Software Engineering Group Project Design Specification

Author:	wgf, deo4, jaj48, jor51
Config Ref:	SE.G17.DS
Date:	2017_04_05
Version:	1.1
Status:	Release

Department of Computer Science

Aberystwyth University

Aberystwyth Ceredigion

SY23 3DB

Contents

1 Introduction

1.1 Purpose of this Document

The purpose of this Document is to show the design of the system for creation of the actual system. It will aid the system by providing a guide to the implementation.

1.2 Scope

This document specifies the design of the system. It will describe the layout and design of the system.

1.3 Objectives

The objective of this document is to aid the implementation of the system. It will provide a guide to implementation the system through the use of sequence diagrams, significant algorithms, and significant data structures.

2 Decomposition Description

2.1 Programs in system

The system that we are designing is all included into one program.

2.2 Significant classes in each program

The CardDeck class takes a stack of cards and shuffles them. We use this for the chance cards and the crew cards.

The ChanceCard class has all the chance card information stored in it. This class calls the ChanceCardHelper which contains the functionality

The ChanceCardHelper class contains all the functionality for the chance cards. It contains all of the methods that work with the effects of the chance cards

The CrewCard class stores all the information about the crew cards. This gets the value, the colour and the image to display for the user.

The TreasureType class stores most of the information about treasures. This includes the treasure name and the value of each treasure.

The Trading class is the graphical interface that shows the display of the trading screen. This screen displays the player, their treasure, the port, the treasure from the port and the accept button to accept the trade.

The GUI classes handles all the graphical interface that shows displays all of the game running. It is split down into their separate classes due to their use.

The GameState class is a class that breaks down the players moves. It will allow the user to do 1 of 5 things with their ship. The first is spin. This allows the ship to rotate without moving. Spin and move is when the ship is at a port and can choose to move diagonally or straight forwards. The next is move and spin. This allows the ship to move and then rotate. The forth is the attacking move. This allows the ship to move and attack a player. The last is trading. This allows the ship to move to a port and trade.

The Position class stores the position of all objects on the board. With the help of the PosistionHelper class it allows ships to move around, interact with objects and not go though islands.

The Score class keeps all the players score. it will also update the score when the player brings treasure back to the port. Once a player returns to the port the Score class will check if the players score is 20 or higher and if it is it will notify the users that the user has just won the game.

The TurnTracker class grabs all 4 players, checks their port and calculates who goes first by starting at London and going anticlockwise around the board.

The FlatIsland class stores treasure and crew cards. When a player is next to the island the player will receive as much treasure as they can hold and the cards on the island.

The PirateIsland class stores all the crew cards passing them to players when they get the chance cards or when they get to Treasure Island.

The TreasureIsland class stores chance and treasure cards. When you land next to this island you can take crew from Pirate Island or some treasure from Treasure Island. If you are unlucky your best pirate will leave your crew and go to Pirate Island.

The Game and GameApp class deals with putting all the classes together and the running of the game. In the Game class it stores the player's usernames and helps the UI with player movement like having arrows and highlighting the squares a player can move. The game app is where all the methods are called to run the game. This is where it navigates though the GUI screens and sets up the game.

The Gameboard class controls all of the locations for the various spaces on the board, the location of the islands and the location of the ports.

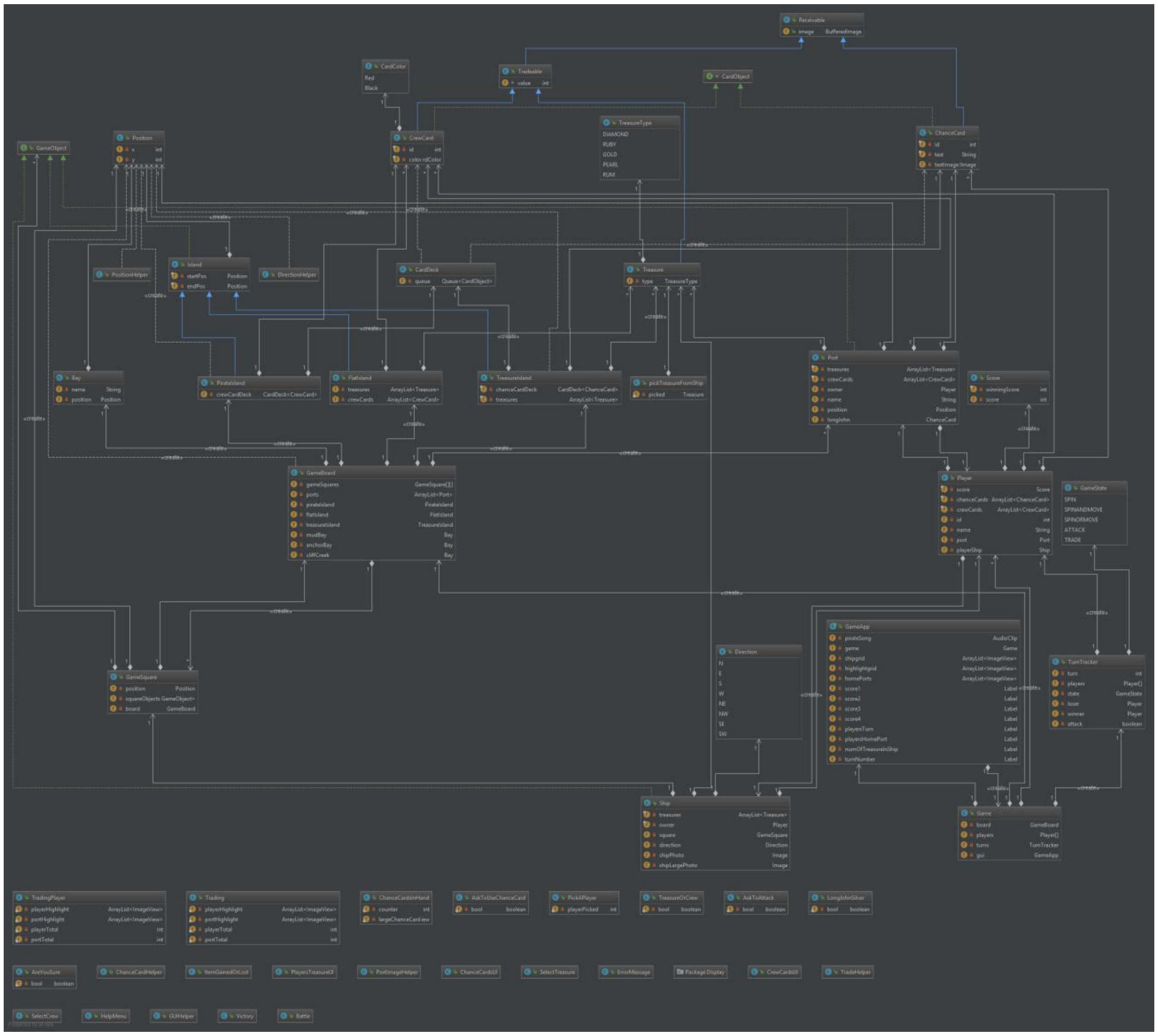
The GameSquare class is the GUI underlay. This stores the locations of the player ships, the islands and the ports and stores them all in one class to make it easier to call upon. This gets most of the x and y coordinates from the GameBoard class.

The Players class is where all the player information will be stored. The player id, the player name, the player score and all the cards they have. This includes crew chance and treasure. This will make it easier to get all the player information from the other classes so the gui can display them with ease.

All the ports will be in one class called Ports. This will store if a player owns the port and what player owns it. It will also store all the crew and treasure stored at that port and the position it is so the game can work out the turn order.

Trading, Tradable, Receivable and ports all handle the trading in the game. The ports contain all the treasure that the player can trade. The player then trades using methods from trading, tradable and Receivable with the ports with equal value or the trade is not valid. The GUI is used throughout the process to show what happens.

2.3 UML Class diagram



2.4 Mapping from requirements to classes

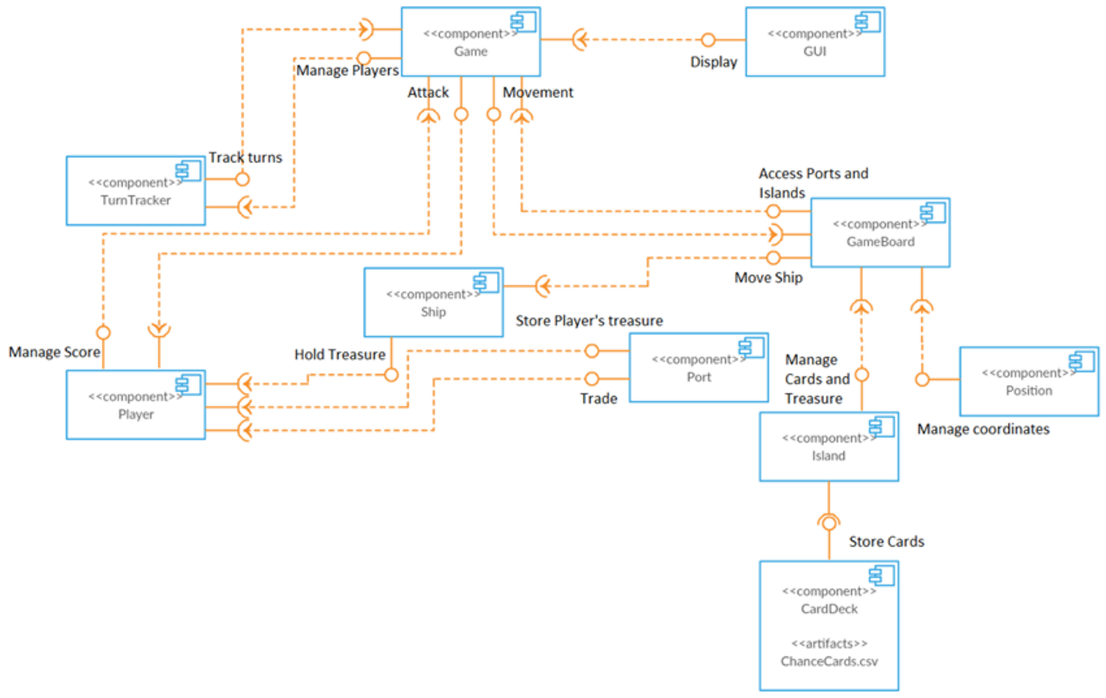
Functional Requirement	Classes providing requirement
FR1 – Player Setup	In the Game App class, we have the game main menu where players can add their usernames to the coloured boxes as prompted by the main menu. Once the players have added the names and clicked the start button the game will then store it and pass it to the Player class where the Player class will pass it around.
FR2 – Port Assignment	In the port class, the Ports will take London and 3 random ports. It will then apply the players to the ports and make the players take their turn in an anticlockwise order starting in London.
FR3 – Crew Card Management	There are 36 crew cards in total, These being worth 1, 2 and 3 in both red and black. In the CrewCard class, we store the card id, the value and the colour which is pulled from the CardColor class. Once we have all the cards we put them into the CardDeck to be shuffled. At the start of the game, the Game Class will deal 5 crew cards to each of the players. as we stored the crew cards as a queue we can use first in last out meaning when a card is placed back into the deck it goes to the bottom of the pile.
FR4 – Chance Card Management	The chance cards have 2 class called ChanceCard and CardDeck class, and a CSV file. There are 28 Chance cards in total in the CSV file. We have got the CSV file and put the cards into a queue so we could use a function called Collections.shuffle((List) queue to shuffle them. When they are drawn we have used switch cases in the ChanceCard class to give them their abilities. Once the card has been used, as we stored the cards as a queue we can use first in last out meaning when a card is placed back into the deck it goes to the bottom of the pile.
FR5 – Treasure Management	The Treasure has its own class called Treasure with getters and setters to the TreasureType class. in TreasureType it contains all the names and values of the treasure putting them into a string. The Treasure class will assign all treasure to start at Treasure Island.
FR6 – Player Management	The Player management is mainly done in the Player Class. In this class it has the player id to help identify the player to the application. It will also have the player name which helps identify the player to the players. The Port class will assign the player a port at the start of the game and will store it in the Player Class. The player card will have 2 array lists; chance cards and crew card. it will contain the ship id to identify to the application who's ship is who's. The player class will also store the player's score. The ship has its own class called Ship and this class can carry 2 pieces of treasure. Once the ship has got to their port the ship will deposit the treasure to the player's score.
FR7 – Port Management	The Port class contains vital information. It stores if the port is player owned and who owns it. The treasure and crew cards stored on that island in an array list. It also

	gets the position of the port from the Position class. The Port class also tells the ship it's starting direction.
FR8 – Flat Island Management	The flat island has a class called FlatIsland that has an array list of treasures and cards. Once a player approaches this island the PositionHelper class will detect that it's docked.
FR9 - Board display	The GameBoard class places all the 400 squares by creating a 20 by 20 grid. It will then display all the ports on the grid giving information like which port is owned or unowned. This class will also get all the ships positions so it can detect when ships are attacking. it will also display all the islands ie treasure, flat and pirate island. The GameApp displays all the highlighted move squares and the rotate buttons.
FR10 – Game Setup	To start of the game will need to take the usernames from the main menu located in the Game class which passes it to Player class (check FR1 for more info). Ports are then set to each player with the Port class which passes it to Player class (check FR2 for more info). Each player is then dealt 5 crew cards (check FR3 for more info). The Treasure Class will then deal out random treasure making sure each player gets a total of 8 points to their port. The ships will then be assigned to their ports. The Port class also tells the ship it's starting direction.
FR11 – Taking turns	The Class TurnTracker deals with assigning players a port making sure London goes first and keeping track of whose turn it is. To do this the TurnTracker class links with GameState. This gives the user the possibility to spin (rotate their ship), move and spin (move their ship then rotate), Move (move without rotating), attacking another ship or trading.
FR12 – Attacking players	When two players are in the same square the PositionHelper class will detect this and make the two ships enter the battle phase with Battle class. This will require both the player's attack power which it calls from the CrewCardUI. It will compare the two and who's ever attack power is higher wins. The losing player gives all treasure to the winner (if there's treasure remaining it goes to treasure island). if the loser has no treasure the losing players will have to give its 2 lowest crew cards to the winner. Once this has been completed the loser will get a free move. If the battle is a draw and both attack powers are the same the attacking ship will get a free move and no treasure is lost.
FR13- Treasure Island	When a player ends their turn on treasure island the TreasureIsland class will run giving the player a Chance card from the top of the deck. this class will deal one chance card from the CardDeck class that will use the ChanceCard switch case to put that card into effect.
FR14 – Flat Island	When a player ends their turn on Flat Island the FlatIsland class will be called upon to award the player a piece of treasure. If the player has room for only one piece of treasure the FlatIsland class will give the player the highest possible treasure it can. Players will also be

	given all cards that are stored on the island. These 2 methods are done with 2 array lists of treasure and crew cards.
FR15 – Arriving at a port	When a player goes to their home port all the treasure will be unloaded and there will be a player win check to see if that player has won (explained more in FR17). When a player arrives at another port they can trade their own treasure and cards. This can only be done in if both values are equal to each other. This is managed by the Trading Class and it gets the values of the Treasure and cards from the Treasure Class linking to the TreasureType class. The crew cards get its values from the CrewCard class linked to the CardDeck class. The trade happens immediately and each affected player gets the appropriate cards.
FR16 – Anchor Bay	Anchor Bay is only for the use of Chance card use of 25 and 26. Once the chance card 25 or 26 has been chosen the switch case will activate. This will store the chance card to the ChanceCardUI which will allow the user to view it in their chance card inventory. Once the player has the chance card and goes to Anchor bay a trade window using the Trading class will pop up and allow the user to get treasure up to the value of 7.
FR17 – Detecting the end game	When a player hits 20 points and returns to the port, They are the winner. We have a Score class that has a WinningScore int that equals 20. Each player has a score and when they are in their own port a Boolean of the player score will run checking that the player's score is 20 or over. If this is true and the score is higher or equal to 20 the screen will change to the Victory screen which will run the Victory class displaying the winner details and will ask the group if they would like to play again.

3 Dependency Description

3.1 Component Diagrams



4 Interface Description

4.1 PositionHelper Interface specification

Type: Public

Public Methods:

static ArrayList<Position> getAvailableMoves(Ship s) – The method calculates the available moves the player can make

static ArrayList<Position> getAvailablePortsMoves(Ship s) – Returns the available moves while in a port

static boolean isPort(Position pos, GameBoard board) – Checks if the current position is a port

static isEdge(int x, int y) – Checks if the position is the edge of the board

isFlatIsland(int x, int y)

isPirateIsland(int x, int y)

isTreasureIsland(int x, int y)

isShip(int x, int y)

static boolean isIsland(Position position) – Checks if the position is part of an island static boolean shouldTurn(Ship ship, Position pos) – Checks if the ship should turn

static boolean moveIsValid(Ship ship, Position pos2) – Checks if the ship move is valid

static boolean moveFromPortIsValid(Ship ship, Position pos2) – Checks if the movement from the port is valid

static boolean moveIsValid(Position pos1, Position pos2) –Checks if the move is not going off the edge of the board or into an island

static int positionToGridID(Position pos) –Is given position returns an ID of the GameSquare in the Grid

static Position gridChange(int x, int y) –Is given x and y coordinates and creates a position object

static boolean isNextTo(Position pos1, Position pos2) –Checks if given positions are next to each other

static boolean isPlusOrMinus(int a, int b) –Used in the isNextTo method to check whether the position is left or right of the position

moveThroughPlayer(ship s, Positon endPos, GameBoard board)

distanceTraveled(Position pos1, Position pos2)

4.2 Game Interface specification

Type: Public

Game(GameApp app) – Constructor of the class objects

getPlayer(int player) – returns the player that is inputted

setPlayer(Player player) – Sets the player in the private players array

createPlayers() – Creates players in the game and assigns them a home port and a ship

dealCards() – Deals cards to the players

assignUsersPort() – Sets the home port of the players

dealTreasure() – adds treasure to the neutral ports up to a value of 7

dealChanceCard() –Deals out the chance cards

nextTurn() –Moves onto the next turn

checkPosition() –Returns the current player's position

oldMoveShip(Ship s, Position pos)

moveShip(Ship s, Position pos) -Moves the ship

onSquareClick(Position pos) – Moves the ship on the click of a highlighted square click or turns it if clicked to the side moveShip(Ship s, Position pos) – moves the ship's position in game and on the board

onUserNameInput(String name1, String name2, String name3, String name4) – Sets the name of the players

onGameBegin() – starts initial start up of the game

addShipsToGUI() – Adds the ships to the board

getCurrentPlayer() –Returns the current player

turnShip(Ship s) –Turns the ship

setInitialGameState() –Sets the game state

getTurnNum() –Returns the turn number

calculateWinner(Player p1, Player p2)

attack(Player winner, Player loser)

playerTreasureToTreasureIsland(Player player)

checkPosition() - Checks the position of the ship

ChanceCard getChanceCard (ArrayList<ChanceCard> cards, int id) - returns ChanceCard with the specified ID from the ArrayList

giveCrewCardsFromAttack(Player recipient, Player giver) - Gives the crew cards from the loser to the winner

setupTradingPorts() – Setups the trading ports

4.3 GameBoard Interface specification

Type: Public

Public Methods:

GameBoard() – Constructor of the class objects

moveShip(Ship ship, Position newPos) – Moves the ship on the board
addSquares() – Creates the squares in the game board

addPorts() – Adds the ports to the board

addBays() –Adds bays to the board

addIslands() –Adds islands to the board

getUnownedPort() –Returns Venice and Amsterdam

ArrayList<Port> getPorts() –Returns the ports getPort(int portID) –Return a port

PirateIsland getPirateIsland() –Returns pirateIsland

FlatIsland getFlatIsland() –Returns FlatIsland

TreasureIsland getTreasureIsland() – Returns TreasureIsland

GameSquare getSquareAt(int x, int y) –Returns the GameSquare at the inputted co-ordinates

getSquareAt(Position pos) –Returns the GameSquare at the position

Bay getMudBay() –Returns Mud Bay

Bay getAnchorBay() – Returns Anchor Bay

Bay getCliffCreek() – Returns Cliff Creek

4.4 TurnTracker Interface specification

Type: Public

Public Methods:

TurnTracker() – Constructor of the class objects

getCurrentTurn() – Returns the current turn

setTurn(int turn) – Sets the current turn

nextTurn() – Sets the current turn to the next turn

getCurrentPlayer() – Returns the current player

addPlayer(Player p) – Adds players into the game and sets the turn order based on the home ports

getState() –Returns the state of the turn

setState() –Set the state of the turn

begin() –Moves onto the next turn

getAttack() – returns if there is an attack

getWinner() - Returns the winner of the battle

setWinner(Player player) – Sets the winner of the battle

getLoser() – Returns the loser of the battle

setLoser(Player loser) Sets the loser of the battle

getPlayerAtIndex(int index) – Returns the player at the inputted index

4.5 Ship Interface specification

Type: Public

Extends GameObject class due to the island being an object in the Game.

Public Methods:

Ship (Player owner) – The ship object constructor

int freeSpace() –Checks if there is free space

addTreasure(Treasure t) –Adds a treasure to the hold

addTreasures(ArrayList<Treasure> t) –Adds 2 treasures to the hold

ArrayList<Treasure> getTreasures() –Returns the 2 treasures from the hold

clearTreasure() –Removes the treasure from the hold

Position getLocation() – returns the position of the ship

setLocation(GameSquare square) – Sets the position of the ship

getSquare() – Returns the gameboard square it is on

setinitalLocation(GameSquare square) – Set the inital location of the ship

Player getOwner() – returns the owner of the ship

setOwner(Player owner) – Set the owner of the ship

Direction getDirection() – Return the direction of the ship

setDirection(Direction direction) – Set the direction of the ship

getShipPhoto() – Return the image that represents the ship

setShipPhoto(String shipPhotoFile) – Set the image the image that represents the ship

removeTreasure(Treasure t) – Removes treasures from the ship hold

ArrayList<Treasure> getTreasures() – Returns treasures from the ship's hold

setDirection(Direction direction) – Sets the direction of the ship

setShipLargePhoto(String shipLargePhotoFile) – Sets the image used for battle, trading etc

getNumOfTreasures() – Returns the number of treasures carried by the ship

calculateValue() - Calculates the total value of the treasures in the hold

4.6 CardDeck Interface specification

Type: Public

Public Methods:

CardDeck() – Constructor of the card deck class

addCard() – adds a card to the deck

setQueue() – Sets the card deck

CardObject removeCard() – Removes a card from the deck

shuffle() –Shuffles the deck

importFromFile() – Imports the cards from a CSV

genCrewCards() –Creates crew cards

getSize() – Returns the size of the card deck

4.7 Player Interface specification

Type: Public

Public Methods:

Player(int id, String name) – Contractor of the class

setPlayerShip(Ship playerShip) – Sets the ship of the player

Port getPort() – returns the port

setPort() – returns the port

getId() – returns the id of the object

setScore(Score score)

Score getScore()

setId() – sets the id of the object

getName – returns the name private variable

setName(String name) – Sets the name private variable

addCrewCards(ArrayList<CrewCard> crewCards) – adds crew cards to the arraylist

ArrayList<CrewCard> getCrewCards() –Returns player’s crew cards

addChanceCards(ArrayList<ChanceCard> chanceCards) – Returns the crew card arraylist

ArrayList<ChanceCard> getChanceCards() –Returns player’s chance cards

int getMoveStrength() –Returns the amount the player can move

getAttackStrength() –Returns the attack strength

Ship getPlayerShip() – Returns the ship

removeCrewCard(CrewCard crewCard) – Removes a crew card from the player's hand

removeSingleCrewCard() – Removes a random crew card

removeChanceCard (int id) – Removes a chance card from the player's hand

ChanceCard getLongJohn () – Returns Long John silver

4.8 Position Interface specification

Type: Public

Public Methods:

Position(int x, int y) – Constructor for the position class

getY() – Returns the y private variable

setY(int y) – Sets the y private variable

set(int x, int y) – Sets both of the y and x private variables

getX() – Returns the x variable

setX(int x) – Set the x variable

boolean isIsland() – Checks if the position is part of an island

String toString() - Displays the value of the x and y

containsShip(GameBoard board)

isEdge() –Checks if position is edge of the board

isPort(GameBoard board) –Checks if position is a port

isNextToOrOnIsland(Island island) –Checks if position is an island or next to one

equals(Object o) – Compares positions to see if there are the same

isNextToOrOnAnyIsland(ArrayList<Island> is) – Checks if is next to or on any island

isBay(Bay b) – Checks if the position is a bay

4.9 Crew Card Interface specification

Type: Public

Extends CardObject due to the fact that the crew and chance cards have some common functionality

Public Methods:

CrewCard(int id, CardColor color, int value) – The constructor for the crew cards

getID() – returns the value of the id private variable

getColor() – returns the value of the color private variable

loadImage() –Loads the image of crew cards

getID() –Returns the id of the card

4.10 Chance Card interface specification

Type: Public

Extends CardObject due to the fact that the crew and chance cards have some common functionality

Public Methods:

getID() – returns the value of the id private variable

getText() - returns the value of the text private variable

executeChanceCard() – executes the effect chance card based on the id of the chance card e.g. chance card with id of 1 executes chance card 1

Image getTextImage() – Loads the text image of the chance card

void loadImage() – Loads the picture image of the chance card

4.11 Score Interface specification

Type: Public

Score – Constructor for the class

Int getScore() – Returns the value of the score private variable

setScore(int score) – Sets the value of the score private variable

addToScore(int score) – Adds the inputted score to the score private variable

hasWon() – check if the player has won

String toString() – Displays the value of the score

4.12 Flat Island Interface specification

Type: Public

Extends Island class due to the fact that there are multiple island class that contain the same functionality

Public Methods:

ArrayList<Treasure> getTreasures() – Returns the Treasures on the island

ArrayList<CrewCard> getCrewCards() – Returns the crew cards on the island

addTreasure(Treasure treasure) – Adds treasure to the island

addCrewCard(CrewCard crewCard) – Adds crew cards to the island's deck

ArrayList<Treasure> getTreasures() – Returns the treasures

ArrayList<Treasure> getAndRemoveTreasure() – Clears the treasure on Flat Island

ArrayList<CrewCard> getCrewCards() – Returns the crew cards on Flat Island

trade(Player currentPlayer) - Handles trading on Flat Island

sortTreasure() – Sorts the treasure on flat island

4.13 Pirate Island Interface specification

Type: Public

Extends Island class due to the fact that there are multiple island class that contain the same functionality

Public Methods:

PirateIsland(Position startPos, Position endPos) – Constructor for the class

CrewCard getTopCard() –Returns the top card from the deck

returnCrewCard(CrewCard card) – Returns a crew card

CardDeck<CrewCard> getCrewCardDeck() – Returns the crew card deck on Pirate Island

4.14 Treasure Island Interface specification

Type: Public

Extends Island class due to the fact that there are multiple island class that contain the same functionality

Public Methods:

ArrayList<Treasure> getTreasures() – Returns the Treasures on the island

addTreasure(Treasure treasure) – Adds a treasure

ChanceCard getTopCard() –Returns the top of the deck getTreasureQty() –Returns the amount of treasure

genTreasures() –Generates the treasures

removeTreasure(Treasure treasure) –Removes the treasure from the island

addChanceCard(ChanceCard card) – Adds chance card to Treasure Island

ArrayList<Treasure> getTreasures() – Returns the treasures on Treasure Island

getTreasureOfType(TreasureType tt) – Returns the type of treasures on Treasure Island

qtyOfValue(int value) – Returns the number of treasure on treasure

treasuresOfValue(int value) - Adds treasure to Treasure Island

4.15 Game Square Interface specification

Type: Public

Public Methods:

GameSquare(int x, int y, GameBoard board) - The class constructor

Position getPosition() – Returns the position private object

Port getPort() – Returns the port private object

setPort() – Sets the port private object

Island getIsland – Returns the Island private object

setIsland – Sets the island private object

remove(Ship ship) – Removes the ship object from the GameSquare

remove(GameObject o) – Removes the GameObject object on that square

GameBoard getBoard() – Returns the board private object

setBoard(GameBoard board) – Sets the board private object

boolean containsShip() –Checks if the gamesquare contains a ship

add(GameObject o) –Add a game object to the GameSquare

GameBoard getBoard() –Returns the GameBoard

setBoard(GameBoard board) –Sets the GameBoard

containsShip() – Checks if the GameSquare contains a ship

4.16 Island Interface specification

Type: Public

Extends GameObject class due to the island being an object in the Game.

Public Methods:
Island(Position startPos, Position endPos) – The constructor for island

Position getStartPos() –The start position of the island

Position getEndPos() –The end position the island

4.17 DirectionHelper Interface specification

Type: Public

Static void highlightTurns(Ship s, GameApp par) – Highlights the turns

Static in directionToAngle(Direction dir) – Returns the angle values of each point on the compass

static Direction positionToDirection(Position pos1, Position pos2) - Takes 2 directions and returns the direction of the second pos from the first

Position getNextPos(Position pos, Direction dir) –Get the next direction

numToDir(int num) – converts the angle number to a direction on the compass

isSameDirection(Position start, Position end, Direction dir) - Returns if the direction is the same

turnIsValid(Ship ship, Direction direction) – Checks if the turn is a valid one

4.18 Treasure Interface Specification

Type – Public

Public Methods:

TreasureType getType() –Returns the type the of treasure

setType(TreasureType type) –Sets the type of the treasure

getFriendlyName() –Returns the name of the treasure

loadImage() – Loads the image of the treasure

4.19 CardObject Interface Specification

Type: Interface

Public Methods:

getID() – Returns the id of the object

4.20 Port Interface Specification

Port(String name, GameSquare s) – Constructor of the class objects

getOwner – Returns the owner of the port

setOwner – Sets the owner of the port

Boolean isOwned() – Checks to see if the port is owned by a player

storeTresure（ArrayList<Treasure> treasures） – The owner stores all their treasure in the port

Position getLocation() – Returns the position of the port

setLocation(Position position)

trade() – Allows the player to trade at a neutral port

String toString() – Displays the name and position of the port

String getName() – Returns the name of the port

getLongJohn() – Returns long john silver

setName(String name) – Sets the name of the port

ArrayList<Treasure> getTreasures() – Returns the treasure in the port

getCrewCards() – Returns the crew cards in the port

getWaterFace() - Calculates which way to point the ship when the game stops

getLocation() - Returns the location of the port

setLocation(Position position) - Sets the location of the port

int getCardValue() – Returns the total CrewCard values

getValue() - Returns the total value of the port

getTreasureValue() - Returns the value of the treasure

4.21 GameApp Interface Specification

Type: Public

Extends Application javafx

start(Stage window) – Starts the gui and game handle(MouseEvent e) –Check if it contains the mouse event

playSound() – Plays the music on a loop

muteSound() – mutes the music

updateTurnNumber() –Upadtes the turn number

updatePlayersTurn() –Update the players turn

updateScores() –Update the scores of the game

setShipDirection(buccaneer.enumData.Direction direction, buccaneer.helpers.Position position) - Changes the direction of the ship in the current location

setShipPosition(Ship ship, buccaneer.helpers.Position position) – Sets the ship's position

moveShip(Ship ship, Position moveTo) – moves the ship from the its current position to the new position

void highlight(ArrayList<buccaneer.helpers.Position> positions) – Displays the highlighted squares where the ship can move

highlightDirection(Position highlightPosition, Direction arrowDirection) –Highlights the direction

dehighlight() – removes the highlighting of the squares

setHomePortOnBoarder(Player player) - sets the home port of the player on the border in the correct colour aligning to the player

updateSideBar() – Updates the sidebar with information from the game

4.22 GameObject Interface Specification

Type: Interface

Public Methods

4.23 DirectionHelper Interface Specification

Type: Public

Public Methods

highlightTurns(Ship s, GameApp par) –Highlights the turn squares on the board

directionToAngle(Direction dir) –Converts north, south, east, west etc.. to angles

Direction numToDir(int num) –Returns north south east west

Direction positionToDirection(Position pos1, Position pos2) –Rotates the ship

isSameDirection(Position start, Position end, Direction dir) –Checks if it is turning to the same direction getNextPos(Position pos, Direction dir) –Get next compass point

turnIsValid(Ship ship, Direction direction) – Checks if the turn is valid

4.24 TreasureType Interface Specification

Type: enum

Public Methods

getValue() –Return the value of the treasure

getName() –Return the name of the treasure

4.25 Battle Interface Specification

Type: public

Public methods:

display(Player player1, Player player2) – displays the battle pop up window

4.26 ChanceCardUI Interface Specification

Type: Public

Public Methods:

display() – displays the Chance card pop up window

4.27 CrewCardUI Interface Specification

Type: Public

Public Methods:

display(buccaneer.main.Player player) – displays the Crew Cards with all the movement and attack values

4.28 ItemGainedOrLost Interface Specification

Type: Public

Public Methods:

display(ArrayList<Receivable> items, boolean gained, String name) – displays the pop up window when you receive crew cards or treasure

4.29 PlayersTreasureUI Interface Specification
Type: Public

Public Methods:

display(buccaneer.main.Player player) – displays the players current treasure in their hold

4.30 Trading Interface Specification
Type: Public

Public Methods:

display(Player player, Port port) – displays the Trading pop up window

addToGrid(ArrayList<Tradeable> tradeablesList, ArrayList<ImageView> highlight, GridPane grid, GridPane highlightGrid)

4.31 SelectTreasure Interface Specification
Type: Public

Public Methods:

display(int maxValueAllowed, int numOfTreasuresAllowed, ArrayList<Treasure> treasures, Ship playerShip) –Displays if there the ship's hold is full

display2(int maxValueAllowed, int numOfTreasuresAllowed, ArrayList<Treasure> treasures, Ship playerShip) – Handles what happens when someone selects treasure

getImage: -Returns the treasure's image

4.32 Victory Interface Specification
Type:Public

Public Methods:

display(Player player) –displays the victory screen

4.33 AreYouSure Interface Specification
Type:Public

Public Methods:

display() – The GUI for the safety net to check if the user really wants to close certain, important windows

4.34 AskToAttack Interface Specification

Type:Public

Public Methods:

display(Player toask, Player moving) –The GUI for when a player moves through another and asks if he wants to attack

4.35 AskToUseChanceCard Interface Specification

Type:Public

Public Methods:

display(ChanceCard chanceCard, String name) –The GUI for when a player is in a port and uses a chance card

4.36 PickAPlayer Interface Specification

Type:Public

Public Methods:

display(Player player , Player[] players) – The GUI for picking another player

4.37 TradingPlayer Interface Specification

Type:Public

Public Methods:

display(Player player, Player port) –The GUI for trading with a player

addToGrid((ArrayList<Tradeable> tradeablesList, ArrayList<ImageView> highlight, GridPane grid, GridPane highlightGrid) - Displays the players tradable treasure in the player tradable grid

4.38 ErrorMessage Interface Specification

Type:Public

Public Methods:

display(String message) –The GUI for when there is an error

4.39 GUIHelper Interface Specification

Type:Public

Public Methods:

Image getImage(TreasureType treasure) – Returns the image for the different types of treasure

Font getPirateFont(int size) – Returns the custom pirate font

4.40 HelpMenu Interface Specification

Type:Public

Public Methods:

display(Player player) –displays the victory screen

4.41 TreasureOrCrew Interface Specification

Type:Public

Public Methods:

display() – The GUI for the help menu

4.42 LongJohnSilver Interface Specification

Type:Public

Public Methods:

display() –The GUI for Long John Silver

4.45 SelectCrew Interface Specification

Type:Public

Public Methods:

display(int maxValueAllowed, ArrayList<CrewCard> crewCards, Player player) – The GUI for selecting CrewCards

4.46 PickTreasureFromShip Interface Specification

Type:Public

Public Methods:

display(Ship ship) – Allows the player to pick a treasure from another player’s ship

4.47 DisplayFlatIsland Interface Specification

Type:Public

Public Methods:

display(Player player) –The GUI for displaying the contents of Flat Island

4.48 DisplayPirateIsland Interface Specification

Type:Public

Public Methods:

display(Player player) – The GUI for displaying the contents of Pirate Island

4.49 DisplayTreasureIsland Interface Specification

Type:Public

Public Methods:

display(Player player) – The GUI for displaying the contents of Treasure Island

4.49 DisplayPort Interface Specification

Type:Public

Public Methods:

display(Player player) – The GUI for displaying the contents of a port

4.50 PortImageHelper Interface Specification

Type:Public

Public Methods:

Image getPortImage(int id) – Returns the different port colour for each player

4.51 Tradable Interface Specification

Type:Public

Public Methods:

getValue() – Returns value of the tradable

setValue() – Sets the value of the tradable

4.52 Receivable Interface Specification

Type:Public

Public Methods:

Image getImage() – Returns the image of the receivable

chanceCard1(Game game) – Handles chance card 1 functionality

chanceCard2(Game game) – Handles chance card 2 functionality

chanceCard3(Game game) – Handles chance card 3 functionality

chanceCard4(Game game) – Handles chance card 4 functionality

chanceCard5(Game game) – Handles chance card 5 functionality

chanceCard6(Game game) – Handles chance card 6 functionality

chanceCard7(Game game) – Handles chance card 7 functionality

chanceCard8(Game game) – Handles chance card 8 functionality

chanceCard9(Game game) – Handles chance card 9 functionality

chanceCard10(Game game) – Handles chance card 10 functionality

chanceCard11(Game game) – Handles chance card 11 functionality

chanceCard12(Game game) – Handles chance card 12 functionality chanceCard13(Game game) – Handles chance card 13 functionality

chanceCard14(Game game) - Handles chance card 14 functionality

chanceCard15(Game game) – Handles chance card 15 functionality

chanceCard16(Game game) – Handles chance card 16 functionality

chanceCard17(Game game) – Handles chance card 17 functionality

chanceCard18(Game game) – Handles chance card 18 functionality

chanceCard19(Game game) – Handles chance card 19 functionality

chanceCard20(Game game) – Handles chance card 20 functionality

chanceCard21(Game game) – Handles chance card 21 functionality

chanceCard22(Game game) – Handles chance card 22 functionality

chanceCard23(Game game) – Handles chance card 23 functionality

chanceCard24(Game game) – Handles chance card 24 functionality

chanceCard25&26(Game game) – Handles chance cards 25 and 26's functionality

chanceCard27(Game game) – Handles chance card 27 functionality

chanceCard28(Game game) - Handles chance card 28 functionality

get4CrewCards(Player player, PirateIsland pirateIsland) – gives 4 crew cards from Pirate Island

gainCrewCards(Game game, int numOfCards) – Gives Crew Cards

ArrayList<CrewCard> loseNumOfCrewCards(buccaneer.main.Player player, int numOfCards) – The player looses crew cards

getNumOfCrewCads(buccaneer.main.Player player) – Returns the number of the player's crew cards

getClosestPlayer(Game game) – Returns the closest player

getOtherPlayersAtTreasureIsland(Game game) – Returns the other players docked at Treasure Island

sendCrewCardToFlatIsland(Game game, buccaneer.cards.CrewCard card) – Sends a crew card to Flat Island

treasureORcrew(Player player) – Returns treasure or crew

reduceCrewCardToValue(int value, Player p, Game g) – Reduces the value of the crew cards to a specific number

takeCrewCards(Game g, int crew) – Takes crew cards from Pirate Island

takeTreasureOrCrew(Game g, int treasure, int crew) – Selects between treasure or crew cards

getLowestCard(ArrayList<CrewCard> cards) – Returns lowest card getHighestCard(ArrayList<CrewCard> cards) – Returns highest card

4.52 Bay Interface Specification

Type: public

Public Methods

getName() – Returns the name

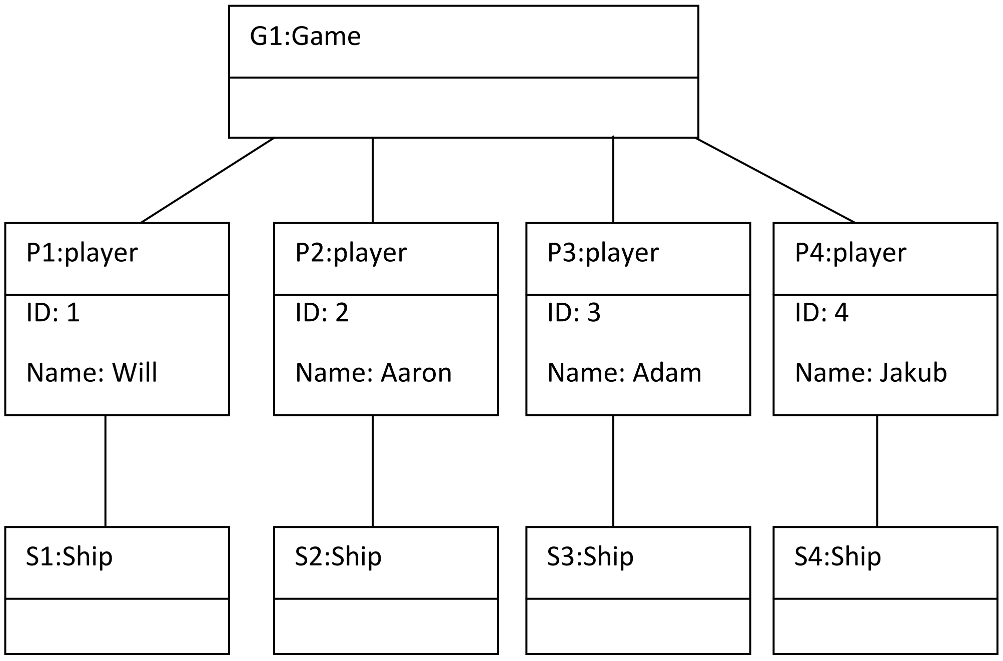
setName(String name) – Sets the name

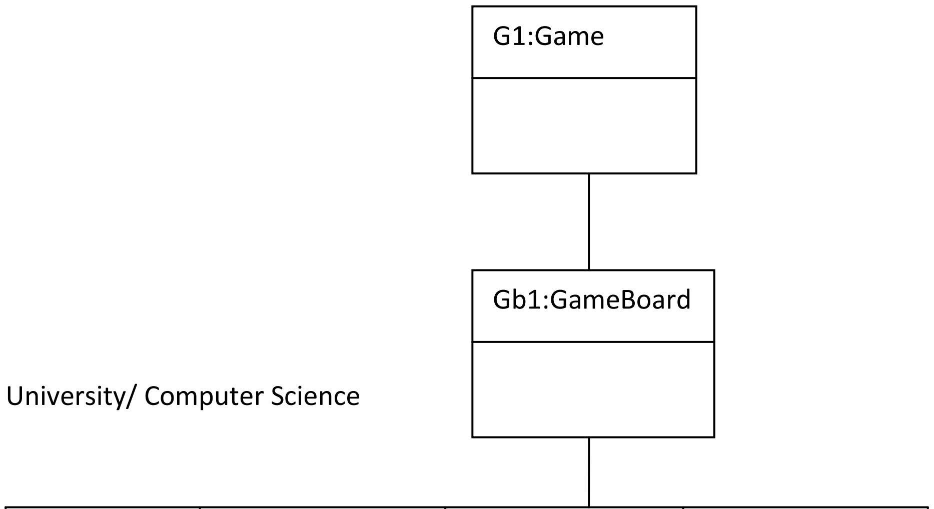
getPosition() – Returns the position

setPosition(Position position) – Sets the position

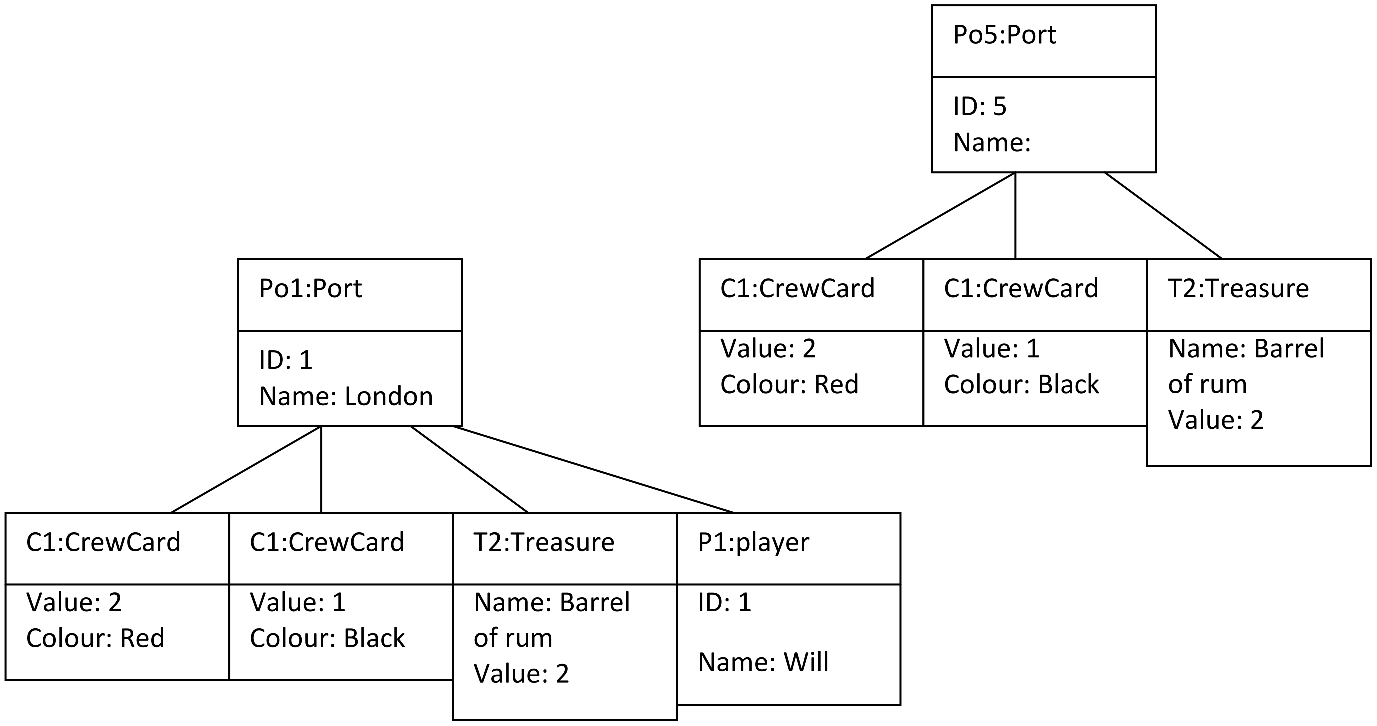
4.50 Object Diagrams

Game and GameBoard

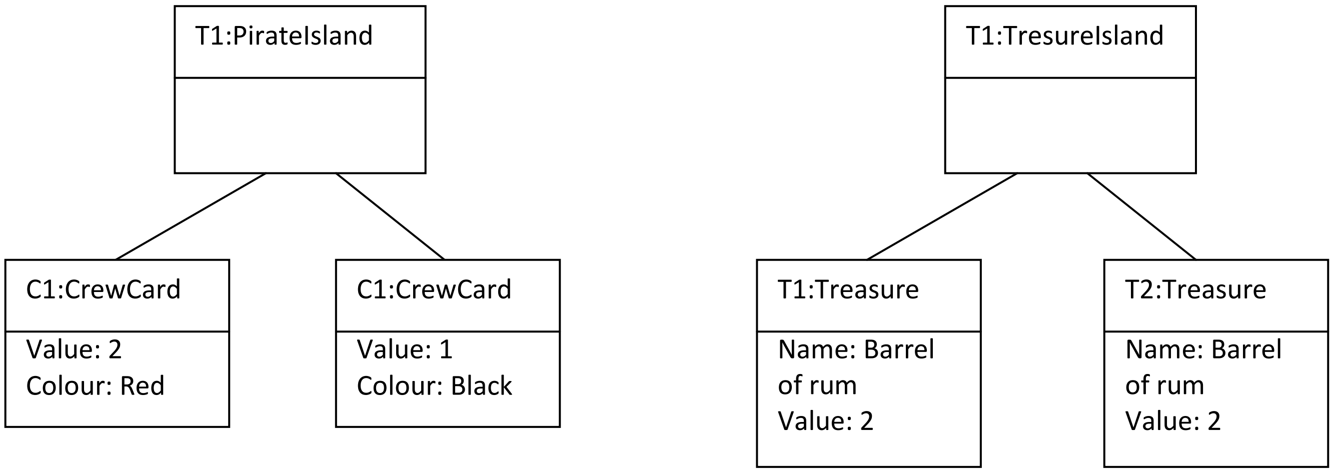


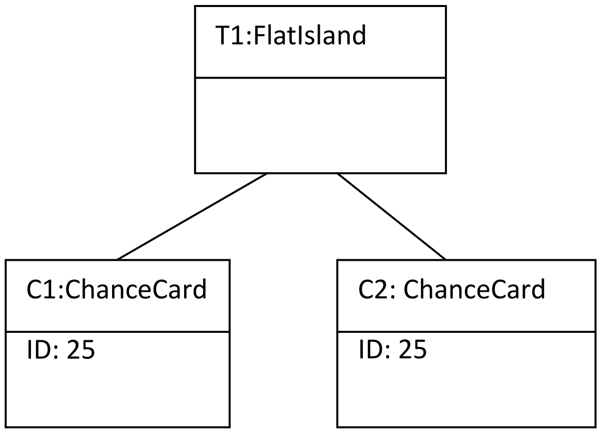


Port with owner and without

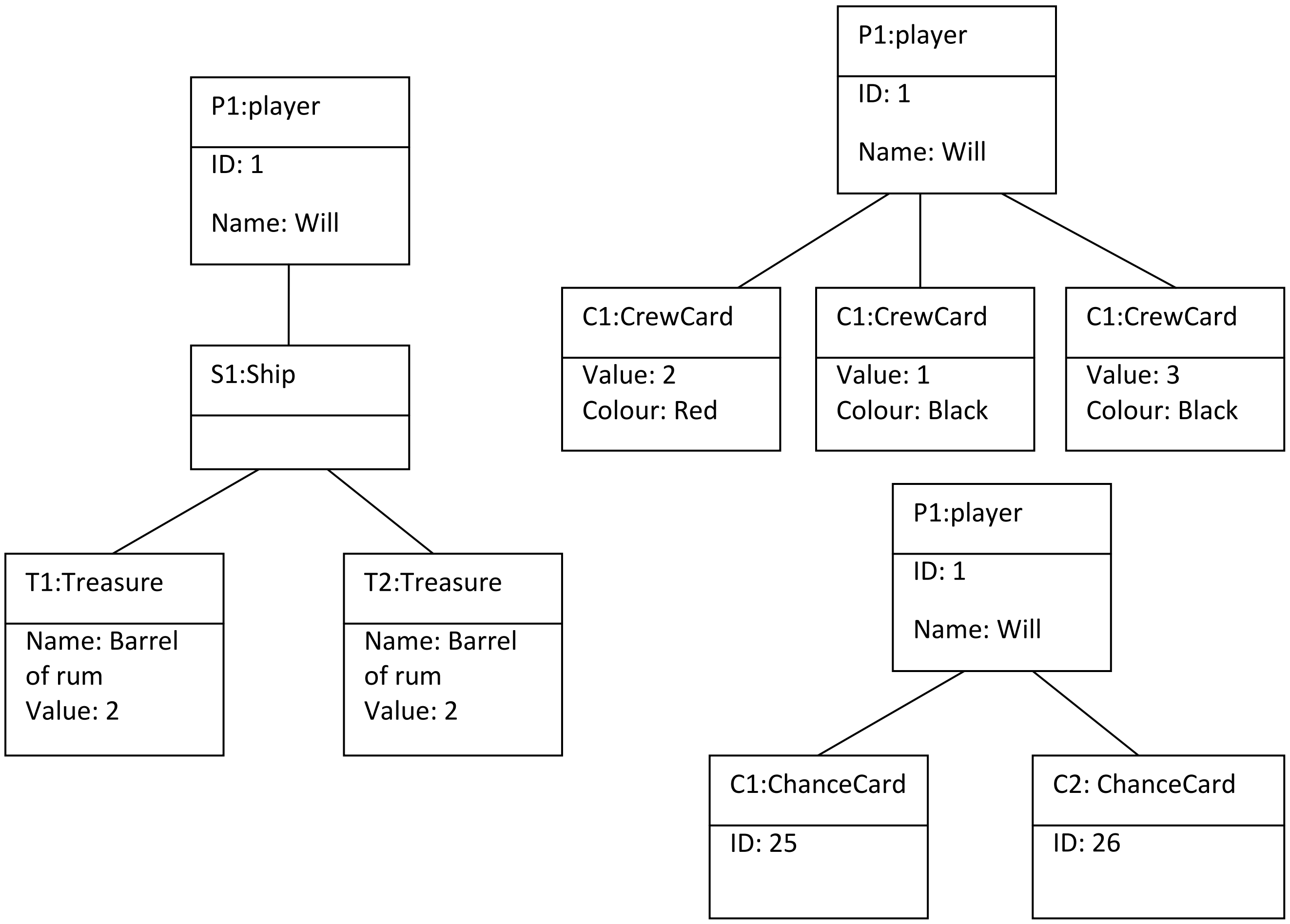


Islands




T1:FlatIsland
C1:ChanceCard
ID: 25
C2: ChanceCard
ID: 25

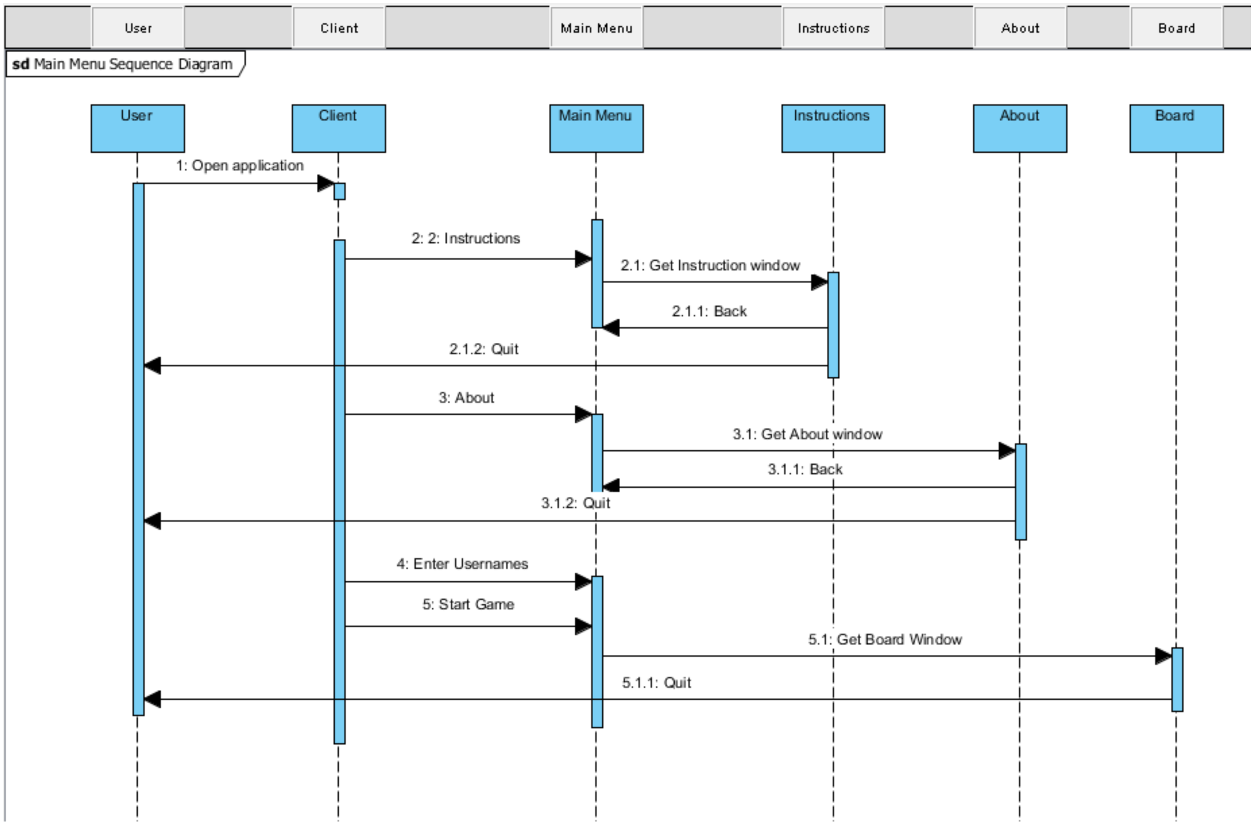
Player


P1:player
ID: 1
Name: Will
S1:Ship
T1:Treasure
Name: Barrel of rum
Value: 2
T2:Treasure
Name: Barrel of rum
Value: 2
P1:player
ID: 1
Name: Will
C1:CrewCard
Value: 2
Colour: Red
C1:CrewCard
Value: 1
Colour: Black
C1:CrewCard
Value: 3
Colour: Black
P1:player
ID: 1
Name: Will
C1:ChanceCard
ID: 25
C2: ChanceCard
ID: 26

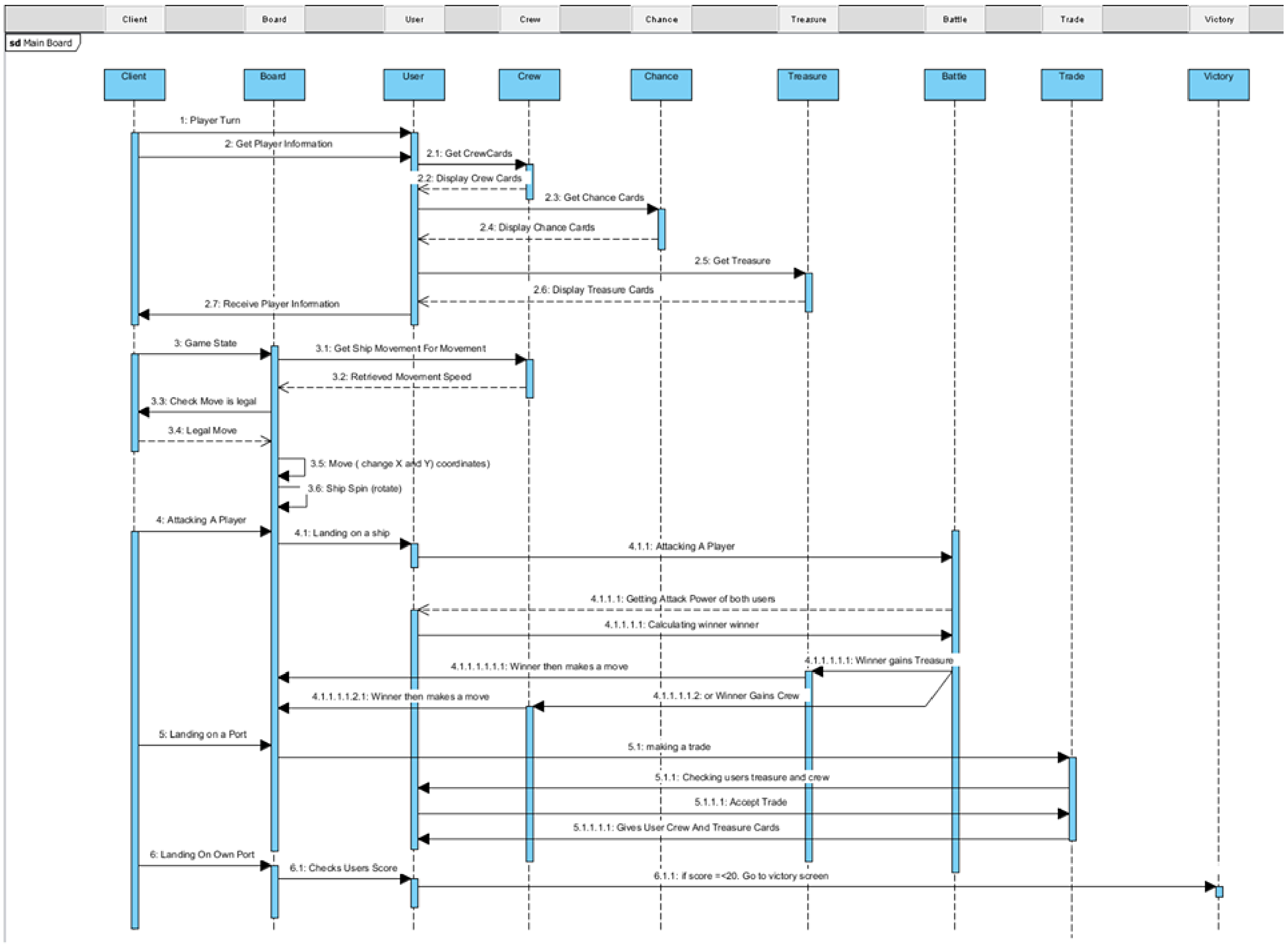
5 Detailed Design

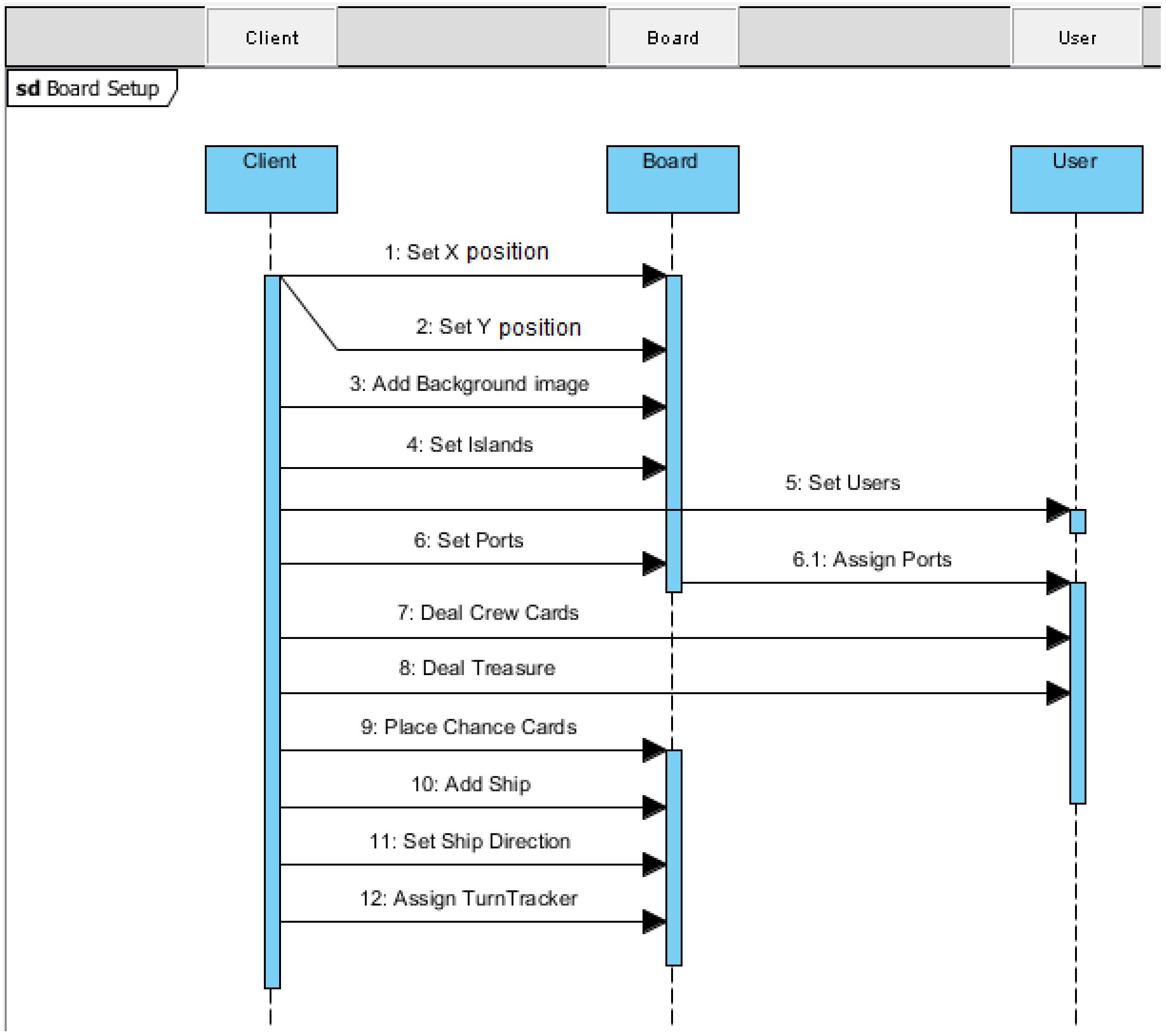
5.1 Sequence diagrams

Main menu



Displaying player information, Attacking and Trading.

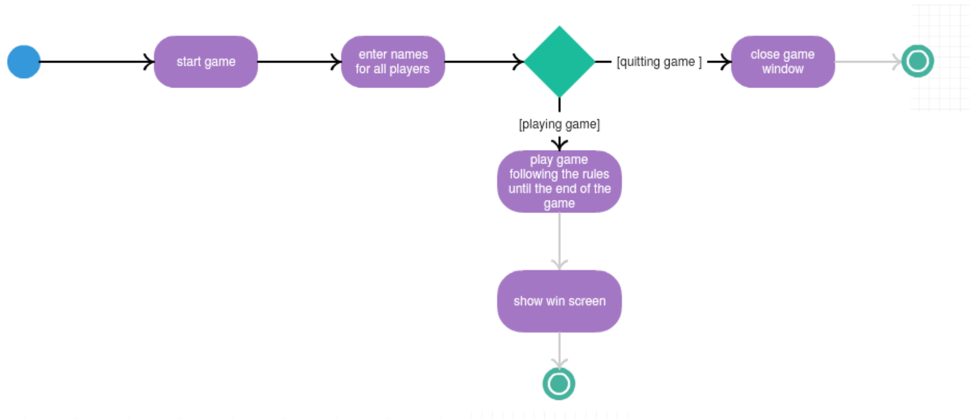

Client
Board
User
Crew
Chance
Treasure
Battle
Trade
Victory
sd Main Board
1: Player Turn
2: Get Player Information
2.1: Get CrewCards
2.2: Display Crew Cards
2.3: Get Chance Cards
2.4: Display Chance Cards
2.5: Get Treasure
2.6: Display Treasure Cards
2.7: Receive Player Information
3: Game State
3.1: Get Ship Movement For Movement
3.2: Retrieved Movement Speed
3.3: Check Move is legal
3.4: Legal Move
3.5: Move (change X and Y) coordinates)
3.6: Ship Spin (rotate)
4: Attacking A Player
4.1: Landing on a ship
4.1.1: Attacking A Player
4.1.1.1: Getting Attack Power of both users
4.1.1.1.1: Calculating winner winner
4.1.1.1.1.1.1: Winner then makes a move
4.1.1.1.1.1: Winner gains Treasure
4.1.1.1.1.2.1: Winner then makes a move
4.1.1.1.1.2: or Winner Gains Crew
5: Landing on a Port
5.1: making a trade
5.1.1: Checking users treasure and crew
5.1.1.1: Accept Trade
5.1.1.1.1: Gives User Crew And Treasure Cards
6: Landing On Own Port
6.1: Checks Users Score
6.1.1: if score =<20. Go to victory screen



Game Setup

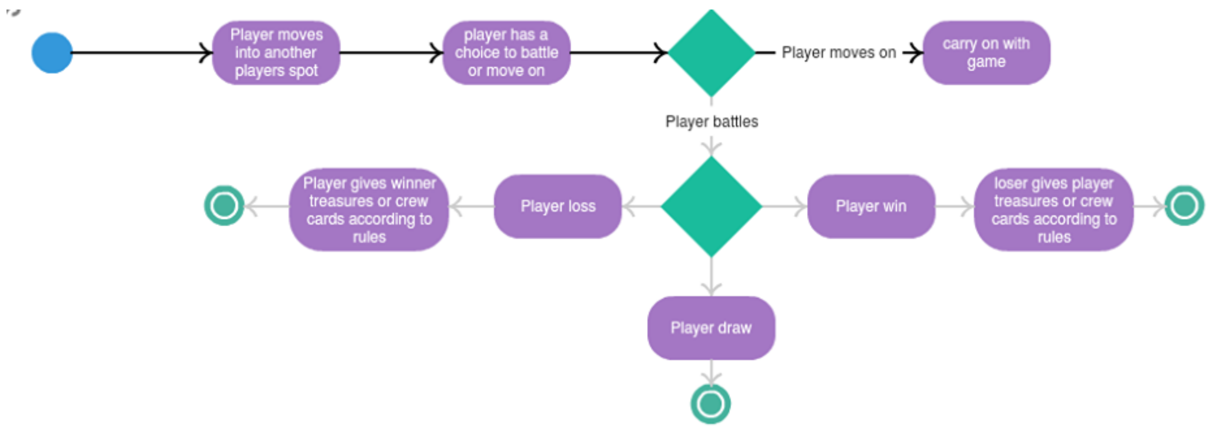
5.2 Flow Diagrams

General Game

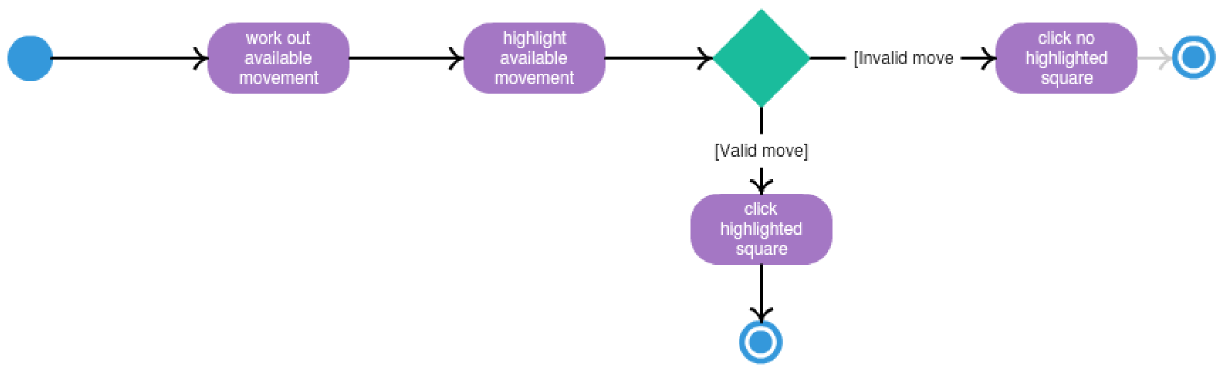


start game
enter names for all players
[quitting game]
close game window
[playing game]
play game following the rules until the end of the game
show win screen

Attacking


Player moves into another players spot
player has a choice to battle or move on
Player moves on
carry on with game
Player battles
Player gives winner treasures or crew cards according to rules
Player loss
Player win
loser gives player treasures or crew cards according to rules
Player draw

Moving


work out available movement
highlight available movement
[Invalid move
click no highlighted square
[Valid move]
click highlighted square

5.3 Significant algorithms

Attacking

The first player who wishes to attack the second player moves to the square that contains another player.

An attack then commences and a pop up is drawn using the AttackGUI class.

The player's crew cards are then added together and if they are more, red crew cards than black then the combined value of the black crew cards are subtracted away from the red crew cards; otherwise the red crew cards are subtracted from the black crew cards.

The values of the players' attack are compared and a winner is decided. If the loser has treasure then the winner is given the choice of the losing player's treasure. If the winner's hold is full then the loser's treasure is returned to treasure island. If the loser has no treasure then the loser gives his 2 lowest crew cards. If the loser only has one crew card then the user gives that card to the other player. If there is a tie then the attacking player makes a legal move. The outcome of the battle is then displayed on the GUI.

The loser can then make a legal move

Moving

The gui draws the highlighted squares where the user can move. This is calculated by the total value of the crew cards that the player has.

The player that wants to move selects the square on the board. The position of the ship is then updated to move the player to the position that is selected

The GUI then updates the ship on the board

Trading

The player that wants to trade moves to the neutral port

On arriving at the port a pop up from the TradingGUI class is drawn. If the user has no treasure or crew cards then he cannot trade. If the user has equal crew cards to the treasure at the port then the GUI displays the option for him to trade. If the user wishes to trade then his crew cards are removed and treasure is added to the hold in their ship.
If the user has treasure he can trade them for crew cards. If the user has equal treasure to the crew cards in the port, then he may trade his treasure for the crew card.

Game Setup

First the Game app class displays a pop up that displays the main menu which contains 4 text boxes for users to input their names.

4 player objects are then created and assigned the name of each of the players. 4 ship objects are created and assigned to each of the players. Then they are assigned a home port at random.

Then the crew cards are assigned to each of the island and ports.
Then the treasure is assigned to each of the islands and ports.

The game board is then displayed showing all of the islands and ports and the user’s ships at each of their respective ports. The player that was assigned to London goes first, and then turns go counterclockwise around the board.

5.4 Significant data structures

Card deck

The Card deck uses a queue as a data structure since a queue acts in the same manner as a deck of cards.

CSV

The Chance card information is adding in using a csv file.

6 Document History

Version	CCF	Changes	Changed by
0.1	N/A	Initial creation	wgf
0.2	N/A	Basic addition	wgf
0.3	N/A	Sequence diagrams and Sections 2 and 3	jor51
0.4	N/A	Component diagram	jaj48
0.5	N/A	Flow Diagrams	deo4
0.6	N/A	Object diagrams and Sections 4 and 5	wgf
1.0	N/A	Checks and changes before release	wgf
1.1	G17-CCF-04	Added component diagram and re-arranged some diagrams	wgf
1.2	G17-CCF-05	Added missing classes for the class interface	wgf