Assignment 2

Name: Rizwan Hussain

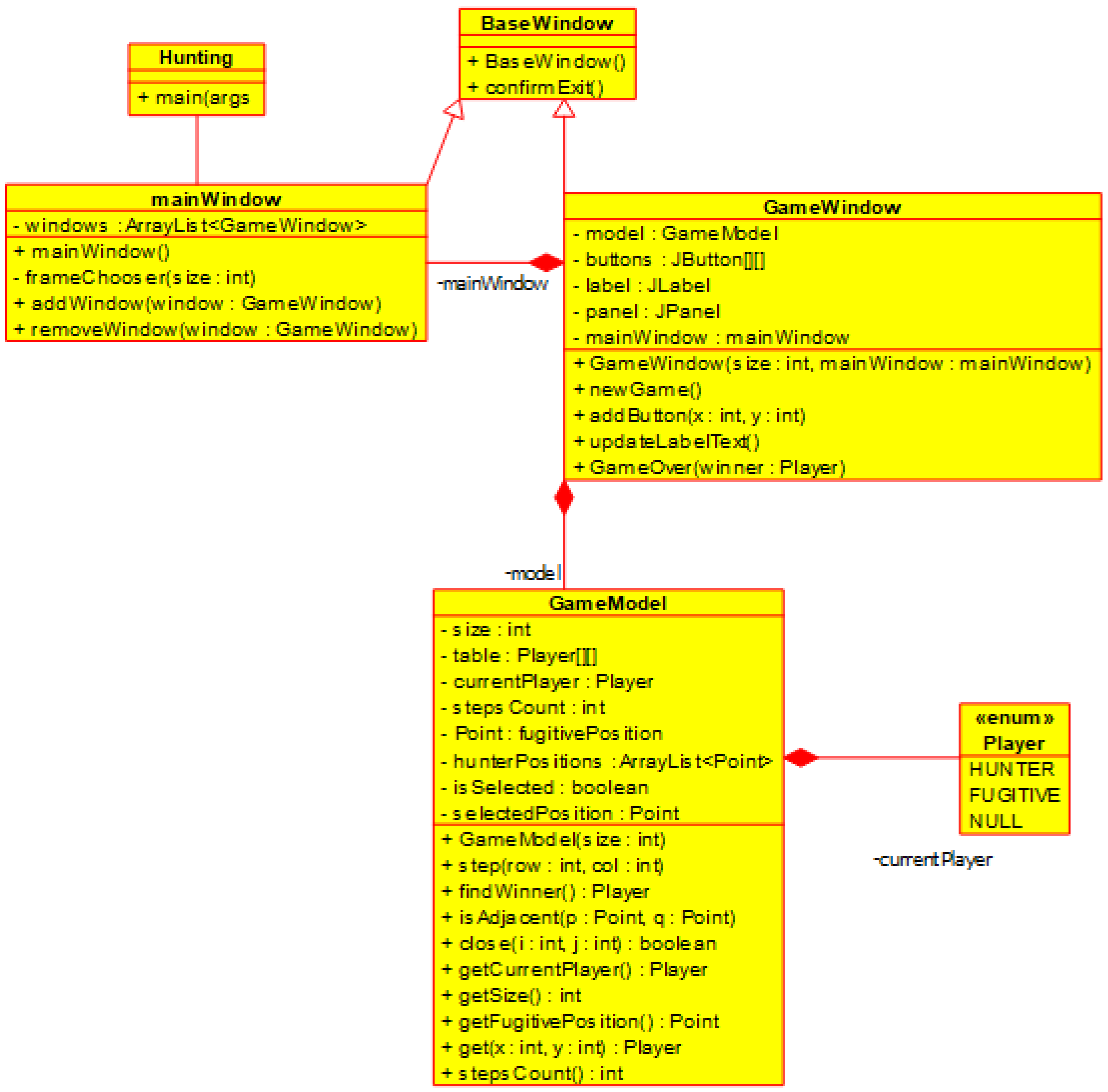
Neptun: ULLBQG

Task Description:

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.

UML Diagram:


BaseWindow
+ BaseWindow()
+ confirmExit()
Hunting
+ main(args
mainWindow
- windows : ArrayList<GameWindow>
+ mainWindow()
- frameChooser(size : int)
+ addWindow(window : GameWindow)
+ removeWindow(window : GameWindow)
-mainWindow
GameWindow
- model : GameModel
- buttons : JButton[][]
- label : JLabel
- panel : JPanel
- mainWindow : mainWindow
+ GameWindow(size : int, mainWindow : mainWindow)
+ newGame()
+ addButton(x : int, y : int)
+ updateLabelText()
+ GameOver(winner : Player)
-model
GameModel
- size : int
- table : Player[][]
- currentPlayer : Player
- stepsCount : int
- Point : fugitivePosition
- hunterPositions : ArrayList<Point>
- isSelected : boolean
- selectedPosition : Point
+ GameModel(size : int)
+ step(row : int, col : int)
+ findWinner() : Player
+ isAdjacent(p : Point, q : Point)
+ close(i : int, j : int) : boolean
+ getCurrentPlayer() : Player
+ getSize() : int
+ getFugitivePosition() : Point
+ get(x : int, y : int) : Player
+ stepsCount() : int
«enum»
Player
HUNTER
FUGITIVE
NULL
-currentPlayer

Description of Methods:

BaseWindows Class:

Constructors:

BaseWindow(): Sets the title, size and the closing operations of the BaseWindow.

Other Methods:

confirmExit(): Confirms if the user really want to exit the game.

mainWindow Class:

Constructors:

mainWindow(): Initializes the windows as empty ArrayList. Adds three buttons to the panel to choose the frame sizes.

Other Methods:

frameChooser(): Takes an int size and creates a new GameWindow with that size and adds it to the windows.

addWindow(): Takes a GameWindow and adds it to the windows.

removeWindow(): Takes a GameWindow and removes it from the windows.

GameWindow Class:

Constructors:

GameWindow(): Takes an int size and a mainWindow and initializes all the fields and add clickable buttons to the panel.

Other Methods:

newGame(): Removes the current window from the windows list and disposes it. Also it adds a new GameWindow to the windows.

addButton(): Takes two integers as coordinates of the buttons and sets the text of the respective button accordingly. Furthermore, it adds actionListener to the button and adds the button to the panel.

updateLabelText(): Updates the above label text with the name of the current player.

gameOver(): Takes a Player winner and announces it in a popup message.

GameModel Class:

Constructors:

GameModel(): Takes an int size and initializes all the fields accordingly. Gives initial positions to the Fugitive and Hunter in table and all other positions are assigned to Null Player.

Getters:

getCurrentPlayer(): Returns the currentPlayer.

getSize(): Returns the size of the board.

get(): Takes two integer as coordinates and returns the Player at that coordinates in the table.

stepsCount(): Returns stepsCount.

Other Methods:

Step(): Takes two integers and completes a step for the current player if there is an empty place and toggles the current Player.

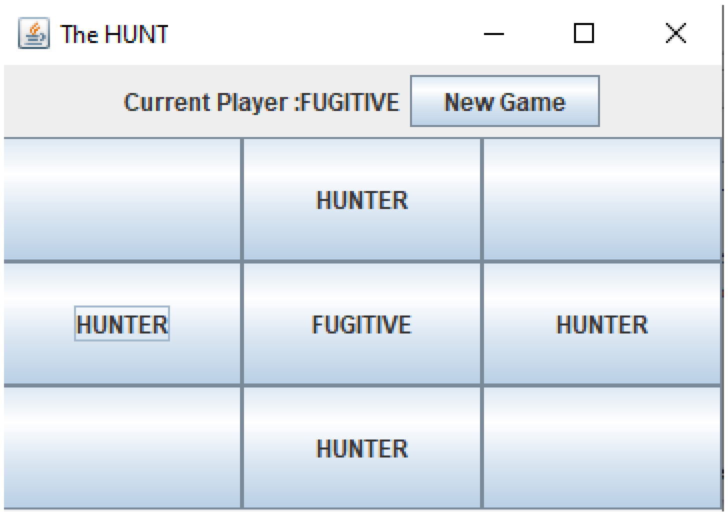
findWinner(): Checks if the stepsCount is less than 4*size and Fugitive is surrounded by the Hunter. If so, then returns the Hunter otherwise returns Null and if stepsCount is more than 4*size then returns Fugitive.

isAdjacent(): Takes two Points and returns if they are not same and are adjacent.

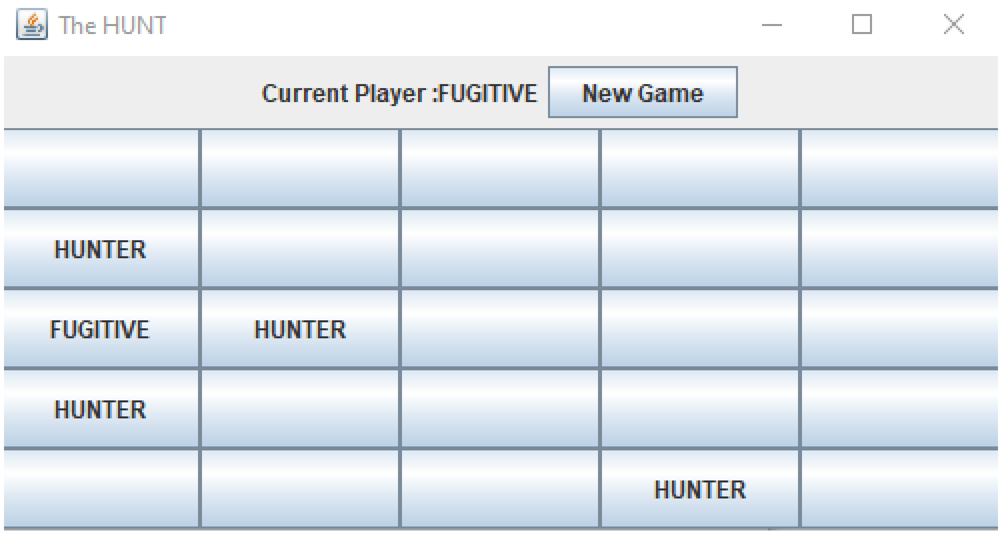
Close(): Takes two integers and decides if their difference is equal to one.

Testing:

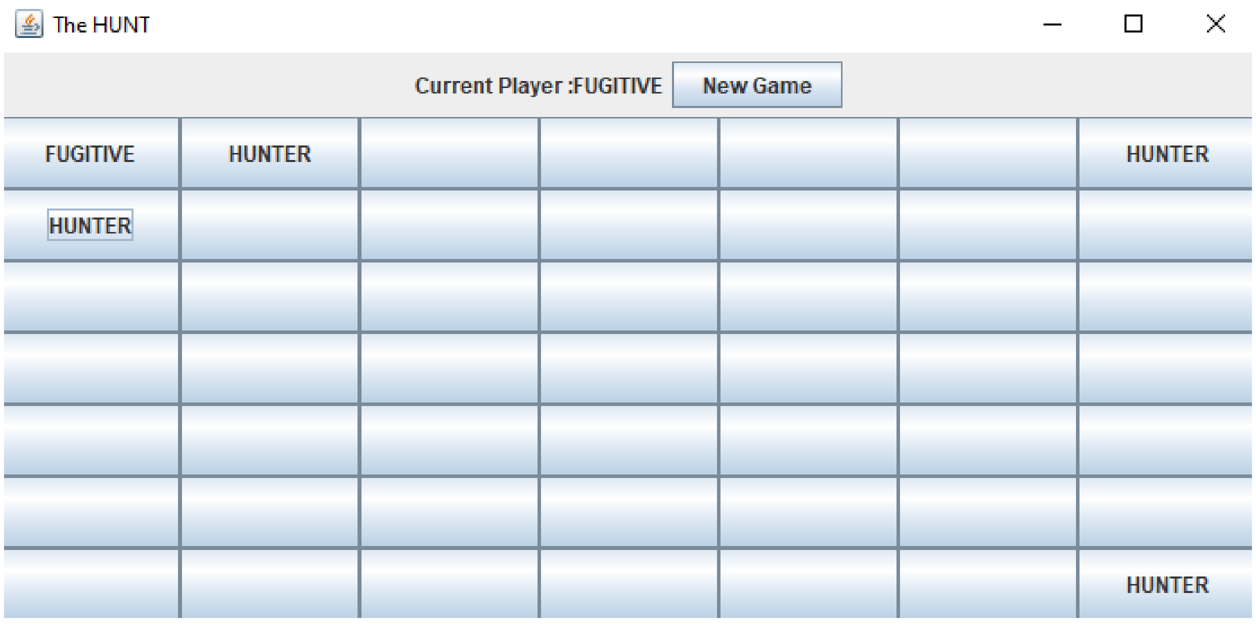
1. When the moves are less than 4*size and Hunter surrounds the Fugitive from four sides. The winner is the Hunter.



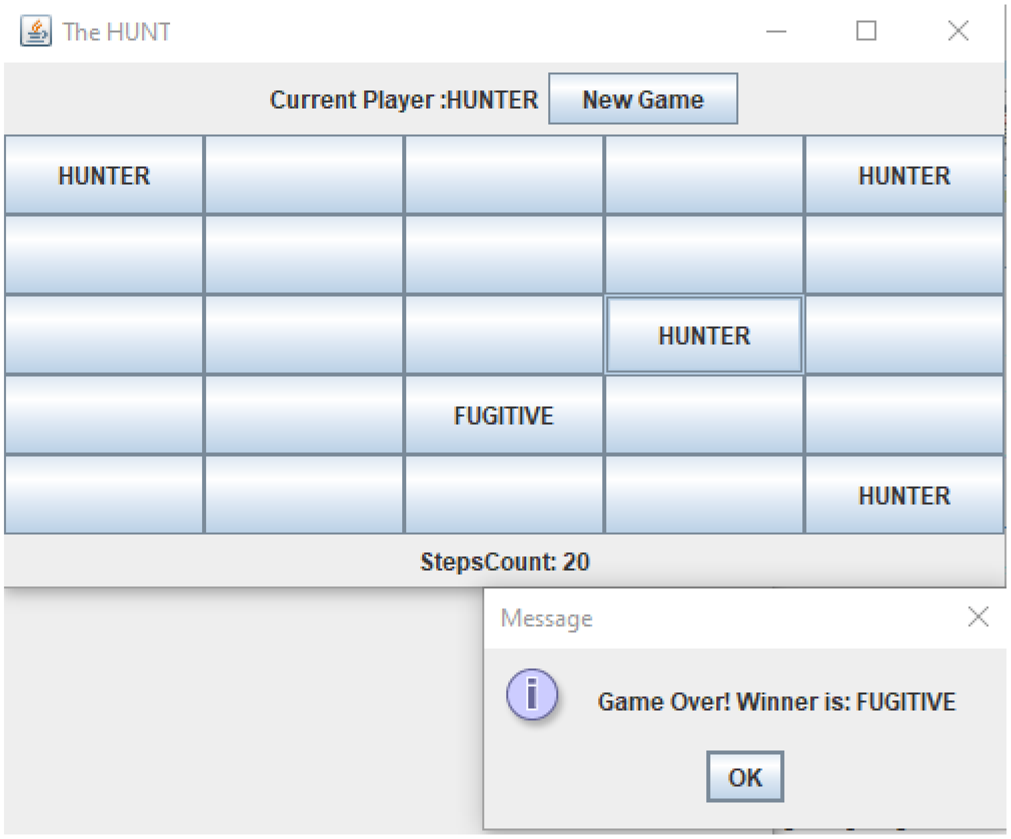
2. When the moves are less than 4*size and Hunter surrounds the Fugitive from three sides. The winner is the Hunter.



3. When the moves are less than 4*size and Hunter surrounds the Fugitive from two sides. The winner is the Hunter.



4. When the moves are more than 4*size and Hunter can’t surround Fugitive. Winner is the Fugitive.



Game Over! Winner is: FUGITIVE