**Introduction:**

The purpose of this assignment is to develop a sudoko game which is consists by a series of 3x3 grids, into each the numbers one to nine must be placed exactly once.Each number can also be used only once in a column, and similarly in a row.In this game player need to create an account, after create account they need to login and select game level then start the game. After complete the game level player able to see the game Statistics.

.

**Task 1**

**Class list and class diagram:**

**Candidate class:**

Initial selection of classes is called the candidate class. It is important to indentity all candidate classes before starting or develop a new system. So that before starting the suduku game I have identify the candidate class from the scenario of this assignment. I have identify final class from those of candidate class for this sudoko game.  
This list of candidate class that are given below:

**Candidate class list:**

**Main** – this will be the main class of this game.

**EasyLevel** – this class will be used for generating easy level for this game.

**MediumLevel** – this class will be used for generating medium level for this game.

**HardLevel** – this class will be used for generating hard level for this game.

**SudokoSolver**– this class will be used solving the sudoko game.

**UnAssisted-** this class will be used full complete the sudoko game and submit for validation.

**Aided-** this class will be used for helping purpose.

**Progress-** To show the player progress

**SudokoValidation-** To check Sudoko is valid or not

**UserActivity** **–** this class will be used for create account.

**Final class diagram:**

Below here are the final class diagram:

|  |  |
| --- | --- |
| **Candidate class name** | **Reason of choose this class** |
| CreateAccount | This class will be help create user account |
| GameMenu | Main window will be show |
| Userstatistics | This class will contain user statistics |
| UserAccesability | This class help user acceess the application. |
| User | This class will contain user details. |
| UserProgress | This class will contain user progress details. |
| Sudoko | This will be the main class. |
| SudokoGenerator | This class generage the sudoko game. |
| SudokoSolver | This class will help to solve the sudoko game. |

**Class diagram:**

On below, here are the whole class diagram of this application with showing associations (with multiplicity), operations and attributes.

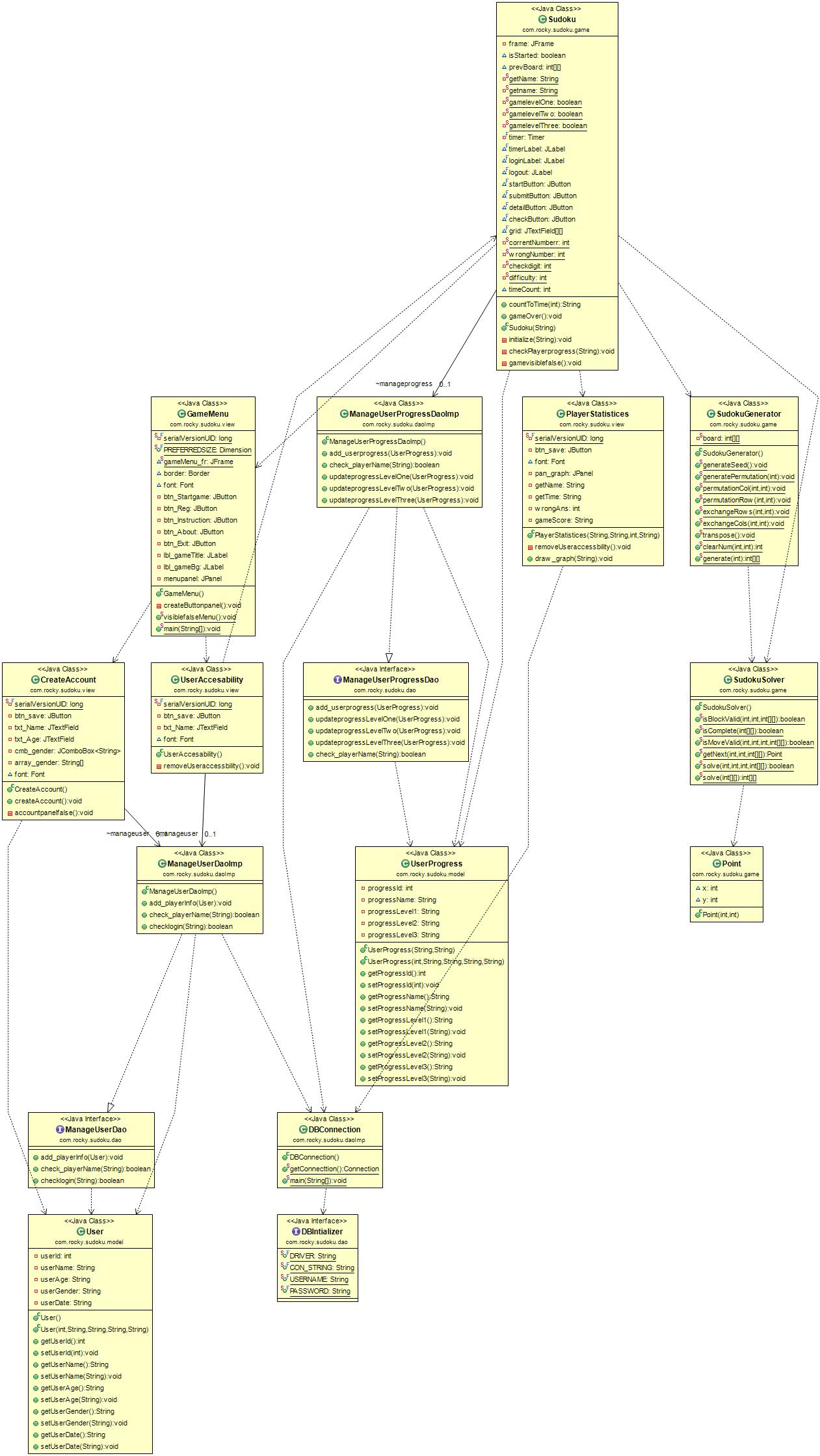


Figure 1: Class diagram for sudoko game

**Task 2 – Activity diagram:**

CreateUseraccount (create user account)

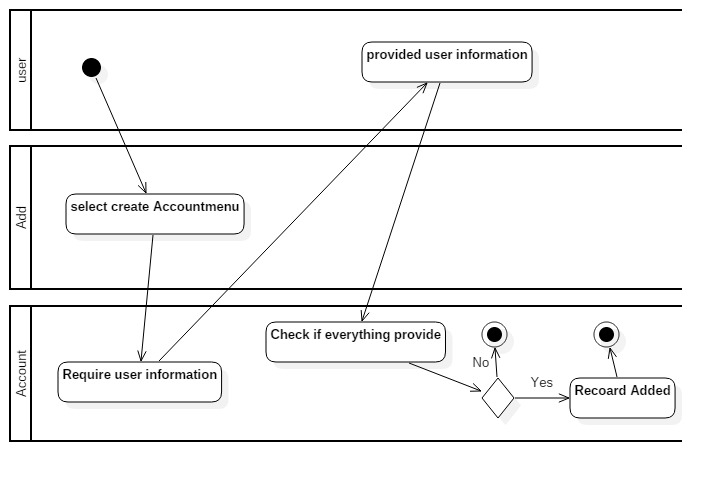


Figure 2: Activity diagram for create account

RemoveuserAccount

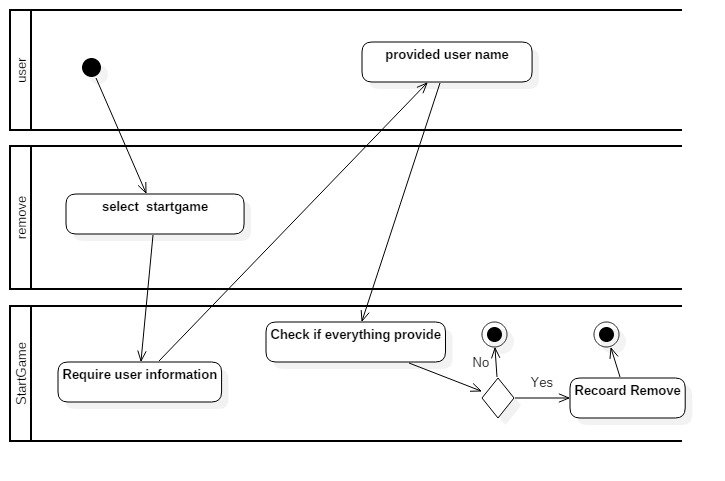
****

Figure *3*: Activity diagram for removeuser Account

Start sudoko game

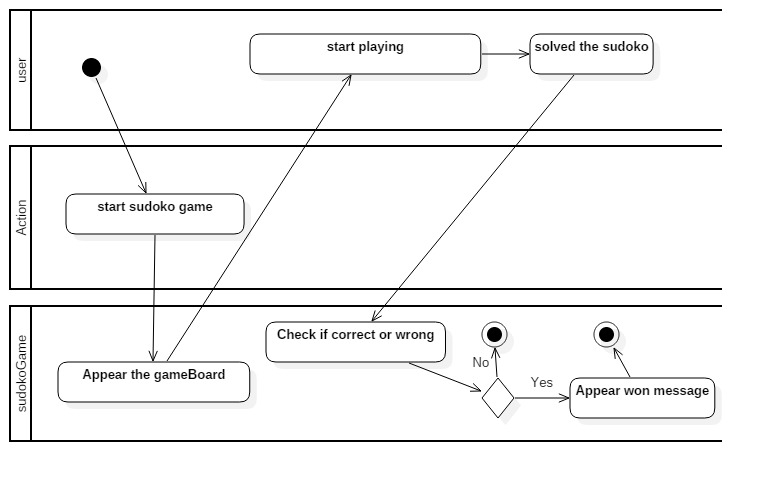


Figure *4*: Activity diagram for sudoko game

Save game progress

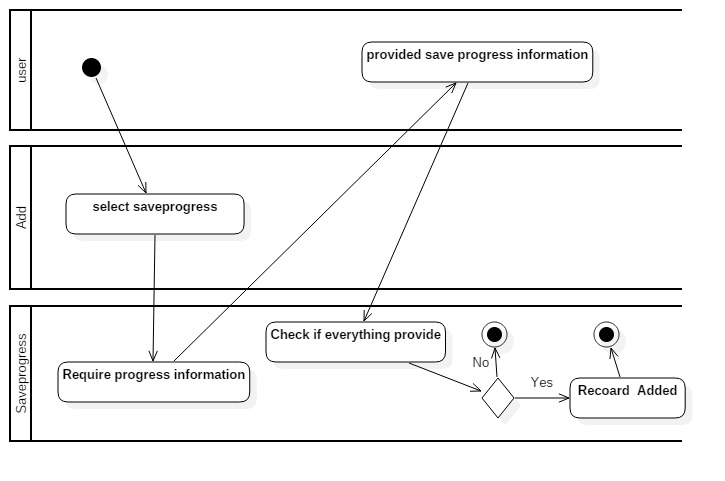
****

Figure *5*: Activity diagram for save game progress

View user statistic

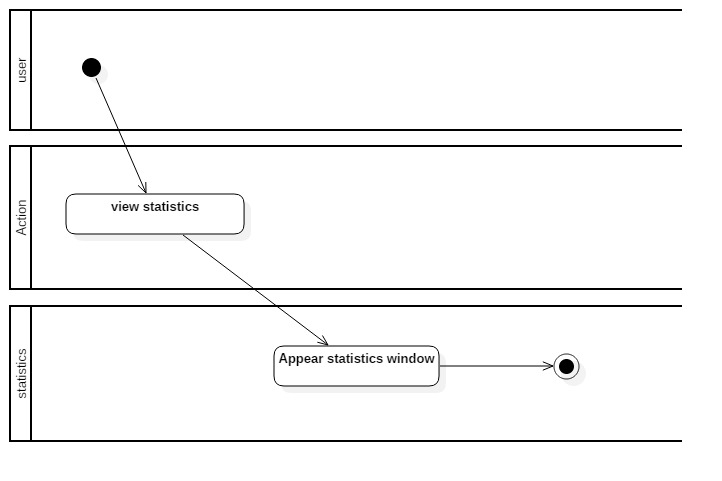


Figure *6*: Activity diagram for view user statistices

**Task 3 – Use case diagrams**

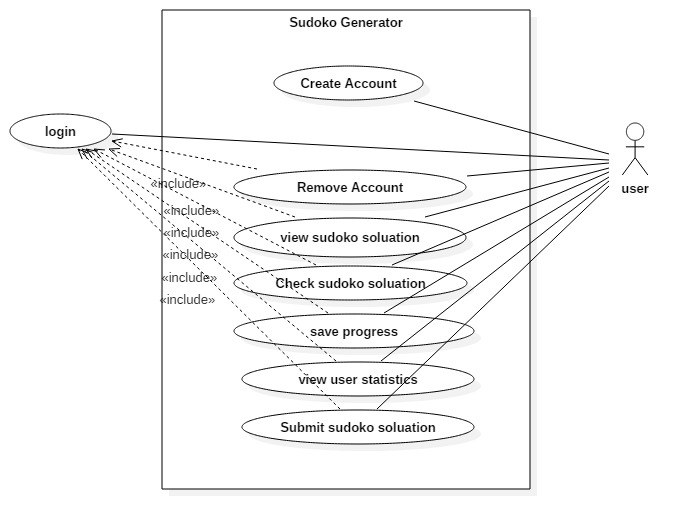


Figure *7*: Activity diagram for use case diagram

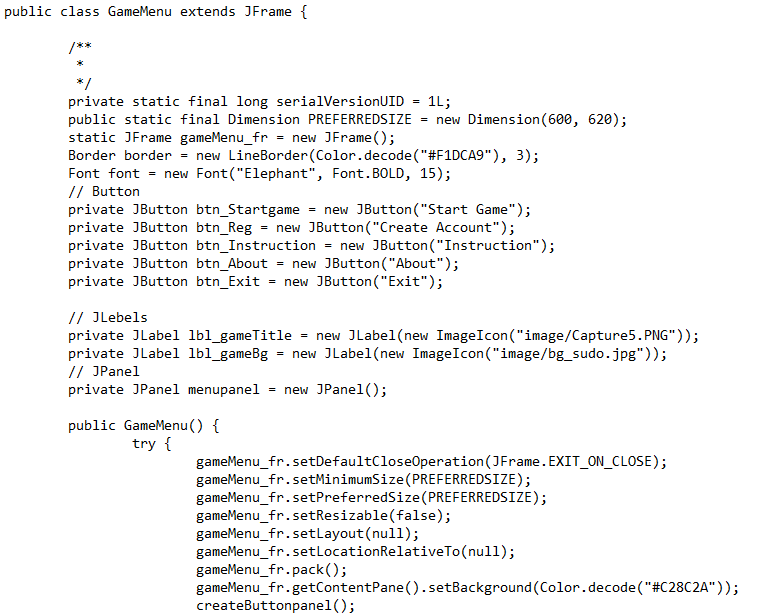
**Use case functional requirement description:**

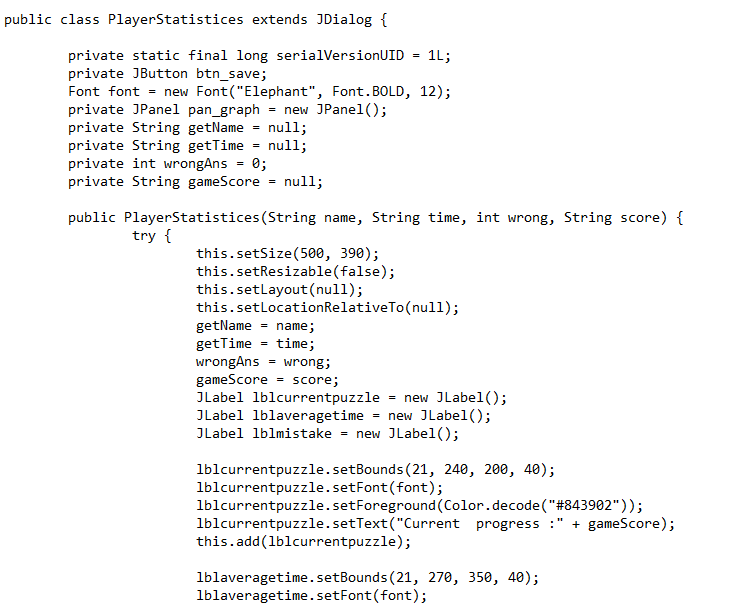
|  |  |  |  |
| --- | --- | --- | --- |
| Requirements identifier | Requirement name | Requirement description | Acceptance criteria |
| RI001(functional) | Register | As a user, he must want to sign up to the application to get login access. | Register can successfully done by user. |
| RI002(functional) | login | As a user he / she must want to login to use the application. | Login can successfully done by user. |
| RI003(functional) | Remove account | As a user,he can remove his account from this application | Remove account successfully done by user. |
| RI004(functional) | View sudoko soluation | As a user he or she can easily see soluation of the sudoko game for different levels. | Soluation of sudoko game is successfully done |
| RI005(functional) | Check sudoko soluation | As a user he or she can easily check the sudoko soluation if he or she want | Checking of sudoko game is successfully done |
| RI006(functional) | Save progress | As a user if he or she can save their sudoko game progress | Game progress of sudoko game successfully done. |
| RI007(functional) | View user statistices | As a user he or she can see their own game statistices | User statistices of sudoko game is successfully done |
| RI008(functional) | Submit sudoko soluation | After complete the sudoko then submit the soluation. | Soluation of sudoko game is successfully checking. |

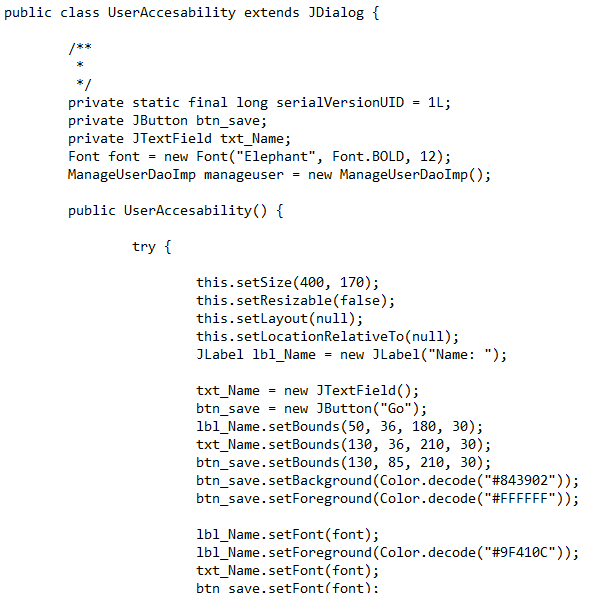
**Task 4 - Code architecture:**

**Code inheritance:**

It is the mechanism in java by which one class is allow to inherit the features (fields and methods) of another class. Here GameMenu extends the JFrame that’s means GameMenu class inherit the features of Jframe class.

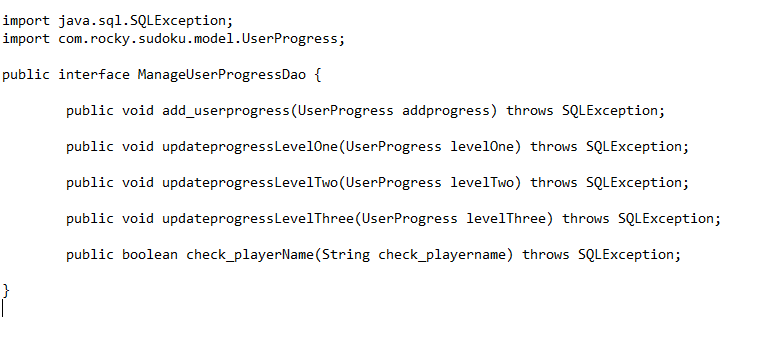


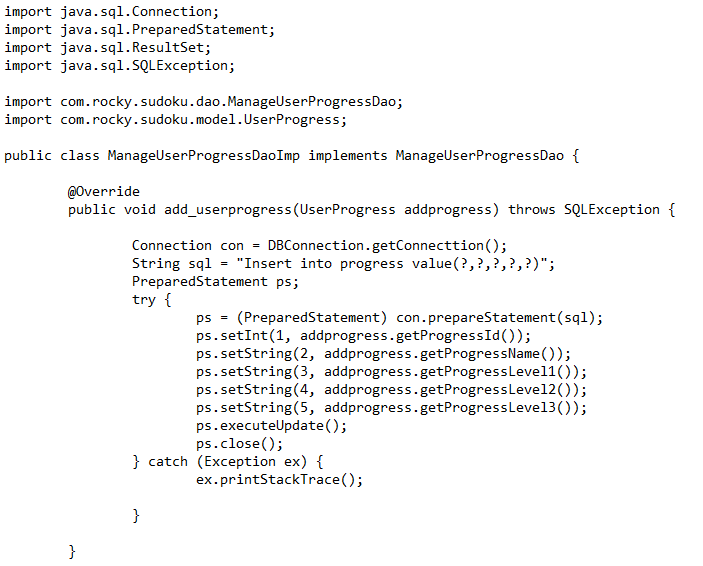


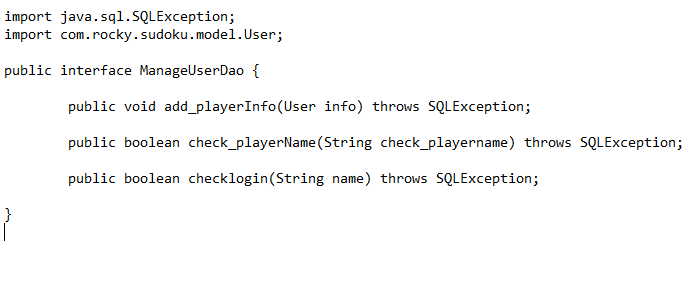


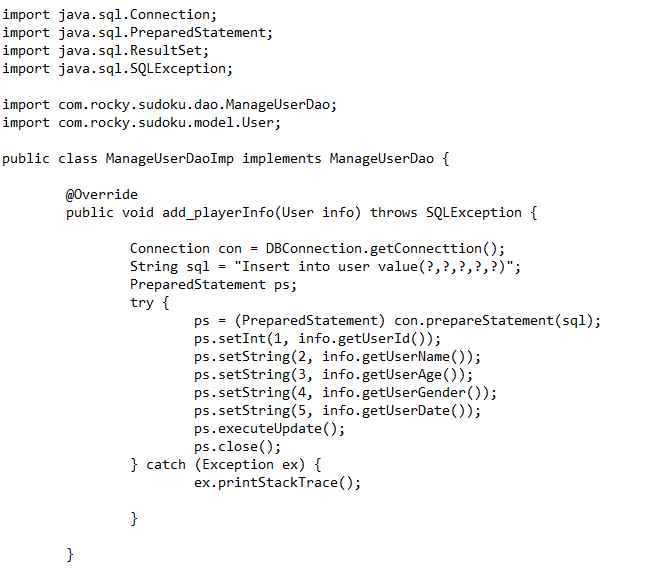
**Polymorphism:**

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.



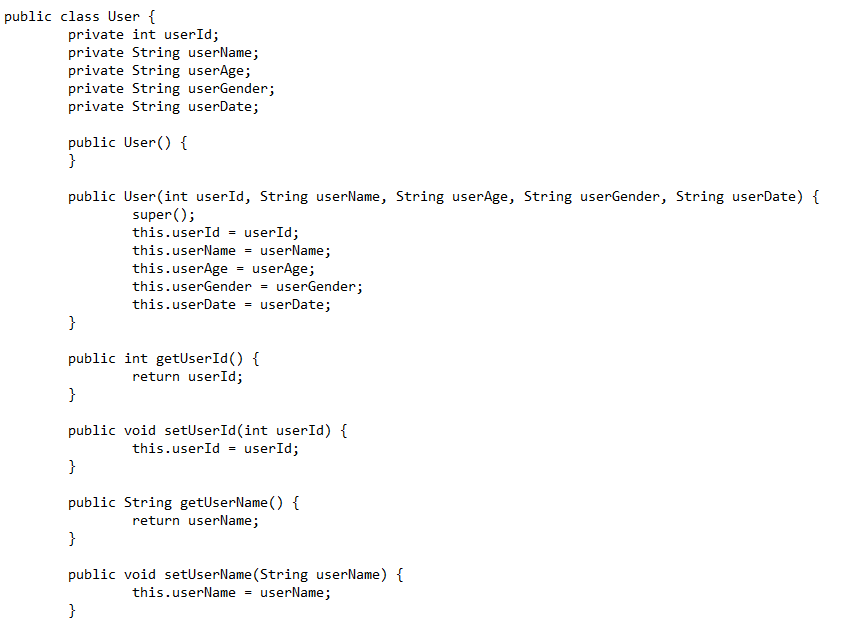


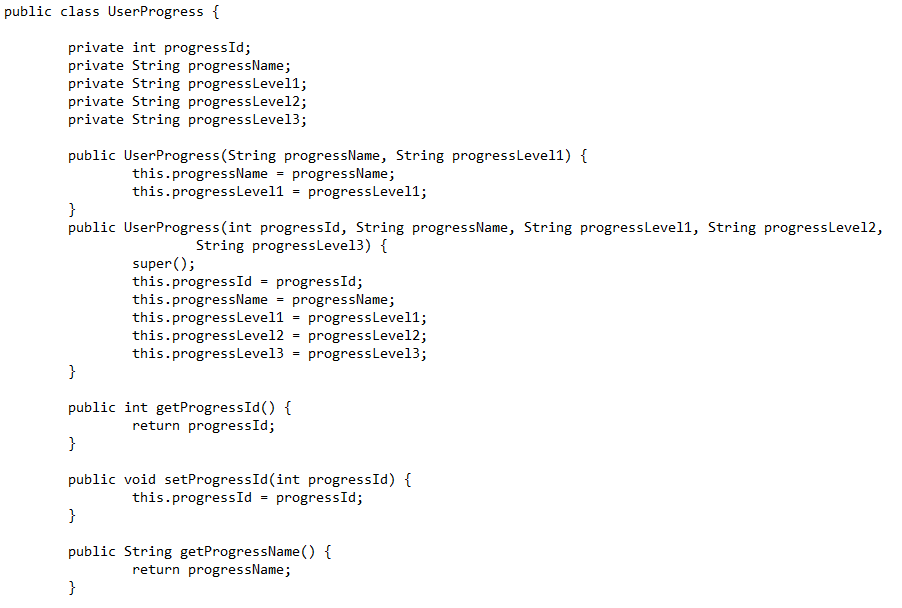




**Encapsulation:**

Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as data hiding.





**Code for user data input:**

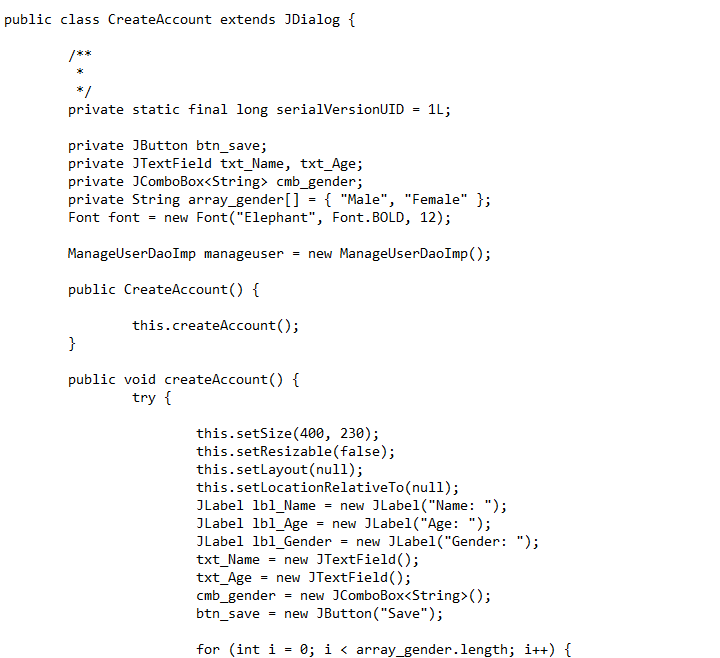
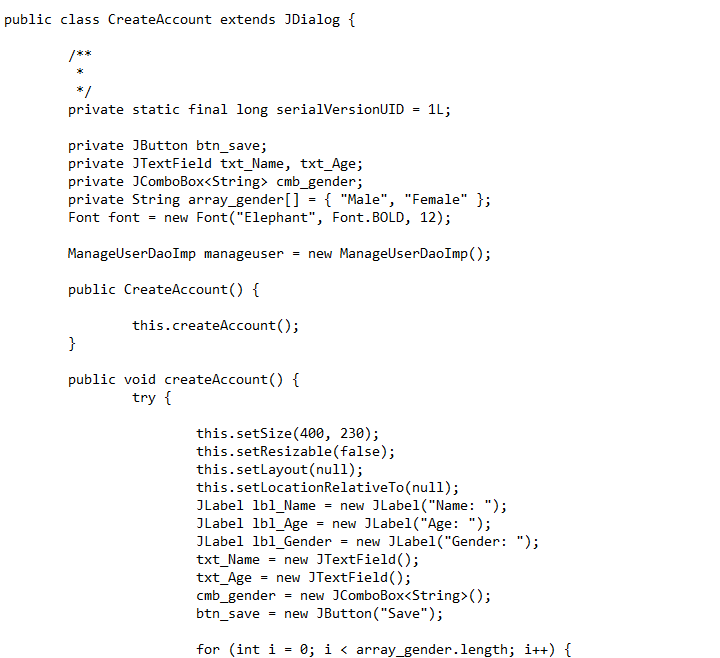




Figure *8*: create user account

**Code for user data output:**



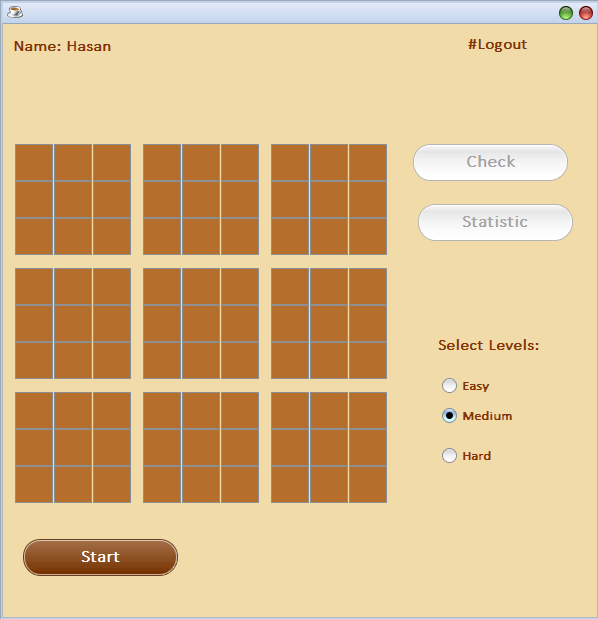
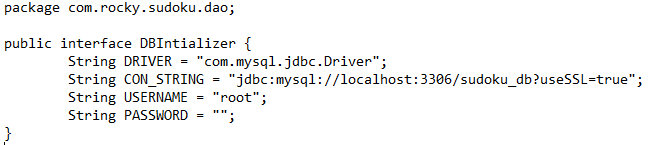


Figure *9*: output user account

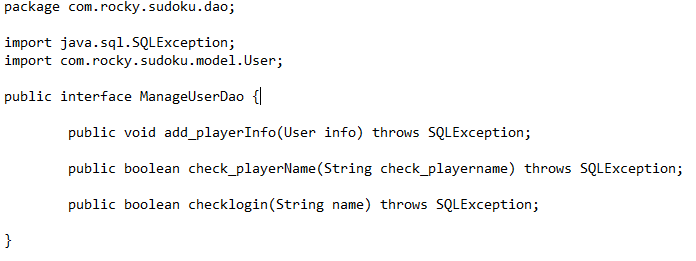
**Task 5 – System implementations:**

Below I provide my class codes include (inheritance, polymorphism, input and output).

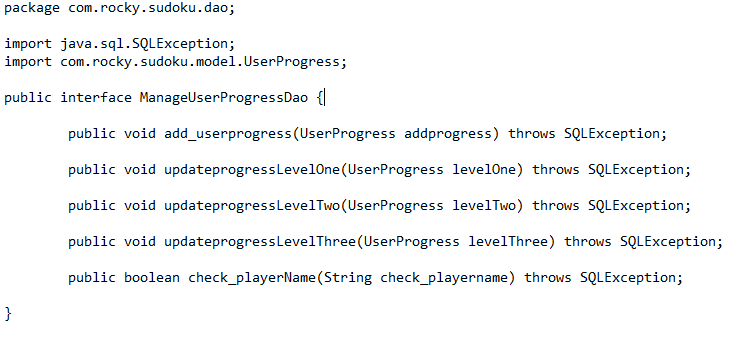
**DBIntializer:**



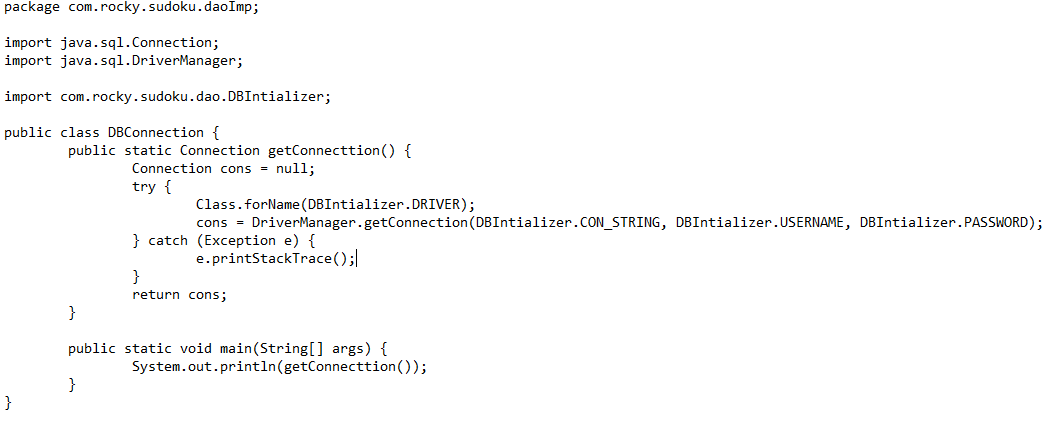
**ManageUserDao:**



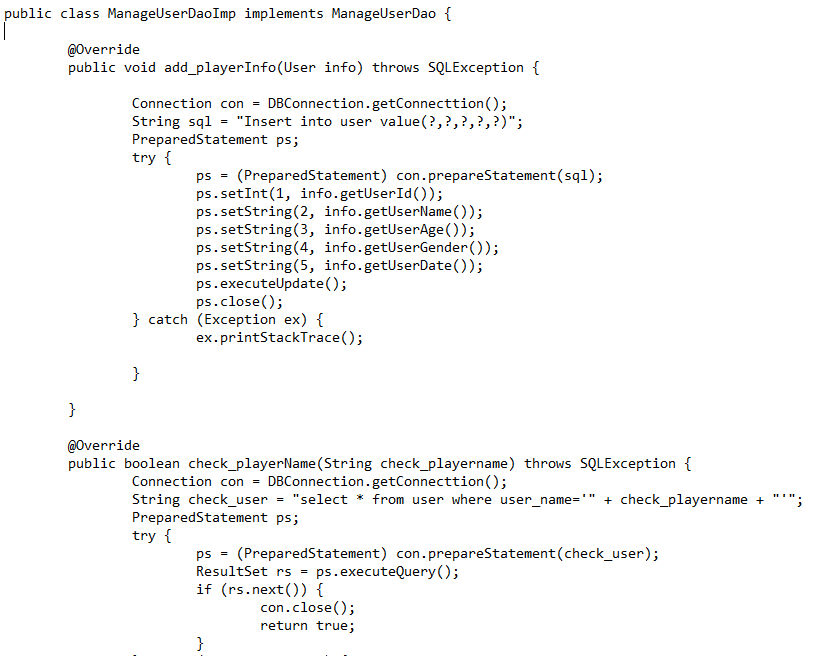
**ManageUserProgressDao:**

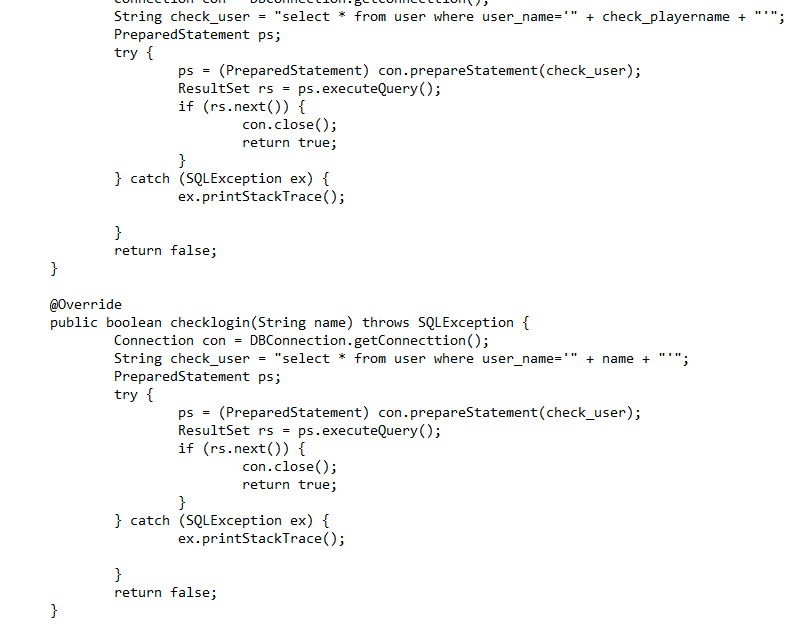


**DBConnection:**



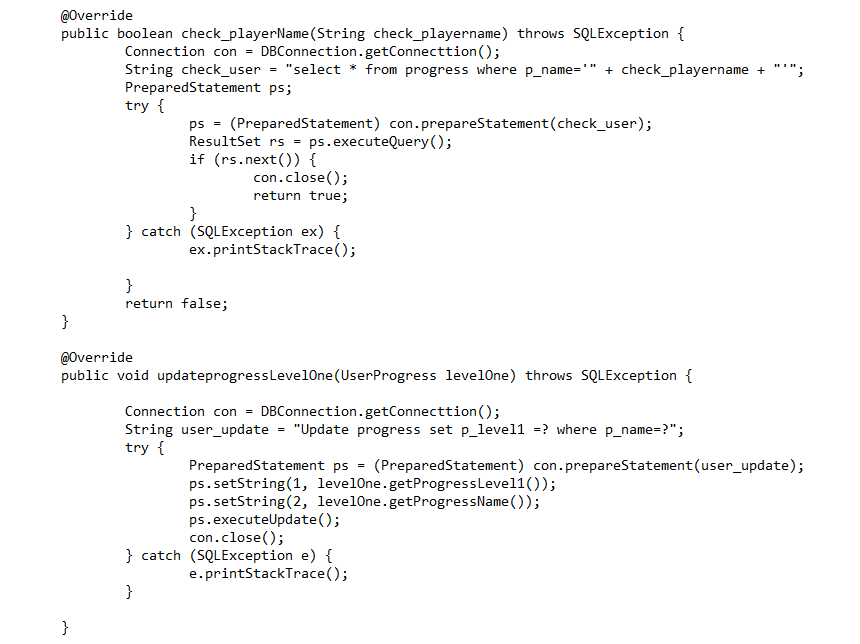
**ManageUserDaoImp:**

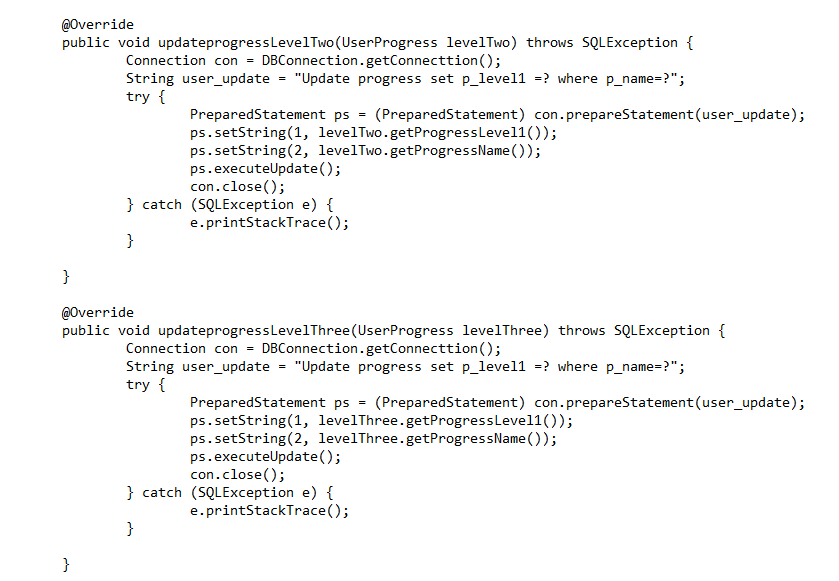




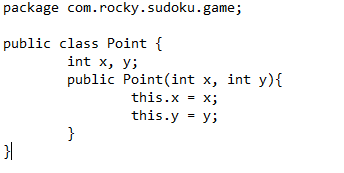
**ManageUserProgressDaoImp:**



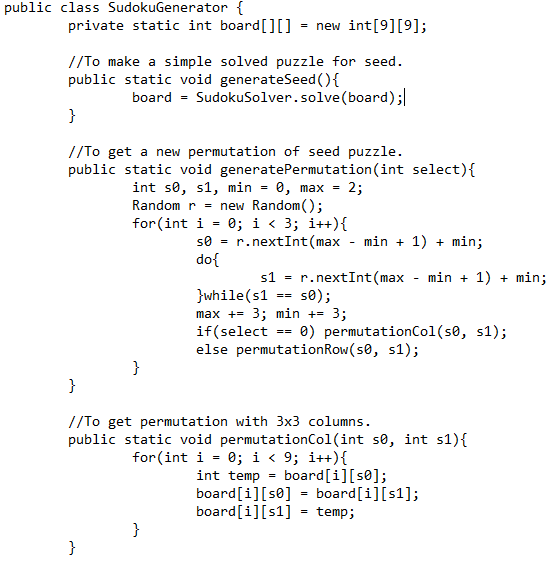


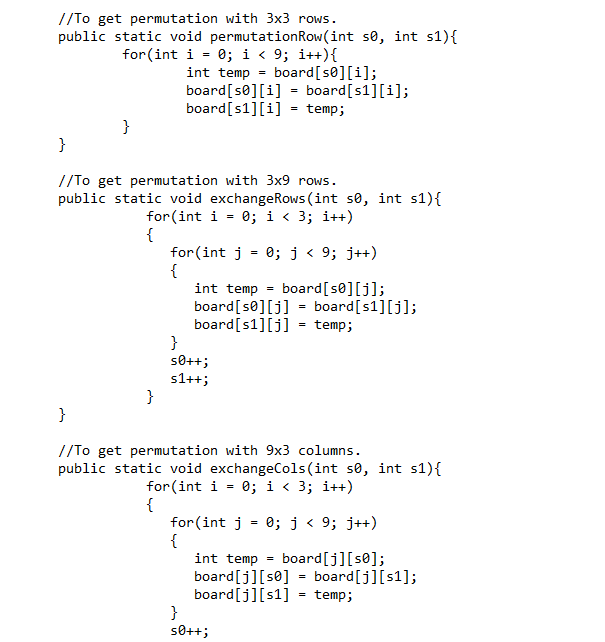


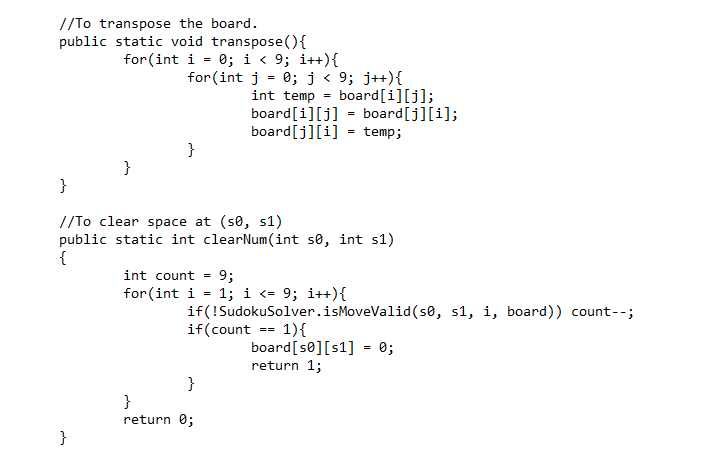
**Point:**



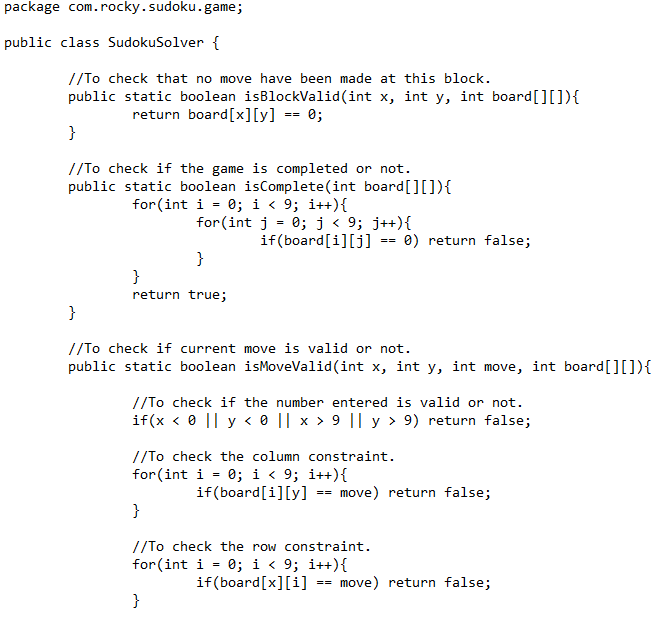
**SudokuGenerator:**

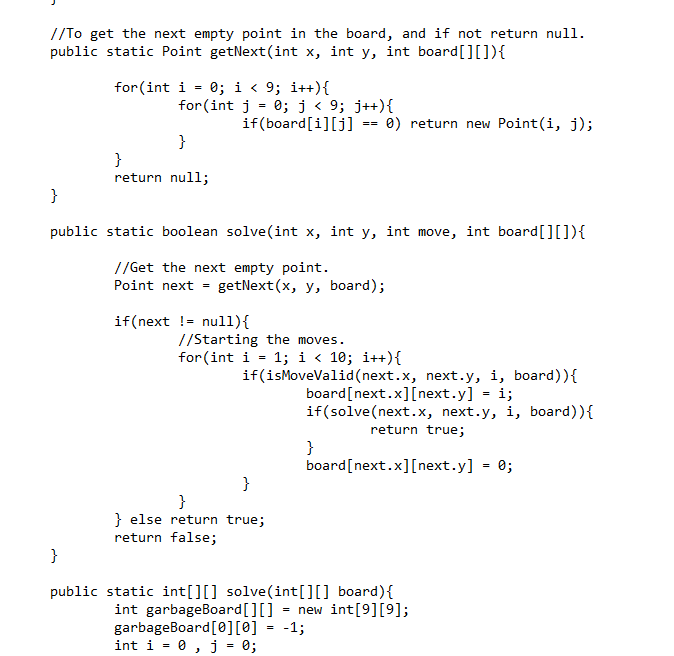




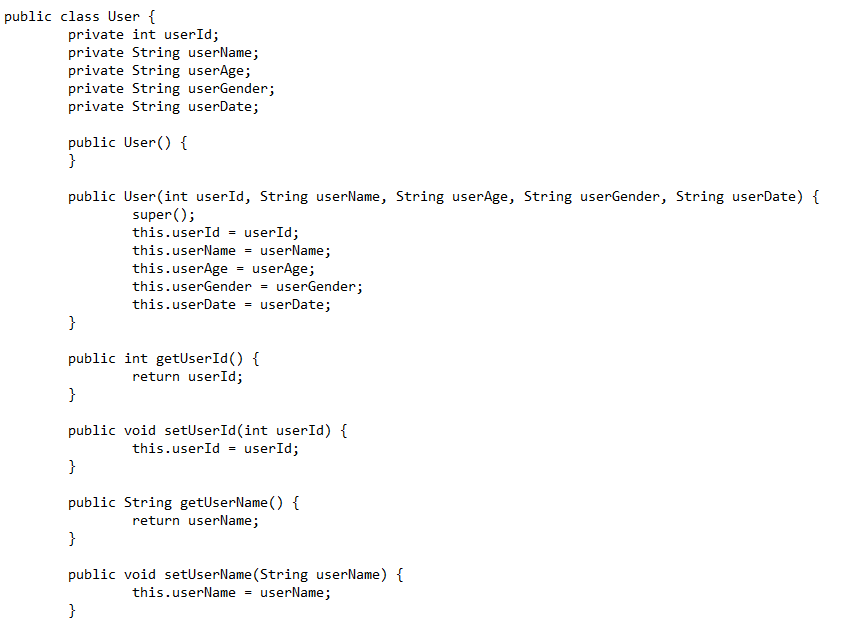


**SudokuSolver:**

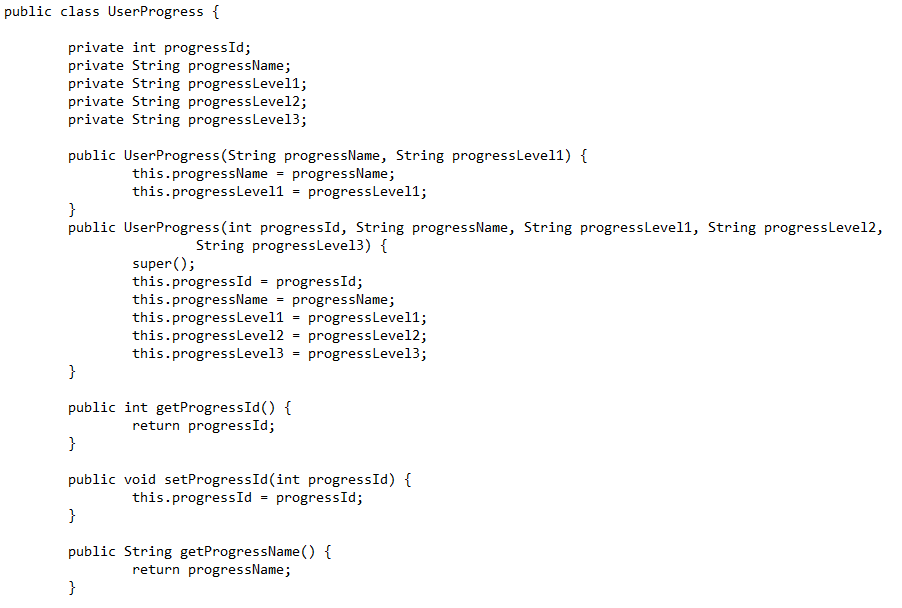




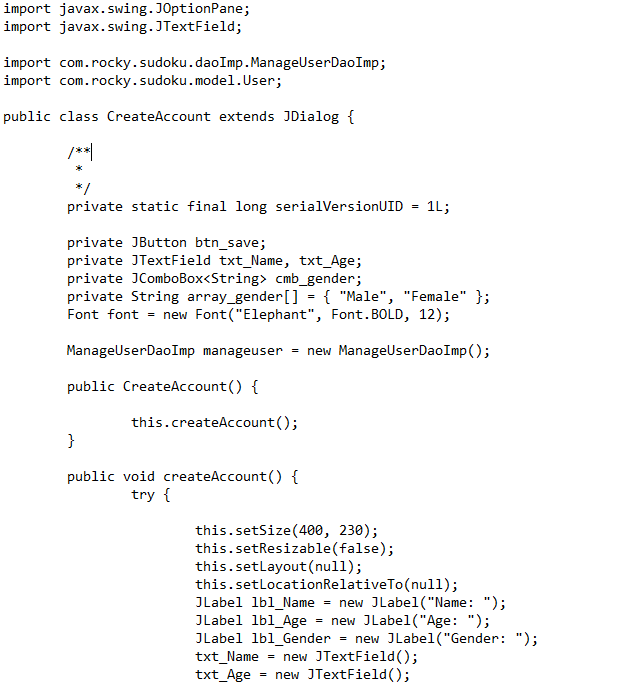
**User:**

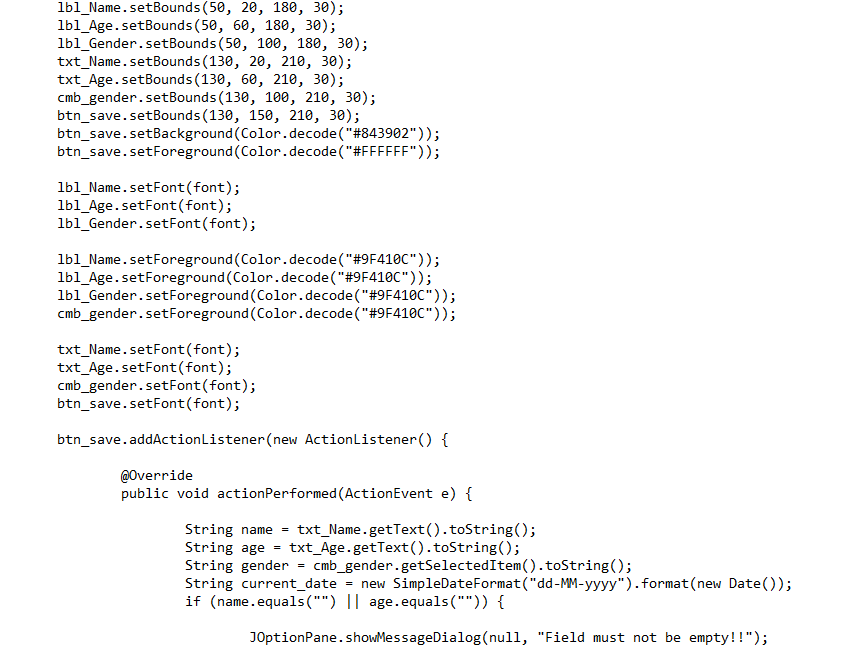


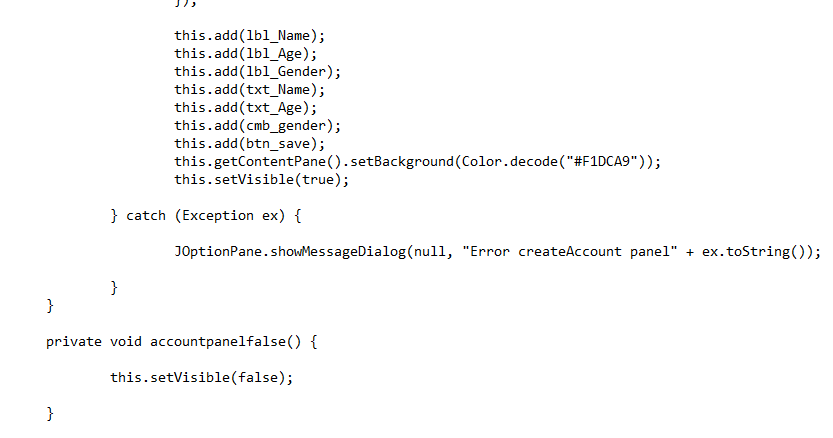
**UserProgress:**



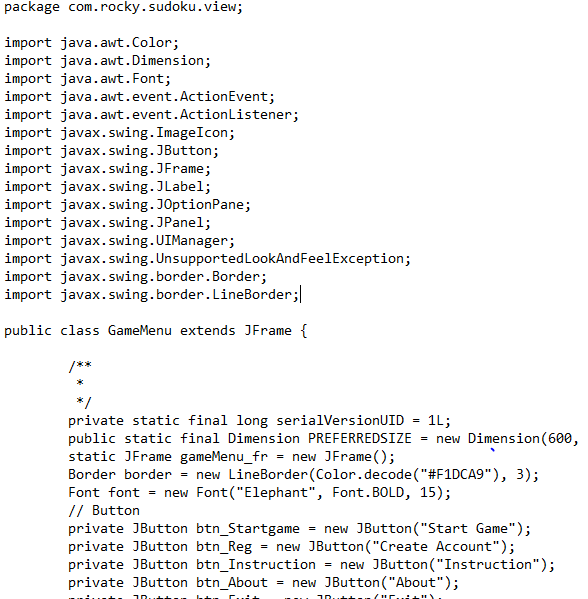
**CreateAccount:**

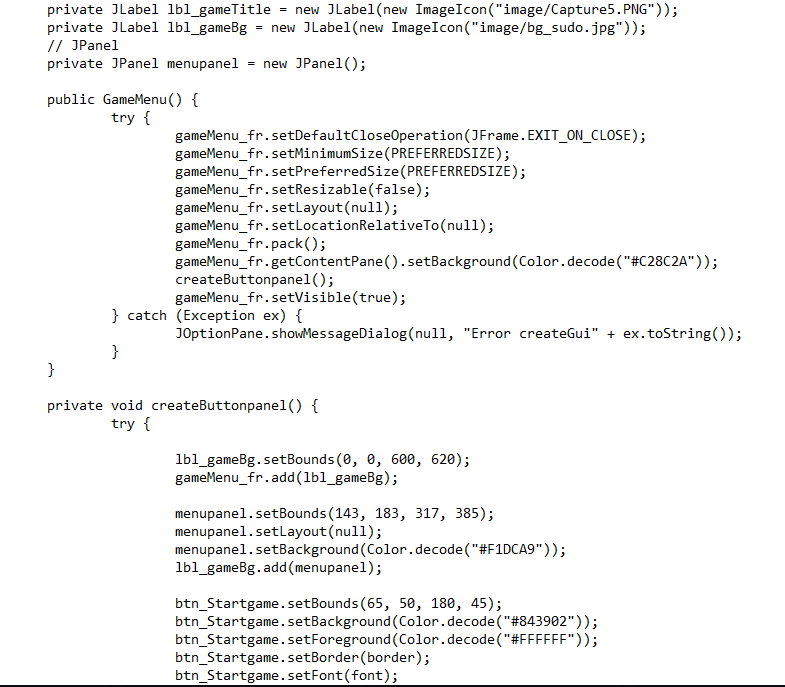


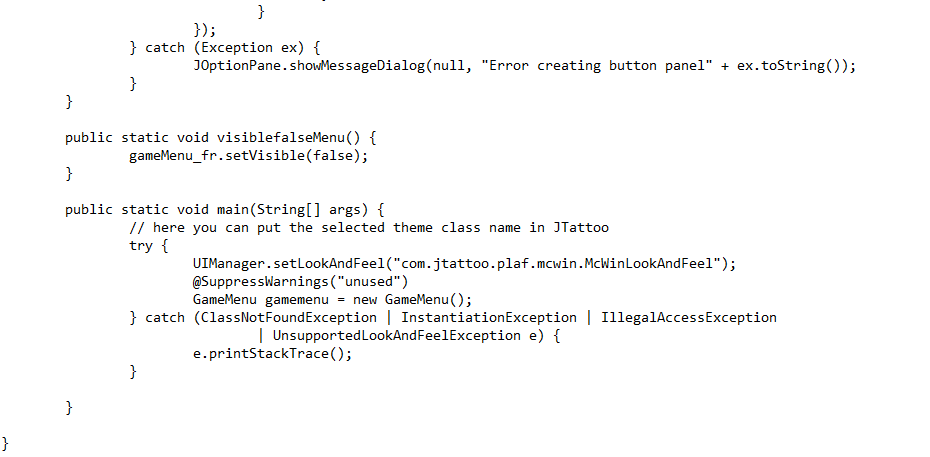




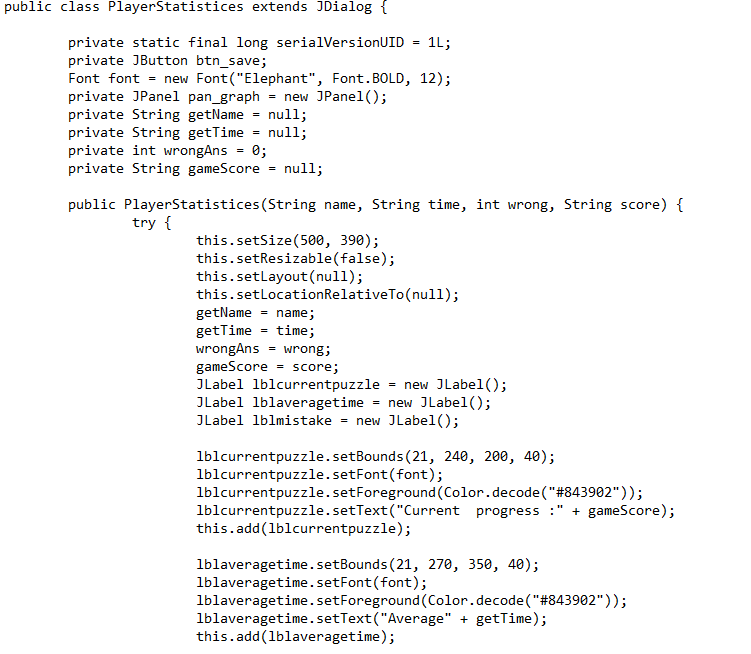
**GameMenu:**

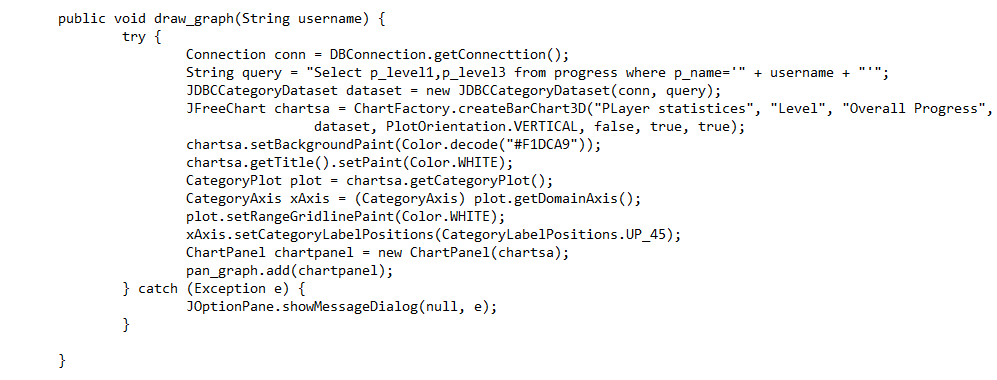




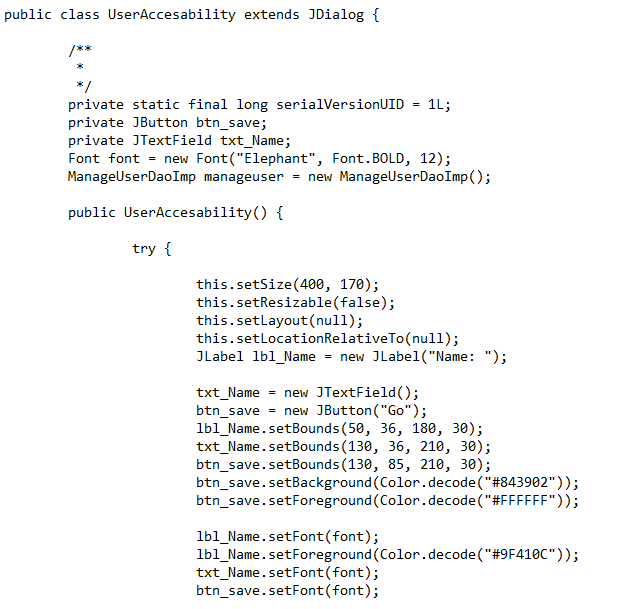


**PlayerStatistices:**





**UserAccesability:**





**Task 5: System implementations**

The implement codes are provided in an individual zip file include the database SQL file.

**Conclusion:**

This documentation cover from task 1 to 4 including all requirements. This assignment could be more improved a lot if I got some more time and deep understanding of some different development pattern.