ALGONQUIN COLLEGE

SCHOOL OF ADVANCED TECHNOLOGY

ICT - Applications & Programming
Computer Engineering Technology – Computing Science

A21

Game MVC (Class Diagram)

Team:
Dylan Boyling - ID: 041042484

Battleship Proposal

Part 1 GUI Definition

1.1. Classes specification

Main

BattleshipGame
properties : properties file
playGame(): loads properties. initializes view, model, and controller

Controller class(es)

Controller
boardState: making calls to board model historyState: making calls to history log playerState: making calls to player state, tracking time or win/loss
validateGuess(x, y): asks model to validate player guess changeDimension(dimension): asks model to change board dimen playGame(): starts timer, starts game changeLanguage(language): changes language of game randomizeBoard(): randomizes ship locations on board placeShip(x, y, isVertical): places ship at x,y coordinates either horizontally or vertically

Figure 1 Driver class and controller classes

View classes

BattleshipView
GridPanel LogPanel OptionsPanel
initializeView(): initializes primary frame as well the other game panels to generate the full game window

LogPanel
HistoryState (used for getting updated history)
initializePanel(): creates panel with history log update(): updates panel when model changes

OptionsPanel
OptionsController (sends updated options to controller)
initializePanel(): creates panel with options or buttons

GridPanel
BattleshipController (used to send data to controller) BoardState (used to get data to display) PlayerState (used to player data, health, name, etc)
initializePanel(): creates panel with grid of buttons for game board update(): updates panel when model changes

Model classes

BoardState
gridDimension: dimension for one side of game board grid: 2D array representing board grid with ship locations ships: list of ships on the board
reset(): clears the grid, sets dimension to default randomizeShipLocations(): creates and places ships randomly based on board dimension resizeGrid(dimension): resizes grid to new dimension getGrid(): returns current state of the board validateGuess(x, y): validates grid guess by checking if ship exists getMaxHitPoints(): gets the max number of hit points for the board getTotalHitPoints(): gets the total number of hit points currently

ShipSquare
isAlive: status of if ship has been guessed or not isPlayer: whether or not it belongs to (human) player size: size of ship, used for displaying on grid (e.g. 4)
setAlive(status): changes status of ship square to alive/not alive

HistoryState (separate or keep with BoardState?)
history: list of events that have occurred
updateHistory(event): adds a new event to the history log

PlayerState
name: name of player wins: number of wins across all matches losses: number of losses across all matches timer: seconds game has been running points: points during current game
addWin(): increments number of wins addLoss(): increments number of losses

Figure 2 View and model classes

1.2. Class diagram

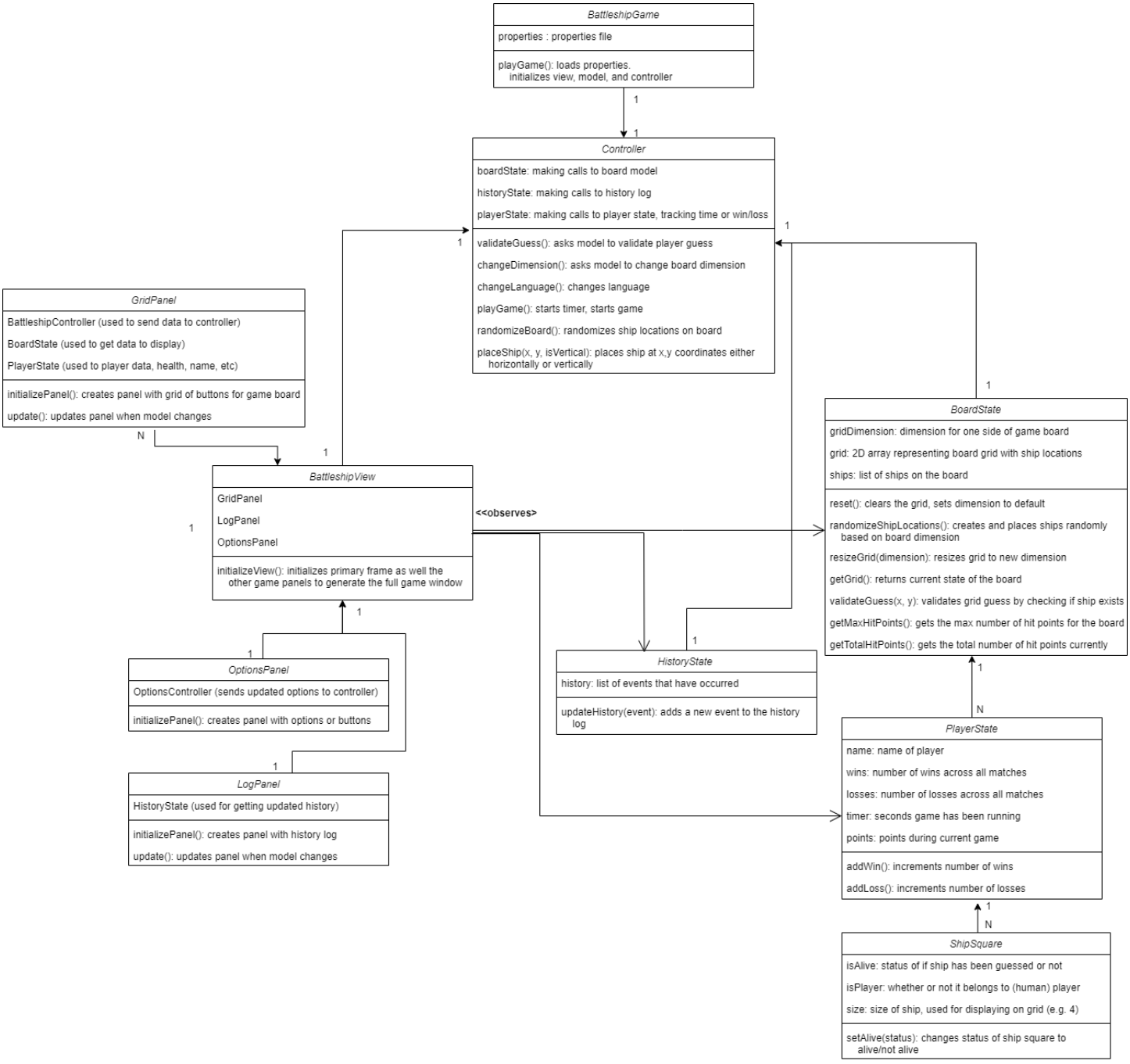

Figure 3 MVC model for Battleship

There is room for improvement in the organization of the MVC model. The view will use the Observer pattern to observe changes in the models, which will be observables. The view is split into four primary panels which include the player panel, the system player panel, the history panel, and the options panel.

Interactions with the view will pass data to the controller which will then be passed to the model. For example, GridPanel is composed of a grid of buttons and clicking one (in play mode) will send the necessary information to validate the ship location's guess such as the x and y coordinates and if it is the player's turn or not. If there is a change in the model, the model's state is set to changed and notifies it's observers so that they may update as needed.

Not all getters and setters were included in the class specification or the MVC model to prevent cluttering but the necessary ones were added. It should be noted that they will be included for the relevant class attributes when they need to be accessed or modified.