Programming Technology

Assignment-2

Documentation for Task-5

BLACK HOLE

Written and implemented by

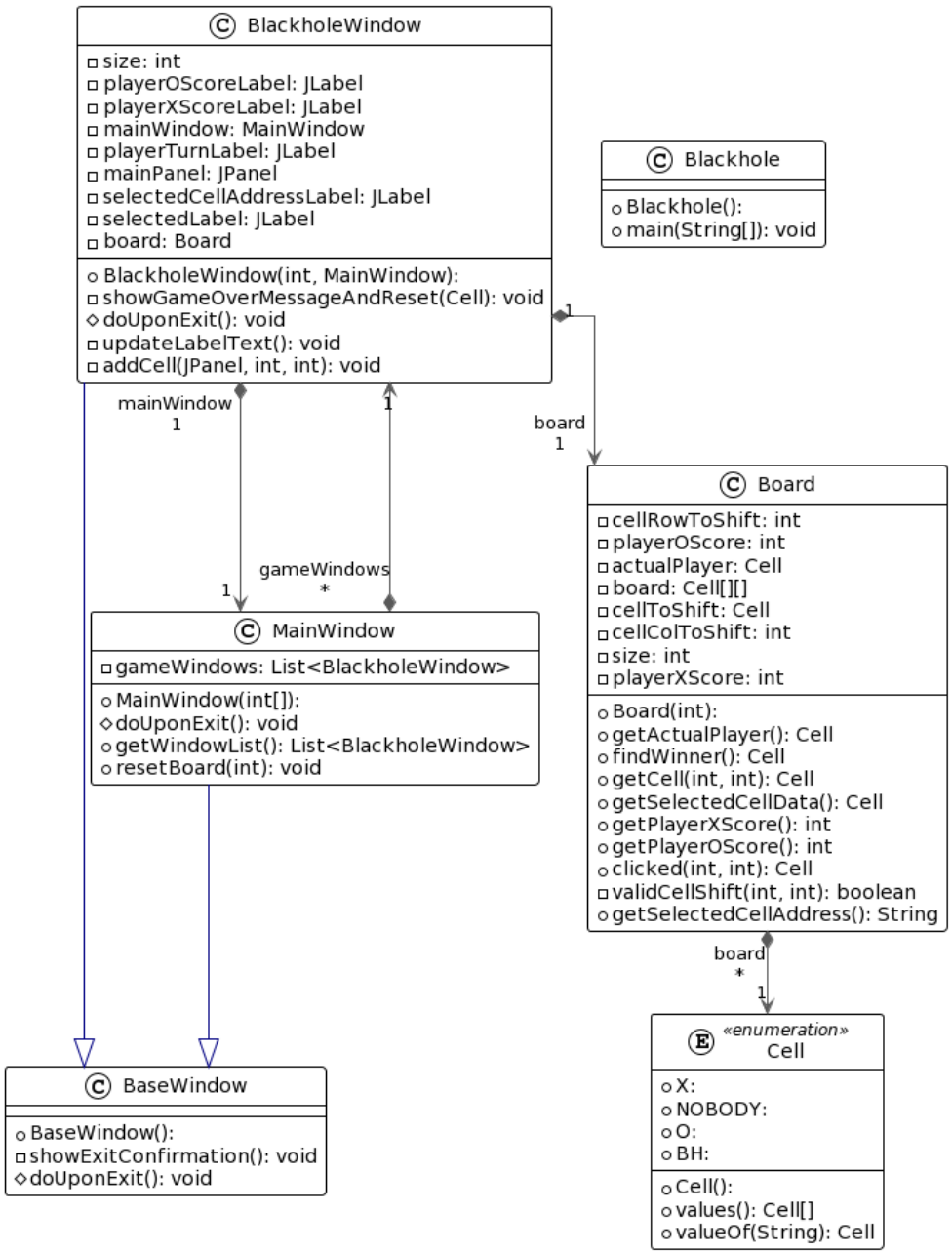
Zayar Htet (DRCVG2)

Task Description

Black hole is a two-player game, played on a board consists of n x n fields, which has a black hole at its center. Each player has n-1 spaceships, which are placed initially in the lower (upper) diagonal of the board (so the same colored spaceships placed on the same side). The players take turns moving one of their own spaceships. The black hole interferes with the navigation system of the spaceships, so they cannot move only one place, but they move until they reach the edge of the board, the black hole, or an other spaceship. A spaceship cannot jump over an other one. A player wins, if he manages to move half of his spaceships into the black hole. Implement this game, and let the board size be selectable (5x5, 7x7, 9x9). The game should recognize if it is ended, and it has to show in a message box which player won. After this, a new game should be started automatically.

Solution

UML Diagram



Classes and Methods

BaseWindow

This class extends JFrame and the parent class of the windows. This will initiate the window dimension and on click exit button feature.

1. **showExitConfirmation() : void** – PRIVATE
   Once the user clicks the exit button, the program terminates and everything has to be shut down.
2. **doUponExit() : void** - PRIVATE
   Will dispose the JFrame and everything.
3. **isInside(p: Point) : Boolean** - PUBLIC ABSTRACT NON-STATIC
   The children of this parent class will implement this method because there are 2 variants of shape, polygon and circular objects, therefore, this will be overridden in child classes.
4. **toString() : String** – PUBLIC ABTRACT
   The toString() method has to be implemented in child classes due to variants of shape.

MainWindow

This class extends the BaseWindow and will create a menu window. There are 5x5, 7x7, 9x9 options in the menu.

1. **resetBoard() : void - PUBLIC**
   Will initialize and reset the board which is creating a new window object.

BlackHoleWindow

Create the board window by adding necessary components and User Interface.

1. **addCell(Integer, Integer) : void - PRIVATE**

   Adding the cell into the board and connects with EventHandler

2. **showGameOverMessageAndReset(Cell) : void - PRIVATE**
   Will show the message dialong indicating that the game is over. Reseting the board after that.
3. **updateLabelText() : void - PRIVATE**
   Updating the labels (player turn, collected spaceship, etc..)

Board

1. **clicked(i,j: int) : Cell** – PUBLIC
   This is the method that will synchronize between the user interface and state model which is a matrix. This is the key main method of the whole project. This will be called once the user clicked the cell.
2. **validCellShift(i,j: int) : boolean** – PRIVATE
   Once the user clicked the cell, this method will check if the clicked cell is valid or not because each cell is allowed to move horizontally and vertically.
3. **findWinner() : Cell** – PUBLIC
   This method will check if there is a winner everytime the user triggered a click.

Cell – Enum of value X, O and black hole (BH), and plain cell (NOBODY)

Events and Event Handler

The whole board is represented with the buttons and the state model is represented with the 2-D array or matrix. There is only 1 actionlistener and eventhandler. The action listener was implemented on each button. Once the user clicked the button/cell, the eventhandler will record the clicked cell data (which player has clicked) in the model and then make necessary restriction for the next click that the user is going to trigger. Moreover, the eventhandler will call a clicked() method from Board, which will make changes to our matrix according to the clicked cell. All the labels are updated and winner is searched in each trigger.