|  |
| --- |
|  |

|  |
| --- |
| **Technical Report** |
| ICTPRG430 – Apply Introductory Object-Oriented Language Skills AT2 |
|  |
|  |
|  |
| **Luke Wait** |
| **4/9/2022** |
|  |

Table of Contents

[Part 1 – Planning Documentation 2](#_Toc113354849)

[Task 1 – Email 2](#_Toc113354850)

[Task 2 – Report 3](#_Toc113354851)

[Proposed data structures 3](#_Toc113354852)

[Object Oriented Language details 3](#_Toc113354853)

[Object Oriented Techniques 3](#_Toc113354854)

[Internal and External Documentation plan 3](#_Toc113354855)

[Proposed Development stages and description 4](#_Toc113354856)

[Part 2 – Application Development 8](#_Toc113354857)

[Task 3 – Application Development 8](#_Toc113354858)

[Selection, Iteration and Sequence Constructs 8](#_Toc113354859)

[Read/Write 9](#_Toc113354860)

[Classes 10](#_Toc113354861)

[Data Types, Operators, Expressions: 14](#_Toc113354862)

[Task 4 – Development Requirements 16](#_Toc113354863)

[Debugging 16](#_Toc113354864)

[Organisational Standards 17](#_Toc113354865)

[Functionality testing with outcomes 19](#_Toc113354866)

[Part 3 – Hand-Over 20](#_Toc113354867)

[Task 5 – Conclusion 20](#_Toc113354868)

# Part 1 – Planning Documentation

## Task 1 – Email

**EMAIL TO CLIENT**

|  |  |
| --- | --- |
| **To:** | Mark Brown |
| **From:** | Luke Wait |
| **Subject:** | Requirements and Features for Gold Coast E-Sports Desktop Application |
| **Message:** | Greetings Mark,  Following our discussion about the development of a prototype application for the management and recording of competition information for e-sports teams I have drawn up a list of required features:   * Must be able to run on Windows 10 laptop or PC * Must provide the ability to add new entries to existing data – this includes: Add a new competition result for a team (date, location, game, team, points earned) Add a new team (team name, contact person, phone, email, player names) * Must be able to update existing team data (change contact person name, phone, email, or add/remove player names in the team) * Must be able to save entered data to disk when the application is closed * Must be able to read previously entered data in from disk when the application is opened * Must be able to display retrieved data in a list or table * Must be able to read/write from CSV files   In addition, I’ve concluded the application should be GUI-based as this will significantly improve ease of use.  Please advise if I’ve understood the requirements in full. When you are comfortable that all considerations have been addressed, I’ll be happy to proceed with development.  Regards,  Luke Wait |

## Task 2 – Report

### Proposed data structures

Classes:

* Competition
* Team
* GC\_EGames\_GUI

Java dynamic data structures:

* ArrayList<Competition> competitions
* ArrayList<Team> teams
* ArrayList<String> players

### Object Oriented Language details

* Java SE (Standard Edition)

### Object Oriented Techniques

* Abstraction (Competition, Team, and Player classes)
* Encapsulation (public methods accessing private data)
* Inheritance (main class extends javax.swing.JFrame)
* Polymorphism (override methods used to write to file – toString())

### Internal and External Documentation plan

Internal:

* Header comments at the beginning of each source code file with date and version
* Comments at the beginning of each method
* Comments to explain complex algorithms or logic
* Comments to declare noteworthy variables and objects

External:

* Javadocs (class documentation)
* GitHub (version control)

### Proposed Development stages and description

**1. Import external files into package**

* competitions.csv
* teams.csv
* players.csv
* Gold Coast E-Sports image

**2. Setup proposed classes and data structures**

**Competition class**

Instance variables:

* String game
* String competitionDate
* String location
* String team

Methods:

* Constructor (parameterised)
* get/set
* toString (override)

**Team class**

Instance variables:

* + String team
  + String contactPerson
  + String contactPhone
  + String contactEmail
  + ArrayList<String> players

Methods

* Constructor (parameterised)
* Constructor (default)
* get/set
* removePlayers
* toString (override)
* toPlayersString ()

**GUI class**

Instance variables:

* ArrayList<Team> teams
* ArrayList<Competition> competitions

**3. Read in external csv files to ArrayList variables**

Methods:

* readTeamAndPlayerData()
* readCompetitionData()

**4. Write ArrayList data to external files**

Methods:

* saveTeamData()
* savePlayerData()
* saveCompetitionData()

**5. Create jFrame and GUI elements in NetBeans visual editor**

* Display banner for Gold Coast E-Sports image
* Team competition results tab:
  + - table to display competitions data
    - button to display team leader board
* Add new competition result tab:
  + - 4 text fields with to input date, location, game, and points
    - combo box to select team
    - button to write data to file
* Add new team tab:
  + - 4 text fields to input team name, contact person, contact phone, and contact email
    - text area to input player names
    - button to write data to file
* Update an existing team tab:
  + - 3 text fields for contact person, contact phone and contact email
    - text area for player names
    - combo box to select team
    - button to write data to file

**6. Team competition results tab panel**

* Display read-in competition data to table
* create customised jTable using methods and read from teams ArrayList to populate
* Display TEAMS LEADER BOARD pop-up when user clicks Display top teams button
* setup totalPoints instance variable in Team class to hold total points earnt
* when reading competitions ArrayList, also add to totalPoints variable of team instance in teams ArrayList when team name is the same in both lists
* add to relevant teams totalPoints whenever a new competition result is saved
* sort teams ArrayList in descending order according to int value of points variable
* display in jOptionPane

**7. Add new competition result tab panel**

* Setup combobox with team names from teams ArrayList
  + add team name from each team in teams ArrayList to jComboBox using method
* Display SAVE NEW COMPETITION RESULT pop-up when user clicks button
  + use method to validate that all fields are not empty, etc, and if user clicks jButton with invalid fields display error jOptionPane outlining what is missing
  + if all fields are valid display jOptionPane with yes/no option
  + if user selects yes add all fields to competitions ArrayList
  + update competition table after successfully saving
  + if user selects no do nothing

**8. Add new team tab panel**

* Display SAVE NEW TEAM pop-up when user clicks button
  + use method to validate that all fields are not empty, etc, and if user clicks jButton with invalid fields display error jOptionPane outlining what is missing
  + if all fields are valid display jOptionPane with yes/no option
  + if user selects yes add all jTextFields to teams ArrayList
  + set each line of the jTextArea to players ArrayList contained in team instance
  + add new team names to both team comboboxes after successfully saving
  + if user selects no do nothing

**9. Update an existing team tab panel**

* Setup combobox with team names from teams ArrayList
  + add team name from each team in teams ArrayList to jComboBox using method
  + populate fields from teams/players ArrayLists using method
* Display UPDATE AN EXISTING TEAM pop-up when user clicks button
  + use method to validate that all fields are not empty, etc, and if user clicks jButton with invalid fields display error jOptionPane outlining what is missing
  + if all fields are valid display jOptionPane with yes/no option
  + if user selects yes set the fields to the variables of team instance when team name matches jComboBox selection, using set methods
  + remove all entries from players ArrayList contained in team instance using method
  + set each line of the jTextArea to players ArrayList contained in team instance
  + is user selects no do nothing

**10. Save data on application exit**

* Display SAVE BEFORE EXIT pop-up
* when user attempts to close application display a jOptionPane with yes/no option
* if user selects yes write to external files competitions.csv, teams.csv and players.csv using methods and close application
* if user selects no don’t save and close the application

**11. Final requirements review and testing**

* Runs on Windows 10
* Read in external file data when app launches (display error pop-up if any problems reading from file)
* Display competition data in table (update when new comp added)
* Display TEAMS LEADER BOARD pop-up when user clicks button (show total points per team, in descending order)
* Display team names in comboboxes (update when new team added)
* Auto populate fields in Update an existing team panel based on combobox selection
* Add new competition results to competitions ArrayList when user clicks button and chooses yes on resulting pop-up (provided all fields are valid, otherwise display error pop-up detailing what is invalid)
* Add new team to teams/players ArrayLists when user clicks button and chooses yes on resulting pop-up (provided all fields are valid, otherwise display error pop-up detailing what is invalid)
* Update an existing team in teams/players ArrayList when user clicks button and chooses yes on resulting pop-up (provided all fields are valid, otherwise display error pop-up detailing what is invalid
* Save ArrayLists data to competitions.csv, teams.csv and players.csv files when user attempts to close application and chooses yes from resulting pop-up

**12. Complete documentation**

* Planning documentation
* Application development
* Hand-Over

# Part 2 – Application Development

## Task 3 – Application Development

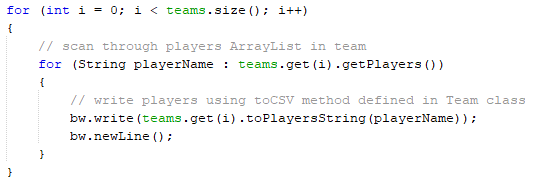
### Selection, Iteration and Sequence Constructs

* Java code demonstrating at least one (1) IF or IF/ELSE construct

Text, letter

Description automatically generated

* Java code demonstrating at least two (2) Iteration or loop constructs



* Java code demonstrating usage of user-created methods

A picture containing text

Description automatically generated

### Read/Write

* Java code displaying what you have used for the program to read from an external CSV file
  + – Include a demonstration of an array used when reading from the file

Graphical user interface, text, application, email

Description automatically generated

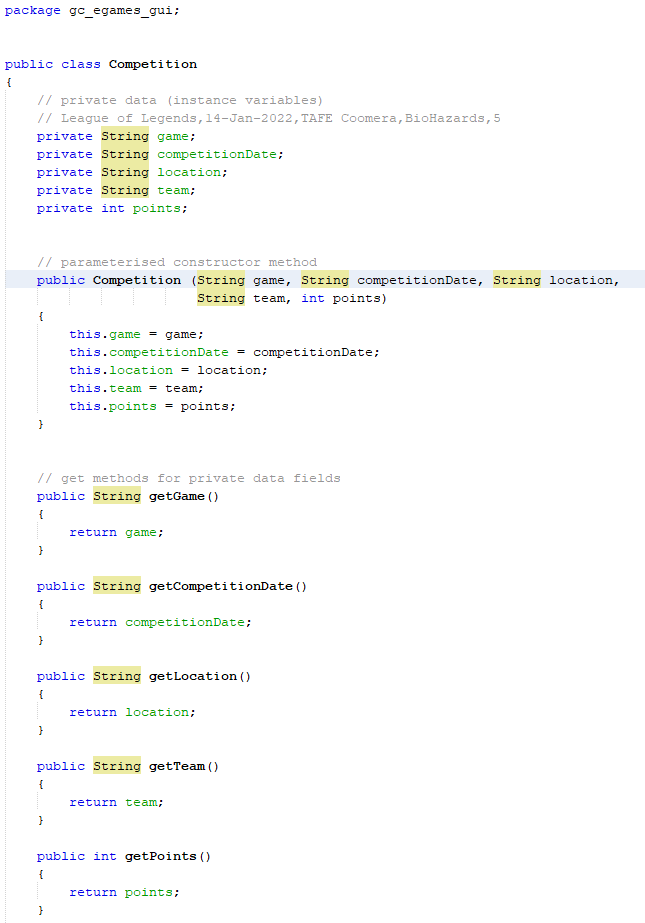
* Java code used to write to the external CSV file
  + – Include a demonstration of an array used when writing to the file

Graphical user interface, text, application, email

Description automatically generated

### Classes

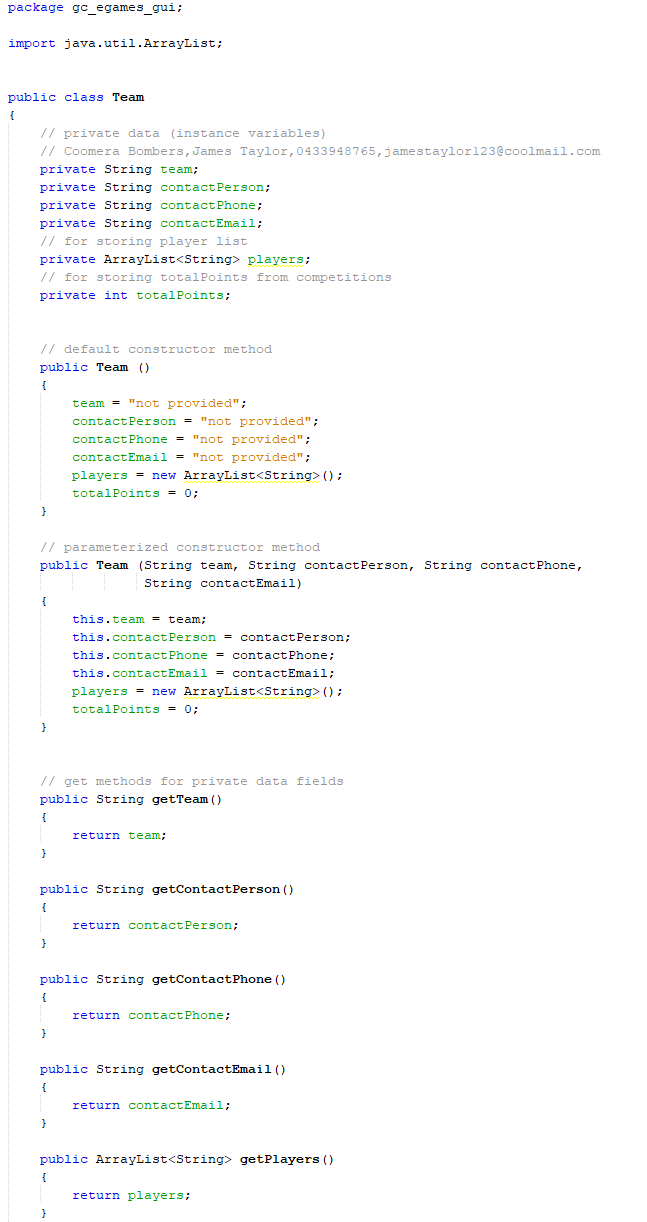
* Java customised (user-defined) class with at least 4 data fields.



Text, application

Description automatically generated

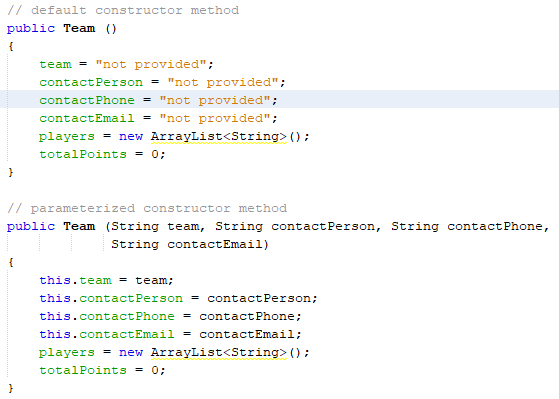
* Java class making use of
  + At least 4 instance variables
  + Method overloading
  + User-defined objects combined into a data structure



Text

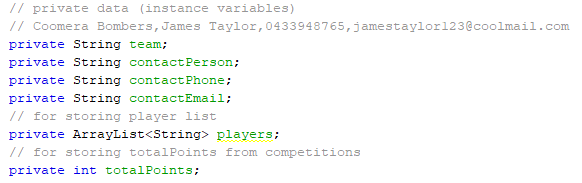
Description automatically generated with medium confidence

* Usage of two constructors in one class

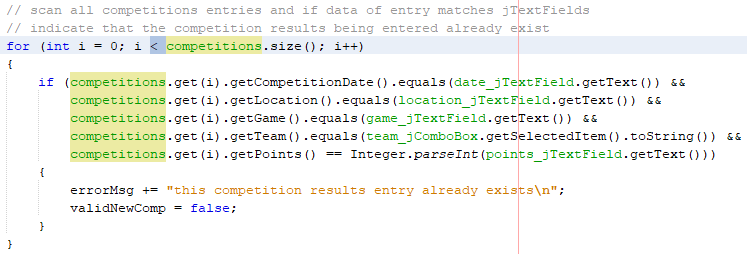


### Data Types, Operators, Expressions:

* Java code demonstrating usage of variables of at least 3 different data types



* Java code showing usage of at least 3 Operators from the following: (arithmetic, assignment, comparison, logical operators)



* Java code demonstrating usage of Expressions on at least 3 different occasions (expressions can be those used in conditional statements for if/else if or loops)

Text, letter

Description automatically generated

A screenshot of a computer

Description automatically generated

Text

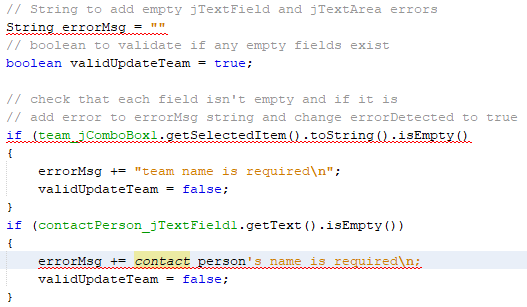
Description automatically generated

## Task 4 – Development Requirements

### Debugging

* (at most 5) syntax and logic errors, displaying usage of breakpoints and watches

3 Syntax errors: missing ; at end of String declaration, missing ) at end of if statement, missing “” for adding String to errorMsg.



This was screenshot was taken during v1 of my application. The method has changed significantly, but it demonstrates use of the breakpoint function helping to solve a logical problem. Turns out I had an i where a j should have been.

Graphical user interface, text, application, email

Description automatically generated

I had to manufacture this one as I didn’t take any more debug screenshots as I was working. It’s the what the previous method became. The leader board is missing a team, and stepping through showed the index was set too high, skipping the first team.

Graphical user interface, text, application

Description automatically generated

### Organisational Standards

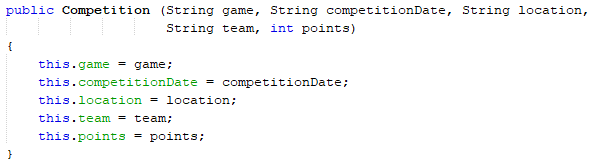
* Java code meeting naming conventions and a description of the applicable naming convention (e.g. variable and method declarations)

package gc\_egames\_gui; // package showing snake\_case

public class Competition // class showing PascalCase

private void readCompetitionData() // method showing camelCase

* Java code meeting layout conventions and a description of the applicable layout convention (e.g. tab indentations, use of curly bracket alignment and spacing between operators and operands in expressions)



* Java code meeting documentation standards and a description of the demonstrated documentation standard (e.g. header and method comments, version control)

Graphical user interface, text, application, chat or text message

Description automatically generated

### Functionality testing with outcomes

Tests were performed at various stages of development, so I included what version the test was conducted during in the bottom left of the Function to test column.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function to test** | **Test Steps** | **Expected Result** | **Actual Result** | **Fix Required** | **Status**  **Pass/Fail** |
| 1. Add a new team (test with no inputs)  v.1 | 1.1 Launch app 1.2 Click ADD NEW TEAM tab 1.3 Enter nothing for any of the text fields 1.4 Click SAVE NEW TEAM button | Pop-up displays 5 error messages – requiring inputs for: - team name - contact name - phone - email - player’s names | Pop-up displays 4 error messages – requiring inputs for: - team name - contact name - phone - email | Player’s names to be validated | FAIL (on first test)  Fix made and retested - PASS |
| 2. Display top teams list  v.1 | 2.1 Launch app  2.2 Click Display Top Teams button | Pop-up displays all teams and total points in descending order | Pop-up displays total points in descending order but team names incorrectly matched | Change order of teams in unison with points array | FAIL (on first test)  Added teams array and mirrored points array, retested - PASS |
| 3. Display read-in competition data to table  v.1 | 5.1 Launch app  5.2 Check if table on home tab displays data from competitions.csv | Table is populated with data from csv | Table is populated with data from csv | No fix required | PASS – the code implemented the function without fault |
| 4. Add a new competition result (test with non-numeric value in points)  v.2 | 3.1 Launch app  3.2 Click ADD NEW COMP RESULT tab  3.3 Enter non-numeric value into points text field  3.5 Click SAVE NEW COMP RESULT button | Pop-up displays error message – Points must be a number >= 0 | Pop-up message only displays error when char is not an integer, allowing negative numbers | Add conditional statement >= 0 | FAIL (on first test)  Added fix and retested - PASS |
| 5. Update an existing team (test with blank line in Player Names field)  v.2 | 4.1 Launch app  4.2 Click UPDATE EXISTING TEAM tab  4.3 Enter nothing for any text fields  4.4 Click SAVE UPDATE button | Pop-up displays error message – requiring: - no blank lines in player names text area | Pop-up displays error message – requiring: - no blank lines in player names text area | No fix required | PASS – the code implemented the function without fault |
| 6. Save entered data to disk when the app is closed (write to files)  v.3 | 6.1 Launch app  6.2 Enter updated data to data fields of any tab with save option  6.3 Attempt to close application  6.4 Check files | Pop up displays option to save session data before closing application:  - if yes write to external files | Pop up displays option to save session data before closing application:  - selecting yes writes to external files | No fix required | PASS – the code implemented the function without fault |

# Part 3 – Hand-Over

## Task 5 – Conclusion

* Summary of the development process including a checklist of developed features:

First thing I did was go over all the support files:

* The scenario.docx provided a list of functional requirements (used for the initial client email) and acted as a guideline for GUI design.
* The developmentplan.docx covered a lot of the proposed development stages, including setup of the classes, variables, and methods. It offered some basic suggestions for how to proceed from there. I rewrote this part, grouping all the requirements by panel and listed how I planned on achieving each. This made an effective checklist during coding (I used this in place of header notes).
* The technicalreport.docx was a handy template for documentation. As such I copy/pasted the development stages I had reworked into the appropriate section and after setting the all the GUI elements in place it was the only document I required.

I developed the app by checking off everything in the proposed development stages (pg.4-6), not necessarily in order the order they are written. I started by finalising the GUI and importing the external files into the project as it made sense to set it and forget it. Then I setup the proposed classes, variables, and methods.

From this point it was all about trying to code the functional requirements based on the plan I had come up with. Initially I was trying a bunch of different ways of achieving a goal and rewriting it over and over, looking for the most efficient way of doing it. After a while of getting nowhere I decided to just get it to work, check it off, and proceed. This approach enabled me to complete the requirements and once I had, I created version 2.

With a better understanding of how everything was integrated I experimented with different approaches, trying to reduce redundancy in the code. After I finished v2 I was happy with it, until you gave the class a 2nd demo and I had some more ideas! So, I revisited and reworked the development plan, made a v3 and implemented the changes. The real breakthrough for me in this assignment was realising the power of an ArrayList of objects and utilizing them as memory.

I tested the methods and functional requirements as they were developed, often with the aid of the debug tool to step through and check variable changes. When a function required the use of multiple methods, I would test their integration to make sure everything was behaving as expected and when the application was complete, I created an executable file that would run on Windows 10 (dist folder of GC\_EGamesv3). I used this to perform system tests which involved going through every requirement and making sure they worked as intended (pg.7).

**EMAIL TO CLIENT**

|  |  |
| --- | --- |
| **To:** | Mark Brown |
| **From:** | Luke Wait |
| **Subject:** | Meeting to discuss and review Gold Coast E-Sports Desktop Application |
| **Message:** | Greetings Mark,  I’m happy to report I have completed software development of the Gold Coast E-Sports Desktop Application prototype!  I hope it meets your expectations and am eager to hear your thoughts. I’d like to request a meeting to provide you with a demo and review the outcome. Please advise a date and time at your nearest convenience.  Regards,  Luke Wait |

* A summary of the feedback from the meeting, including planned solutions to any issues raised in the meeting.

The application project meets all functionality requirements, however there is no security measures. It is recommended that the software includes some form of authentication (via a login and password) to ensure that only authorised users can access the application.

To avoid duplication, I added validation functions to check if competition results or team names already exist. It’s also worth considering additional checks and formatting rules for individual text fields, such as competition date, which would benefit from being consistent. There is also a rather large problem when writing to external files if the comma is allowed in the text fields as it is used as a delimiter, resulting in corrupted data!

Some extra functions could also come in handy, like the ability to delete teams and delete/edit competition results in case a mistake is made. Being able to sort the competitions table by column would be of benefit in retrieving relevant data easily.

A backup system for the external files would also be advantageous in case of writing errors. Maybe when attempting to write to external files it creates new files and moves the existing ones to a backup folder instead of overwriting them.

Despite the many ways it could be improved, it still works as intended. So mission accomplished!