N-Queen Problem with multi-threading project description.

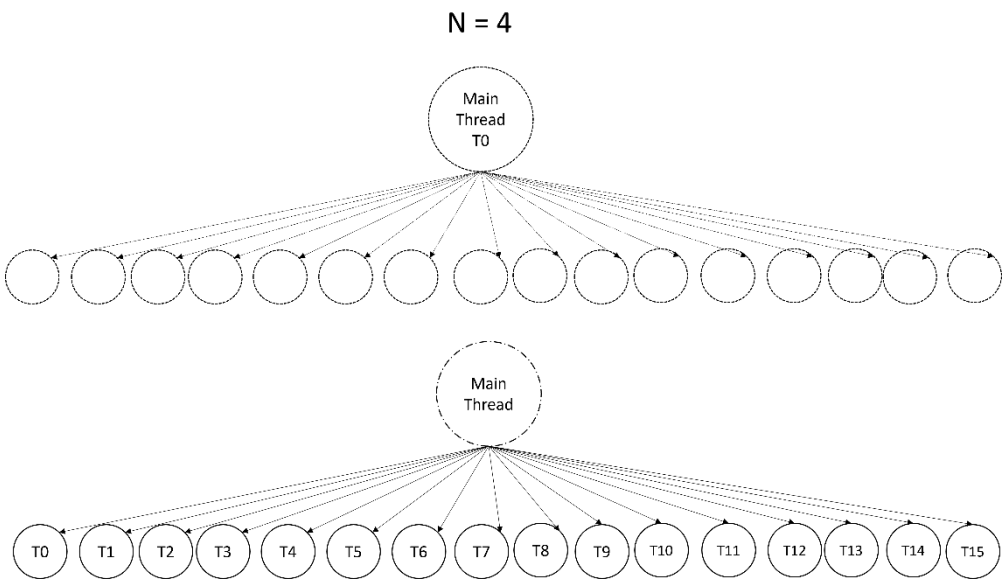
Team members Roles:

MEMBER		ROLE
Name	ID	
احمد عيسى محمود احمد	201900074	**Implementing an efficient algorithm in space and time complexity**
احمد محمد محمد علي	201900095	
أحمد علاء الدين السيد مرتضى	201900065	**Implementing the multithreading part**
احمد عبد المنجي عبد الموجود ابراهيم	201900059	
احمد محمد ابراهيم محمد	201900083	**Implementing the second screen (Solutions Screen)**
عبدالله احمد حسن سلامه	201900452	**Implementing the first screen (Input Screen)**

Quick description of the project idea:

N – Queens problem is the problem of putting N queens on a n x n chessboard such that no queens can attack each other (there is no queen shares the same column or row or diagonal with other queen).

The problem can be solve using different algorithms, but we choose the backtracking one with multithreading.

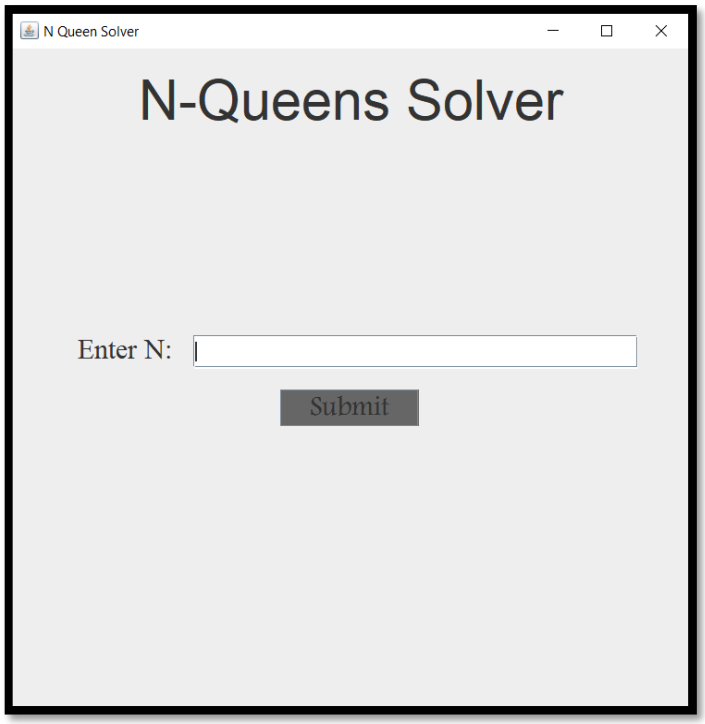


The main idea is instead of make the main thread searching in the whole search tree , we will split the work into n x n sub-search-tree and assign each one of them to a new thread that searches under that sub-tree to find solutions and add it to the main synchronized buffer that we will use it later to show the solutions .

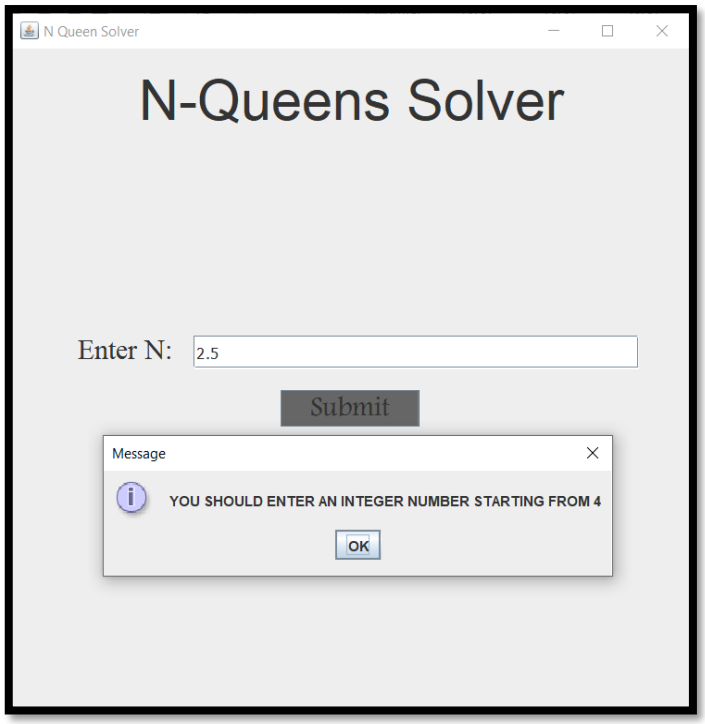
Controls and Roles:

- **When the game starts, it asks the user first to enter the number of queens the user want program to put them in the board.**
- **The number of queens should be an integer number starting from 4.**
- **The user can enter the number by click on Submit button or by pressing Enter from the keyboard.**
- **If the user enters an incorrect number an error massage will pop up and ask the user to enter a correct input.**
- **User should click on "OK" or press Enter on the keyboard on the screen of error massage to continue.**
- **If the user enters a correct number, the title of current screen will change to "N Queen Solver - Processing ... Please Wait.." until all threads of the program finish its work and found the solutions.**
- **If all threads are finished the first screen close and the second screen will open.**
- **The solutions are viewed through the second screen and the total number of solutions and the number of current solution are shown in the title of screen.**
- **User browse the solutions by pressing the left arrow for the past solution and the right arrow for the next solution.**
- **Program will be closed whenever the user clicks on X on the top to close the program.**

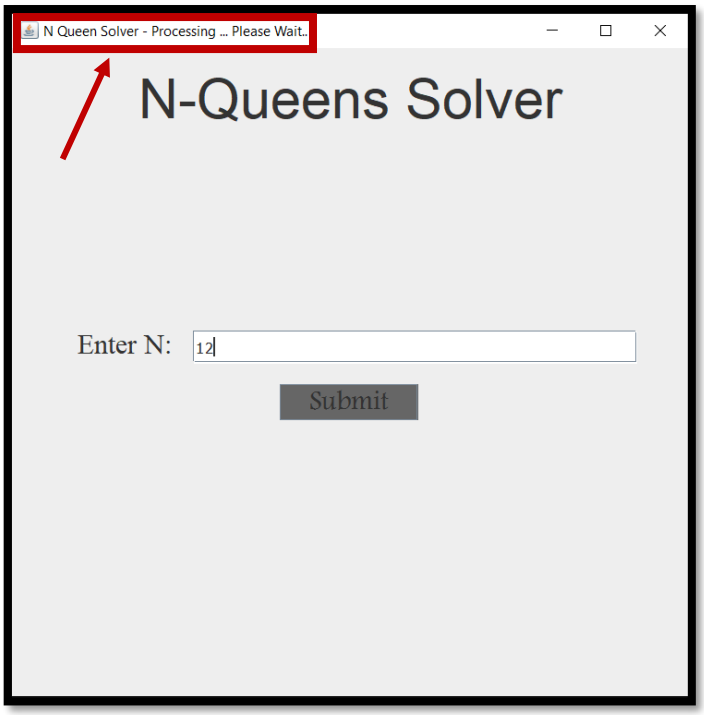
Project GUI:



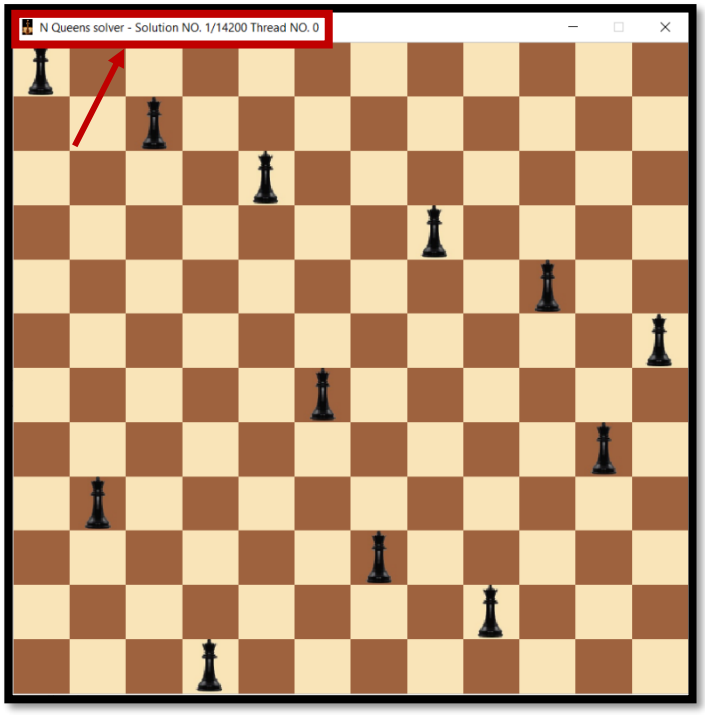
1/ The first screen where the user enters the number of queens (N).



2/ The program receive an incorrect number of queens and pop up an error message to the user.



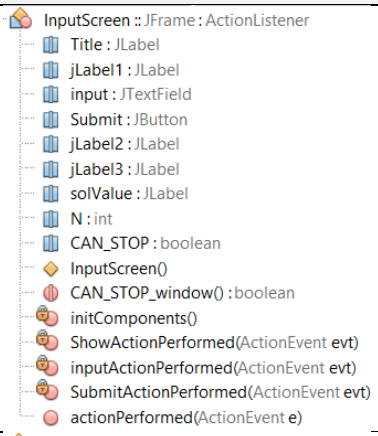
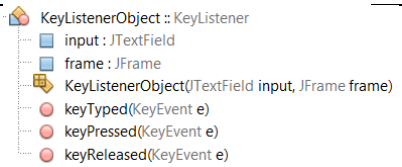
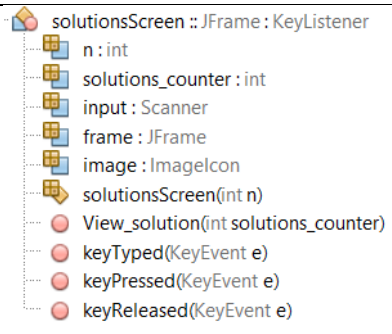
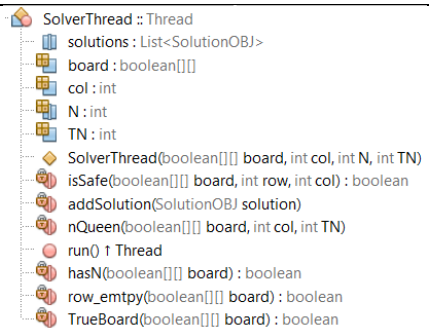
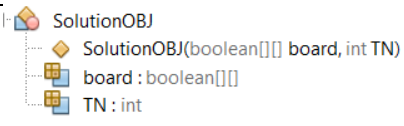
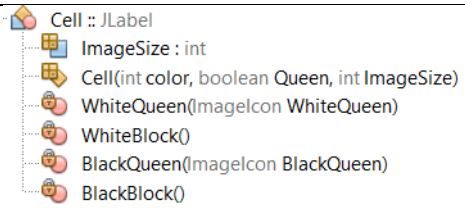
3/ The program receive a correct number of queens and make the user waiting for the program processing.



4/ The first screen closed, and second screen opened (Solutions screen) which shows the solution and its number and the thread number which generate that solution.

Code Documentation:

- **Project contains two main packages: FirstScreenGUI and N_queens_solver_package.**
- **FirstScreenGUI contains classes: KeyListenerObject.java, InputScreen.java.**
- **N_queens_solver_package contains classes: solutionsScreen.java, SolverThread.java, SolutionOBJ.java, N_Qeeens_solver_driver.java,Cell.java.**
- **GUI classes are implemented using Swing and AWT libraries.**
- **InputScreen.java is a class which provide the first input screen in the program, and it uses the second class in its package (KeyListenerObject.java) to enable it to use the keyboard to enter the input.**
- **N_Qeeens_solver_driver.java is a driver class (contains main function) its work is to manage and create the n x n thread and manage the opening and closing of screens.**
- **solutionsScreen.java is the class which provide the second screen (solutions screen) and it implements the keyListener interface to provide to user the control of the solution showed by it ,it also use the Cell.java class to make the cells on the chessboard.**
- **SolverThread.java is the class which extends Thread.java, and it`s contains the backtracking algorithm that solve the N Queen problem in an efficient way.**
- **SolutionOBJ.java is a utility class that used as a data structure to store the solutions, every SolutionOBJ have its own solution board (Boolean 2D array), and the thread number whose generate that solution.**

package	Class	Attributes and methods
FirstScreenGUI	KeyListenerObject.java	
	InputScreen.java	
N_queens_solver_package	solutionsScreen.java	
	SolverThread.java	
	SolutionOBJ.java	
	Cell.java	
	N_Qeeens_solver_driver.java