Vu Ha Tuyen

Q5QHGC

**Assignment 2**

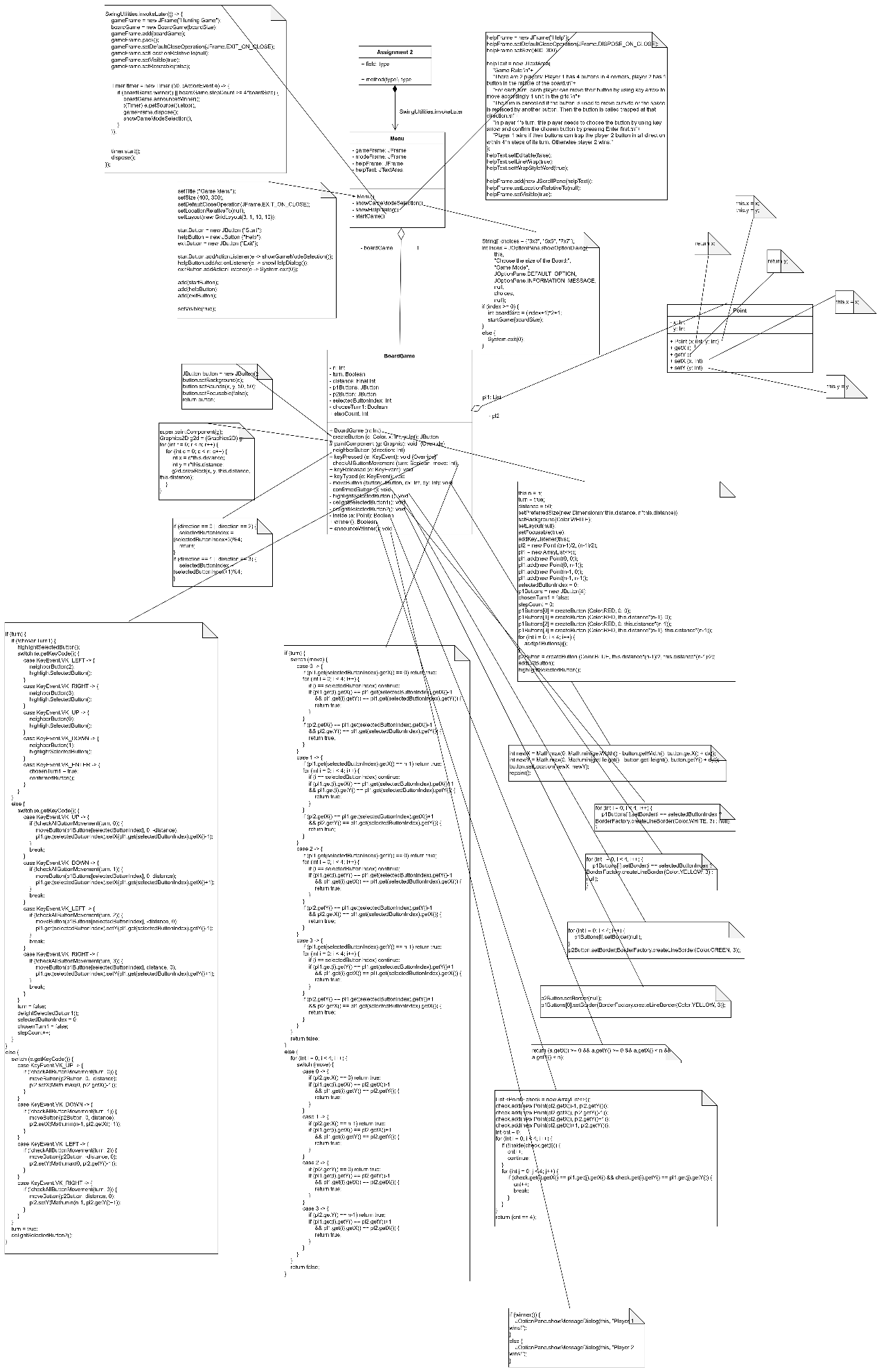
1. Description

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.

1. Class diagram

Source: [Assignment 2.drawio - draw.io](https://app.diagrams.net/#G1XAqxIKyULJcc52xKNpJVCU1TIMo42mK_#%7B%22pageId%22%3A%22VFunhQOyX-4Ljo7fIsTj%22%7D)



1. Methods:

* The boardGame class: Is extended from JPanel and implements KeyListener. Inside this class, there are these methods:
* paintComponent (g: Graphics):

This method renders a grid of rectangles (a square grid) on a JPanel. Each rectangle represents a single cell in the grid. The size of the grid and spacing between rectangles is determined by n and distance.

* neighborButton (Direction: Int):

This method help player 1 choose the button base on key arrow, left or up (0 or 2) for previous button, right or down (1 or 3) for next button.

* keyPressed (e: KeyEvent):

Base on turn (true for player 1, and false for player 2), highlight the selected button. For player 1, we need to choose the button first, after we press Enter to comfirm that button, that chosen button can be moved by using key arrow. For player 2, we just need to move it by using key arrow.

Update the stepCount, a variable to count how many movements the player1 made to check the winning condition.

* checkAllButtonMovement (turn: Boolean, move: Int):

We check for 1 turn, the button can be moved to that direction (0 is up, 1 is down, 2 is left, 3 is right) or not. 1 button can be moved to that appointed direction if there is no button in that direction, and that position is inside the map.

* moveButton (button: JButton, dx: Int, dy: Int):

Set the button to 1 specific direction (dx: from 0 to 1, dy: from 0 to 1, decide the direction of that button, moving 1 unit).

* confirmedButton ():

To confirm that chosen button for the turn of player 1.

* highlightSelectedButton():

Hightlight the button that is chosen in the turn of player 1.

* delightSelectedButton1():

This is for the turn of player 2. Delight buttons of player 1, highlight player 2’s button.

* delightSelectedButton2():

This is for the turn of player 1. Delight the button of player 2, hightlight the first player 1’s button.

* Inside (a: Point):

To check if a Point is inside the board.

* Winner():

Check if the player 1 already won, if player2 are surrounded by walls or buttons in 4 directions (unable to move anymore).

* AnnounceWinner():

Calling this function after fixed movements. If player 1 wins, it annouces player 1 wins. Vice versa, player 2 wins after enough movements, player 1 could not catch him.

* keyReleased (e: KeyEvent) + keyTyped (e: KeyEvent):

Adding these 2 functions to make the implement of KeyListener complete.

* The Point class: Is a class to represent the point with x, y location in the grid. My program does not just have JButton, I also use Point to be easier in saving locations, instead of always getting locations from calculating.
* The Menu class: To create the Menu window, with some elements: Choosing size of the board, Help and Exit.
* Menu():

This is the constructor, it creates the main menu window, sets the name, size, assigns values for start, help and exit buttons.

* showGameModeSelection():

This method is to choose the size of the grid, in turn: 3x3, 5x5, 7x7.

* showHelpDialog():

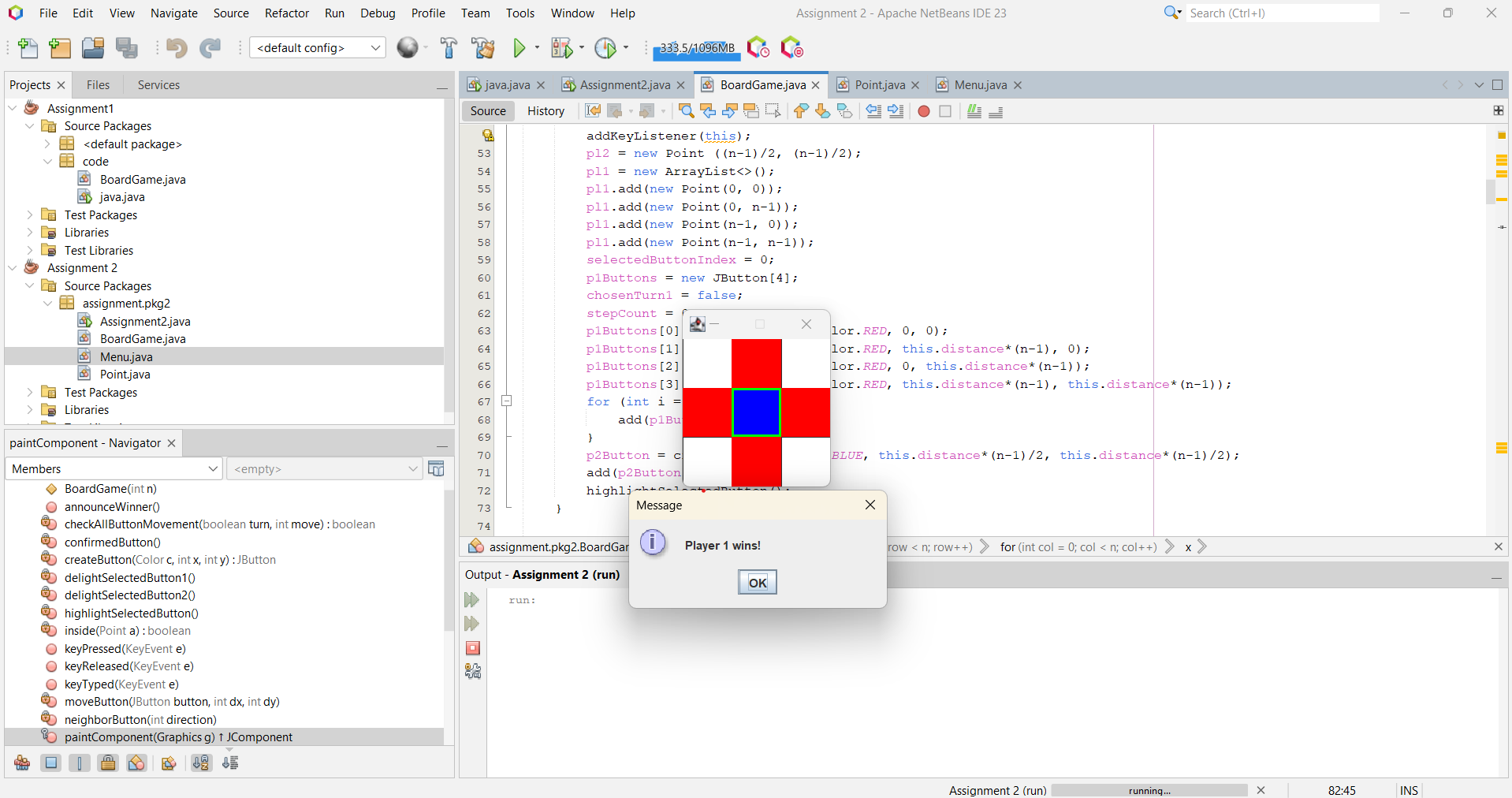
This method shows the game rules.

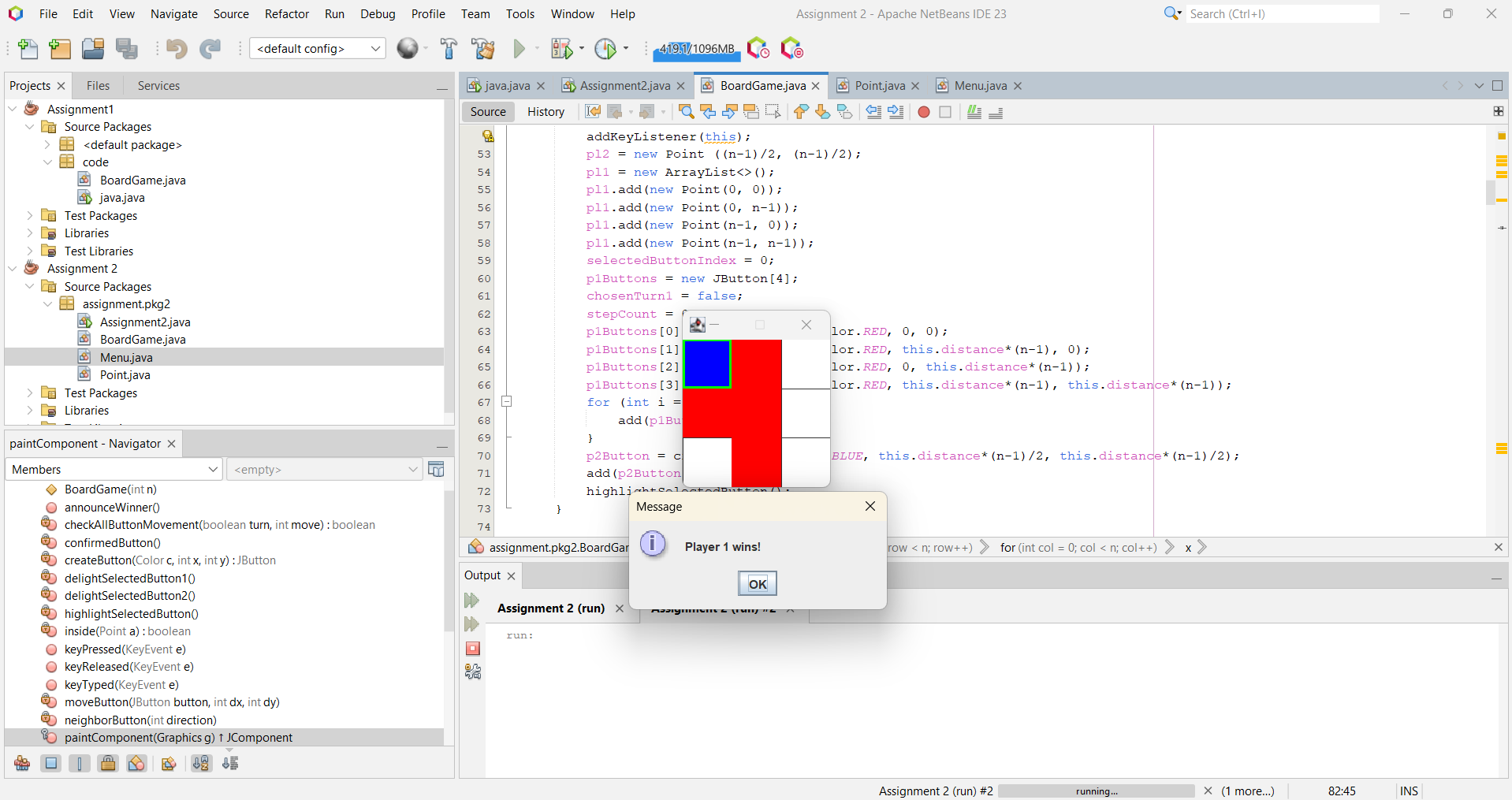
* startGame (boardSize: int):

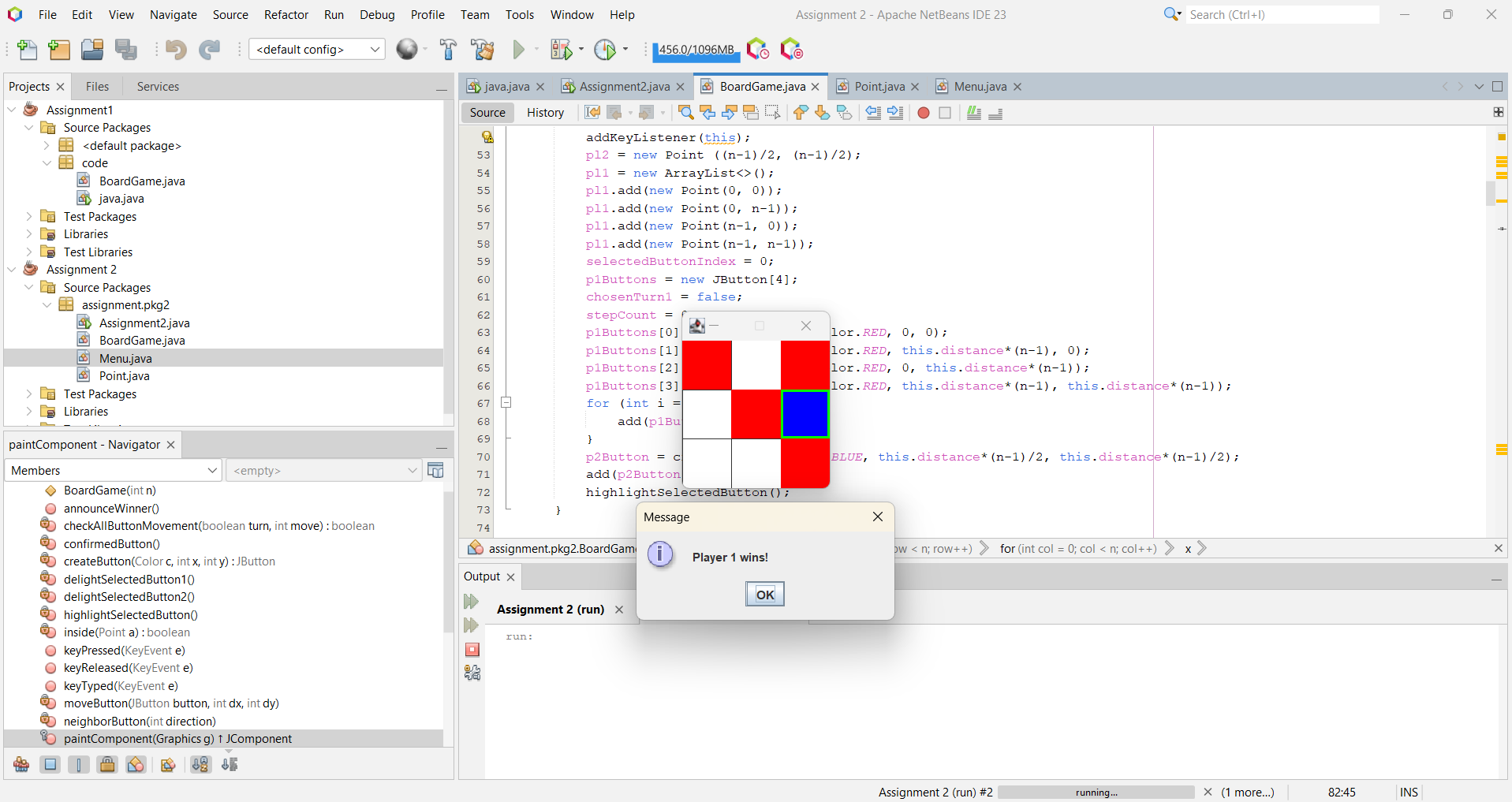
This method creates the game frame and main game loop.

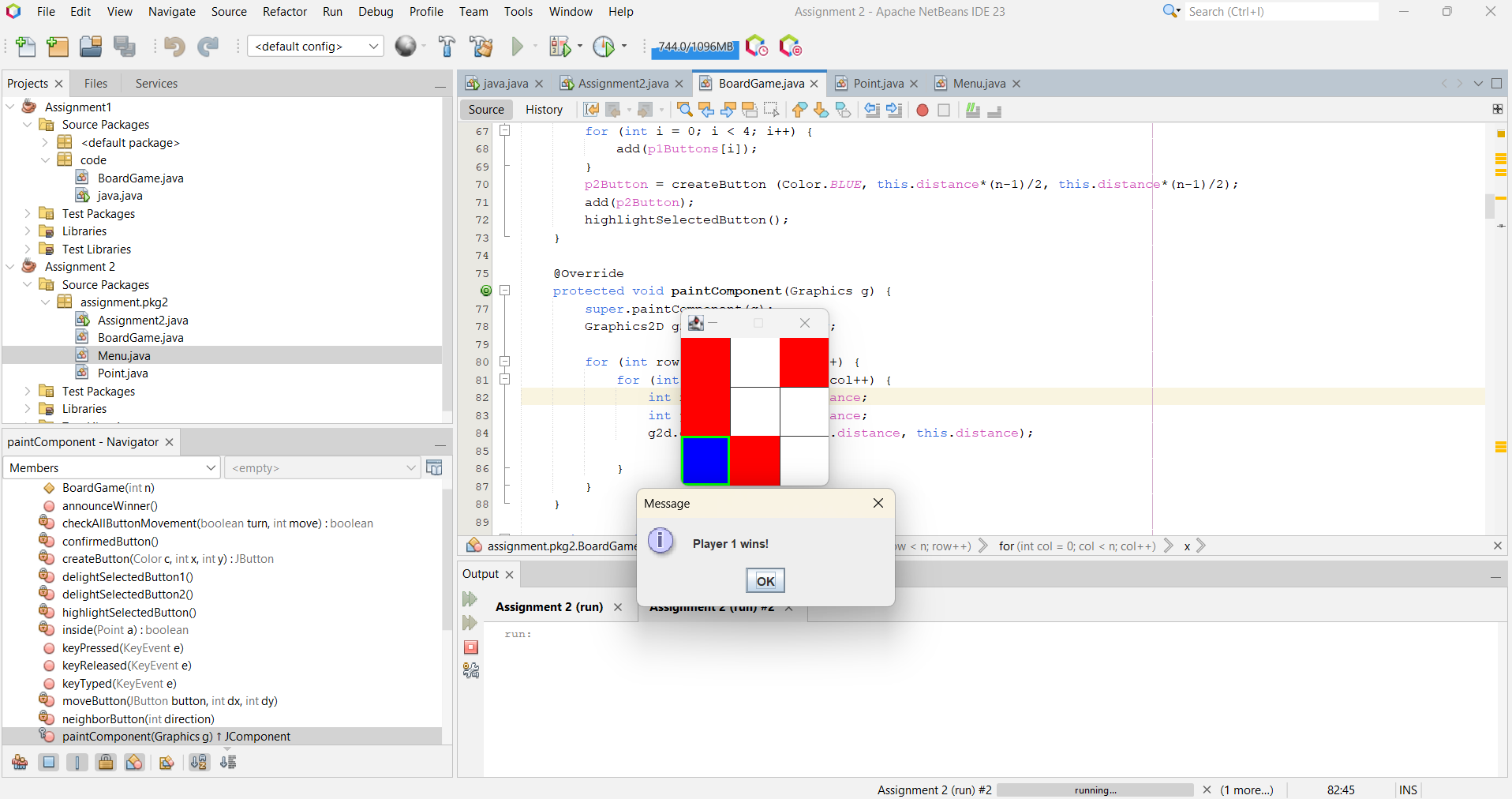
1. List of test cases:

* For 3x3 grid:

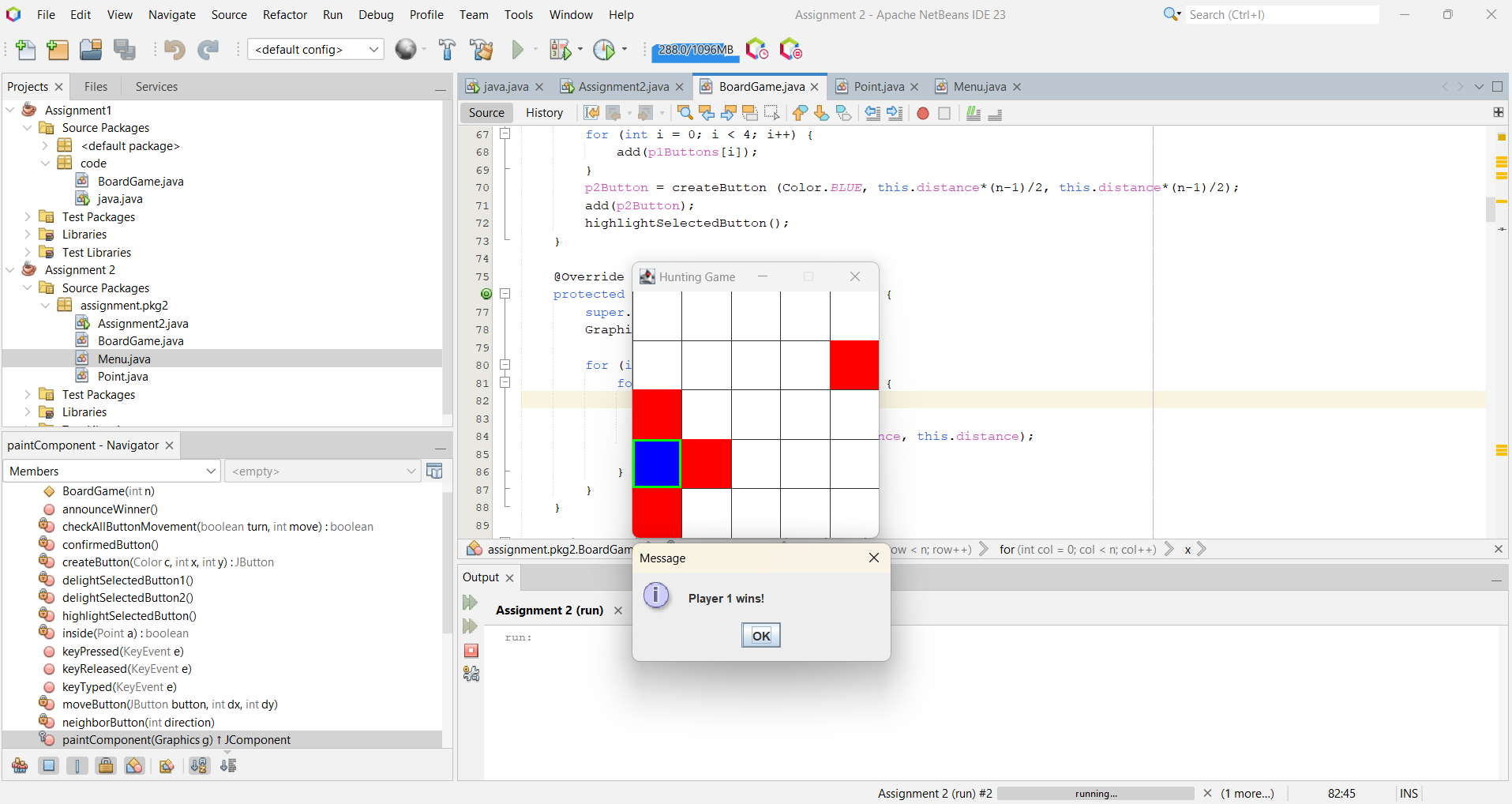


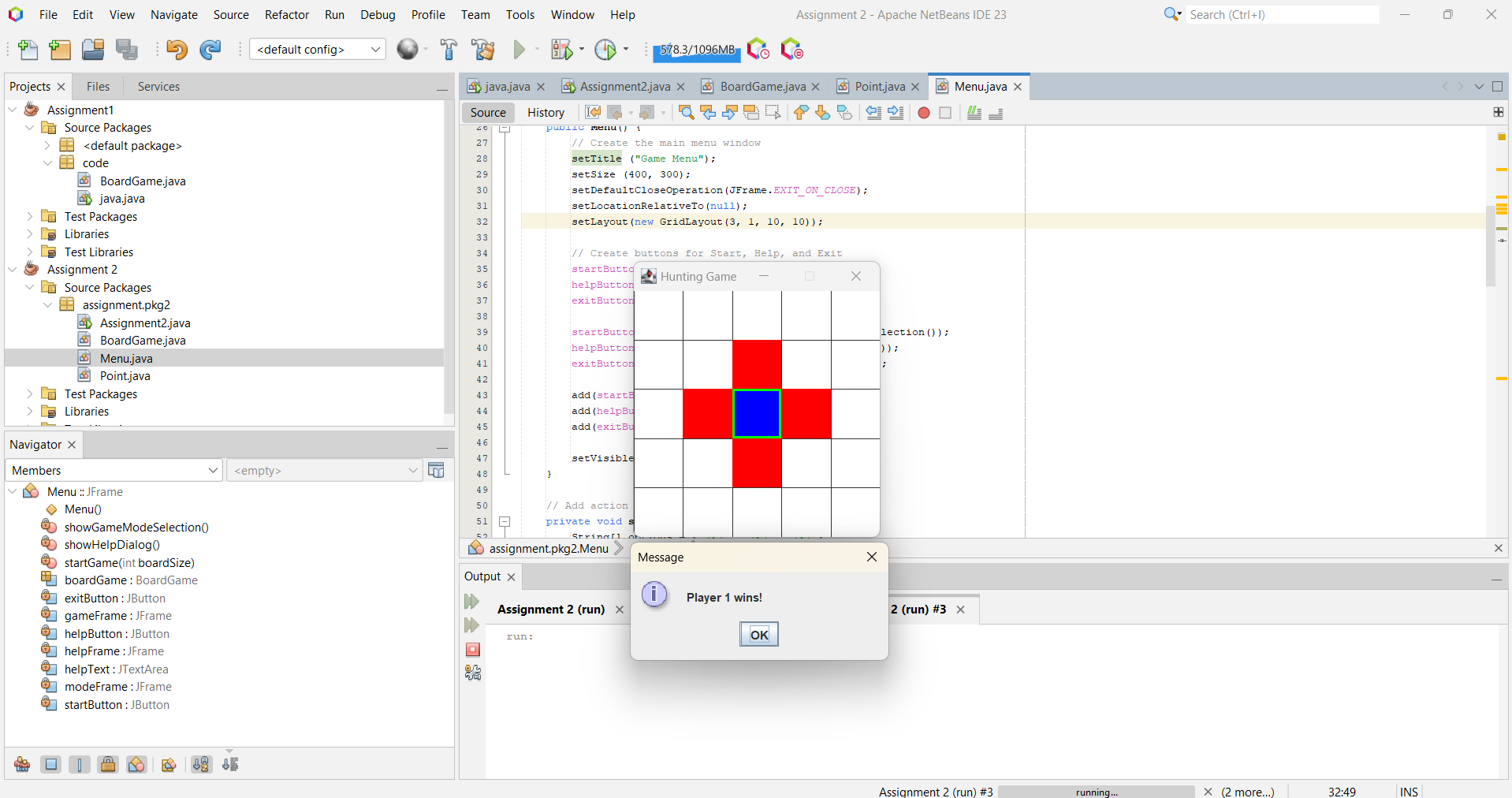


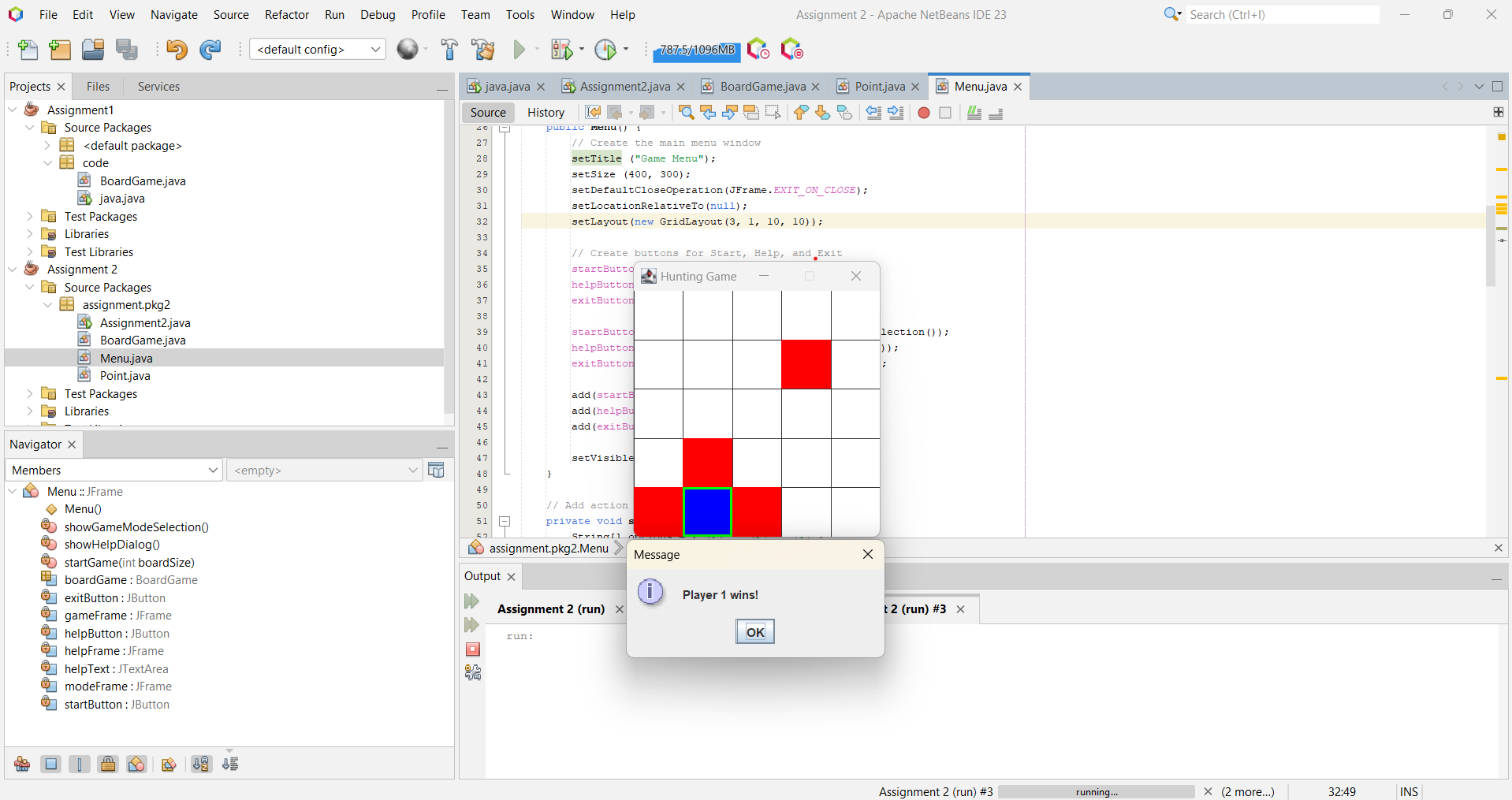


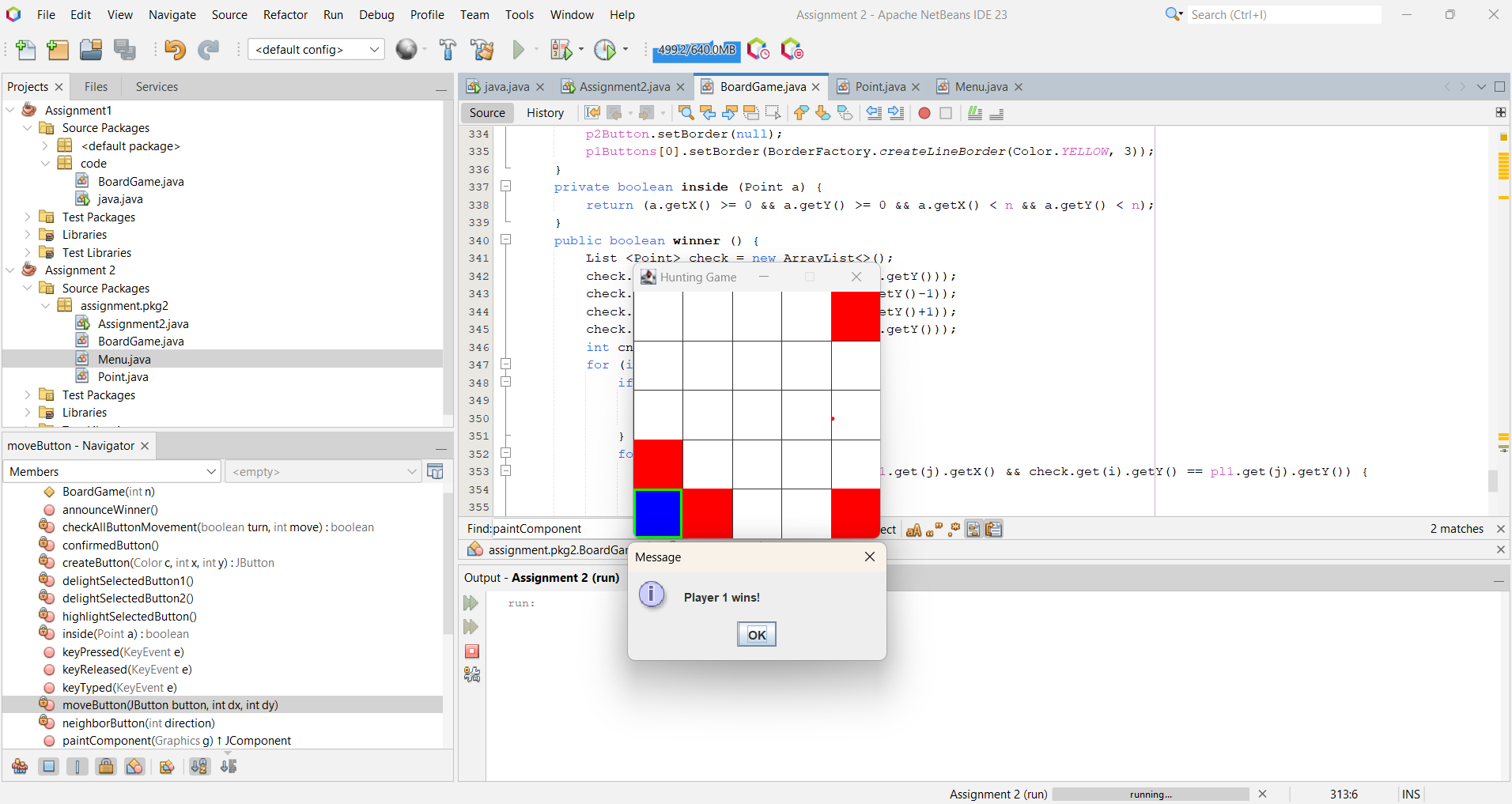


* For 5x5 grid:

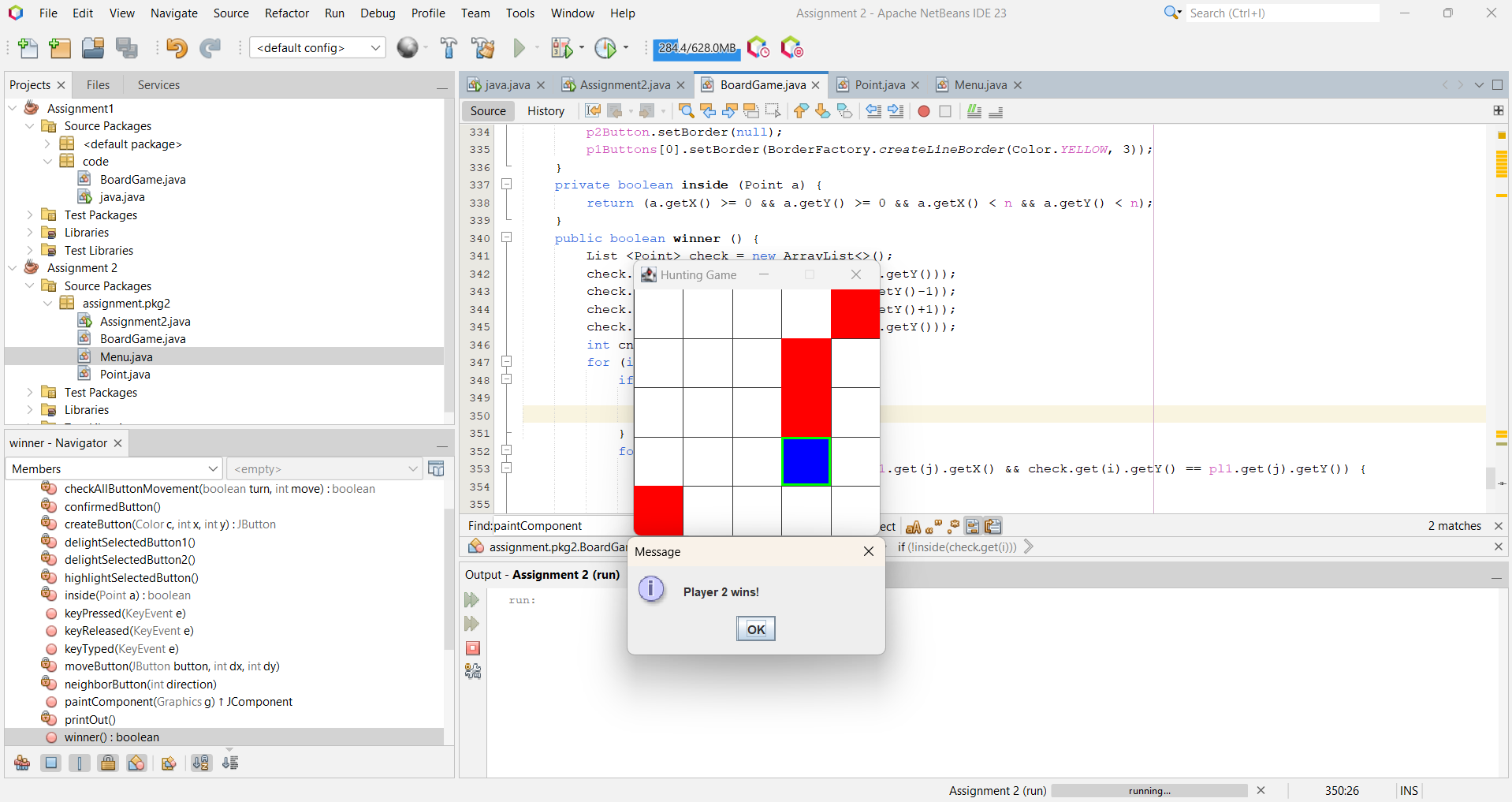




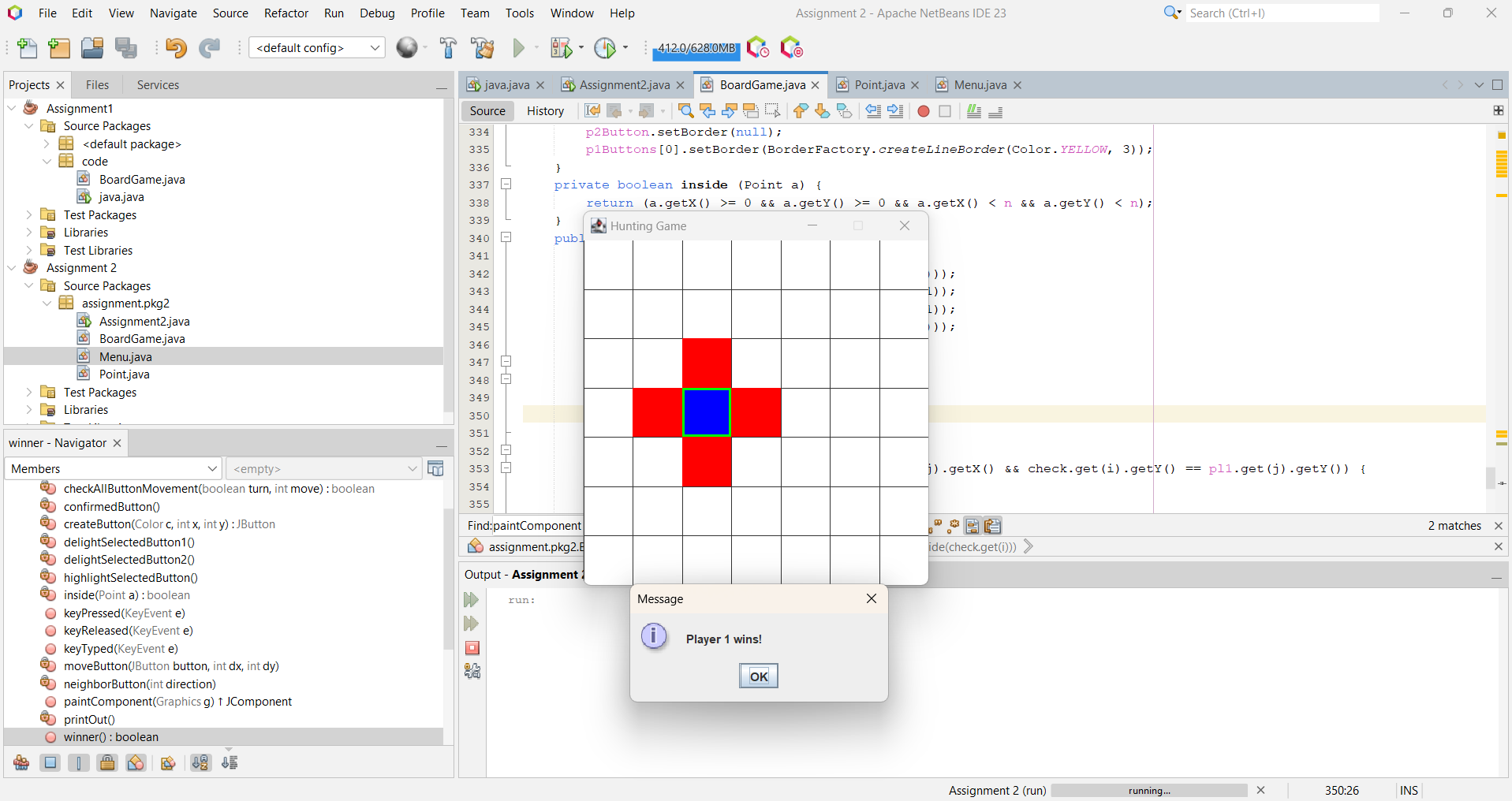


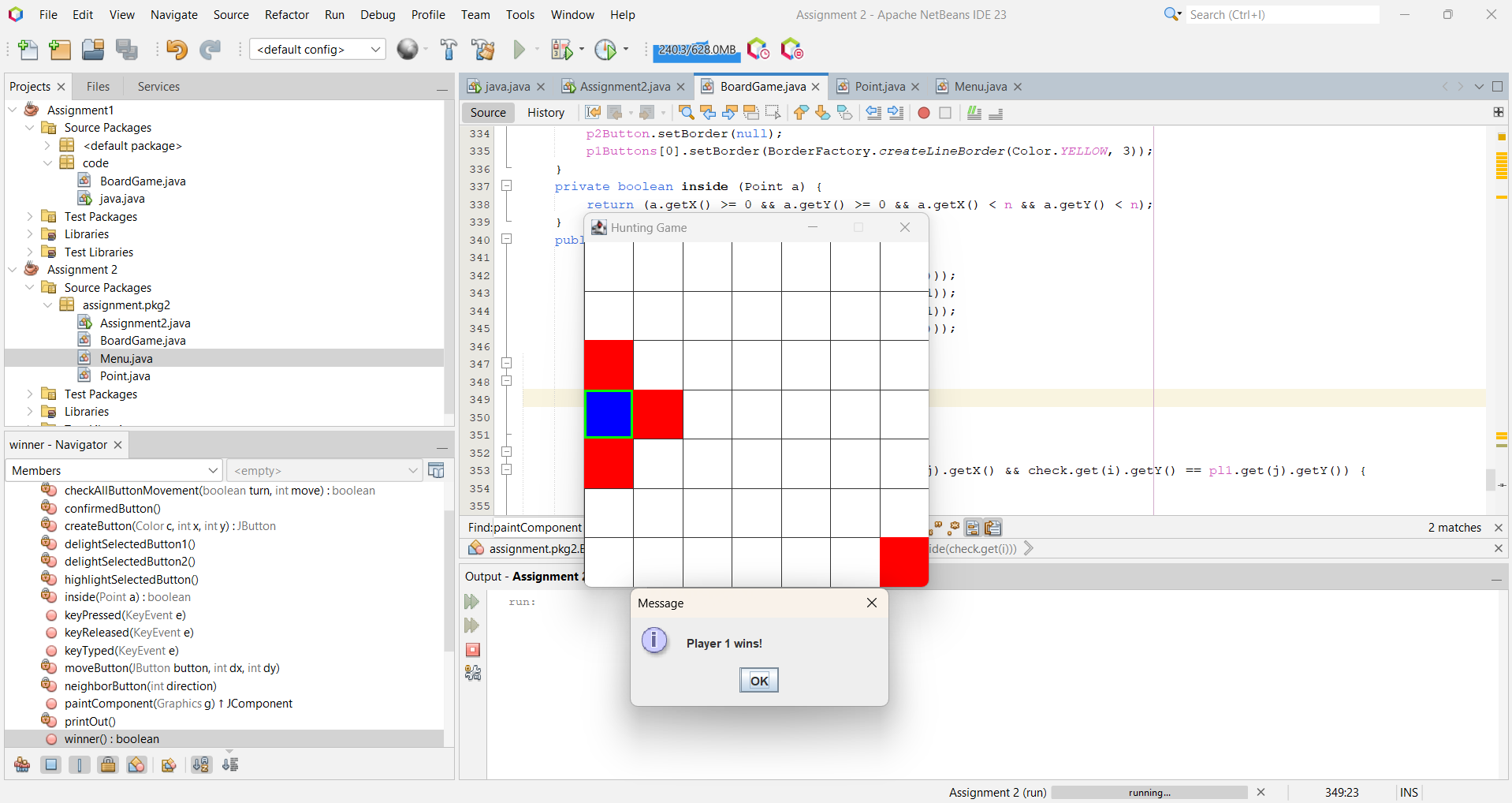


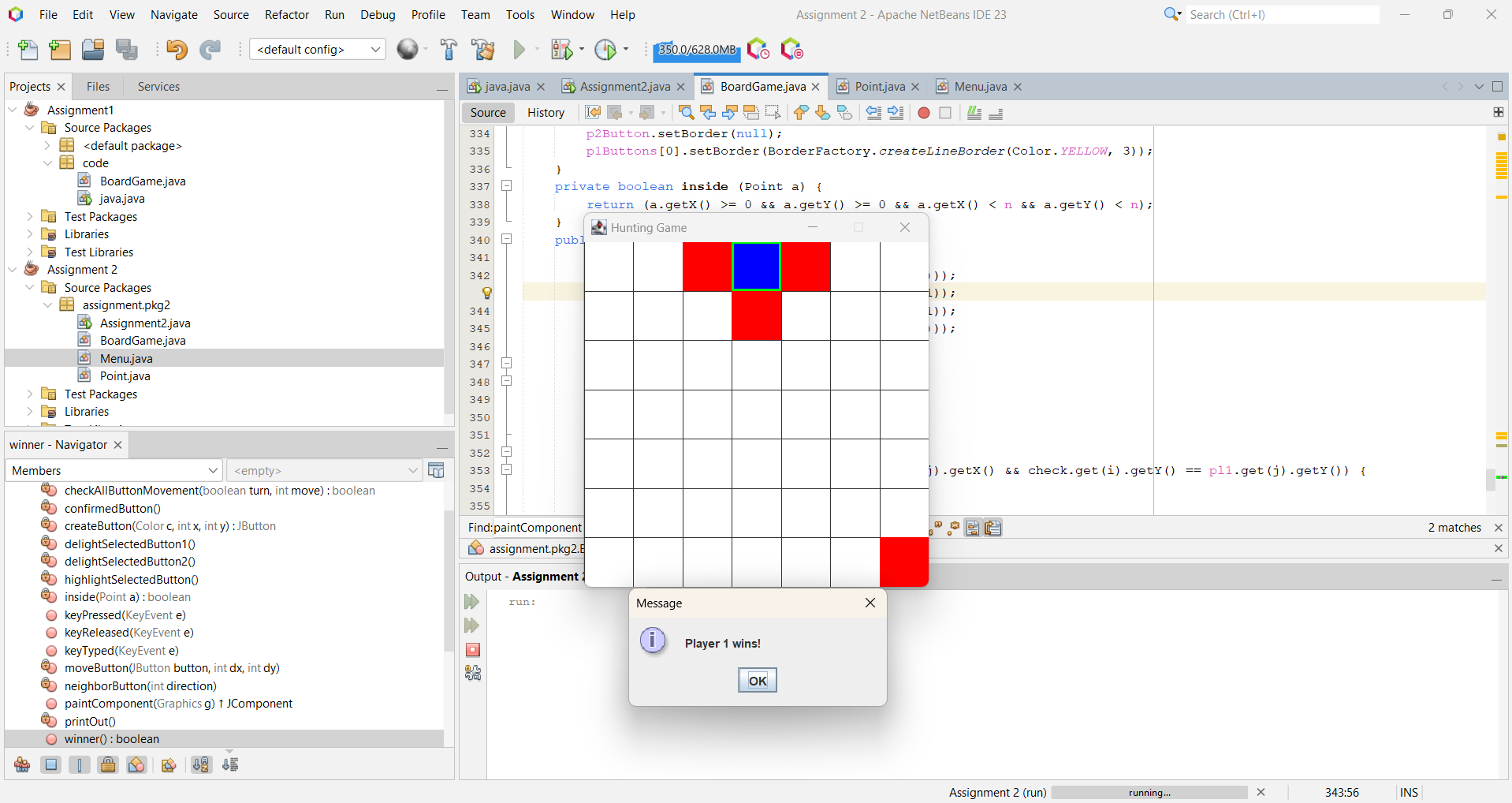
After enough moves (4\*n) of player 1 and he could not catch player 2:

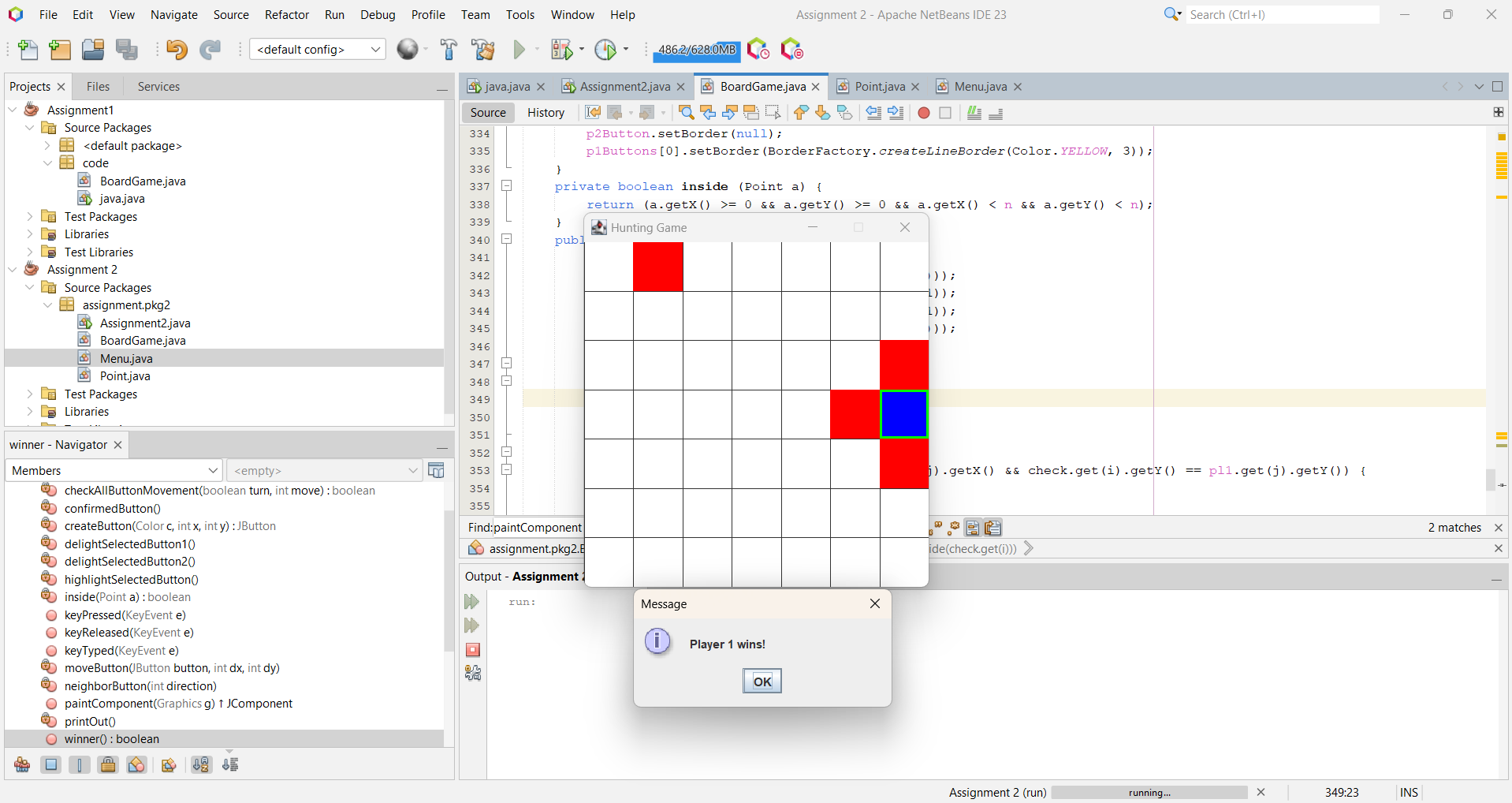


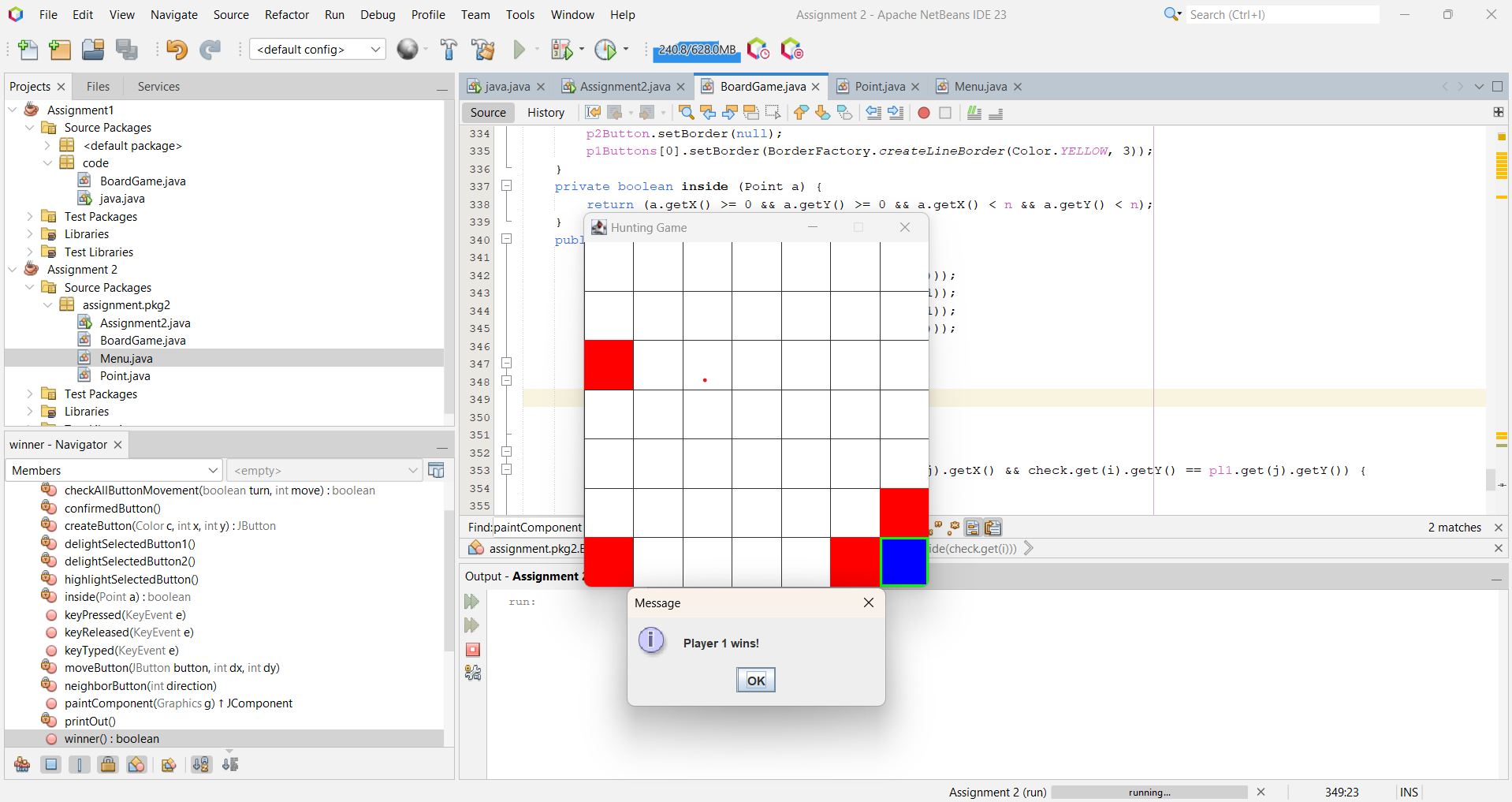
* For 7x7 grid:



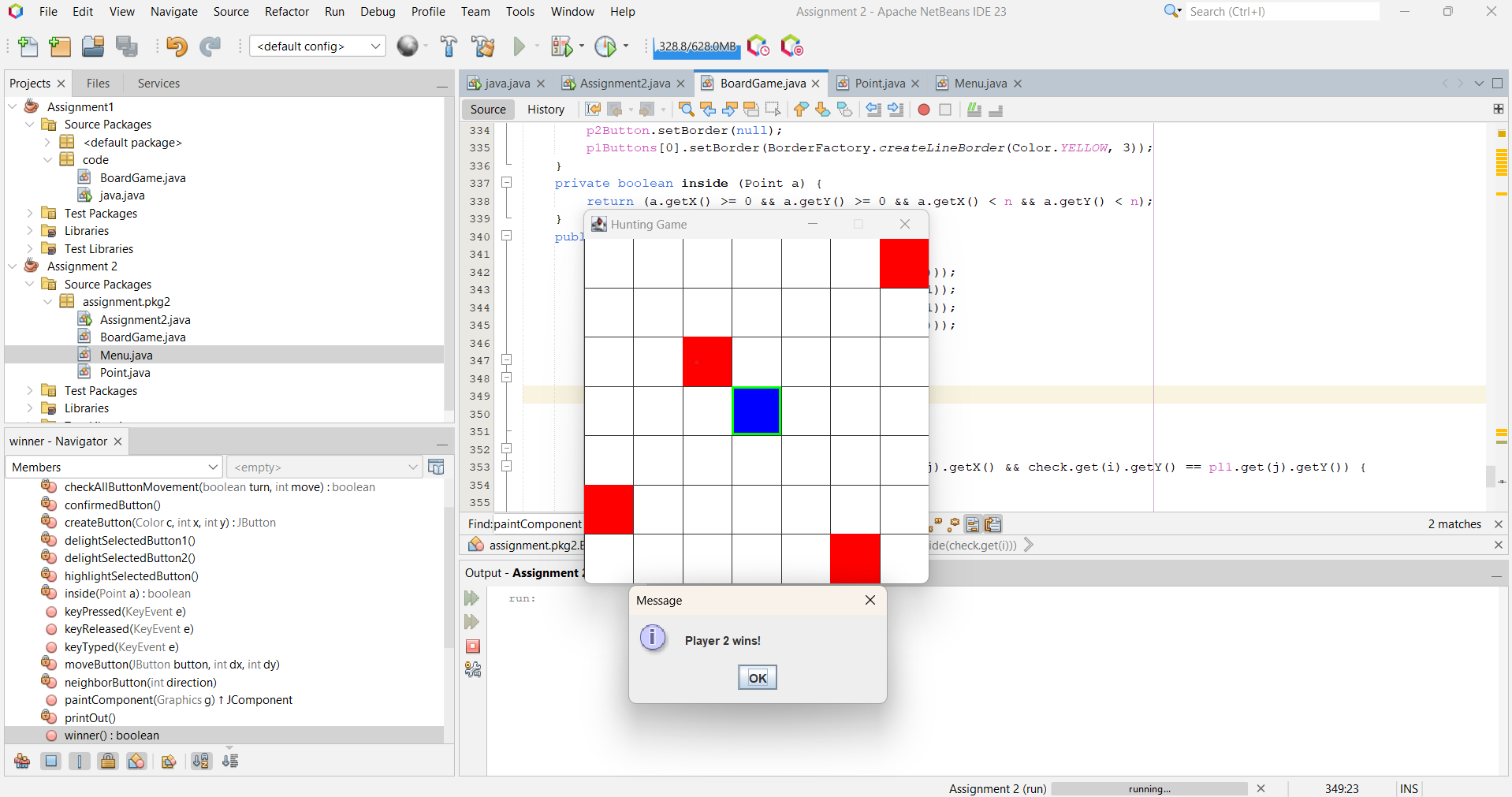








After 28 moves of player 1, he can still not catch player 2:



1. The connections between the events and event handlers:

* keyPressed (e: KeyEvent):

+ e is an event, using as passing argument.

+ The code inside is event handler.

+ The e.getKeyCode() checks which key is pressed and used switch statement to map each key to a specific action.

* 3 buttons startButton, helpButton, exitButton:

+ These buttons’s clicks events are handled by methods described in the code.