Programming 2 – Assignment 2

Battleship Game

Course name: Programming 2

Course code: COSC2082

Lecturer: Kevin Jackson

Assignment 2

Topic: Battleship Game

Student name: Tran Xuan Truong

Student ID: S3393320

Due date: Aug 31st 2012

Submitted date: Aug 31st 2012

1. The Game

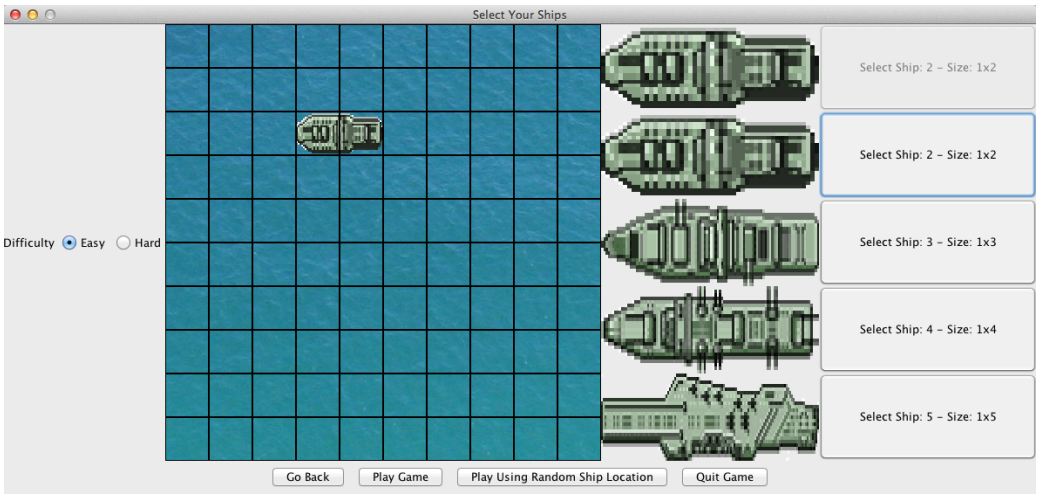
Battleship is a turn-based strategy game for 2 players. In this game, each player has 5 ships placed on a grid. Player cannot see the opponent's ships. The game will end when one of two players has no more ships on his grid.

- **How to play**

Start Game Window:

To play, click "Play Game".
To quit, click "Quit".

Ship Selection Window:



To place ship, click on one of the five buttons on the right to select the ship. After that, click on the grid to place the ship. Repeat for the other 4 ships.

To select difficulty, click on the radio button on the left to select. There are two difficulty levels.

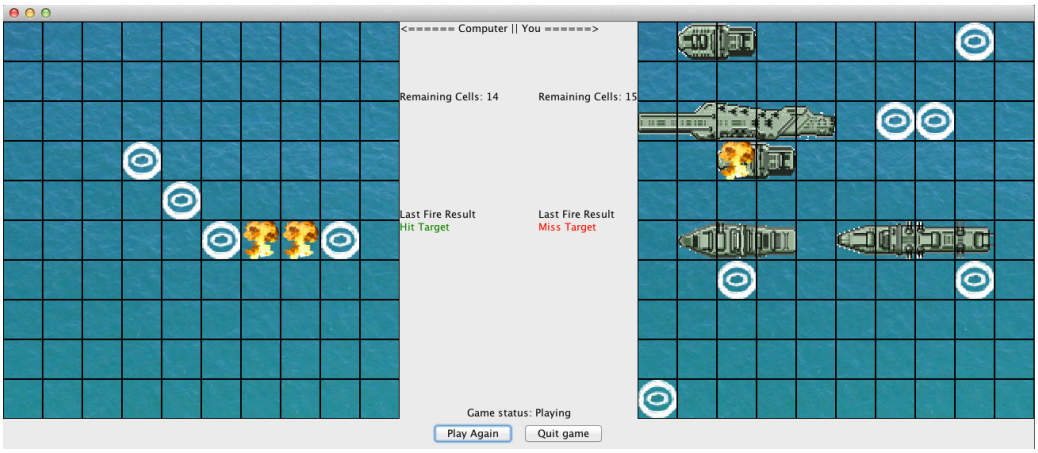
After finishing placing ships, click “Play Game”.

If you do not want to select ship, just want to play game quickly, click “Play Using Random Ship Location”. The computer will generate random location for your ships.

First Turn Dialog:

This dialog demonstrates which player will go first. Click “OK” to play game.

Main game window:



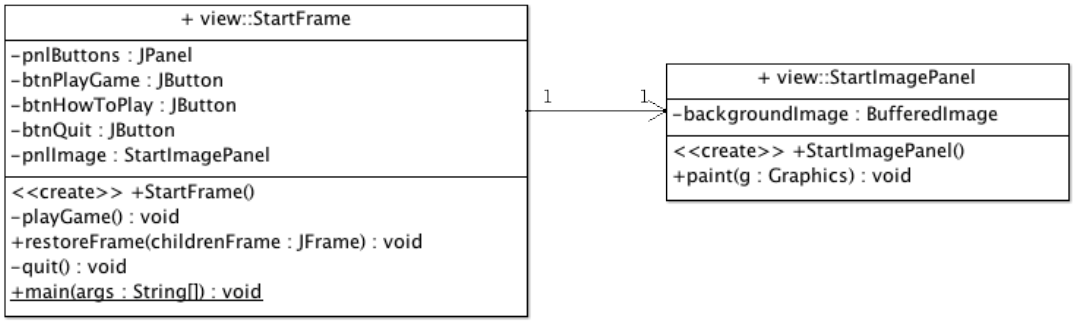
The computer's grid is on the left.
Your grid is on the right.
Click on the cell on the Computer's grid to fire.
The "Remaining Cells" labels indicate the number of cells that contain ship each player has.
The game ends when one of the two has no more ships.
To play again, click "Play Again".
To quit game, click "Quit".

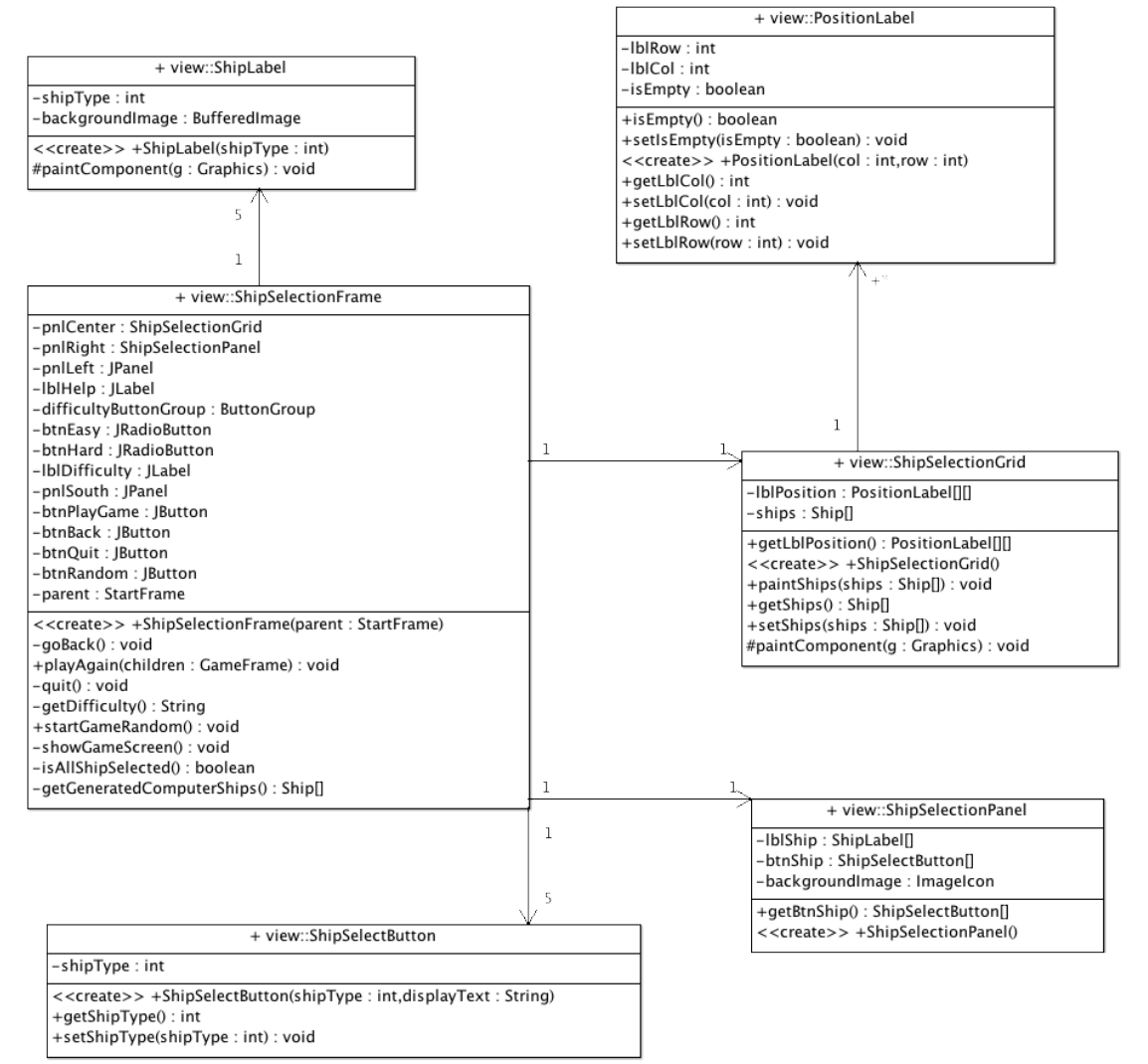
- **Extra feature**

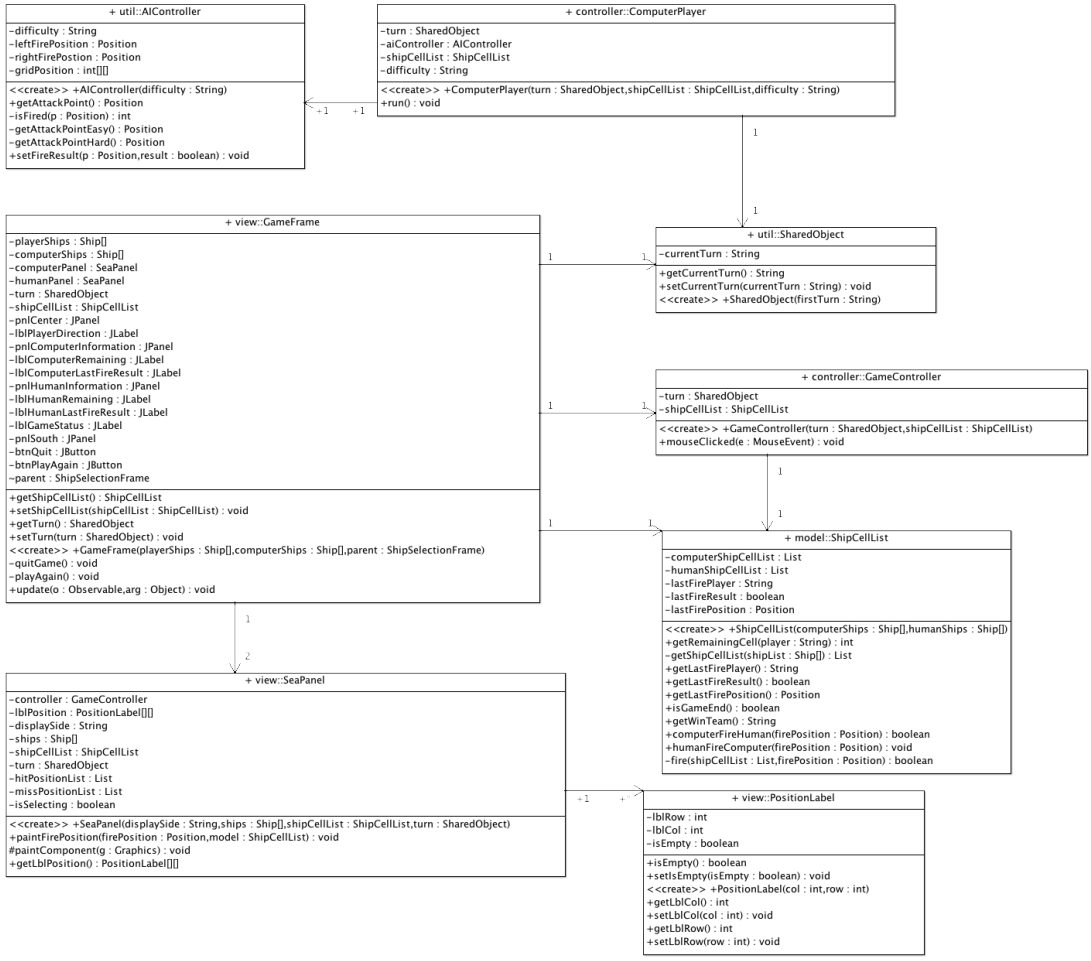
Smart AI:
In "Hard" difficulty level, if the computer hit your ship, it will calculate to fire all cells that contain that ship.

2. Class design

- **Class diagram**




+ view::ShipLabel
-shipType : int
-backgroundImage : BufferedImage
<<create>> +ShipLabel(shipType : int)
#paintComponent(g : Graphics) : void
+ view::PositionLabel
-lblRow : int
-lblCol : int
-isEmpty : boolean
+isEmpty() : boolean
+setIsEmpty(isEmpty : boolean) : void
<<create>> +PositionLabel(col : int,row : int)
+getLblCol() : int
+setLblCol(col : int) : void
+getLblRow() : int
+setLblRow(row : int) : void
+ view::ShipSelectionFrame
-pnlCenter : ShipSelectionGrid
-pnlRight : ShipSelectionPanel
-pnlLeft : JPanel
-lblHelp : JLabel
-difficultyButtonGroup : ButtonGroup
-btnEasy : JRadioButton
-btnHard : JRadioButton
-lblDifficulty : JLabel
-pnlSouth : JPanel
-btnPlayGame : JButton
-btnBack : JButton
-btnQuit : JButton
-btnRandom : JButton
-parent : StartFrame
<<create>> +ShipSelectionFrame(parent : StartFrame)
-goBack() : void
+playAgain(children : GameFrame) : void
-quit() : void
-getDifficulty() : String
+startGameRandom() : void
-showGameScreen() : void
-isAllShipSelected() : boolean
-getGeneratedComputerShips() : Ship[]
+ view::ShipSelectionGrid
-lblPosition : PositionLabel[][]
-ships : Ship[]
+getLblPosition() : PositionLabel[][]
<<create>> +ShipSelectionGrid()
+paintShips(ships : Ship[]) : void
+getShips() : Ship[]
+setShips(ships : Ship[]) : void
#paintComponent(g : Graphics) : void
+ view::ShipSelectionPanel
-lblShip : ShipLabel[]
-btnShip : ShipSelectButton[]
-backgroundImage : ImageIcon
+getBtnShip() : ShipSelectButton[]
<<create>> +ShipSelectionPanel()
+ view::ShipSelectButton
-shipType : int
<<create>> +ShipSelectButton(shipType : int,displayText : String)
+getShipType() : int
+setShipType(shipType : int) : void
5
1
1
1
1
1
1
1
1
5

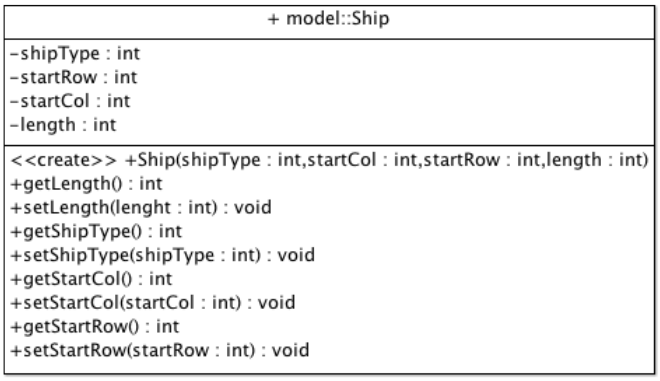

+ util::AIController
-difficulty : String
-leftFirePosition : Position
-rightFirePostion : Position
-gridPosition : int[][]
<<create>> +AIController(difficulty : String)
+getAttackPoint() : Position
-isFired(p : Position) : int
-getAttackPointEasy() : Position
-getAttackPointHard() : Position
+setFireResult(p : Position,result : boolean) : void
+ controller::ComputerPlayer
-turn : SharedObject
-aiController : AIController
-shipCellList : ShipCellList
-difficulty : String
<<create>> +ComputerPlayer(turn : SharedObject,shipCellList : ShipCellList,difficulty : String)
+run() : void
+ view::GameFrame
-playerShips : Ship[]
-computerShips : Ship[]
-computerPanel : SeaPanel
-humanPanel : SeaPanel
-turn : SharedObject
-shipCellList : ShipCellList
-pnlCenter : JPanel
-lblPlayerDirection : JLabel
-pnlComputerInformation : JPanel
-lblComputerRemaining : JLabel
-lblComputerLastFireResult : JLabel
-pnlHumanInformation : JPanel
-lblHumanRemaining : JLabel
-lblHumanLastFireResult : JLabel
-lblGameStatus : JLabel
-pnlSouth : JPanel
-btnQuit : JButton
-btnPlayAgain : JButton
~parent : ShipSelectionFrame
+getShipCellList() : ShipCellList
+setShipCellList(shipCellList : ShipCellList) : void
+getTurn() : SharedObject
+setTurn(turn : SharedObject) : void
<<create>> +GameFrame(playerShips : Ship[],computerShips : Ship[],parent : ShipSelectionFrame)
-quitGame() : void
-playAgain() : void
+update(o : Observable,arg : Object) : void
+ util::SharedObject
-currentTurn : String
+getCurrentTurn() : String
+setCurrentTurn(currentTurn : String) : void
<<create>> +SharedObject(firstTurn : String)
+ controller::GameController
-turn : SharedObject
-shipCellList : ShipCellList
<<create>> +GameController(turn : SharedObject,shipCellList : ShipCellList)
+mouseClicked(e : MouseEvent) : void
+ model::ShipCellList
-computerShipCellList : List
-humanShipCellList : List
-lastFirePlayer : String
-lastFireResult : boolean
-lastFirePosition : Position
<<create>> +ShipCellList(computerShips : Ship[],humanShips : Ship[])
+getRemainingCell(player : String) : int
-getShipCellList(shipList : Ship[]) : List
+getLastFirePlayer() : String
+getLastFireResult() : boolean
+getLastFirePosition() : Position
+isGameEnd() : boolean
+getWinTeam() : String
+computerFireHuman(firePosition : Position) : boolean
+humanFireComputer(firePosition : Position) : void
-fire(shipCellList : List,firePosition : Position) : boolean
+ view::SeaPanel
-controller : GameController
-lblPosition : PositionLabel[][]
-displaySide : String
-ships : Ship[]
-shipCellList : ShipCellList
-turn : SharedObject
-hitPositionList : List
-missPositionList : List
-isSelecting : boolean
<<create>> +SeaPanel(displaySide : String,ships : Ship[],shipCellList : ShipCellList,turn : SharedObject)
+paintFirePosition(firePosition : Position,model : ShipCellList) : void
#paintComponent(g : Graphics) : void
+getLblPosition() : PositionLabel[][]
+ view::PositionLabel
-lblRow : int
-lblCol : int
-isEmpty : boolean
+isEmpty() : boolean
+setIsEmpty(isEmpty : boolean) : void
<<create>> +PositionLabel(col : int,row : int)
+getLblCol() : int
+setLblCol(col : int) : void
+getLblRow() : int
+setLblRow(row : int) : void

- **Class description**

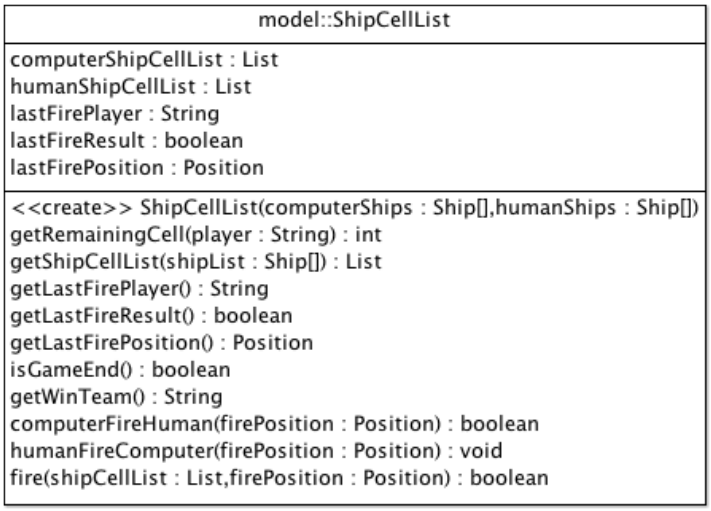
MVC Structure

- model package

Ship: stores the information of the ship (starting row, starting column of the ship and length of the ship)

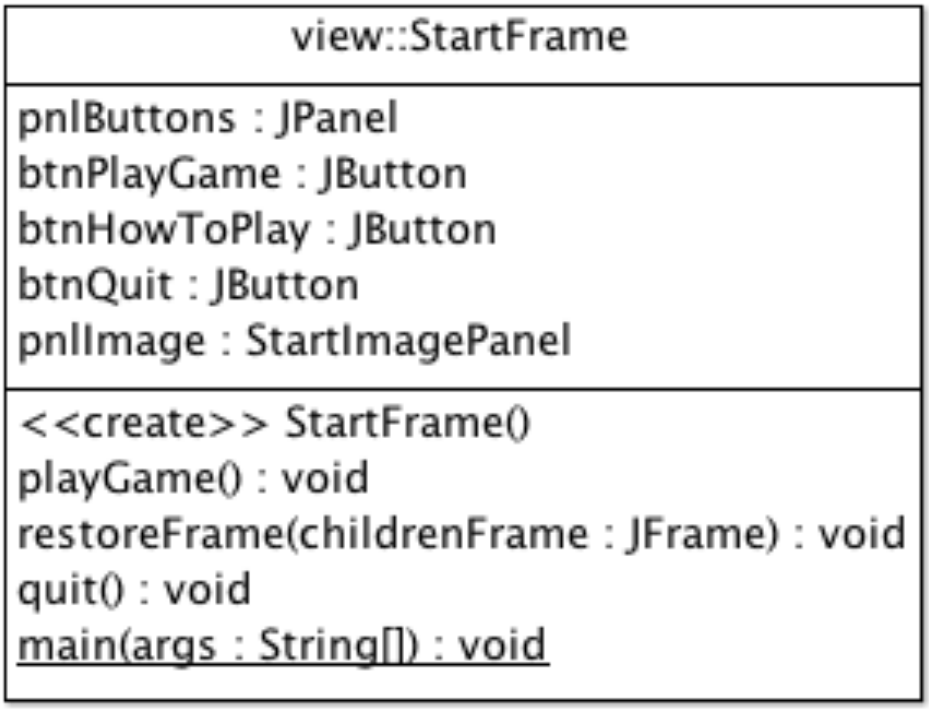


ShipCellList: this is the model for the game. It stores the list of cells that contain ships of both computer and human player. It also has some methods to fire ship for computer and human player. When created, it receive an array of computer and human player's ships (Ship[]) and convert it to the List<Position> list of the cells

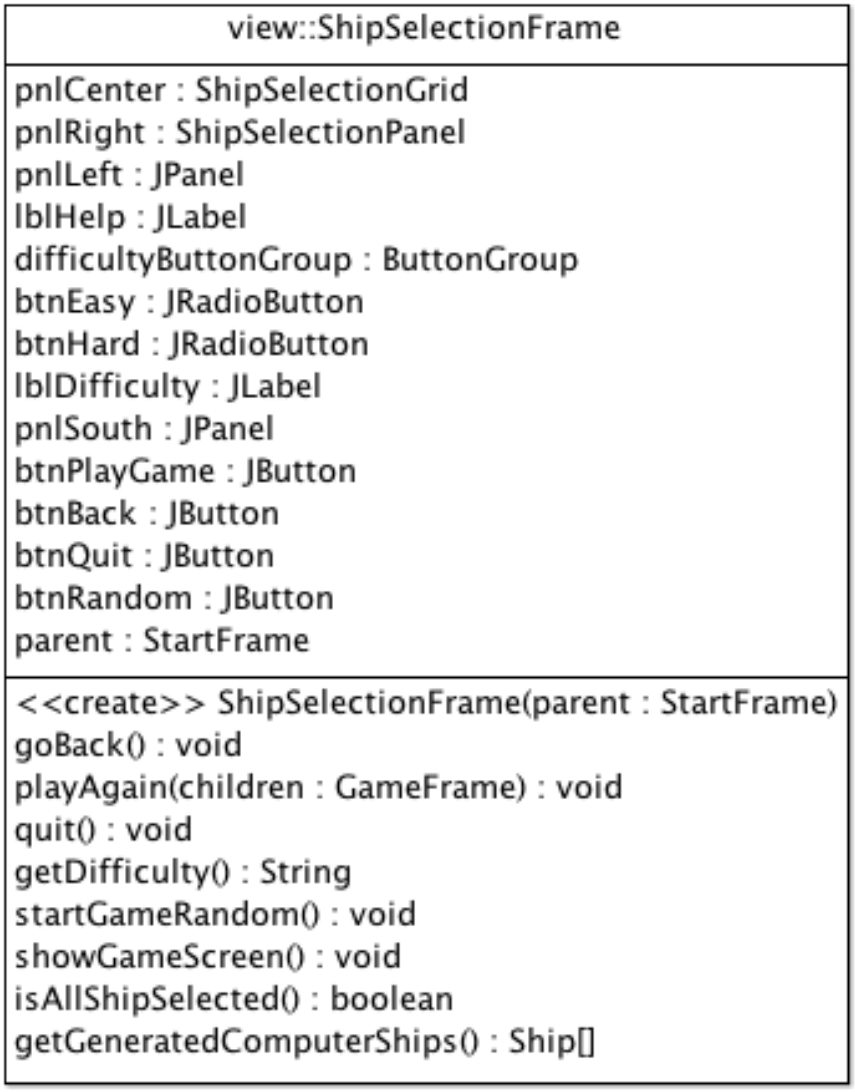


- view package

StartFrame: this class is just to show the start screen of the game



ShipSelectionFrame: this class is used to shows a window for user to choose where to place ther ship on the grid



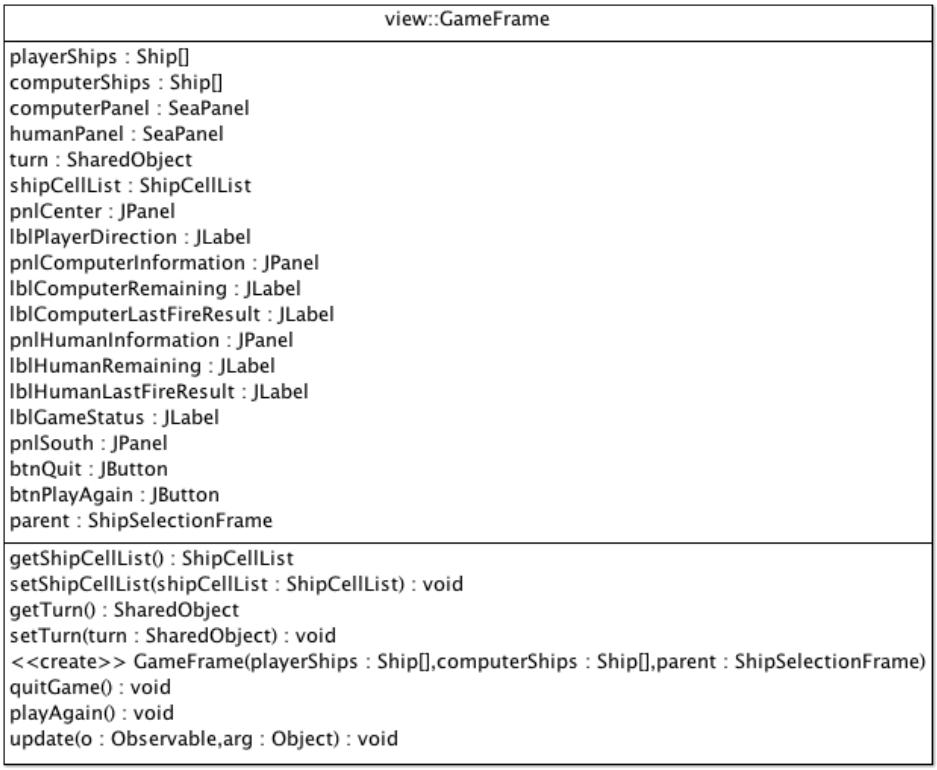
GameFrame: show the main game window.

Variables:

playerShips: Ship[] and computerShips: Ship[]: store the list of ships to paint to the grid and to pass to the model

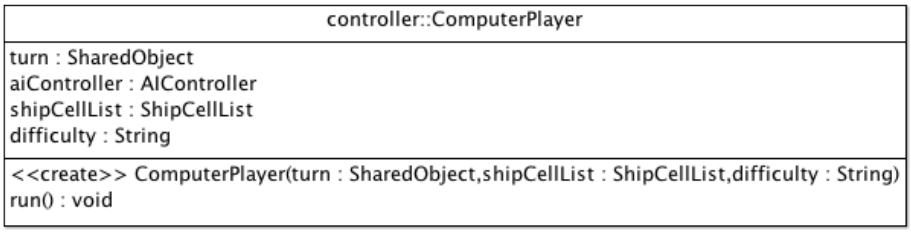
computerPanel: SeaPanel and humanPanel: SeaPanel: display the grid of thw two players

turn: SharedObject: indicates the current turn

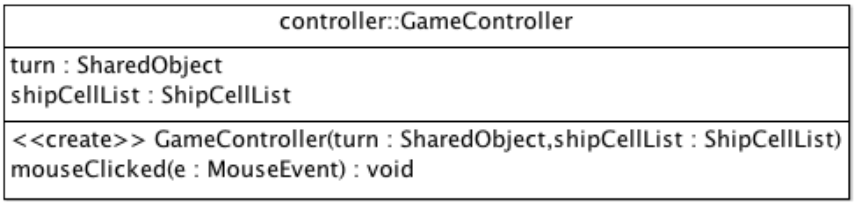


- controller package

ComputerController: a thread act as computer player



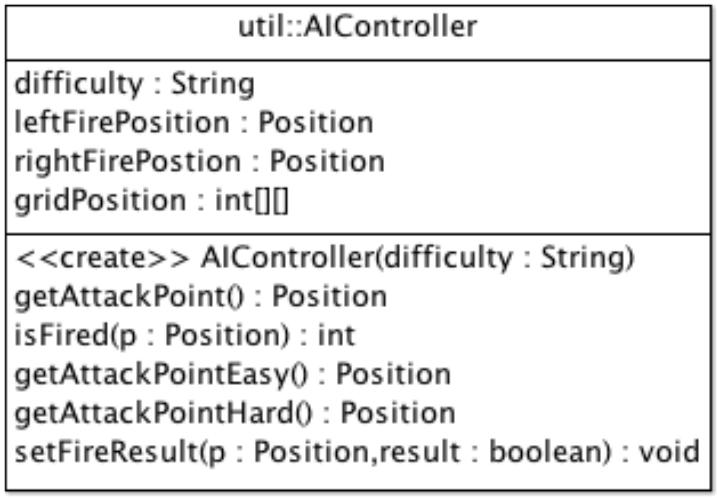
GameController: control the action when users click on the grid to fire



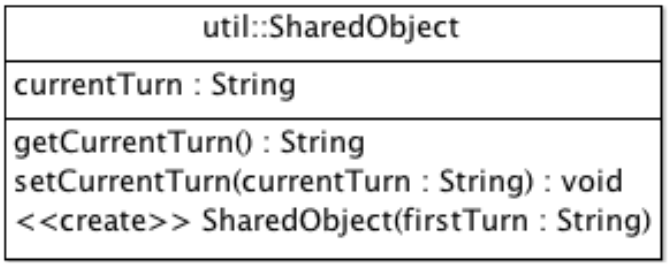
Other classes:

- util package

AIController: controls the AI of computer player.
Variables: difficulty: indicates the difficulty of the game.
Methods: getAttackPoint(): using the difficulty variable, return the corresponding point to fire.



SharedObject: stores the current turn



3. Implementation

MVC Architecture:

View: Game Window (GameFrame class)

This class extends Observer class. When the game starts, it will show the two grid of computer and human player. I used JLabel to display the cells in the grid. After firing, the model will notify this class and call to the update() function. In this function, it will check the remaining cells of both sides as well as the game status (is game playing or stopped) and show to the user.

Controller:

I created one class named GameController that extends MouseAdapter to apply to all those JLabels on the grid. This is really simple, when the user clicks on one cell in the grid, the mouseClicked event will be raised. The Controller just gets the source of the event (the label) and the get some information of the label to know the label's position on the grid. After that, the controller will call the model to fire to that position.

Model: ShipCellList class

This class contains the list of all cells that contain ships of both sides, computer and human player. It also contains the function for each player to fire. After firing, it will notify the view (GameFrame class) so that the view can know whether the fire result is miss or hit and then display it on the frame.

Smart AI:

The game also has a smart AI. At the beginning, the computer picks a random location to fire. When it hits a ship, it will calculate the surrounding cells to find the remaining parts of the ship to destroy it.

Extend capabilities:

The game can easily add more ships because it stores the ships in an array, every time the game runs, it will loop through the array and shows the ships to the user.

Part that can be implemented better

The thread that acts as computer player is programmed not so smart. It uses an endless while loop. It only stops when the game ends. I think I should apply synchronization to make the computer wait until its turn. However, I couldn't do that, I can only apply it to small application but in this game, it's a little difficult for me to do that.

4. References

Images:

Ho Xuan Hieu. SID: s3357671