Assignment 3 – Hunting game

By Saeed Fathallah KhanlooBrise

Neptun Code : F0HAXC

Task

3. Hunting

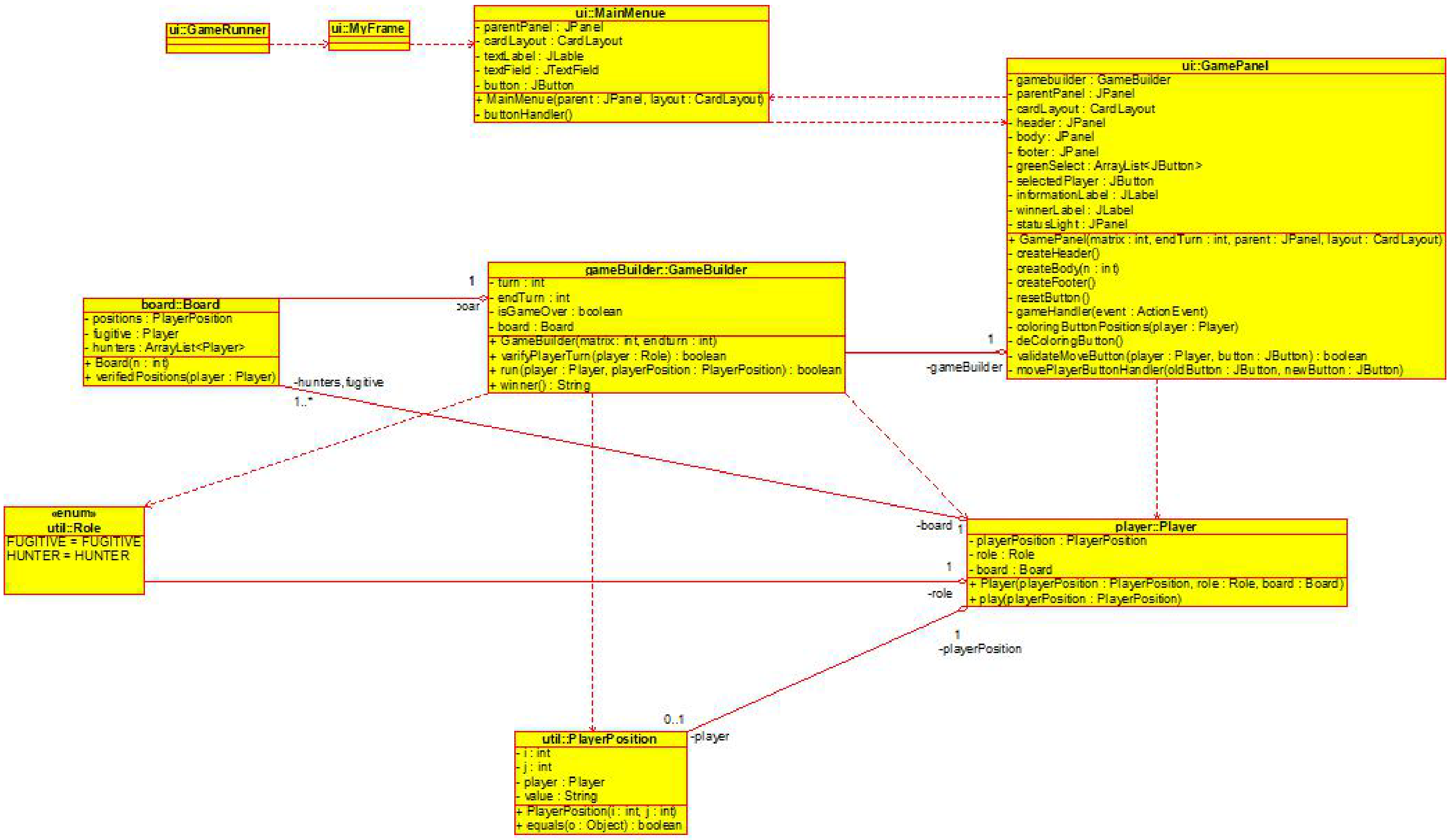
Hunting is a two-player game, played on a board consists of *n* x *n* fields, where the first player (call him fugitive) tries to run away, while the second player (the hunter) tries to capture him/her. Initially, the character of the fugitive is at the center of the board, while the hunter has four characters (one at each corner). The players take turns moving their character (hunter can choose from 4) 1 step on the board (they cannot step on each others character). The objective of the hunter is to surround the fugitive in at most *4n* steps, so it won't be able to move.

Implement this game, and let the board size be selectable (3x3, 5x5, 7x7 → turns are 12, 20, 28). The game should recognize if it is ended, and it has to show the name of the winner in a message box (if the game is not ended with draw), and automatically begin a new game.

First version (v1) we read input for matrix size from input text area , and players can move diagonally

Second version (v2) we read input from radio button only 3 cases of 3x3 , 5x5 , 7x7

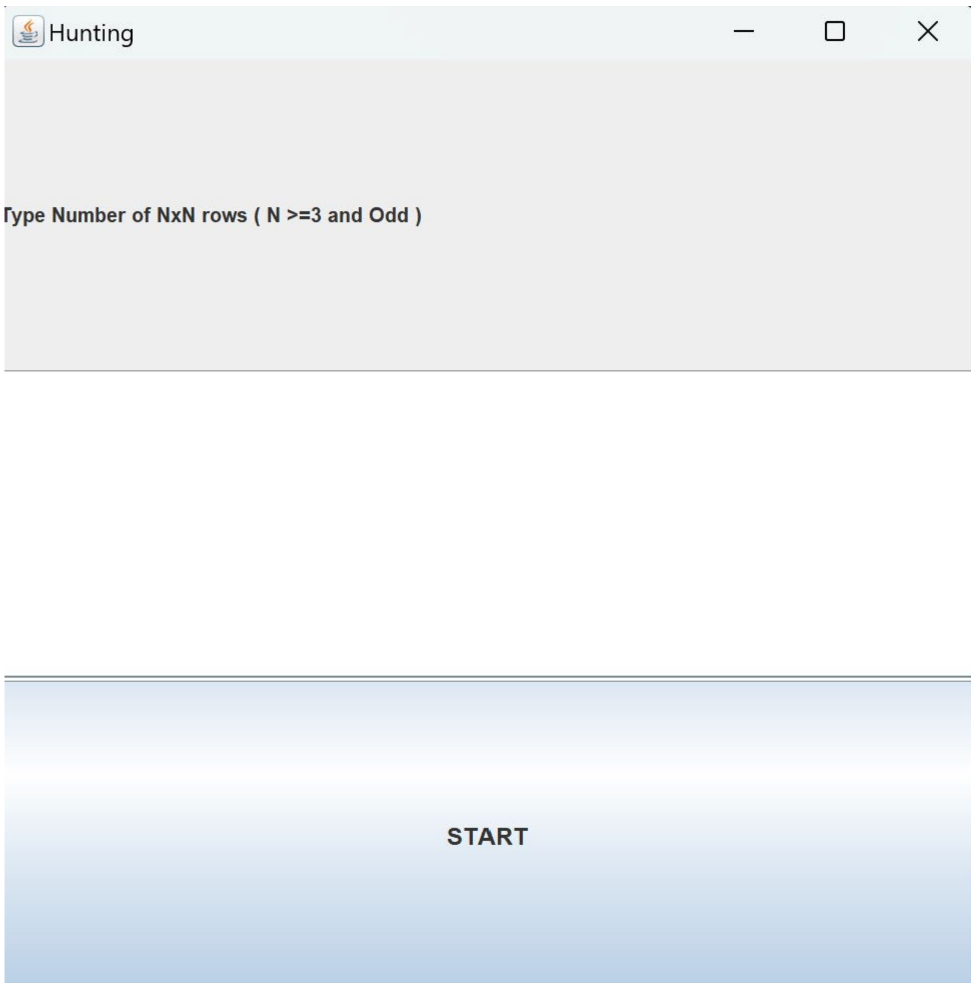
UML

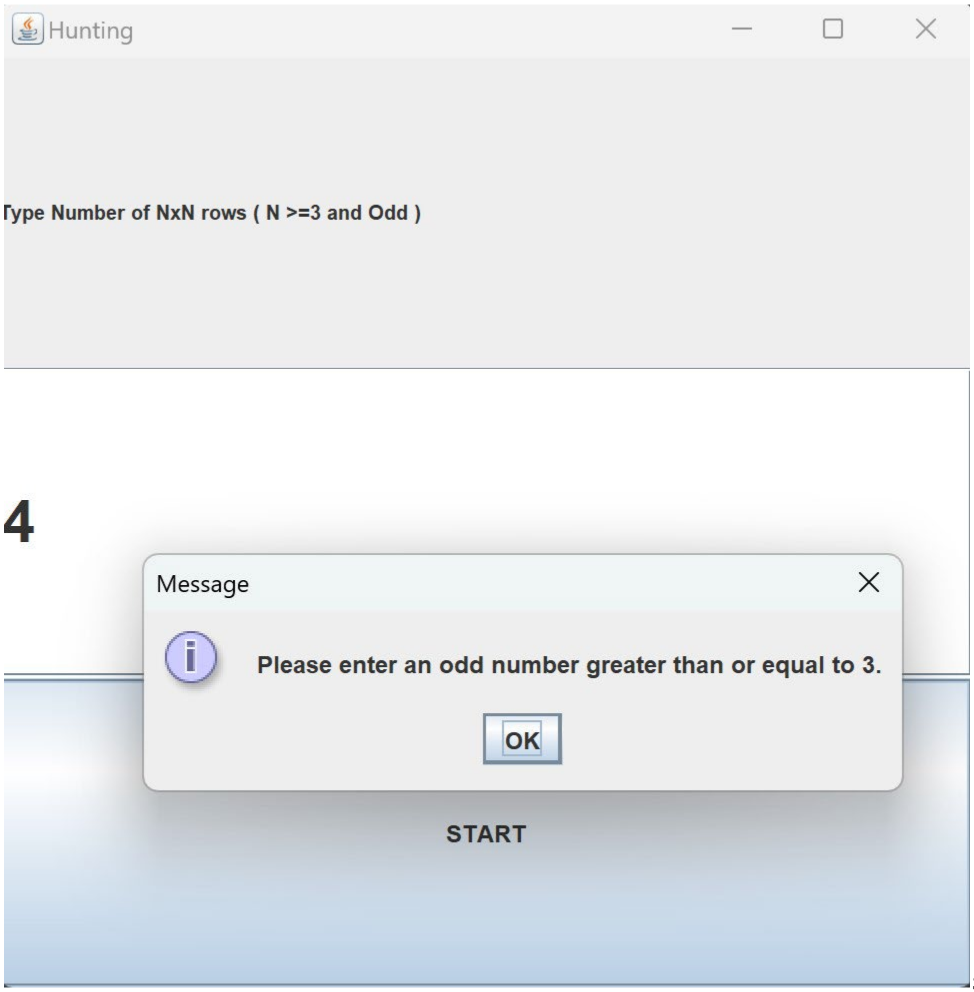

ui::GameRunner
ui::MyFrame
ui::MainMenue
- parentPanel : JPanel
- cardLayout : CardLayout
- textLabel : JLable
- textField : JTextField
- button : JButton
+ MainMenue(parent : JPanel, layout : CardLayout)
- buttonHandler()
ui::GamePanel
- gamebuilder : GameBuilder
- parentPanel : JPanel
- cardLayout : CardLayout
- header : JPanel
- body : JPanel
- footer : JPanel
- greenSelect : ArrayList<JButton>
- selectedPlayer : JButton
- informationLabel : JLabel
- winnerLabel : JLabel
- statusLight : JPanel
+ GamePanel(matrix : int, endTurn : int, parent : JPanel, layout : CardLayout)
- createHeader()
- createBody(n : int)
- createFooter()
- resetButton()
- gameHandler(event : ActionEvent)
- coloringButtonPositions(player : Player)
- deColoringButton()
- validateMoveButton(player : Player, button : JButton) : boolean
- movePlayerButtonHandler(oldButton : JButton, newButton : JButton)
gameBuilder::GameBuilder
- turn : int
- endTurn : int
- isGameOver : boolean
- board : Board
+ GameBuilder(matrix : int, endturn : int)
+ varifyPlayerTurn(player : Role) : boolean
+ run(player : Player, playerPosition : PlayerPosition) : boolean
+ winner() : String
board::Board
- positions : PlayerPosition
- fugitive : Player
- hunters : ArrayList<Player>
+ Board(n : int)
+ verifiedPositions(player : Player)
1
boar
1
-gameBuilder
-hunters, fugitive
1..*
«enum»
util::Role
FUGITIVE = FUGITIVE
HUNTER = HUNTER
-board
1
1
-role
player::Player
- playerPosition : PlayerPosition
- role : Role
- board : Board
+ Player(playerPosition : PlayerPosition, role : Role, board : Board)
+ play(playerPosition : PlayerPosition)
1
-playerPosition
0..1
-player
util::PlayerPosition
- i : int
- j : int
- player : Player
- value : String
+ PlayerPosition(i : int, j : int)
+ equals(o : Object) : boolean

Sample Input

MainMenu Frame when starting the program ->

VERSION ONE




Hunting

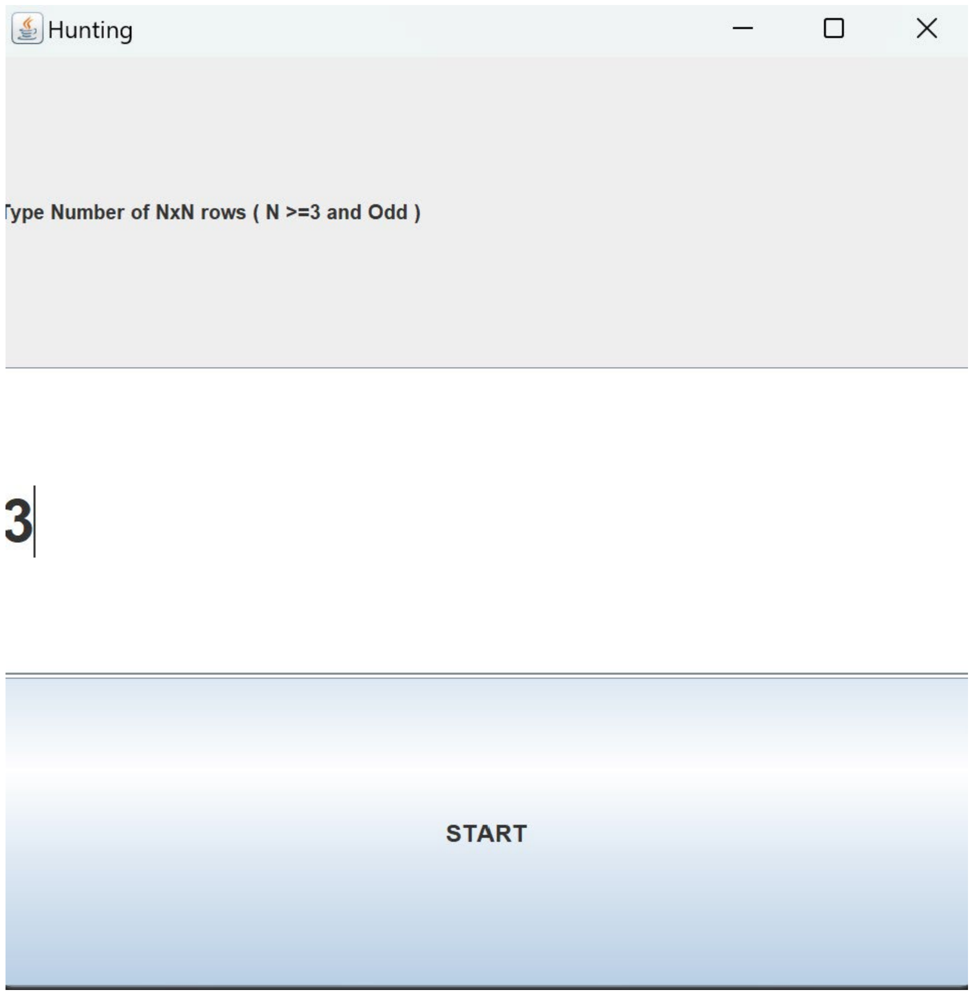
Type Number of NxN rows (N >=3 and Odd)

4

Message

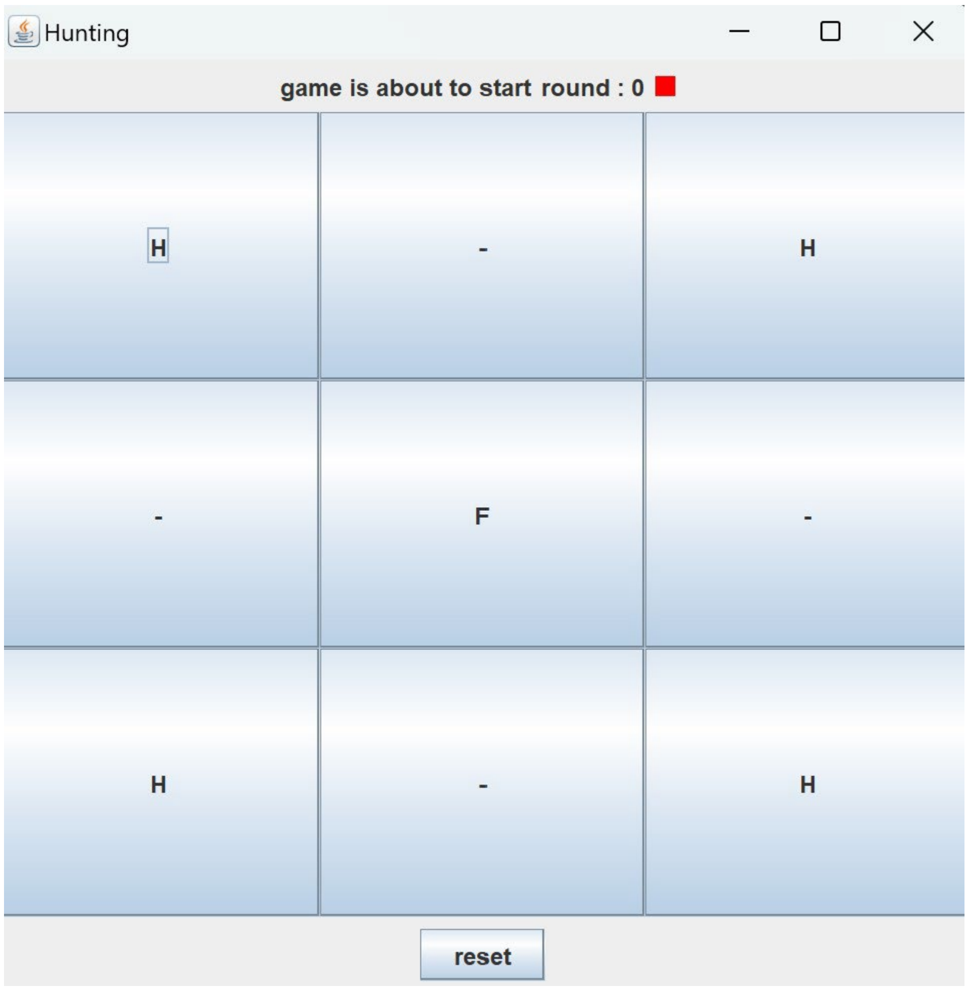
Please enter an odd number greater than or equal to 3.

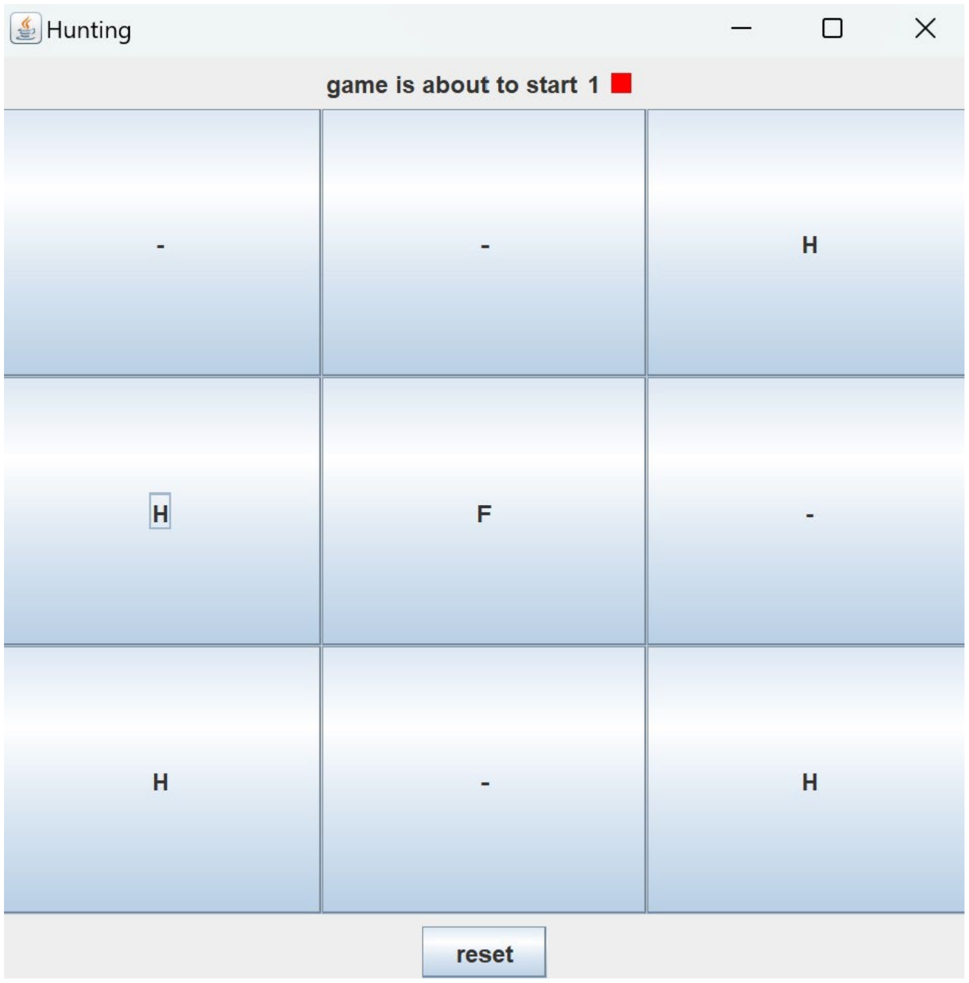
ssdsddsdsdds



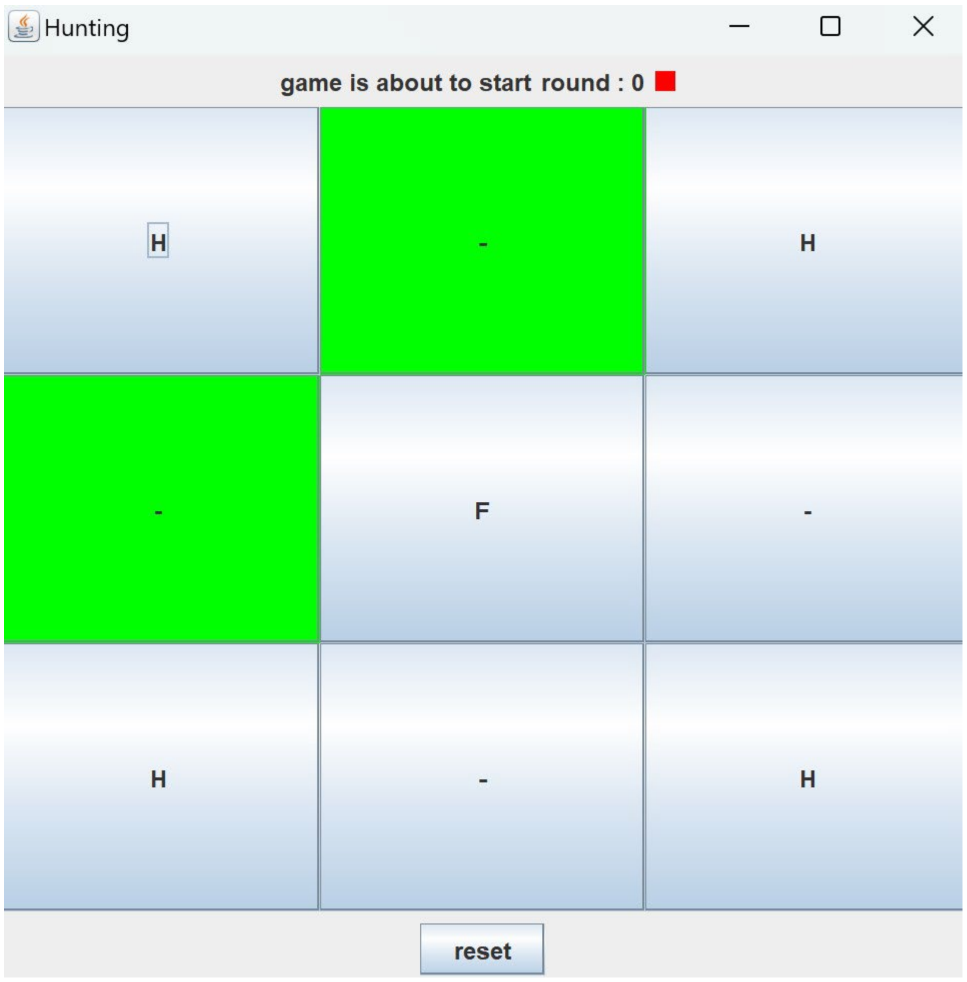
Type Number of NxN rows (N >=3 and Odd)

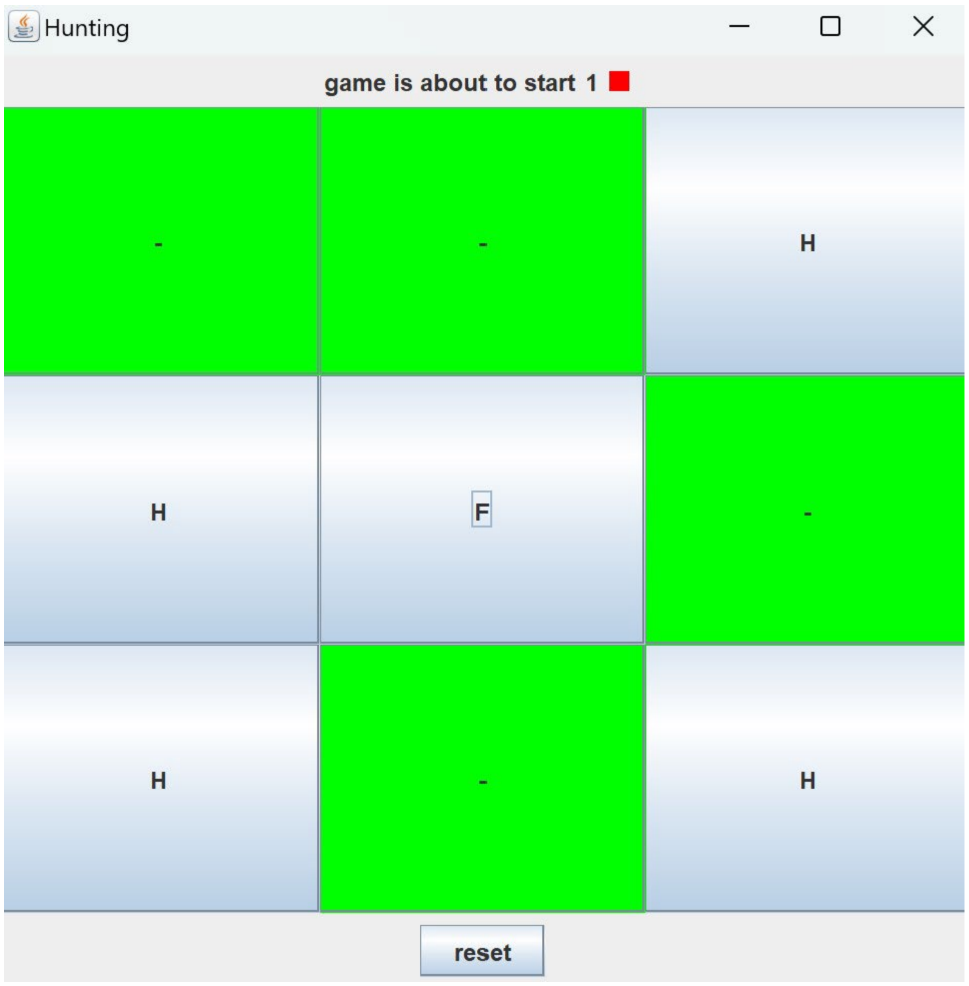
3


Hunting
game is about to start round : 0 🟥
H
-
H
-
F
-
H
-
H
reset

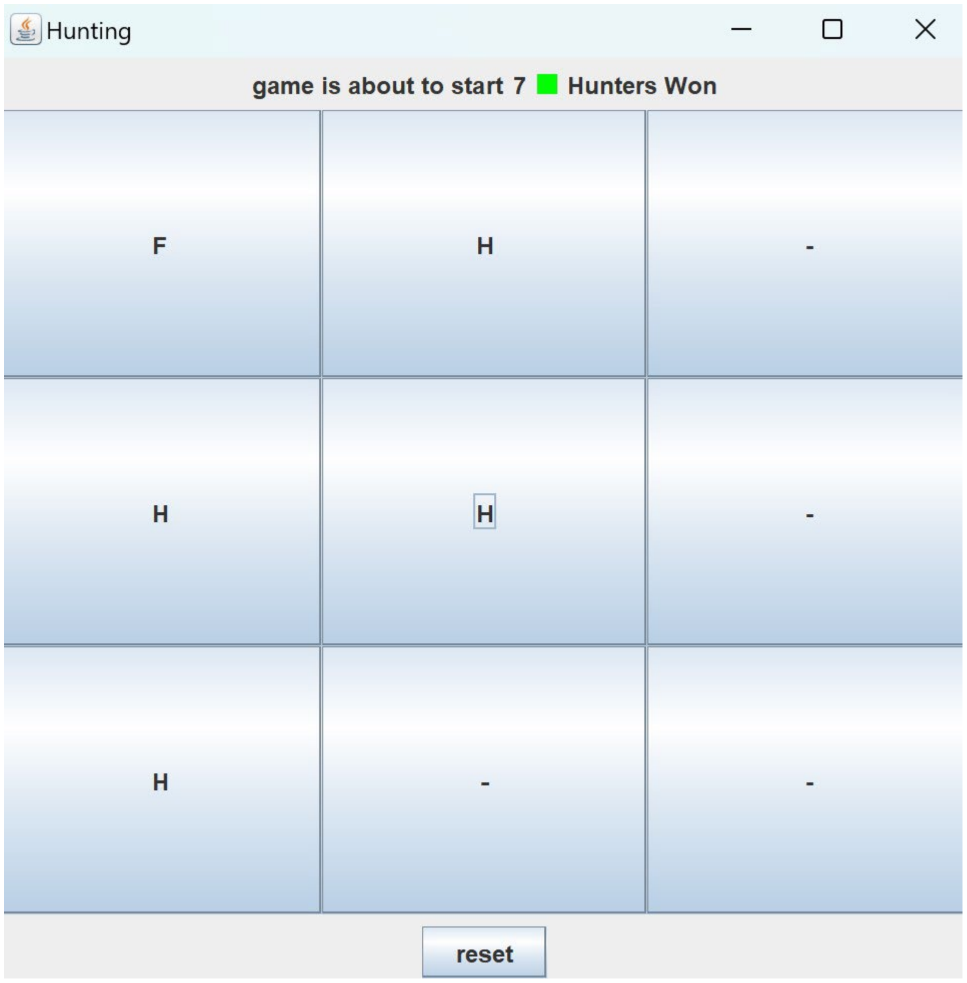


-	-	H
H	F	-
H	-	H


Hunting
game is about to start round : 0
H
-
H
-
F
-
H
-
H
reset

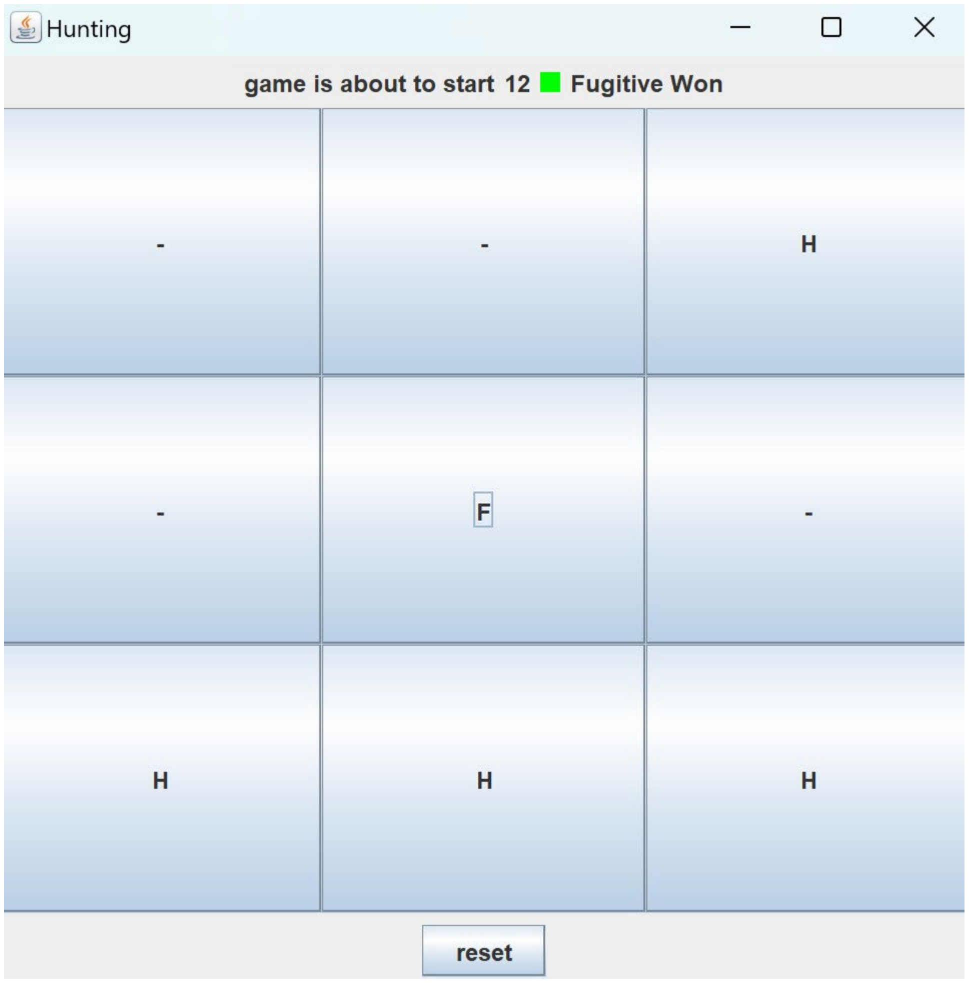

Hunting
game is about to start 1
-
-
H
H
F
-
H
-
H
reset

result output



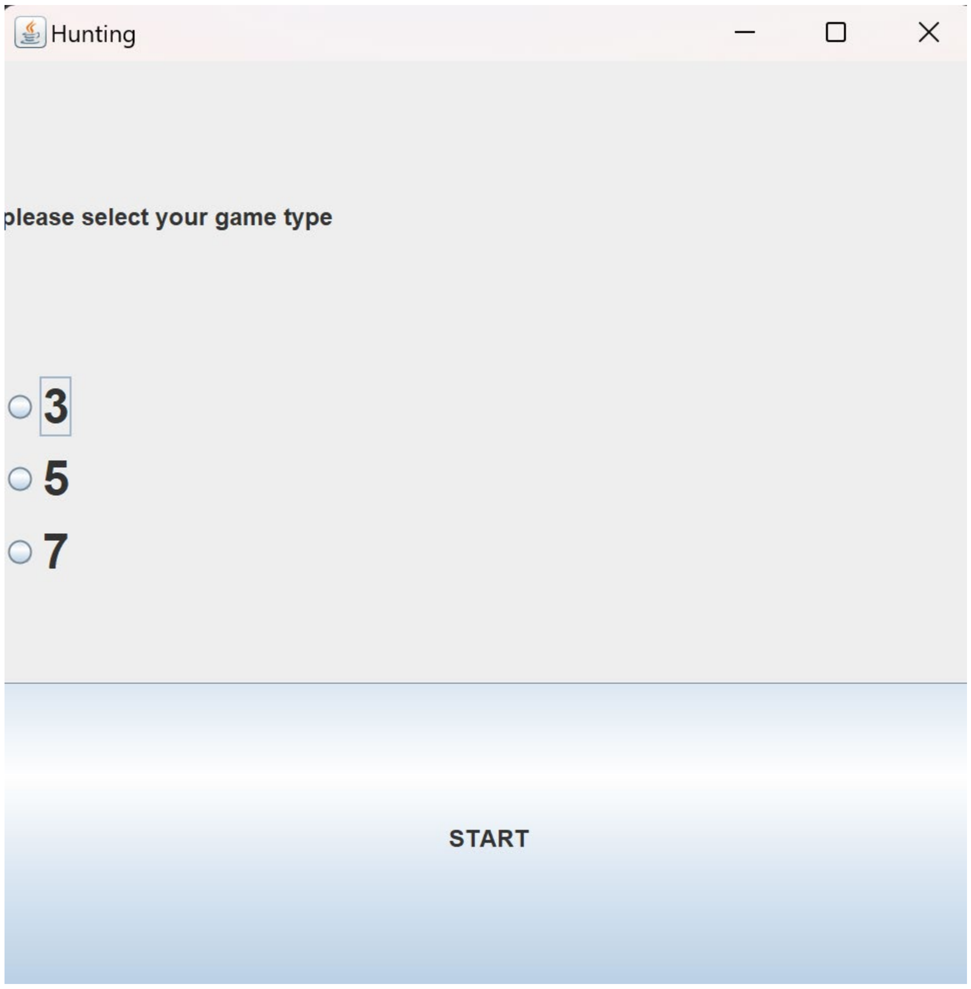
game is about to start 7 🟩 Hunters Won

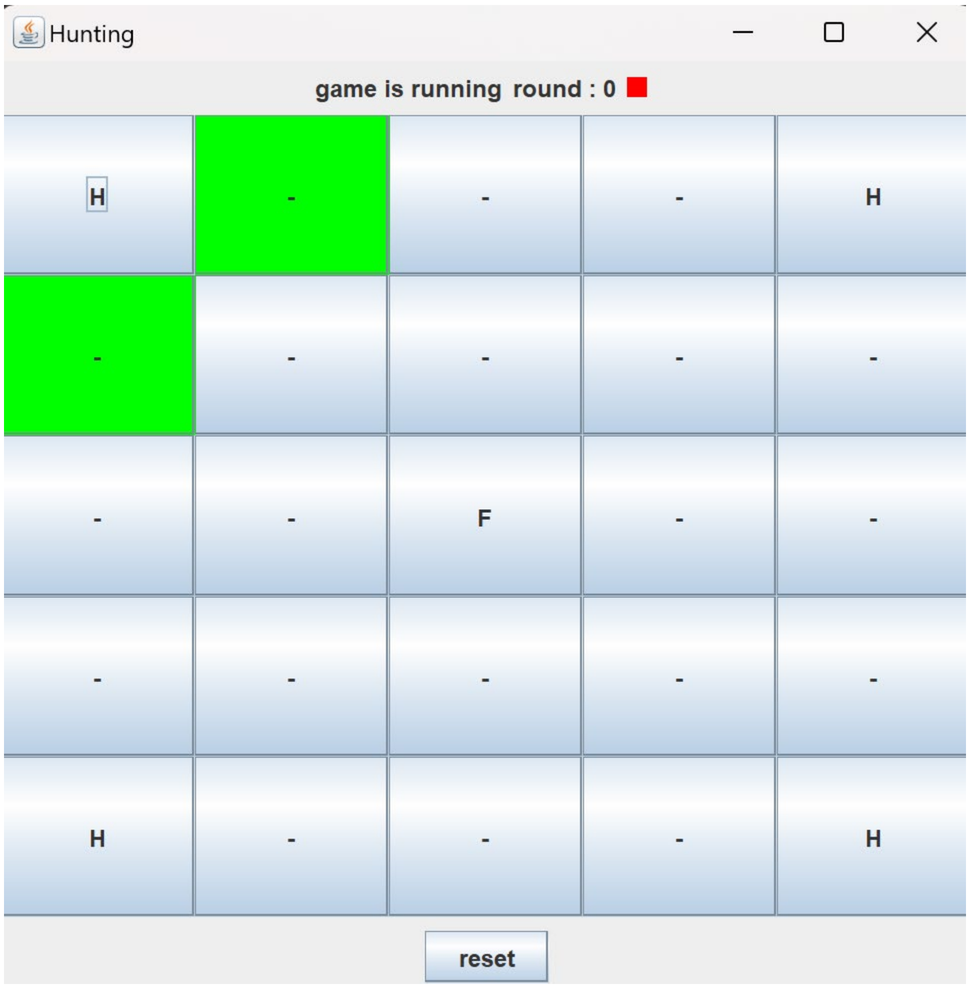
F	H	-
H	H	-
H	-	-



VERSION 2

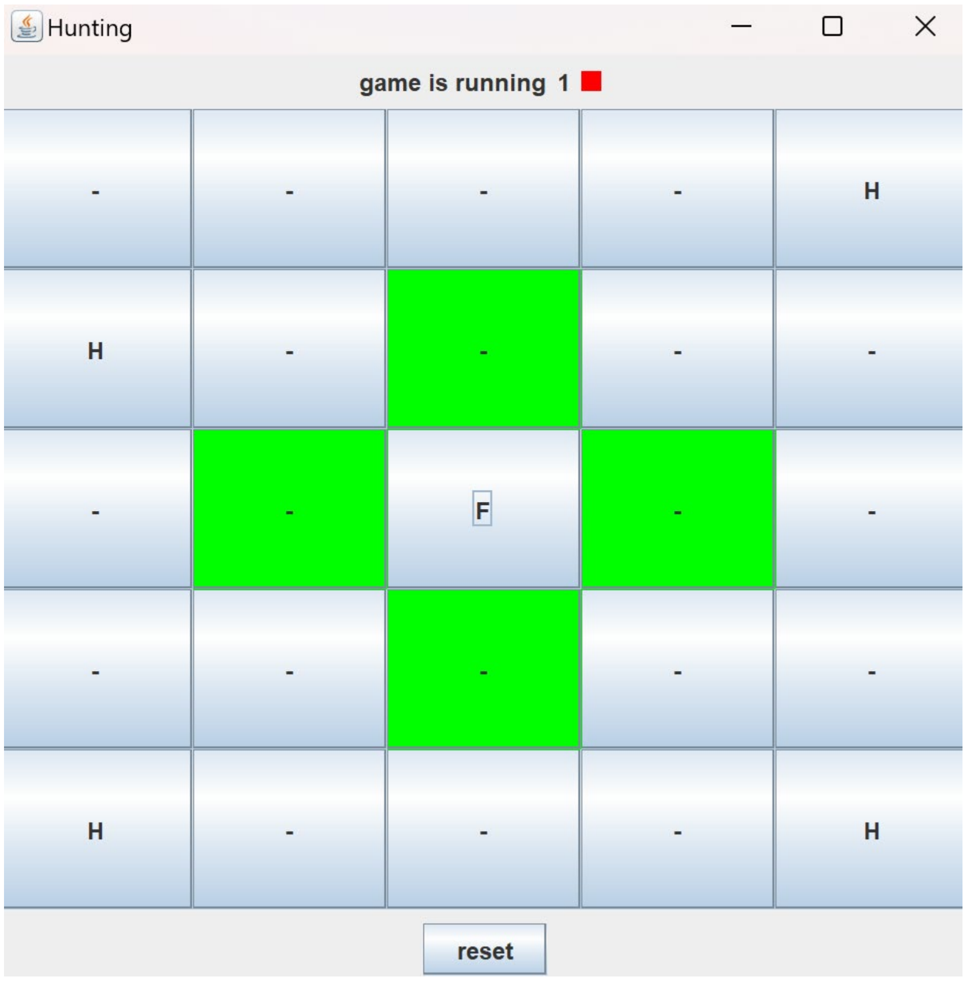
---------------------------->

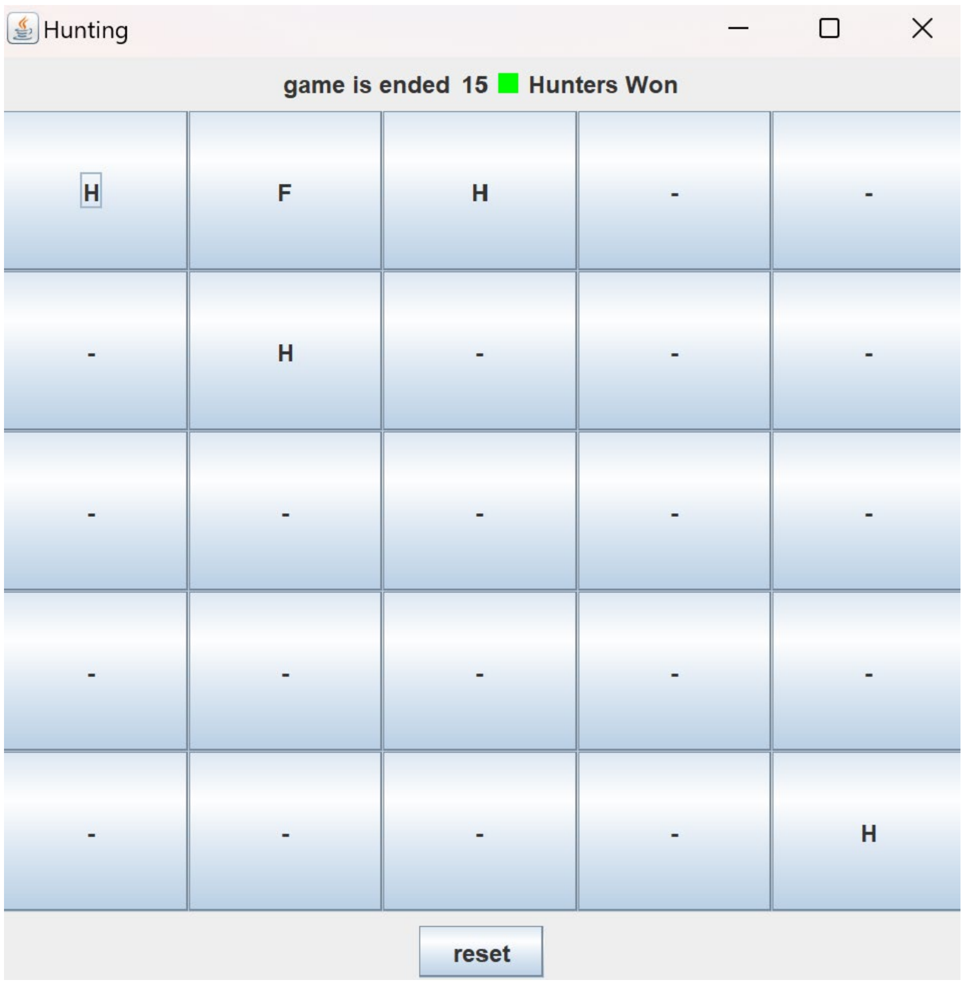

Hunting
please select your game type
3
5
7
START



game is running round : 0 🟥

```
H - - - H
- - - - -
- - F - -
- - - - -
H - - - H
```


Hunting
game is running 1 🟥
- - - - H
H - - - -
- - F - -
- - - - -
H - - - H
reset



Test Documentation for capitalyGame Project

1. PlayerPositionTest.java

Purpose: to manage position of players and position of the buttons on screen

Test Case Overview:

ConstructorTest:

Objective: To validate the constructor's input handling, ensuring that it correctly initializes throws exceptions for invalid inputs.

Details:

This test covers the following scenarios:

- - Creating position with negative value.
- - checking I an J .
- - checking value .

Expected Outcome:

The constructor should throw an IllegalArgumentException for invalid configurations,

Like negative value and check if I J and value is set correctly.

Example Assertion:

assertThrows(IllegalArgumentException.class, () -> new PlayerPosition(-1,-1));

assertEquals("-",position.getValue());

2. PlayerTest.java

Class: Player

Purpose: The Player class represents players in the game, managing their state, actions, and interactions with the board.

Test Case Overview:

ConstructorTest:

Objective: To validate the Player constructor's input handling, ensuring that it correctly initializes .

Expected Outcome:

Throw exception if inputs are null

Example Assertion:

assertThrows(IllegalArgumentException.class,()->new Player(null, Role.*FUGITIVE*,null));

playTest:

Objective: To ensure that the player's movement on the board is accurately tracked and that their positions updates correctly according to game rules.

Details:

This test covers the following scenarios:

- first we create a board
- then we select the player at 0,0 position which is a hunter
- play to a position 0,1
- play to a position 1,1 where some one else is there
- move to a null position

Expected Outcome:

Check if the move was correct

And check player position after moving to some one position

Or if parameter is null

Example Assertion:

assertTrue(player.play(board.getPositions()[0][1]));

assertEquals(board.getPositions()[0][1],player.getPlayerPosition());

3. BoardTest.java

Purpose: *class for creating the board where players positions are and where the matrix (the data) will be saved*

Test Case Overview:

constructorTest:

Objective: To verify that the board initializes correctly .

Details:

Checking to see if we can create the the board with even value or negative or zero

Expected Outcome:

It should throw exception and the board and players on it

Should be correctly initiated and have the right values (“F”) for fugitive and (“H”) for hunters

Example Assertion:

assertThrows(IllegalArgumentException.class,()->new Board(4));

assertEquals("F",board.getPositions()[1][1].getValue());

VarifiedPositionTest:

Details:
Checking to see where can a player move

And if its null position should throw exception

Expected Outcome:
It should throw exception if the position is null

And should be equals with generated sample positions

Example Assertion:
assertThrows(IllegalArgumentException.class,()->board.verifiedPositions(null));

assertEquals(expected,board.verifiedPositions(board.getFugitive()));

4. gameBuilderTest.java

Purpose: This class represents Builder class for the game who can do the rules of the game

And run the board and players methods

Test Case Overview:

ConstructionTest:
Objective: To initate the class with right parameters

Details:
Giving values of negative for end turn and positive value for matrix size .

Expected Outcome:
Should throw exception if that happends

Example Assertion:
assertThrows(IllegalArgumentException.class,()->new GameBuilder(3,-2));
assertThrows(IllegalArgumentException.class,()->new GameBuilder(4,10));

varifyPlayerTurnTest:
Objective: To test if base on turn and role its players time

Details:
Creating game builder and then since its first round check who's turn is it

Expected Outcome:
The first round hunter need to play so it should be true

Example Assertion:

assertTrue(gameBuilder.varifyPlayerTurn(Role.HUNTER));

winnerTest:
Objective: check at this point of the game who is the winner and return the name

Details:
Moving players and checking the outcome of the game at beginning and the end

Expected Outcome:
First round we check the method to see if we have winner and what is outcome of the game at this point of the game and at end we make hunter win and check again

Example Assertion:
assertEquals("Hunters Won",gameBuilder.winner());
assertNull(gameBuilder.winner());
assertFalse(gameBuilder.isGameOver);