# Program Specifications

Samantha Rydzewski, Kaitlin Hoang, and Kyler Kopacz

We essentially started over on our project so almost everything from our previous specifications document has been changed.

Description of how the Classes Interact –

The program will run the main method of the GameInterface Class which will bring up the home screen. The GameInterface class will contain an instance of the Drawer class and call the Drawer's class run method. This method will prompt the game to start and open a new GUI containing the battleship grid. The Drawer class will contain an instance of the Level class and whether the level is an Easy object, Intermediate object, or Hard object (which will be determined by the GameInterface class) and this object will pass the information to the Drawer class. The Drawer class will contain an instance of the GameLogic class and call methods on that instance to run the game. The Drawer class and GameLogic class both contain instances of the Boat class and the Square class.

Boat Class

The Boat class represents the boats in the game. The class keeps track of the location of the boats, the length of them, and if they've been sunk. When the program is running there multiple instances of boats, the number depends of what the player selected as there desired number of boats for their game.

The member variables are boolean sunk, int length, and ArrayList<Point> locations. Sunk is a boolean that represents if the boat has been sunk. Length is an int that represents the length of the boat that has not been hit, for example a length of two means the boat has two squares that have yet to be hit. Locations is an ArrayList of type Point that represents the points the boats cover in their grid. All the member variables are private as no other classes have a need to directly change them.

There is one constructor that takes an ArrayList of Points in as a parameter which represents the location of the boat. There is only one constructor because the only information a Boat object needs to be constructed is its location.

The class has the method checkHit that takes a Point as a parameter. The method checks if the point the player or computer guessed is the same as a point the boat is located at and if so it decreases the length of the boat as it's been hit. The method is public so other classes can call it on a Boat object and the method returns a boolean, true if the boat has been hit and false if it hasn't.

The method isSunk returns the value of the sunk variable so it returns a boolean. The method checkSunk is void and checks if the length is zero and if so sets sunk equal to true. The method getLength is public and returns an int which is the length of the boat. The method getLocations returns the variable locations. These methods are public so other classes can call it on a Boat object.

Updates: We updated the Boat class to work with Point objects and got rid of the setLocation method as it no longer had a function when we reworked our game. We also got rid of the Grid class the Boat class was using as it was easier to just pass in an ArrayList of type Point. We also got rid of the SmallBoat class, MedBoat class, and BigBoat class because all they did was set the length of the Boat which we could do by passing in an ArrayList of type Point.

Square Class
The Square class represents the squares in the grids in the game. The class is used to make a two-dimensional array which represents a grid, either the player's or the opponent's. When the program is running there are many instances of the Square class as they make up the grids.

The member variables are int x, int y, boolean isShot, and boolean containsBoat. X represents the x-coordinate of the Square in the grid and y represents the y-coordinate of the Square in the grid. The boolean isShot keeps track if this Square has been shot, true means it has been shot and false means it hasn't been shot. ContainsBoat is a boolean which is true if a boat is located on that Square and false if the Square is empty. (caps) All the variables are

There is one constructor that takes in two ints and two booleans which are the values of x, y, isShot, and containsBoat. There is only one constructor.

The method getShot returns the value of isShot. The method getX returns the value of x. The method getY returns the value of y. The method setShot takes in a boolean and sets the value of isShot to it. The method setContainsBoat takes in a boolean and sets containsBoat to it. The method getContainsBoat returns the value of containsBoat. The methods are all public so other classes can call them on a Square.

Updates: We added the Square class which essentially replaced the Box class and the GridSquares class. We did this in order to streamline our code and have a class that would make it easy to make a grid and compare spots guessed.

GameLogic Class
The GameLogic class runs the logic of the game, aka generating boats, updating the value of the boats and squares, and taking shots. The purpose of this class is to contain the information that will be used by the Drawer class to display the grids and update them. When the program is running there will be one instance of the GameLogic class.

The member variables are boolean gameOver, int playerTurn, Square[][] humanGrid, Square[][] compGrid, Random rand, ArrayList<Boat> humanBoats, ArrayList<Boat> compBoats, and boolean boatsAccepted. GameOver represents if someone has won the game, true meaning someone has one and false meaning the game is still going. PlayerTurn represents whose turn is it currently, 1 means it's the players turn and 2 means it's the opponent's turn. HumanGrid represents the Squares in the player's grid and compGrid represents the Squares in the opponent's grid. HumanBoats is a list of all the Boats the player has on their grid and compBoats represents all the Boats the opponent has on their grid. BoatsAccepted is

All the variables are public so they are easily accessible by the Drawer class.

There is one constructor that takes no parameters. The constructor initializes the variables and fills the grids with Squares.

The method convertPoints takes in an int x and an int y and returns a Point which represents the location of the x and y in the GUI.

The method generateRandomBoat is void takes in an int, Square[][], and and ArrayList of Boats as a parameters. The int represents the desired size of the boats, Square[][] is the grid the Boats are being places on, and ArrayList<Boat> is either the human's boat list or the opponent's boat list to which the Boats will be added. The method will create a Boat of the desired size and randomly place it on the grid.

The method shoot is void. It takes a Point, a Square[][], and an ArrayList of Boats as parameters. The Point is the place that is being shot, the Square[][] is the grid that's contains the spot that's being shot, and the ArrayList<Boat> is the list of Boats on that grid. The method sets the value of isShot of the Square shot to true and goes through the Boats and updates them if they were shot.

The method switchTurns is void and takes in no parameters. The method changes the value of playerTurn so it is the other player's turn.

The method areYaDeadYet takes in an ArrayList of Boats as a parameter. The method checks if all the Boats are sunk and returns true if they are and false is they aren't.

The method generateAllBoats is void and takes in an int which represents how many Boats it should generate. The method then uses the method generateRandomBoat to populate each grid with the desired number of Boats.

The method startGame

All the method are public so the Drawer class can call them on a GameLogic instance.

Updates: We added this class to our project in order to organize our code better and contain the logic behind the game in one class. Having the values of each Square and Boat on each grid stored in another class also made it easier to update the GUI in the Drawer class.

Drawer Class
The Drawer class is what will display the game on the computer and allow the user to click the keyboard and the mouse and play the game. The class extends JPanel and implements MouseListener and KeyListener. When the game runs there will be one instance of the Drawer class.

The member variables are int WIDTH, int HEIGHT, Level computer,  and GameLogic gl. WIDTH and HEIGHT are final and they store the width and height for the panel. Computer represents which level opponent the player is playing against. Gl represents the GameLogic object that the Drawer class will use. The variables are public so they are easily accessible.

There is one constructor that take no parameters. The constructor sets the size of the panel and initializes the member variables. There is only one constructor because only one is needed for the function of the game.

We implemented MouseListener and only added to MouseClicked. In MouseClicked, the player passes in a Point object, and this Point object is passed into the GameLogic method shoot, along with the computer's Grid and the computer's Boat array. All of these updates are displayed in PaintComponent.

KeyPressed will register any key and switch to display the correct grid. The method will also cause the computer to take its turn. This method is void and doesn't take any parameters.

The method paintComponent takes in a Graphics object as a parameter and is void. The method will draw the appropriate grid, shots, and boats in the GUI. The method will also stop updating and display a GAME OVER screen once the game is over.

The method drawGrid takes a Graphics object as a parameter and is void. The method then draws the basic grid with no boats or hits being shown.
All the methods are public so they can be called easily.

The main method creates the JFrame, displays the GUI, and creates the GameLogic object.

The method DrawBoat takes in a Graphics object and a Boat object as parameters and is void. The method draws the boat.

The method DrawBoats takes in a Graphics object and an ArrayList of Boats as parameters and is void. It loops through the array and draws each boat using the DrawBoat method.

The DrawShots method takes in a Graphics object and a 2D array of Squares as parameters. The method loops through the array and checks if each Square is shot and if each Square contains a boat. If the array contains a shot and a boat, it draws a green shot. If not, it draws a red shot.

Updates: We reworked our entire Drawer class as our previous one wasn't functioning.

Battleship Class
The battleship class only contains one main method. It creates a new instance of the Game Interface class.

Level Class
The Level Class contains one field, an ArrayList of Squares called guesses. The constructor contains no parameters and initializes the Squares on guesses. The method turn takes in no parameters and returns a single Point (0,0).

Easy Class
The Easy Class extends Level. The Easy class has one constructor that takes in no parameters. It overrides the method turn. Its turn method guesses a random point on the grid and returns that Point object.

Intermediate Class
The Intermediate Class extends Level. The Intermediate class has one constructor that takes in an ArrayList of Boats. The class has the member variable rand of type Random and an ArrayList of Boats named activeHits which keeps track of the hits the computer has made that haven't sunk a ship. The method turn overrides the method turn from the Level class and returns a Point which is the computers guess. The method pickGuess returns a Square which will be used by the turn method. PickGuess either randomly picks a Square or make an intelligent guess from its previous hits. The method listContains is overloaded with one taking in an ArrayList of Points and a Point and the other taking in an ArrayList of Squares and a Square. ListContains returns a boolean representing if the list passed in contains an object with the same x and y values. The method removeFromList is void and takes in an ArrayList of Squares and a Square. RemoveFromList removes the Square in the list with the same x and y as the Square passed in. The methods isHorizontal, isVertical, and oneHit are used by the pickGuess method to make an intelligent guess. The three methods return a Square. All the methods are public.

Updates: Made the class work with the Drawer class and made the code cleaner.

Hard Class
The Hard class extends Intermediate. The Hard class has one constructor that takes in an ArrayList of Boats. The method changeGuesses takes in an ArrayList of Boats and modifies the guesses variable by removing 32 Squares from it that do not contain a Boat. This way the computer is a harder opponent to beat as it has a higher change of randomly guessing the location of a boat. The method is public and void.

Updates: Had the Hard class extend the Intermediate class as they shared a lot of code.

GameInterface
This is the class that loads up the main menu so the user can pick the difficulty of the game and launch the game. It extends JFrame, and implements ActionListener. The constructor loads the image that is provided in the images folder, and runs the mainMenu method. Everything in it is public. The mainMenu method is a public method that does not return anything, and it displays all the information of the start screen. It has a lot of Jpanels and a button to start the game, and a JComboBox to choose the difficulty. All of the information that I needed was found here https://docs.oracle.com/javase/tutorial/uiswing/components/index.html I added

ActionCommands to buttons and such, and those actions were sorted to do different things in the actionPerformed method. It starts up the main games and can launch multiple games at once.