PES University, Bengaluru

(Established under Karnataka Act 16 of 2013)

Department of Computer Science & Engineering

Session: Jan - May 2022

UE19CS353 – Object Oriented Analysis and Design with Java

Theory ISA (Mini Project)

Report on

Sudoku Game in JAVA

By:

Nivedhitaa R – PES2UG19CS268

Nithin Francis –PES2UG19CS267

Riya Puri – PES2UG19CS334

6th Semester E

Table of Contents

1.Project Description

Business Logic description:

The sudoku.java class is responsible for handling all the activity on the sudoku

board including getting a free cell list, resetting the board, and calculating whether there are available cells.

It also handles the overall logic of filling the board with exact solutions for the puzzle and sets the board with appropriate values for level easy, medium and hard.

We have used the algorithm similar to the N-queens problem to identify the correct answer and each time the game is reset the values will also be changed according to random numbers generated by the random number generator function in class sudoku.

Persistence Logic Description:

The first implementation was the Highscore. Highscore was created by creating a textfield through which a player could insert their Name and converting the already inbuilt Timer of the game into a form of Score.

We have implemented the top 5 high scores as well as once a player finishes the game, their previous best would be previewed.

The second implementation was the Load/Save Game. Since Sudoku coincidentally works on Rows and Columns, I was able to insert the rows and columns into a Database Table.

Two separate Tables were made, one to store the Grid (Actual values of the grid) and another to store the User's inputs + Values pre-filled for User.

Following the Principle of Single Responsibility, Each of a) Highscore, b) Saving and Loading the Grid

And c) Saving and Loading the User Inputs were made as separate Classes which conduct their own individual functionalities.

Following the Principle of Open Close, when creating the Database, extra effort was made to not break any existing functionality but to extend it instead.

Front UI description:

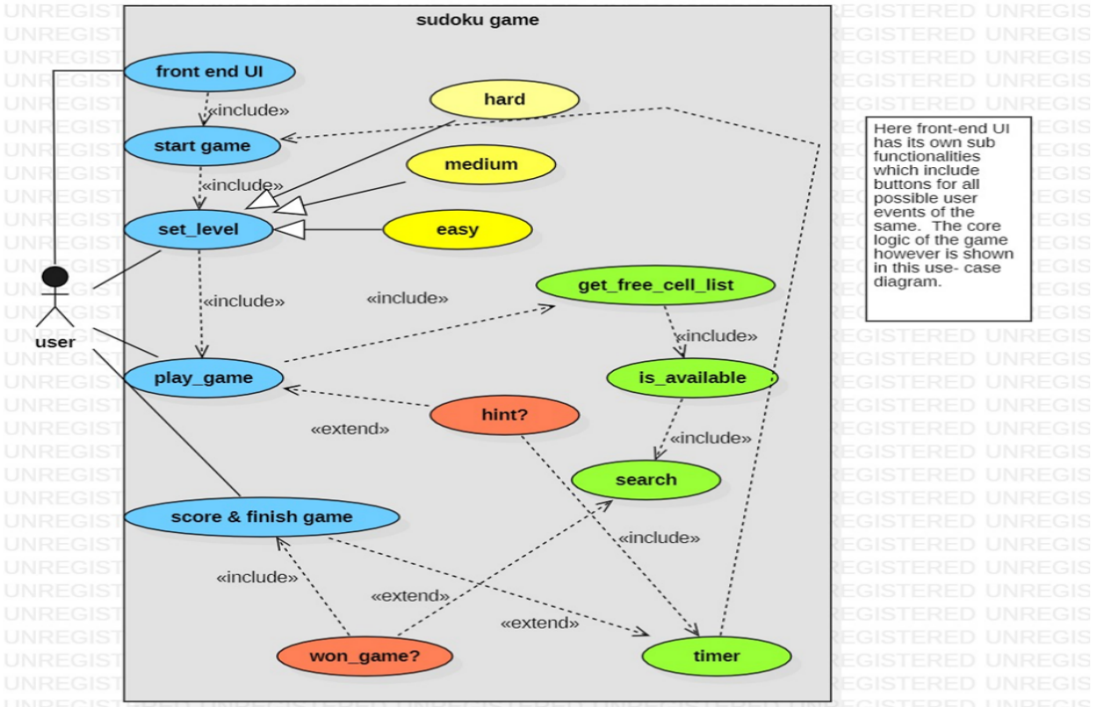
Panal.java forms the crux of the application. Since we have implemented the Facade Design pattern, the Panel is responsible for being the main class interacting with the user. Panel also interacts with the persistence (GridLayout, saveLoad) and game logic (sudoku.java).

Making use of java frameworks such as Swing and AWT, The UI has been created to change colors on the event of a mouse click or hover.

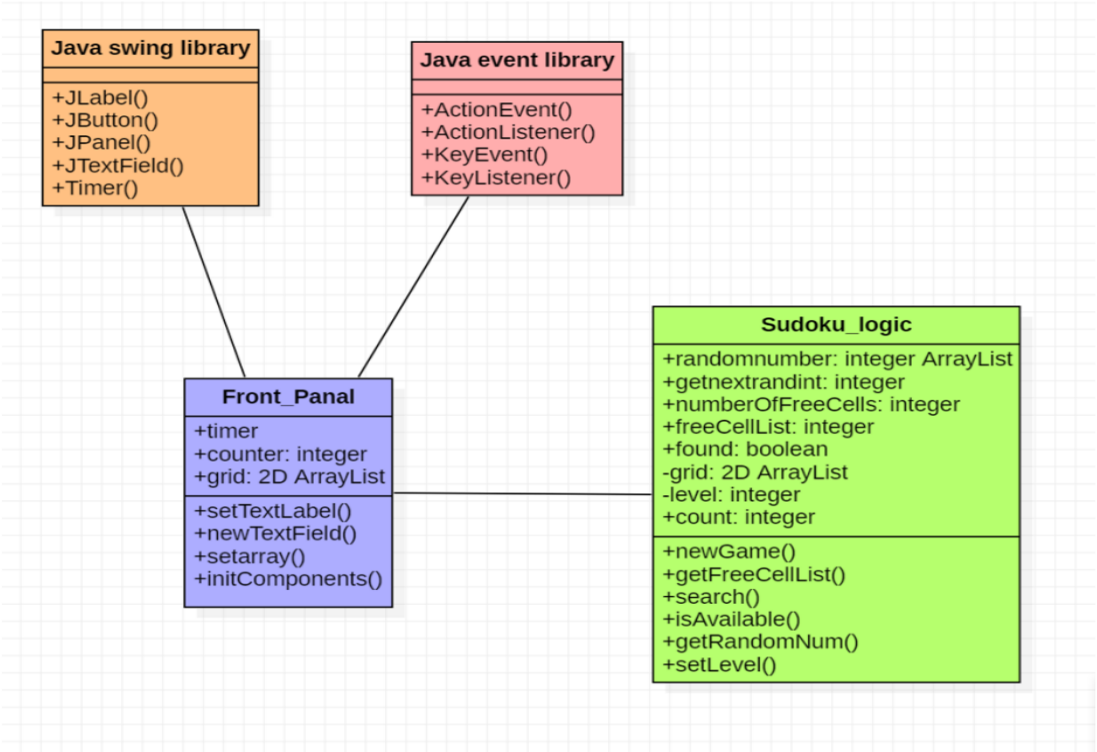
The panel also helps create the empty grid layout and interacts with the sudoku class to add and check values. It displays dialog boxes on submission or whenever there are errors by the user.

2.Analysis and Design Models:

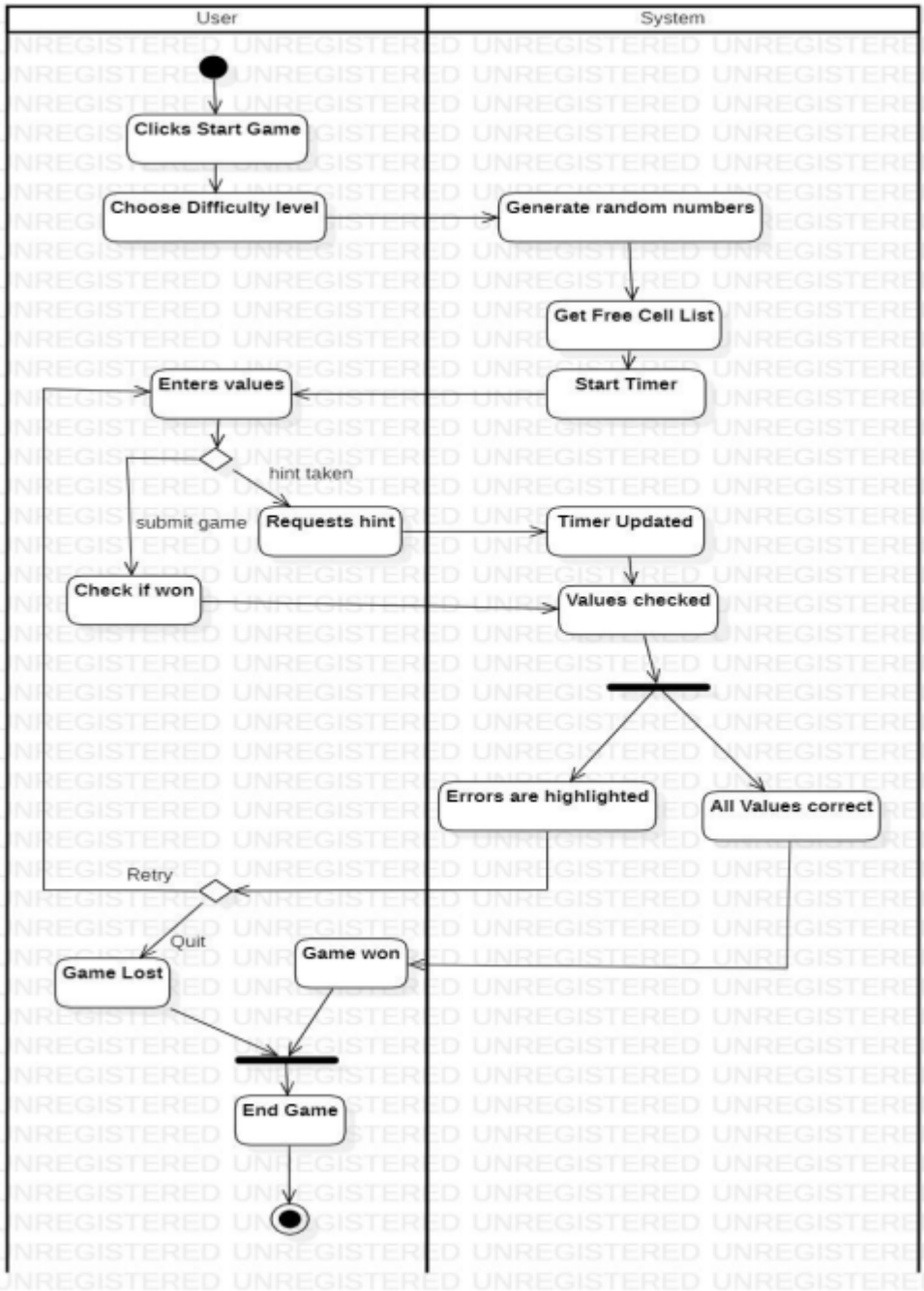
Use cases diagram:



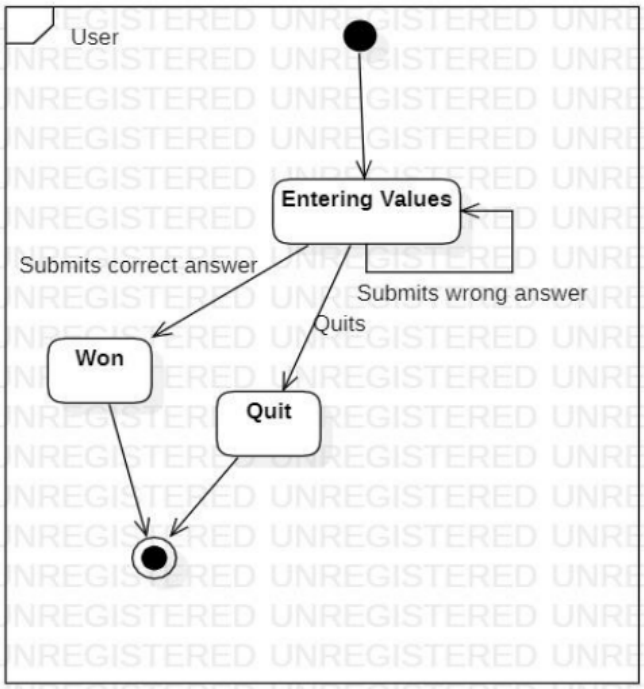
UML CLASS DIAGRAM:



Activity diagram

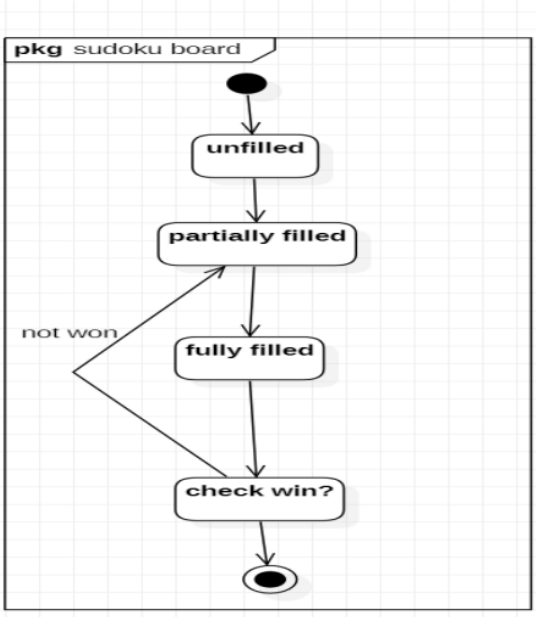


User state diagram:



State diagram:

Sudoku board:



3.Tools and Frameworks Used

Model-View-Controller framework applied.
View: Panal.java (User Interface)
Model: GridLayout and SaveLoad.java (Database and connecting classes)
Controller: Sudoku.java (Logic of the game)

Tools and frameworks used:
SQLITE: connect the UI panel to the database (persistence logic)
Java Libraries :
Swing API for the GUI
AWT: Abstract Window Toolkit used as listener abstract class for mouse and key events

4.Design Principles and Design Patterns Applied

Design Principles:

SOLID PRINCIPLES:

1. Single Responsibility principle throughout the sudoku. java class.

Open close principle hasn't been violated, we have used extension over rewriting..java package.

2. Open-close principle hasn't been violated considering that we only extended features of base classes and never went for refactoring of code for any requirement.

Design Pattern:

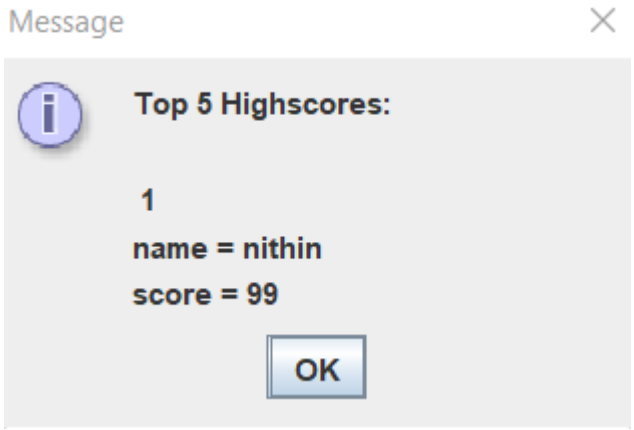
STRUCTURAL DESIGN PATTERN:

Facade, because our entire game's focal point is the swing UI wherein the business logic, persistence and client interact with each other.

The client interacts solely with the game panel (swing API) which further communicates with the persistence and game logic.

5.Application Screenshots (3-4 important pages)


</Suduko Game>
0 3 8 0 0 4 9 0 0
4 0 5 0 0 9 0 0 3
2 9 7 0 0 0 4 0 1
0 0 2 0 0 9
8 0 0 0 1 0
0 0 4 0 8 0
0 0 3 0 0 0 1 5 0
0 8 1 3 0 7 0 0 0
6 0 0 8 0 0 2 0 7
Message
Saved
OK
Easy
Medium
Hard
Load Game
Check Game +30s
Exit
Enter your Name :
asd
Submit
Timer :00 : 00 : 48
HighScore


Message
Top 5 Highscores:
1
name = nithin
score = 99
OK

6.Team member contributions

Nivedhitaa R	**PES2UG19CS268**	**Business Logic (sudoku.java)**
Nithin F	**PES2UG19CS267**	**User Interface (swing, awt)**
Riya Puri	**PES2UG19CS334**	**Persistence Logic (sqllite)**

7.Conclusion

There are a lot of scope of improvement for this project:

1. Using an architectural model wherein there is lesser chance of the anti pattern blob existing considering that we are using MVC.
2. We could also decentralize the application into a microservice architecture which could in turn reduce coupling between modules. Currently each module is very cohesive but even coupling between the modules is high.

8.References

javax.swing (Java Platform SE 7) (oracle.com)

Lesson: Learning Swing with the NetBeans IDE (The Java™ Tutorials > Creating a GUI With Swing) (oracle.com)

Designing a Swing GUI in NetBeans IDE (apache.org)

Designing a Swing GUI in NetBeans IDE (apache.org)

Types of Classes in Java - GeeksforGeeks

Java SQLite Example - javatpoint

SQlite Java - How To Use JDBC To Interact with SQLite (sqlitetutorial.net)