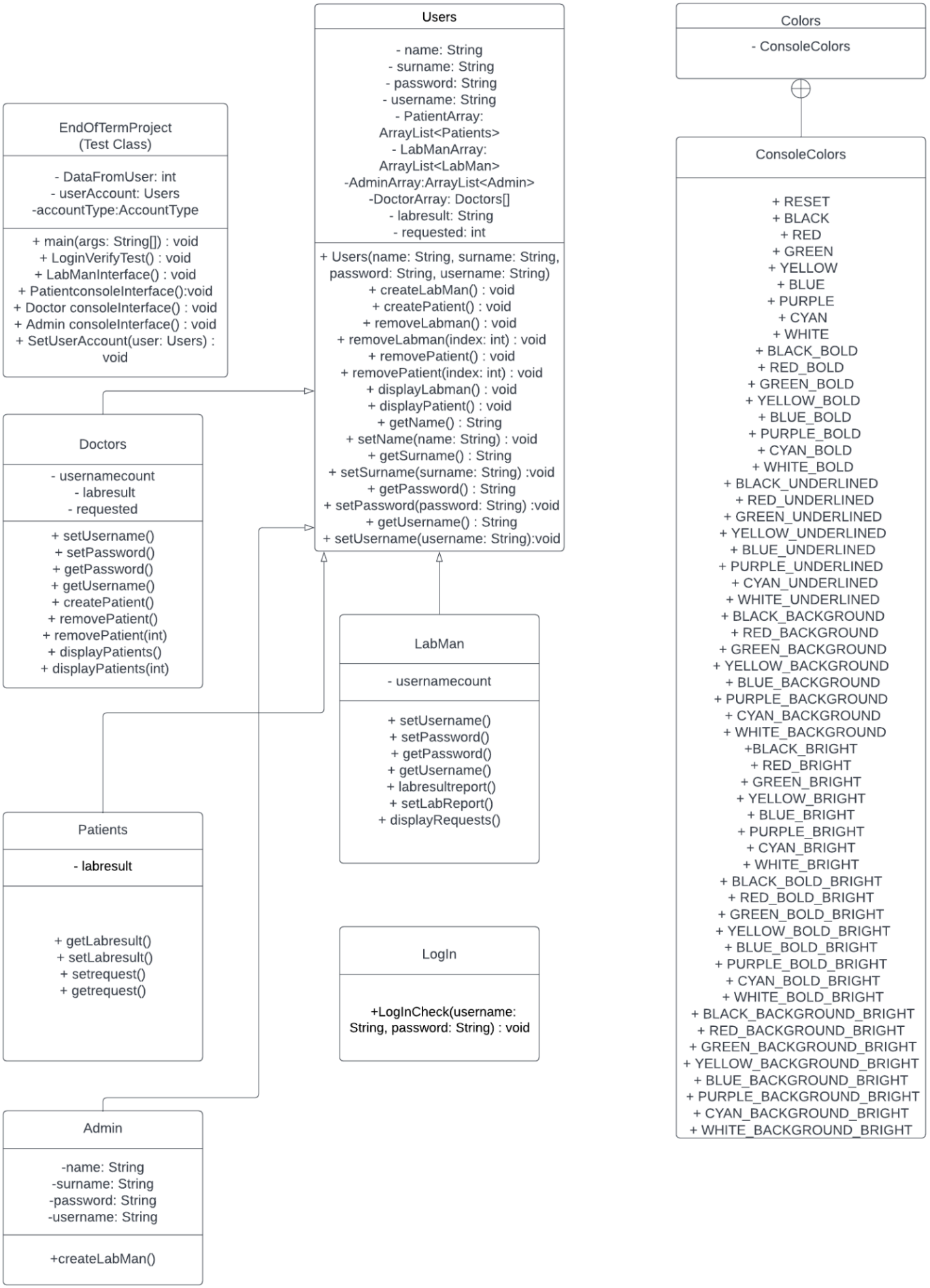
Class diagram of project on UML format.



Information for diagram

 :It is for showing the inheritance between classes.

 :It is for showing the nested class situation.

- \+ : This symbol which is in front of the method name indicates that it is public.
- − : This symbol which is in front of the method name indicates that it is private or in the attitudes indicates that it is protected or private.

General Information Of Project

In the project, there are several classes: EndOfTermProject, LogIn,Colors, Users, Doctors, LabMan, Patients, and Admin.

The EndOfTermProject class serves as the main class, where the program begins its execution. It contains a static Scanner object that allows the program to receive input from the user, and a static integer variable that stores the input received from the user. It also has an enum called AccountType, which lists four possible account types: PATIENT, DOCTOR, LABMAN, and ADMIN. Additionally, there is a static variable called accountType, which will store the type of account that the user has logged into, and a static Users object called userAccount, which will store the user's account information.

The EndOfTermProject class contains several methods, including main(), SetUserAccount(), and LoginVerifyTest(). The main() method is the entry point of the program and contains a while loop that continues to execute until the program is terminated. The SetUserAccount()

method allows the program to set the userAccount variable to a given Users object, while the LoginVerifyTest() method prompts the user to enter their username and password and then verifies the login information using the LogIn class.

The LogIn class contains a single method called LogInCheck(), which receives a username and password as input and checks if they match an existing account. If a match is found, the method sets the static accountType variable in the EndOfTermProject class to the appropriate account type and sets the userAccount variable to the matching account.
The Users class is an abstract class that serves as a base class for the Doctors, LabMan, Patients, and Admin classes. It contains variables for storing a user's name, surname, password, and username, as well as several abstract methods for setting and getting these variables. It also contains a static ArrayList called AdminArray, which is used to store Admin objects, and a static method called VerifyAccountMethod() which receives a username and password and checks if they match an existing account.

The Doctors, LabMan, Patients, and Admin classes all extend the Users class and therefore inherit its variables and methods. They each contain their own unique methods, which are used to perform specific tasks related to their respective roles.

The Colors class is a utility class that contains several constants for different colors that can be used in the console output of the program. It is used to add visual appeal to the program's user interface.
Finally, the LabMan and Patients classes both contain variables for storing lab test results.

The LabMan class contains a method for setting the lab test results for a patient, while the Patients class contains a method for retrieving the lab test results for a particular patient.
In summary, the provided code is for a program that simulates a lab automation system. It allows for the creation and management of user accounts for different roles, such as doctors, lab technicians.

Detailed Information Of The Project

EndOfTermProject Class (The test class)

- The EndOfTermProject is the main class of the program. It contains the main method which is the entry point for the program. It has a static Scanner object called 'sc' which is used to take input from the user. It also has a static variable called 'DataFromUser' which is used to store the integer value entered by the user.

- The class has an enum called 'AccountType' which has four elements: PATIENT, DOCTOR, LABMAN, and ADMIN. It also has a static variable called 'accountType' of the AccountType enum type. This variable is used to store the type of the user's

account (i.e. whether the user is a PATIENT, DOCTOR, LABMAN, or ADMIN). The class also has a static variable called 'userAccount' of the type Users. This variable is used to store the user's account object.

- The class has a static method called 'SetUserAccount' which takes a Users object as a parameter and sets the value of the 'userAccount' variable to the object passed to it. The main method first creates an Admin object called 'admin' and adds it to the 'AdminArray' list. It then creates an object of the EndOfTermProject class and enters an infinite loop. Inside the loop, if the 'accountType' variable is null, it calls the 'LoginVerifyTest' method. Otherwise, it enters a switch statement with 'accountType' as the expression. Depending on the value of 'accountType', it calls one of four methods: 'LabManInterface', 'PatientconsoleInterface', 'DoctorconsoleInterface', or 'AdminconsoleInterface'.

- The 'LoginVerifyTest' method first clears the buffer of the 'sc' Scanner object and then prompts the user to enter their username and password. It then calls the 'LogInCheck' method and passes the username and password entered by the user as arguments. If the 'LogInCheck' method returns an object of the Users class, the 'accountType' variable is set to the appropriate value (i.e. PATIENT, DOCTOR, LABMAN, or ADMIN) and the 'SetUserAccount' method is called to set the 'userAccount' variable to the object returned by the 'LogInCheck' method. If the 'LogInCheck' method returns null, it means that the username and password entered by the user do not match any existing accounts, and the user is prompted to try again.

- The casting that occurs in EndOfTermProject is used to treat a user object as a specific subclass of Users. For example, if the userAccount field is an instance of Doctors, it will be cast to a Doctors object in order to call methods that are specific to the Doctors class. This is an example of downcasting.

- Also, the casting in the EndOfTermProject class is an example of polymorphism. Polymorphism is a feature of object-oriented programming languages that allows objects of different types to be treated as if they are the same type. This is achieved through the use of inheritance and method overriding.
-
- In the EndOfTermProject class, polymorphism is achieved through the use of the instanceof operator and the switch statement. The instanceof operator is used to determine the type of an object, and the switch statement is used to execute different blocks of code depending on the type of the object.

- In the EndOfTermProject code, the try-catch statement is used to catch any exceptions that may occur when casting the userAccount object to the LabMan or Patients class. If an exception occurs, it will be caught and handled, allowing the program to continue executing instead of crashing.

Users Class

- The Users class is a base class that provides common properties and methods for the other classes that extend from it, such as Patients, Doctors, and LabMan.

- One of the properties of the Users class is a username which is a unique identifier for each user. There is also a password property for each user which is used to authenticate the user when they log in. The name and surname properties represent the first and last name of the user respectively.

- There are also several methods in the Users class. The setUsername() method is used to set the value of the username property for a user. The setPassword() method is used to set the value of the password property for a user. The getUsername() and getPassword() methods are used to retrieve the values of the username and password properties, respectively.

- The Users class also has several static properties and methods that are shared by all instances of the class. The PatientArray, DoctorArray, and LabManArray properties are arrays that are used to store instances of the Patients, Doctors, and LabMan classes respectively.

- The AdminArray is an ArrayList of Admin objects. It is used to store instances of the Admin class. The AdminArray appears to be a static member of the Users class, so it is accessible from anywhere in the code without the need to create an instance of the Users class.

- The VerifyAccountMethod is a method in the Users class that checks if the entered username and password match an existing account in the system. It takes in two parameters: a username and a password. It iterates through the DoctorArray, LabManArray, and PatientArray lists and checks if any of the objects in these lists have a matching username and password. If a match is found, the method returns the object. If no match is found, the method returns null.

LabMan Class

- The LabMan class represents a user with the role of a lab manager in the system. It extends the Users class, which means it inherits the attributes and behaviors of the Users class, but it also has additional attributes and behaviors specific to lab managers.

- One behavior specific to lab managers is the labresultreport() method, which allows a lab manager to update the lab result of a patient. It takes in a Patients object as an argument and sets the labresult attribute of that object to "+".

- Another behavior specific to lab managers is the displayRequests() method, which allows a lab manager to view all requests made by patients. It iterates through the PatientArray list and displays the username and request status of each patient. If the request status is less than 1, it prints a message indicating that there are no requests for that patient.

- The LabMan class also has a behavior for updating the lab results of patients, called setLabReport(). This method takes in a username and lab result as arguments and searches the PatientArray list for a patient with a matching username. If it finds a match, it sets the lab result of that patient to the specified value and sets the requested attribute to -1.

- The LabMan class also has an overridden version of the setUsername() and setPassword() methods from the Users class. These methods add a prefix to the specified username and password, respectively, before setting them for the lab manager object. This is likely done to ensure that the username and password of a lab manager are easily identifiable and distinct from those of other types of users.

Doctor Class

- The Doctor class extends the Users class, which means it inherits all the attributes and methods of the Users class. The Doctor class has a default constructor and also has a constructor that takes four arguments, name, surname, password, and username, which are passed to the superclass's constructor to initialize the inherited attributes.

- The Doctor class has a method called "setLabReport" which takes two arguments, a username and a labpresult. This method searches for a patient with the given username in the PatientArray list and sets the labresult attribute of that patient to the given labpresult.

- The Doctor class also has a "displayRequests" method which iterates through the PatientArray list and displays the usernames and requests of all patients. If a patient has no request, it displays a message indicating that there are no requests for that patient.

- The Doctor class also has getter and setter methods for the name, surname, password, and username attributes inherited from the Users class.

- Overall, the Doctor class represents a user with a Doctor account and provides methods for setting lab results for patients and displaying patient requests.

Patient Class

- The Patient class is a subclass of the Users class and represents a patient in the system. It contains several fields and methods specific to patients.

- The fields of the Patient class include labresult, which represents the lab results of the patient, and requested, which represents the number of times the patient has requested their lab results.

- The Patient class also has several methods. The setUsername and setPassword methods allow the patient to set their own unique username and password for logging into the system.

- The getUsername and getPassword methods return the patient's username and password, respectively.

- The getLabresult and setLabresult methods allow the patient to retrieve and set their lab results, respectively. The setrequest and getrequest methods allow the patient to set and retrieve the number of times they have requested their lab results.

- Overall, the Patient class plays an important role in the system by representing and managing the information and actions of patients within the system.

Admin Class

- The Admin class is a subclass of the Users class and is used to represent an administrative user in the system. It has several methods that allow the administrator to perform various tasks, such as creating new LabMan or Doctor accounts, or modifying the information of existing accounts.

- One of the methods in the Admin class is the createLabMan method, which allows the administrator to create a new LabMan account. This method takes four arguments: name, surname, password, and username. It then creates a new LabMan object with these arguments and adds it to the LabManArray list, which is a list of all the LabMan accounts in the system.

- The Admin class also has a method called setName, which allows the administrator to set the name of the Admin object. Similarly, there are methods for setting the surname, password, and username of the Admin object.

- The Admin class also has methods for getting the name, surname, password, and username of the Admin object. These methods are called getName, getSurname, getPassword, and getUsername, respectively.

- In addition to these methods, the Admin class also has a constructor that takes four arguments: name, surname, password, and username. This constructor is used to create a new Admin object with the specified name, surname, password, and username.

LogIn Class

- The LogIn class is a Java class that is responsible for verifying a user's login credentials. It has a single method called "LogInCheck" which takes in a username and a password as input parameters.
- The method first calls the "VerifyAccountMethod" of the Users class, which searches for a user in the list of users and returns the user object if found. If the user object is not null, it checks the type of the user object and sets the "accountType" and "userAccount" variables of the EndOfTermProject class accordingly.
- The "accountType" variable is an enumeration that can take on the values "PATIENT", "DOCTOR", "LABMAN" or "ADMIN". The "userAccount" variable is a reference to a user object.
- If the user object is null, it means that the login credentials are invalid and the "accountType" and "userAccount" variables are not set. The user is then prompted to try again.
- The LogIn class is used to verify the login credentials of a user before allowing them access to their account and the features it provides. It is a crucial part of the overall system as it ensures that only authorized users can access the system.

Color Class

- The Colors class appears to be a utility class that defines a set of constants representing different console colors. These constants are used to change the color of text displayed in the console.
-
- The consoleColors inner class defines a set of constants that represent different colors: BLACK, RED, GREEN, YELLOW, BLUE, PURPLE, CYAN, and WHITE. Each of these constants is a string that contains the escape codes necessary to change the color of text in the console.
-
- The RESET constant is a string that contains the escape code to reset the text color to its default value.