation of the MedicineList class.

**3. SignUp Class:**

**Attributes:**

* - userName: Represents the username for user login.
* - passWord: Represents the password for user login.
* - user\_Name: Represents a predefined username for comparison.
* - pass\_Word: Represents a predefined password for comparison.

**Methods:**

* - Constructors: Initializes the SignUp object with the provided username and password.
* - Getters and setters: Allow access to and modification of the attribute values.
* - method(): Throws an "IOException" with a message stating "Incorrect password input.".
* - toString(): Returns a string representation of the SignUp object.

4. **AgentMedicine Interface:**

**Methods:**

* - toString(): Declares a "toString()" method to be implemented by the AgentMedilist class.

5. **AgentMedilist Class:**

* - Inherits the MedicineList class and implements the AgentMedicine interface.
* - Constructors: Initializes the AgentMediList object with the provided values and calls the parent constructor.
* - Overridden methods: Overrides the getter methods of the parent class.
* - toString(): Overrides the "toString()" method of the parent class.

**6. Main Class:**

* - Contains the "main" method, which serves as the entry point of the program.
* - Handles user login by creating a SignUp object and checking the provided username and password.
* - The main method starts by creating a new instance of the Scanner class to read user input.
* - The program prompts the user to enter their username and password for login authentication.
* - The SignUp class is used to validate the username and password provided by the user. If the login is successful, the program proceeds to the next steps; otherwise, it displays an error message and terminates.
* - If the login is successful, the "userMenu" method is called. This method handles the main menu functionality, such as adding a new medicine, showing the medicine list, removing a medicine, or exiting the program.
* - Inside the "userMenu" method, a list called "medicineList" is created to store instances of the AgentMediList class, representing the medicines.
* - A while loop is used to repeatedly display the menu and process the user's choices until the user chooses to exit.
* - Depending on the user's choice, the program either adds a new medicine to the "medicineList", shows the current medicine list, removes a medicine from the list, or exits the loop.
* - The "addNewMedicine" method is responsible for collecting input from the user and creating a new instance of the AgentMediList class to represent a medicine. The newly created medicine object is then added to the "medicineList".
* - The "showMedicineList" method reads and displays the contents of the "medicine\_list.txt" file, which contains the medicine list.
* - The "removeMedicineList" method allows the user to specify a medicine name to be removed from the "medicineList". It reads the file, searches for the specified medicine name, removes it from the list, and updates the file accordingly.
* - The "removeMedicineList" method is called to write the updated "medicineList" to the "medicine\_list.txt" file, effectively saving the changes made during program execution.
* - The "readMedicineList" method is responsible for reading and displaying the contents of the "medicine\_list.txt" file.
* - Overall, the Main class handles user authentication, provides a menu-driven interface for managing the medicine list, and ensures that the changes made to the list are reflected in the file storage.