Software Engineering Group 17

The Project Maintenance Manual

|  |  |
| --- | --- |
| Author: | deo4 |
| Config Ref: | SE\_N66\_xxx\_xx |
| Date: | 02/05/2017 |
| Version: | 0.4 |
| Status: | Release |

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Copyright © Aberystwyth University 2017

CONTENTS

CONTENTS 2

1. Program description 3

2. program structure 4

2.1 flow charts 4

2.2 pseudo code 5

2.3 reference to design specification 6

3. Algorithms 6

4. the main data areas 6

5. files 6

6. interfaces 6

7. suggestions for improvements 6

8. things to watch for when making changes 6

9. physical limitations of the program 7

10. rebuilding and testing 7

10.1 Building with ANT 7

10.2 Building as a Java Application 7

11. REFERENCES 8

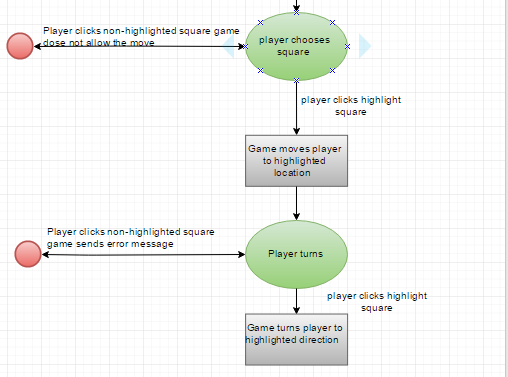
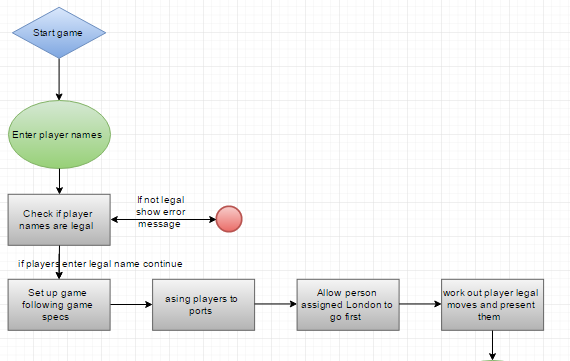
dOCUMENT HISTORY 8

# Program description

This program is adapted from the board game Buccaneer, the program needed to allow to have four people to play the game the same time on the same computer with the computer prompting each player to make moves when appropriate this is handled in the main game function by using an array to store the four players and this allows them to enter their names on the start screen which is a GUI .The program will need to keep track of what the player is holding in terms of cards, treasure and what position on the board position the player is currently located this is done in the player method. The game needs to also only allow a player to have two pieces of treasure on the ship at a time while displaying the state of the game to the players on screen this is done using arrays. The program needs to set up the start position of the game, assigning home ports to players randomly and dealing out crew cards out to players randomly along with applying crew cards and treasure to ports as appropriate. The program then needs to detect a player reaching Treasure Island, Flat Island or the Anchor Bay, then take any appropriate action this is done using a player tracker. When a player choses to take a chance card at Treasure Island, the program will implement the consequences of the chance card selected this is done using a document that stores the chance cards and what they do. The program should be able to manage and exchange of treasure at the ports according to the rules of the game, the game does this by checking where a ship is after its move if it’s in the player’s home port it deposits the treasure, otherwise it shows a trading screen. The game Should be able to indicate legal moves to a player that is about to move from the rules of the game this is done by using the position helper. The game will implement the attacking rule when to ships cross this is done by checking if the squares it's moving through has a player if yes asks them if they want to attack, if the player says yes then it changes the position there moving to then the other player's position, it then moves the player and checks the position they’ve moved to if that has another player if there is then it does an attack if not then the move is done. Finally, the game should detect when a player has won once it has 20 or more points worth of treasure at their home port this is done with the get score class in the player method

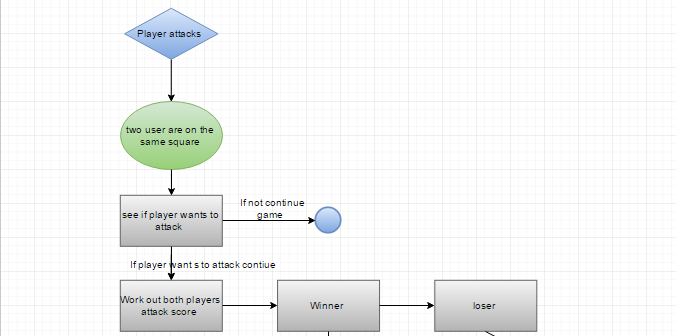
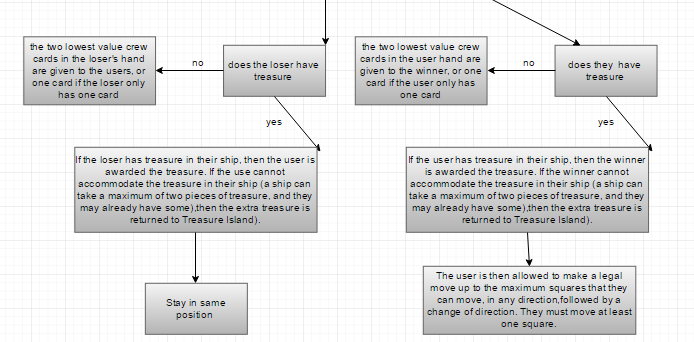
# program structure

## flow charts



The program structure mainly using flow charts, below is a flowchart on moving and turning  that shows the start of the game where the users have to enter their names then the game checks if the names entered by the user are legal if there are not a pop up will come up saying one of the names are not legal, then you enter the names again after legal names are entered the game will start to set up following the specs given it will first assign the players to ports after this it will check what user has been assigned the London port as they are first to move the game will present available moves with showing highlighted squares so if a player clicks a non-highlighted square the game does nothing, but when the user does lick a highlights square it will move to that location on the board. Then the player gets the option to turn and will have the available turns highlighted if they chose a non-highlighted the game presents an error message but when the play clicks a highlighted square they turn in that direction

This is a flowchart shows attacking in the game. So, first two of the users in the game are on the same square the game then prompts the attractor seeing if they want to attack if they don’t the game will continue but if they do the game will then work out both of the players attack score, then it will check if the attacker wins or loses   if the attacker wins the game then checks if the loser has any treasure if they don’t the two lowest value crew cards will be given to the winner, if the loser has only one crew card then only that will be given to the winner, but then if the loser has treasure on the ship then the winner is awarded the treasure provided they have another room on their ship if they cannot the extra treasure is returned to treasure island after this is done the attacker stays in the same position, the loser also goes through this sequence but doing the opposite but after all that is done the user is then allowed to make a legal move up to the maximum squares that are available they must at least move at least one square.



## pseudo code

Inputting user name

Start-up game  
Enter names  
Check if names are legal   
Load up game board

## reference to design specification

In sections:

* 2.2 – 2.4
* 3.1
* 4.1 – 4.33
* 5.1 – 5-2

# Algorithms

The algorithms used in this project can be found in the design specification in sections 5.3 and 5.4 one of the main algorithms used in the game is for attacking, moving, trading and attacking these are explained in the design document in section 5.3.

# the main data areas

One of the main data areas is a svc file that contains all the chance cards that we then can call up on when they are needed.

# files

# interfaces

# suggestions for improvements

Some ideas our group came up with for improvements that are possible to make for this game are to make better junit tests this would have been helpful as it would have made the testing stage much easier, another thing is to create a JAR which is a file that holds all the images, data, code needed for the program and store them all in one file for distribution. One of the other features someone could add to the game would to be implementing 3D graphics, also to have pop up boxes to appear inside of the same widow and have sound play when certain actions happen for example attacking. The main thing we think would be good for the game is allowing it to support different resolutions of monitors.

# things to watch for when making changes

# physical limitations of the program

To play this game you would need at least 1gb of ram, 1gb of storage on your hard drive also you would need …… screen resolution you are also required to have a mouse and keyboard and if you want sound you would need speakers which is optional.

# rebuilding and testing

Building can be carried out in one of two ways:

* Ant Build File
* Standard Java Application Build

## Building with ANT

Note that the program does not officially support building Jar File artifacts due to the way that resources are accessed - Whilst some resources are access via input streams (which can be accessed from within a JAR file), others are not, and are access as external files.

To build the project with the Ant Build File, select build.xml as the ant build file. Normally you can change to the project root directory (not the source directory) and run ant. To build the project, use ant compile.

Ant supports automated running of JUnit tests - to test the program, run ant test.

Ant also supports auto generation of JavaDoc Documentation. To generate this, run ant doc, and open the resulting index.html file within the docs folder.

If you wish to compile, test and generate docs, you can use the main build target to do so. This will also run the dist target, which generates a .jar artifact. However, this is not likely to work

The Ant build system can be integrated with CI systems, and it is recommended to do so while under active development. Configure your CI system to run ant test. Many CI systems (such as travis ci) already do this anyway, so they should not be too much to configure.

## Building as a Java Application

Building as a java application is slightly more involved, but may be quicker; The following steps are for the IntelliJ IDEA IDE, but steps should be similar for other IDE’s.

Download the project, or clone it from github. If the project is in an archive file, then unzip it, and store it in an appropriate directory. Open IDEA, and select Import Project - find the appropriate directory and select it. On the import project screen, select create project from existing sources, and press next. Accept the default Project name and Location, and press next. On the next screen, untick the test directory that is marked “test”, and then click next. Ensure that the lib folder is selected as a Library. Click next, select Buccaneer as a module, and then click on next. If asked to use Buccaneer. iml as a module file, click Reuse. Select java 1.8 as a project SDK (ensure that Java 1.8 JDK is installed and configured with idea), and then click on next, and then finish. Open the Project window, open the src folder and right click on src/buccaneer/main/GameApp.java, and select Run. The application should build, and then run.

To run JUnit test, open the src/test folder, and right click on the test you want to run, and click on test. The test will run, and the status of the test should be displayed.

# REFERENCES

[1] Software Engineering Group Projects: General Documentation Standards. C. J. Price, N. W. Hardy, B.P. Tiddeman. SE.QA.03. 1.8 Release

dOCUMENT HISTORY

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Version* | *CCF No.* | *Date* | *Changes made to document* | *Changed by* |
| 0.1 | N/A | 02/05/2017 | Layout of document and start of program description needs how it does | Deo4 |
| 0.2 | N/A | 04/05/2017 | Program structure and algorithms almost done | Deo4 |
| 0.3 |  | 05/05/2017 | Explained flow charts in the program structure section and explained where to find information on algorithms started main data areas section | Deo4 |
| 0.4 |  | 08/05/2017 | Suggested improvements filled out the rebuilding and testing section | Aaw13 Adl12 |