

## CSC7053 Peer Assessment: Technopoly

<sup>1</sup>Values: 1 = Less than average; 2 = Slightly less than average; 3 = Average; 4 = Slightly more than average; 5 = More than average

<sup>2</sup>This value should consider contributions in the round – direct contributions to required deliverables, and contributions that have made

<b>Evaluation</b>				
Group Number: 11				
Name	Contribution of time and effort <sup>1</sup>	Contribution to team- working and motivation <sup>1</sup>	Contribution to the deliverables <sup>1,2</sup>	Peer Score (Range 85 – 115)
James Campbell	5	5	5	100
Neill Calvert	5	5	5	100
Nial Daly	5	5	5	100
Tom Mills	5	5	5	100
Andy Wilson	5	5	5	100

the deliverables possible.

### Declaration

"I declare that I have read the Queen's University regulations on plagiarism, and that any contribution I have made to this assignment is my own original work, except for any elements that I have clearly attributed to third parties. I understand that this assignment will be subject to an electronic test for plagiarism and will also be subject to the University's regulations concerning late submission if it is received after the deadline."

Name	Date	Confirmation
James Campbell	13/03/2019	I agree to the terms of the declaration
Neill Calvert	13/03/2019	I agree to the terms of the declaration
Nial Daly	13/03/2019	I agree to the terms of the declaration
Tom Mills	13/03/2019	I agree to the terms of the declaration
Andy Wilson	13/03/2019	I agree to the terms of the declaration



**QUEEN'S  
UNIVERSITY  
BELFAST**

**SCHOOL OF  
ELECTRONICS,  
ELECTRICAL  
ENGINEERING AND  
COMPUTER SCIENCE**

**Electronics, Electrical Engineering and Computer Science**

**Group 11 Technopoly Report**  
**CSC – 7053 Software Engineering**

Author: Neill Calvert [40150199]  
James Campbell [12535052]  
Nial Daly [40132152]  
Tom Mills [40148781]  
Andy Wilson [40110221]

Module Supervisor: Dr Ian O'Neill  
Programme: MSc Software Development  
Date: 14<sup>th</sup> March 2019

1. Introduction.....	5
1.1 Project Overview .....	5
1.2 Synopsis of Process and Scope .....	5
1.3 Project Delivery Timeline.....	5
2. Requirements Analysis .....	5
2.1 Project Kick-Off.....	5
2.2 Determination of Core System Functionality Requirements .....	5
2.3 Game Guide .....	8
2.4 Use Case Diagram.....	10
3. System Realisation.....	14
3.1 Discussion of Sequence Diagrams Developed .....	14
3.2 Developed Sequence Diagrams .....	16
4. System Design – UML Class Diagram.....	21
4.1 UML Rationale .....	21
4.2 Maintainability, extensibility and Key Design Decisions.....	21
5. Project Delivery Process .....	22
5.1 Project Management .....	22
5.2 Software Development Process .....	23
5.3 Test Plan.....	23
Appendices.....	24
Appendix A – Testing.....	24
Test Conditions .....	24
Test Cases .....	29
Test Procedures .....	34
JUnit Testing .....	37
Appendix B – Project Delivery Programme .....	38
Appendix C – Project Values and Rent Costs .....	39
Appendix D – Meeting Minutes .....	40

## Glossary of Terms

**CurrentPlayer:** The player who is taking their turn at that time.

**NeutralArea:** A square that if landed on will not incur any cost or action required by the player. In this game, these are the GO and VACATION areas.

**PropertyArea:** The square on the board corresponding to an area of technology.

**Field:** Several areas make up a field on the board that correspond to a city on the globe.

**Bank:** The original default owner of each area on the board.

**DevelopmentLevel:** The level of investment in a particular PropertyArea from level 1 to level 4.

**Desk:** Initial development level of a PropertyArea.

**Floor:** Development levels 1, 2 and 3.

**Office:** Development level 4.

# 1. Introduction

NC

## 1.1 Project Overview

The project entailed creating a text-based Monopoly game using the Java programming language. The project required the game to have a theme based on fields of the IT industry such as 'Artificial Intelligence' and required players to compete for possession of these fields. Basic Monopoly functionality including the paying of rent, buying of property and upgrading of property was also implemented in accordance with the project guidelines. This document provides an overview of the development process of this 'Technopoly' game, including requirements, realisation and design.

## 1.2 Synopsis of Process and Scope

The process followed during the project has been acquired from the project specification and expanded upon using external material (Rumbaugh, Jacobsen and Booch, 1999). This process is outlined below.

- **Requirements Analysis:** Use case descriptions and accompanying use case diagrams that concentrate on the main sets of sequences of actions that the system will be comprised of.
- **Realisation:** An accurate UML Sequence diagram that outlines events.
- **System Design:** A UML class diagram outlining the components of the system.
- **Design Process:** The software testing and verification process including a suitable set of minutes presented at the end of the report.
- **Development:** The implementation process providing a working system while taking into account the initial requirements.

AW

## 1.3 Project Delivery Timeline

The initial kick-off meeting yielded a project delivery timeline visualised in a Gantt chart located in appendix B showing the main tasks and workflows necessary for meeting the project objectives.

# 2. Requirements Analysis

JC

## 2.1 Project Kick-Off

The main purpose of the project kick-off meeting was to understand the requirements, expand upon any requirements that were deemed vague and create an acceptable requirements document from which all members could work from.

The main theme of the game surrounding fields of the IT industry was also expanded upon with the aim to allow players to initially buy a desk on a floor, before upgrading their desk to encompass that floor in popular technological locations around the globe.

ND

## 2.2 Determination of Core System Functionality Requirements

The project specification outlined several core requirements shown in the table below. These requirements were expanded upon and placed into 5 key contexts: **Game Startup**, **Board Setup**, **Moving Position**, **Managing Properties** and **Paying for Services**. A total of 34 requirements were developed within the bounds of these contexts that would aid the design and testing of the project.

JC, ND, NC, TM, AW

### 2.2.3 Requirements

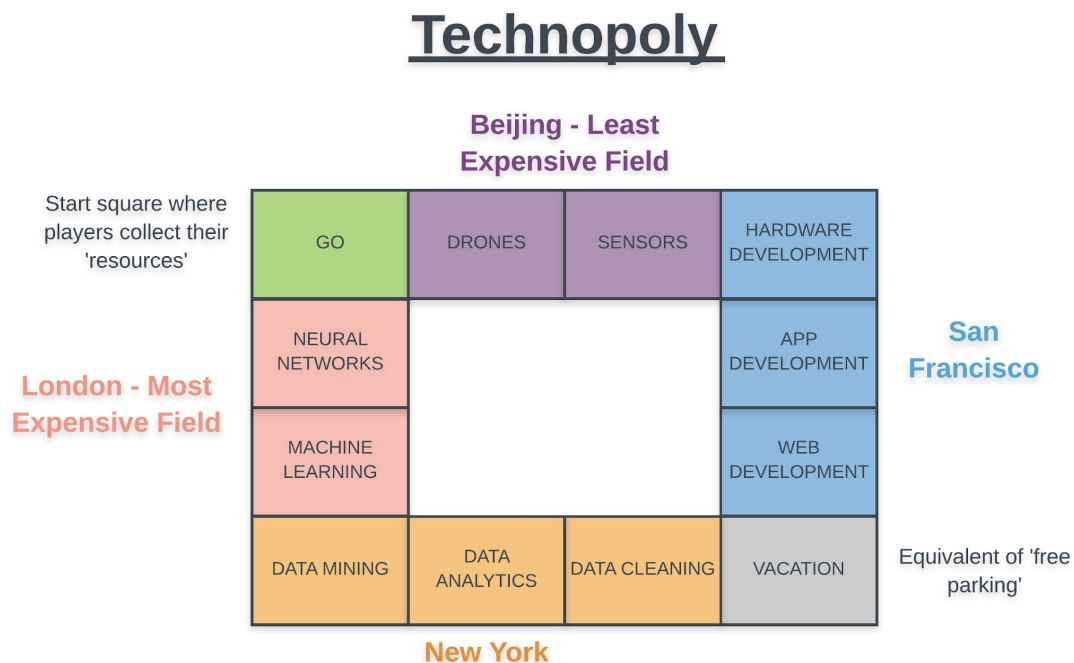
<u>Requirement /Context</u>	<u>Description</u>
<b>Context:</b>	<b>Game Start Up</b>
R1.	The game has at least two and up to four players.
R2.	When a player is added to the game, their names will be entered.
R3.	If a player enters a name that has already been assigned, a warning message will appear, asking the player to choose a different name.
R4.	The order of turns will be randomised.
<b>Context:</b>	<b>Board Setup</b>
R5.	There will be a start area where the player will pick up money each time they pass over or land on that area.
R6.	There will be a free parking area where nothing will happen if the player lands on it.
R7.	The remaining 10 areas will be split between 4 'fields'; 2 fields will have 3 areas; 2 fields will have 2 areas.
R8.	One of the two-area fields will be the most expensive field on the board to acquire and resource.
R9.	The other two-area field will be the least expensive fields to acquire.
<b>Context:</b>	<b>Taking turns – moving position</b>
R10.	Players will take turns in the order that they were shuffled into at the beginning of the game.
R11.	All players will start on the same (Go) area with the same initial amount of money (£500).
R12.	Each player will throw 2 virtual dice.
R13.	The player will move position by the number given by the dice.
R14.	If the player passes through or lands on the Start/Go area during their position movement, they will receive £200.
R15.	The player will be informed about which area they have landed on and its status.
R16.	If the area is available to buy the player will be offered the chance to purchase the area to become its owner.
R17.	If the area is owned by another player the current player must pay the appropriate fee.

R18.	If the area is owned by the current player or is a neutral area no further action is taken.
<b>Context:</b>	<b>Taking turns – managing resources</b>
R19.	After the player moves position, they will have the opportunity to invest their money in one of their areas.
R20.	Before you can develop an area, you must own all areas within the same field.
R21.	The current player must have the opportunity to manage their resources during a turn.
R22.	The current player must have the appropriate balance to invest in their chosen area.
R23.	A player develops an area by purchasing a floor.
R24.	Only one investment may be made during each turn.
R25.	Three floors are needed to invest in an office.
R26.	The development level is capped at an office.
R27.	If a player's resources have changed, the system indicates the reason for the change and announces the player's new 'balance'.
R28.	If a player invests in an area the fee for that area will be updated appropriately.
<b>Context:</b>	<b>Taking turns – paying fees</b>
R29.	When the current player lands on an area owned by another player they must pay the appropriate fee.
R30.	When the current player has paid a fee to another player both accounts will be appropriately updated.
R31.	When one player runs out of money, the game is over.
R32.	If one player no longer wants to play, the game is over.
R33.	When the game is over the amount of money each remaining player has will be displayed.
R34.	When the game is over the player with the most remaining money will be declared the winner.

## 2.3 Game Guide

### 2.3.1 Game Board

TM



The Board layout and game rules were made to cooperate with the core functionality requirements outlined in the requirements table.

The game requires a start area which is represented as the traditional 'Go' area. Players will pick up money each time they pass over this area. A 'Free Parking' area is also needed called Vacation. Both these neutral areas have no owner or fees.

The theme of the game incorporates a number of Technology fields. 2 fields will have 2 property areas while the other 2 fields will have 3 property areas. The board annotations highlight the respective cost: Beijing is the cheapest field, London is the most expensive, while New York and San Francisco are of an equal value.

The relative worth of each property area will be evident in the cost of acquiring and developing it.

The areas within a field are linked to the city where they are most prominent. For example, the fields of Drones and Sensors are both associated with Beijing.

JC

### 2.3.2 Game Rules

- The aim of the game is to become the wealthiest player through the buying and renting of properties.
- The game is for between 2 and 4 players.
- Players are assigned to a random order in each iteration of the game after they input their names.
- This order is maintained throughout the game.
- All players begin on the 'Go' area.
- All players begin with £500.
- The range allowed by rolling the dice is between 2 and 12.



- Each turn consists of movement across the board based on the dice roll and an opportunity for players to manage their resources.
- The opportunity to manage resources (invest in an area) will occur once per turn.
- Players may only select one area to invest in per turn.
- Players may quit the game at the start or at the end of each turn.
- If a player chooses to quit they are removed from the game, remaining player balances are displayed and the player with the highest balance is declared the winner.

### 2.3.3 Gameplay

NC
----

#### Landing on Areas (General)

- If the player lands or passes over the Go area they receive £200 in their balance.
- When the player lands on the Vacation area no changes are made to the players balance or status.
- When the player lands on an area the relevant details (owned/or available to buy desk) of the area will be displayed.

#### Additional Gameplay Features Associated with Property Areas

- If the current Player lands on an owned area, the owner and fee will be displayed and the appropriate fee will be removed from the current Player's balance.
- If the current Player lands on an owned area and has sufficient money to pay the fee then their balance will be reduced appropriately.
- If the current Player lands on an owned area and does not have sufficient money to pay the fee then the game will end.
- If the player lands on an unowned area they will be offered the chance to buy that area.
- If the current Player has sufficient funds to buy a desk in the area their balance will be deducted by the appropriate amount and they will own the area.
- If the player has insufficient funds they may not buy a desk in the area (own the area).
- If the current Player lands on an area that they already own, no further action/change required.

#### Additional Gameplay Features Associated with Managing Resources (investment)

- If the current Player owns all areas in a field and has sufficient money they will be able to invest in a selected area from that field. Their balance will be deducted by the investment price and the development level of the area will be increased.
- If the current Player does not own all areas in a field they cannot invest in any area in that field.
- If the area selected for investment has 3 floors on it, an office will be the next development level available.
- If the area selected for investment has an office on it, no further investment on this area will be permitted.
- Costs of properties, development levels and fees are shown in Appendix B.

#### Additional Quality of Life Gameplay Mechanics

- When prompted for an input the game will recognise an invalid input and explain the issue to the player (exception or otherwise) and prompt the player to confirm their choice

## 2.4 Use Case Diagram

### 2.4.1 Use Case Diagram Description

Initial work began on the drafting of a simple use case diagram to provide the group with a plan for the creation of the game. This draft diagram was then extended when use case descriptions were completed, before a final use case diagram was constructed. Overall, the use case diagram underwent several remodels with the finalised version presented below. Use case descriptions and their alternate flows are shown in section 2.4.2.

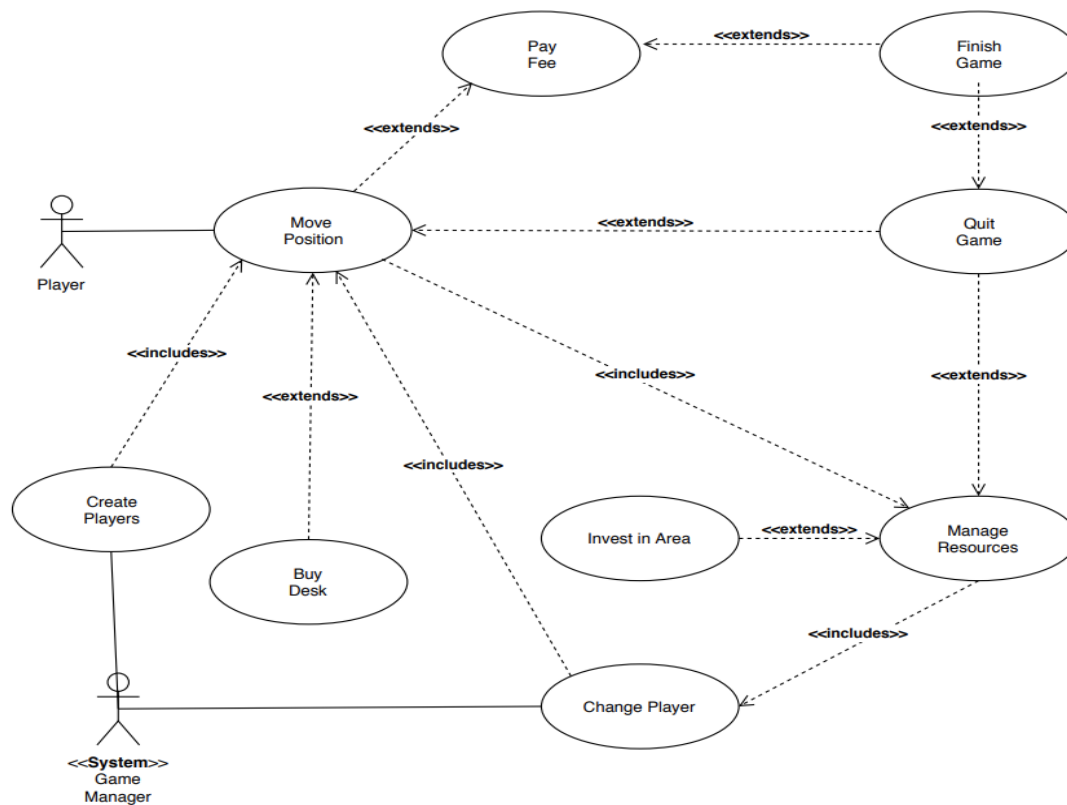


Figure 1 - UML Case Diagram

## 2.4.2 Use Case Descriptions

JC, ND, NC, TM, AW

Flow of Events for the <i>Create Players</i> Use Case (NC)	
<b>Objective</b>	To record player details
<b>Precondition</b>	1. Game has been loaded
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. At the start of the game the user shall be prompted to enter the number of players</li> <li>2. User shall be prompted to enter player names</li> <li>3. When all (max of four) players details have been entered, the order of play will be randomised and the players will be informed of the order of play</li> <li>4. First player will be set as the current player</li> <li>5. Inclusion point → Move Position use case</li> </ol>
<b>Alternative Flows</b>	At 1: If an invalid number of players is entered the user will be prompted to enter a valid number
	At 3: If user enters a duplicate or invalid name they will be prompted to enter a new, distinguishing name.
<b>Post-Condition</b>	<ul style="list-style-type: none"> <li>• Current player about to start game</li> </ul>

Flow of Events for the <i>Move Position</i> Use Case (ND)	
<b>Objective</b>	To move the piece corresponding to the player to another Area on the board
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. It is the current player's turn</li> <li>2. The current position/area of the player on the board is known</li> </ol>
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The player confirms they want to take their turn</li> <li>2. Player rolls dice</li> <li>3. The player will move position/area on the board based on the number obtained from the rolled dice and their previous position</li> <li>4. The player is informed of their new position/area on the board</li> <li>5. The player is informed of the status of the new area (neutral area, owned by current player, available to buy desk or if fee is required to be paid)</li> <li>6. If the area is a neutral area/owned by current player no change occurs</li> <li>7. The player is told that their movement turn is over</li> <li>8. Inclusion point→ Manage Resources use case</li> </ol>
<b>Alternative Flows</b>	At 1: If the player chooses to quit on initial menu Extension point → Quit Game use case
	At 3: If the player position has moved through/landed on the Go area the player is informed of this and their account balance is increased by the appropriate amount.
	At 6: If area is available to buy desk Extension point→ Buy Desk use case
	At 6: If current area is owned by another player Extension point→ Pay Fee use case
<b>Post-Condition</b>	<ul style="list-style-type: none"> <li>• The player has moved position on the board. Move to manage resources use case</li> </ul>

Flow of Events for the <i>Buy Desk</i> Use Case (AW)	
<b>Objective</b>	The current player will 'own' a desk at the current area
<b>Precondition</b>	1. The current area is not 'owned' by another/current player
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The player will be asked if they wish to purchase a desk at the property</li> <li>2. The player indicates they want to buy a desk</li> </ol>

	<ol style="list-style-type: none"> <li>If player has enough money in their account the player's account has the desk amount of money for this area removed from their account</li> <li>The player will be informed of their new balance</li> <li>Return to Move Position use case main flow point 8</li> </ol>
<b>Alternative Flows</b>	At 1: The player indicates they do not want to buy a desk. Move to 5
	At 3: If the player does not have enough money in their account they are informed that they cannot buy the desk. Move to 5
<b>Post-Condition</b>	<ul style="list-style-type: none"> <li>The player now has a desk on the current area (owns the property)</li> </ul>

<b>Flow of Events for the <i>Pay Fee</i> Use Case (AW)</b>	
<b>Objective</b>	<b>Player will pay fee to the Area owner</b>
<b>Precondition</b>	<ol style="list-style-type: none"> <li>The current player is on an area owned by another player</li> </ol>
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>The area is checked for its owner and desk/floor/office status</li> <li>If the player has enough money in their account for the fee it is deducted from their account</li> <li>The owner of the area will have their account increased by the fee amount</li> <li>The player will be informed of their new balance</li> <li>Return to Move Position use case main flow point 8</li> </ol>
<b>Alternative Flows</b>	At 2: If the player does not have enough money to pay the fee. Extension point → Finish Game use case
<b>Post-Condition</b>	<ul style="list-style-type: none"> <li>The players account will be deducted by the correct amount and their movement turn will be over. Move onto manage resources use case</li> </ul>

<b>Flow of Events for the <i>Manage Resources</i> Use Case (JC)</b>	
<b>Objective</b>	<b>To allow the current player to choose an area to invest in</b>
<b>Precondition</b>	<ol style="list-style-type: none"> <li>The current player has completed the movement phase of their turn</li> </ol>
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>The current player is presented with the areas they own</li> <li>The current player chooses to invest in an area Extension point → Invest in area use case</li> <li>Player confirms they want to continue playing Inclusion point → Change player use case</li> </ol>
<b>Alternative Flows</b>	At 2: The current player chooses to not invest in an area. Go to 3
	At 3: The current player chooses to quit the game extension point → Quit Game Use Case
<b>Post-Condition</b>	<ul style="list-style-type: none"> <li>Current player has chosen an area to invest in</li> </ul>

Flow of Events for the Invest in Area Use Case (JC)	
<b>Objective</b>	<b>The current player invests in chosen area</b>
<b>Precondition</b>	1. The current player has chosen an area to invest/develop
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The player owns all areas in the field of the chosen area</li> <li>2. The area does not have an office developed on chosen area</li> <li>3. If the player has enough money the investment price for that area will be deducted from their account</li> <li>4. The player will be informed of their new balance and the new fee/level of development of the area</li> <li>5. Return to Manage Resources use case point 3</li> </ol>
<b>Alternative Flows</b>	At 1: If the player does not own all areas in the chosen field they will be informed they cannot invest in this area Move to 5
	At 2: If the player already has an office in the chosen area they are informed that they may not invest anymore in this area. Move to 5
	At 3: If the player does not have enough money in their account they are informed they cannot invest in the area Move to 5
<b>Post-Condition</b>	The current player has invested in the chosen area. Return to Manage Resources use case

Flow of Events for the <i>Quit Game</i> Use Case (TM)	
<b>Objective</b>	<b>To exit the game if player wishes to do so</b>
<b>Precondition</b>	1. Current player at start (before movement)/end (after manage resources) of their turn
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. Player chooses to not continue with the game</li> <li>2. The player confirms choice</li> <li>3. Extension point → Finish Game use case</li> </ol>
	At 1: current player chooses to continue game return to Move Position use case main flow point 1 (if before movement) or Manage Resources use case main flow point 3 (if after manage resources)
<b>Alternative Flows</b>	At 2: Current player cancels quit game choice. Current player goes back to 1
<b>Post-Condition</b>	<ul style="list-style-type: none"> <li>• Current player has quit game and Finish Game use case initiated</li> </ul>

Flow of Events for the <i>Finish Game</i> Use Case (TM)	
<b>Objective</b>	<b>To finish the game with a summary of player(s) resources</b>
<b>Precondition</b>	1. A player has run out of resources or a player has chosen to quit the game
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. A summary of all the remaining player(s) resources is displayed</li> <li>2. The player with the highest amount of money in their account will be declared the winner</li> </ol>
<b>Alternative Flows</b>	At 2. If two or more remaining players have the highest amount of money in their account no overall winner will be declared
<b>Post-Condition</b>	The game has finished

Flow of Events for the <i>Change Player</i> Use Case (NC)	
<b>Objective</b>	To exit the game if the current player wishes to do so
<b>Precondition</b>	1. It is the end of the current player's turn. They have chosen to continue playing the game
<b>Main Flow</b>	1. Current player is changed to the next player in the order of play 2. Inclusion Point→ Move Position use case
<b>Alternative Flows</b>	
<b>Post-Condition</b>	<ul style="list-style-type: none"> <li>Current player has changed to the next player in the randomised order and they are about to begin their movement turn.</li> </ul>

### 3. System Realisation

AW

In order to determine the classes and objects that are interacting with one another to be included in the finished system, sequence diagrams were developed for the significant sequences and processes.

The following are a set of sequence diagrams, that display the most significant functions of the finished system according to system requirements:

- Player Movement
- Paying a Fee
- Buying a Desk
- Management of resources
- Play Game

#### 3.1 Discussion of Sequence Diagrams Developed

NC

##### 3.1.1 Player Movement

The movement of the player is dependent on the value returned from the diceRoll method. This returned value determines the new position of the current Player. The new position of the player will be checked to determine if the player has passed through the 'Go' area(their balance would be updated appropriately). The current area would now be assessed by the CheckArea method. This method initially checks if the player's current area is a neutral area by using the Enum field attribute (all neutral areas are set to have the neutral field). If this condition is true the manageResources method will be invoked (as a player cannot buy a neutral field).

If the condition is false this means the current area is a PropertyArea. The checkArea method will then check the owner attribute of the current area. There are three possible flows, depending on the owner attribute of the current area; if the owner is the bank (default owner) the buyDesk method is invoked (see 3.1.2). if the owner is another player the PayFee method is invoked (see 3.1.3). If the owner is the current Player then no further action is required and the manageResources method is invoked (see 3.1.4).

### 3.1.2 Buying a desk

ND

When a player's current area is a PropertyArea owned by the bank the buyDesk method is invoked. This method will start by seeking user input to confirm they want to proceed in buying a desk (owning the area). If they do not wish to continue the manageResources method is invoked.

If the player does confirm they want to buy a desk then their balance will be compared to the deskPrice attribute of the current area. If their balance is greater than the deskPrice then the player balance and the current area owner attributes are updated appropriately. The manageResources method will then be invoked. If the player balance is less than the deskPrice then no further changes occur and the manageResources method will be invoked (see 3.1.4).

JC

### 3.1.3 Paying a fee

When a player's current area is a PropertyArea owned by another player the payFee method is invoked. The system will retrieve the players balance and the current fee of the current area.

If the current Player's balance is greater than or equal to the current Fee the current Player's balance and the balance of the owner player are updated appropriately. If the current Player's balance is less than the current fee the quitGameEndTurn method is invoked.

TM

### 3.1.4 Management of Resources

The manageResources method will be invoked after the checkArea method. It will create and display an array of PropertyAreas owned by the current Player.

If the current Player selects an area (investArea) they want to invest in the player selects that they do not want to invest in a PropertyArea the checkFieldOwner method will be invoked. If no area is selected the quitGameEndTurn method will be invoked.

The checkFieldOwner method will check if the current Player owns all the areas in the same field as the investArea. This method will return a Boolean value. If false the quitGameEndTurn method will be invoked. If true, the balance of the current Player and the investmentPrice of the investArea will be compared. If the investmentPrice is greater than the current Player balance the quitGameEndTurn method will be invoked. Otherwise the developmentLevel of the investArea will be compared to the maximum investment level boundary. If the developmentLevel is the same as the maximum the quitGameEndTurn method will be invoked. Otherwise it will be less than the maximum. This is the final check before there is a successful investment. The current Player balance and investArea investmentLevel attributes will be updated appropriately. The new developmentLevel will be used in the updateCurrentFee method for the investArea to change the currentFee attribute of the investArea. The quitGameEndTurn method will be invoked.

AW, JC

### 3.1.5 Main Flow of the Game

The playGame method will begin the game and through the use of a Boolean (endGame), will enable players to quit the game (in two methods) or will ensure that if a player cannot pay the fee for the current PropertyArea the game will end. The createPlayers, displayPlayerList and setCurrentPlayer methods will enable players to enter their details and display the randomised order of play. The Boolean (endGame) will then initially be set to false. While this endGame Boolean is false the game will loop through the following methods: quitGameStartTurn, movePlayer, checkArea, manageResources, quitGameEndTurn, rotatePlayer, setCurrentPlayer.

Three methods will return a Boolean value which could change the endGame value: the quitGameStartTurn, quitGameEndTurn and checkArea methods. All these methods will have their initial return Boolean value set to false. If the player chooses to quit the game in the

quitGameStartTurn or quitGameEndTurn methods, the return Boolean will change to true. This will change the flow of events shown in the sequence diagram and invoke the endGameSummary method. The checkArea method will return a Boolean value which will depend on the Boolean returned by the payFee method; if the current Player cannot pay the fee, the Boolean returned by the payFee method to the checkArea method will be changed to true. This will in turn change the Boolean returned by the checkArea method to true and will change the flow of events shown in the sequence and invoke the endGameSummary method.

NC, JC, ND

### 3.2 Developed Sequence Diagrams

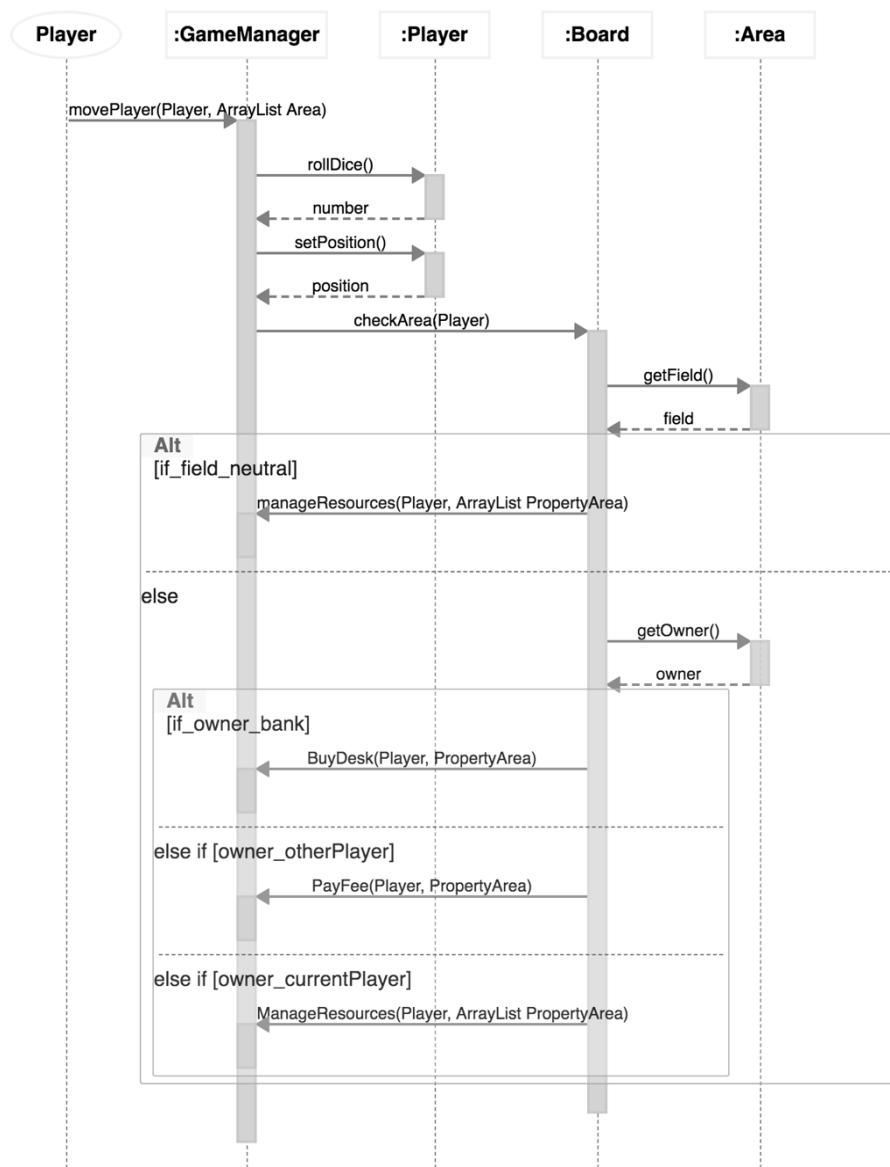
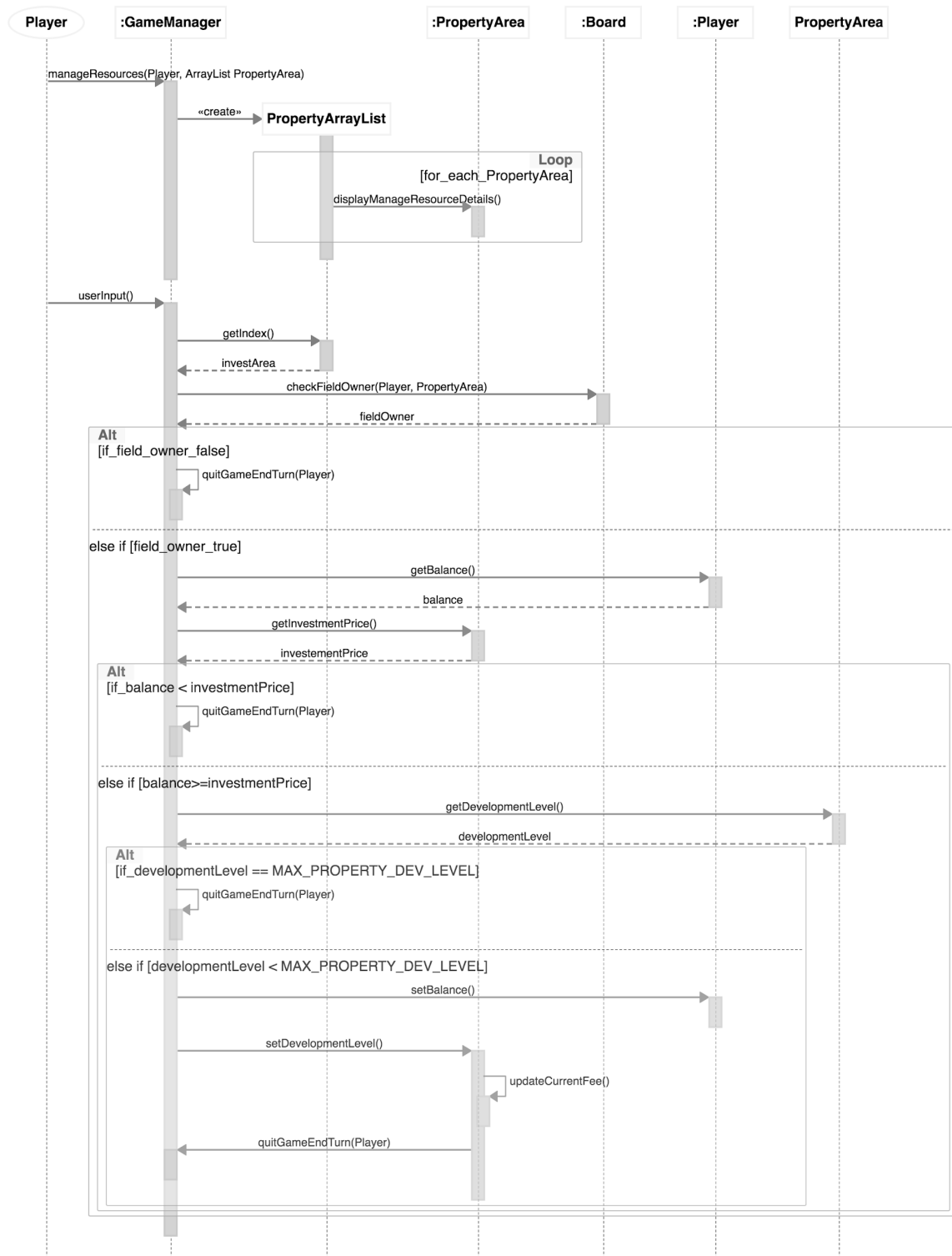
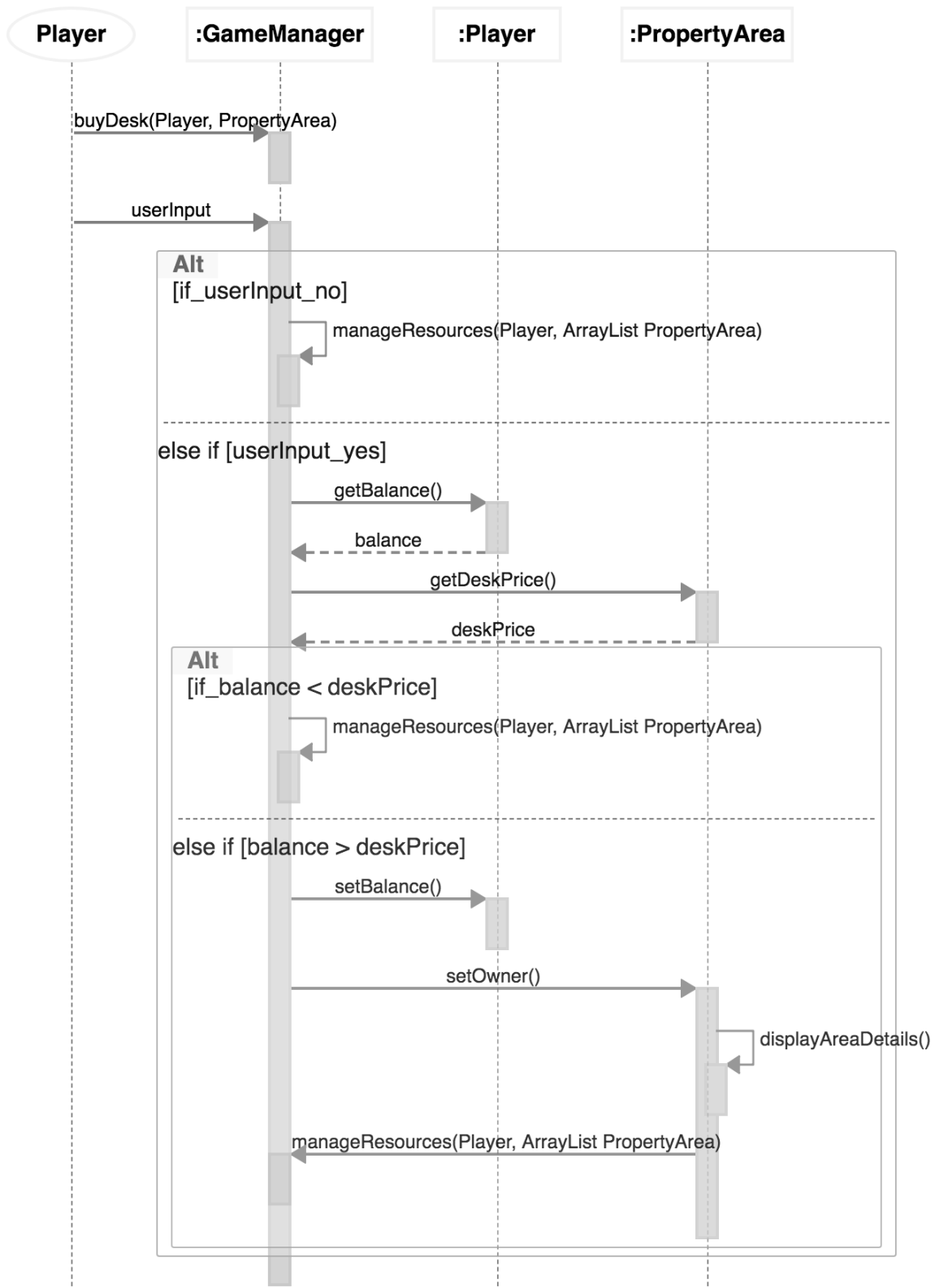


Figure 2 - Sequence Diagram showing the Player Movement

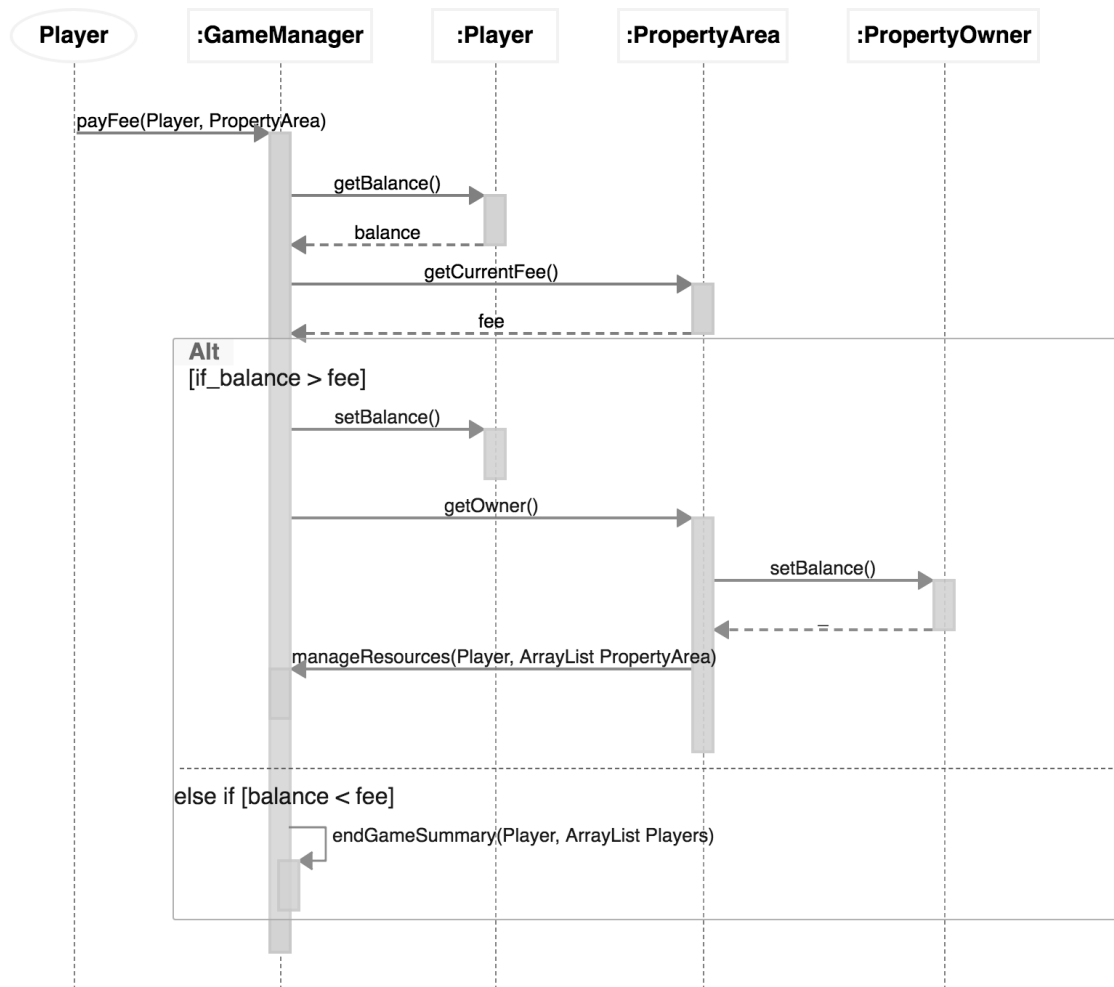




**Figure 3 - Sequence diagram showing the Management of Resources**



**Figure 4 - Sequence diagram showing the Buy Desk process**



**Figure 5 - Sequence diagram showing the Paying Fee process**

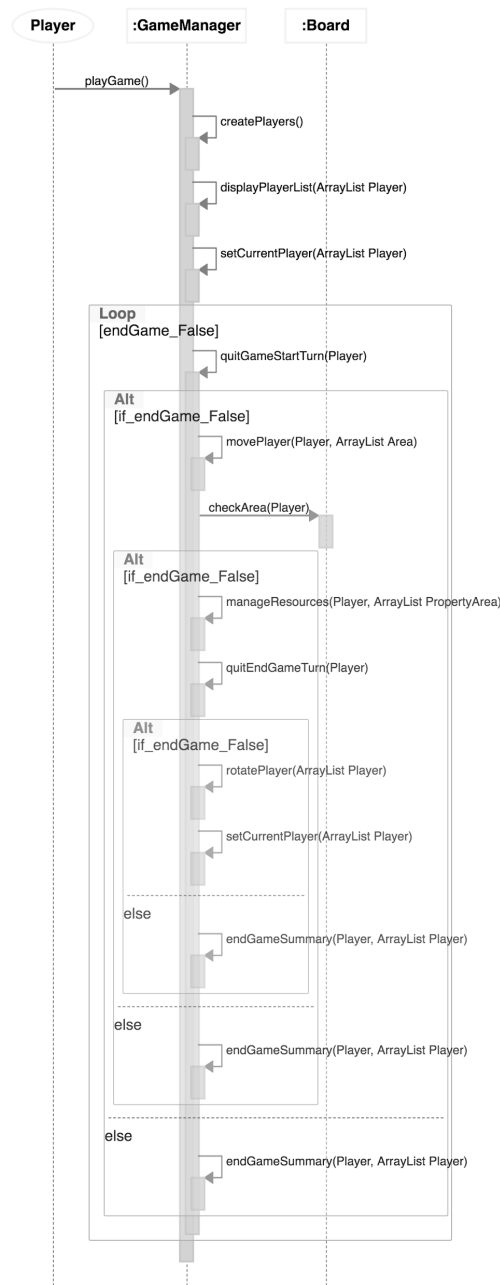


Figure 6 - Sequence diagram showing the Play Game process

## 4. System Design – UML Class Diagram

JC, NC, TM, ND, AW

### 4.1 UML Rationale

The sequence diagrams discussed in the previous section and the interactions between the objects that were modelled in the sequence diagrams themselves were translated into a class diagram. The structure of this class diagram adheres to the UML principles.

All objects and methods that have been identified in the sequence diagrams were taken and integrated into a range of Java classes. The development group used these now identified components of the sequence diagrams to in-turn, provide the necessary system functionality as required by the project specification requirements.

Where necessary, the relationships between classes have been clearly annotated to describe and define the multiplicity arrangements between them. Along with these 'multiplicity arrangement annotations' the relationships are labelled with relationship descriptors and an arrow, to establish which direction that particular relationship descriptor should be read.

This class diagram was subsequently used to provide the foundation for the beginning of the coding process. The objects, classes and methods that were identified here initially provided the original framework for the system coding to take form.

Using this class diagram as a reference point for the developers helped to maintain consistency, specifically, in areas where the relationships between the various classes were maintained and enhanced.

This approach also further encouraged consistency across code development as the naming conventions for the methods were clear to developers. This ensured that any methods referenced by the various developers would function properly when the system was merged and implemented.

### 4.2 Maintainability, extensibility and Key Design Decisions

From the project kick off meeting it was decided that maintainability and extensibility would be high priorities. To meet this requirement a number of constants were used when possible, as shown throughout the UML. A key example of these constants includes the minimum and maximum area boundaries of the Area class. Should a future development team wish to increase or decrease the number of squares for a project they may simply change one of these constants in one place, rather than altering multiple lines of code.

A key example of maintainability of code was the use of enums for the fields of the board. These enums make the class more self-documenting and prevented incorrect labelling of fields by individuals when working on the project. Secondly, these enums can be easily extended by adding a new field that could prove useful for further development of the project.

A further example of maintainability of code was the introduction of the abstract area class that prevents programmers from simply creating an area object that would not be useful. Instead this area class can be extended, as shown with the neutral and property area classes in the UML below. Further development of the game could involve the extension of this area class to include many more types of area on the board for future games.

Lastly, the use of a comparator from the Java collections framework was key to compare players to one another and sort them based on their balance. This allowed for the endGameSummary method to be created, presenting the remaining players sorted by their balance at the end of the Game.

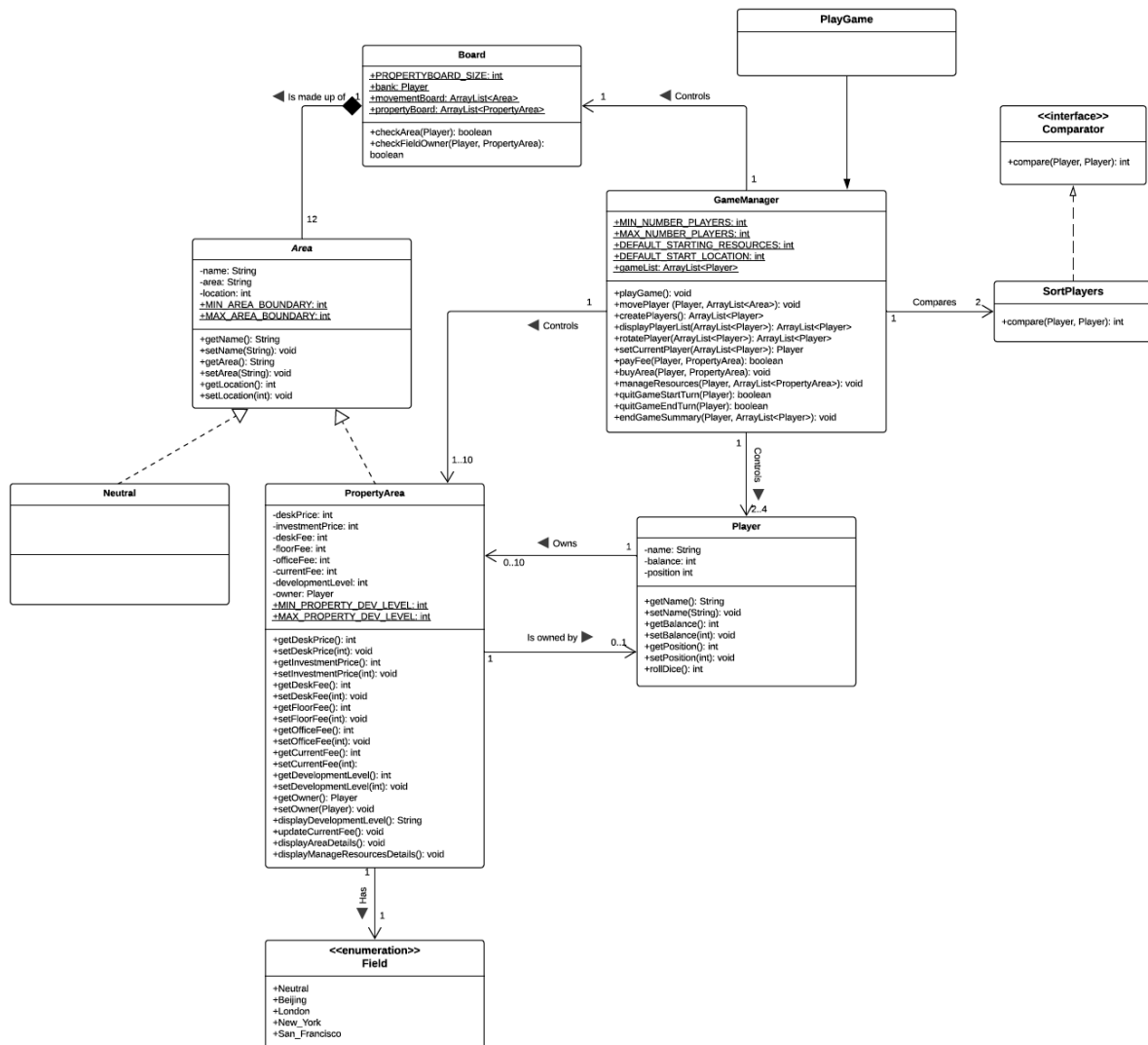


Figure 7 - UML Class Diagram

## 5. Project Delivery Process

ND

### 5.1 Project Management

The Technopoly project progressed sequentially, along with the development programme outlined in our Gantt chart in Appendix B.

Team meetings were conducted regularly, gradually increasing in frequency as the deadline approached. This was done to maintain a high degree of communication between members of the group and establish what work needed to be carried out on the software. Although there was a high volume of formal meetings, documented in the set of recorded minutes (found in Appendix D) there was also a high number of informal meetings.

Deadlines for important milestones were not set in stone, however, key goals were made clear at the end of each meeting to instil a sense of urgency and importance. Upon the date of these deadlines members met to review individual work, before submitting to GitLab as a whole.

TM

## 5.2 Software Development Process

The initial design tasks of the system (Requirement Analysis, Realisation and Design) led the group to take a sequential approach toward developing. In this, an iterative incremental development methodology (IID) approach was applied when developing the code. This provided greater flexibility in the process of producing the Technopoly game.

In applying the IID methodology, three key components of the process were identified: system coding/programming, unit testing and functionality testing.

These three components would be what our development process of the final, working system would revolve around. The IID approach enabled each group member to code and test each class within the system simultaneously along with the other developers. This would lead to a more efficient testing process as testing wasn't only implemented at the end of the coding process. This meant that the 'Big Bang' effect for example at the end of a 'Waterfall' approach did not occur.

This IID approach, in combination with the effective project delivery planning that was put into place, allowed the developers to effectively meet the requirements of the project without implementing a 'stressful' final 'Sprint' as the deadline approached.

The full software development process can be visualised in the Gantt chart located in Appendix B.

## 5.3 Test Plan

ND

"Testing is the systematic attempt to find faults in a planned way in the implemented software." (Bruegge & Dutoit, 2009).

It was imperative to the development of the Technopoly game software, that test planning is initiated early in the development process.

Test planning and implementation of said tests began as soon as the group use case model was agreed upon. This early implementation allowed the group to reduce the severity of any impacts regarding defects, identified along the development process. Early testing also provided the group with sufficient time, to address and correct defects. Although exhaustive testing is impossible, allowing for as much testing time as possible for the group increased the test thoroughness and effectiveness.

The testing methods that the group has applied to the development process includes component testing, led by whichever developer was responsible for their stage of coding. This, as well as static testing and a video recording of the system being used were implemented as test methods.

NC, AW

## 5.4 Test Results

During functional testing a total of 5 test cases failed, amounting to 14.7% of test cases. These test cases were inspected and found largely to be due to human error. This was the case for the deduction of the desk fee method in which the desk price was deducted from a current Player instead of the desk fee. These issues were fixed immediately after testing, before regression testing was carried out to ensure no further harm was done to the system during debugging. The results of this functional testing can be viewed in Appendix A, while component JUnit tests may be viewed in the source folder 'test' within the project submission.

# Appendices

## Appendix A – Testing

JC, ND, NC, TM, AW

### Test Conditions

Test Condition ID	Description	Source	Priority	Priority Justification	Comments
T_CON1	To show that the game can have 2 to 4 players	R1	High	Models of the System	
T_CON2	To show that the player can enter their names	R2	High	Models of the System	Provided their name has not already been entered
T_CON3	To show when a player enters a name that has already been entered the system prevents entering the name	R3	High	Models of the System	
T_CON4	To show that the order of players is randomised	R4	Medium	Models of the System	
T_CON5	To show that the 'Go' Area exists and players start there.	R5	High	Models of the System	
T_CON6	To show that the free parking area exists	R6	High	Models of the System	
T_CON7	To show that the board has 10 property Areas	R7	High	Models of the System	
T_CON8	To show that the most expensive Area on the board is Neural Networks in London	R8	High	Models of the System	



T_CON9	To show that the Areas in Beijing are the cheapest	R9	High	Models of the System	
T_CON10	To show that players will take turns in the order that they were shuffled into at the beginning of the game	R10	High	Models of the System	
T_CON11	To show that all players will start on the same (start) Area with the same initial amount of resources	R11	High	Models of the System/Risk	
T_CON12	To show that players can throw two virtual dice that provide movement of between 2 and 12 Areas	R12	High	Models of the System/Risk	Two virtual dice with a combined value of 12
T_CON13	To show that the player will move position by the number given by the dice.	R13	High	Models of the System/Risk	
T_CON14	To show that if the player passes through or lands on the Start/Go area during their position movement, they will receive £200.	R14	High	Models of the System/Risk	
T_CON15	To show that the player will be informed about which area they have landed on and its status	R15	High	Models of the System/Risk	

T_CON16	To show that If the area landed on by a player is available to buy, the player will be offered the chance to purchase the area to become its owner.	R16	High	Models of the System/Risk	
T_CON17	To show that if the area is owned by another player the current player must pay a fee	R17	High	Models of the System/Risk	
T_CON18	To show that if the area is owned by the current player or is a neutral area no further action is taken.	R18	High	Models of the System/Risk	
T_CON19	To show that after the player moves position, they will have the opportunity to invest their money in one of their areas.	R19	High	Models of the System/Risk	
T_CON20	To show that before you can develop an area you must own all areas within the same field.	R20	High	Models of the System/Risk	
T_CON21	To show that the current player must have the opportunity to manage their money during a turn.	R21	High	Models of the System/Risk	
T_CON22	To show that the current player must have the appropriate	R22	High	Models of the System/Risk	

	balance to invest in their chosen area.				
T_CON23	To show that a player develops an area by purchasing a floor.	R23	High	Models of the System/Risk	
T_CON24	To show that only one investment may be made each turn.	R24	High	Models of the System/Risk	
T_CON25	To show that three floors are needed to invest in an office.	R25	High	Models of the System/Risk	
T_CON26	To show that the investment level is capped at an office.	R26	High	Models of the System/Risk	
T_CON27	To show that if a player's money have changed, the system indicates the reason for the change and announces the player's new 'balance'	R27	High	Models of the System/Risk	
T_CON28	To show that if a player invests in an area its fee will be updated appropriately.	R28	High	Models of the System/Risk	
T_CON29	To show that when the current player lands on an area owned by another player they must pay the appropriate fee.	R29	High	Models of the System/Risk	
T_CON30	To show that when the current player has paid a fee to another player	R30	High	Models of the System/Risk	

	both accounts will be appropriately updated.				
T_CON31	To show that when one player runs out of money, the game is over.	R31	High	Models of the System/Risk	
T_CON32	To show that if one player no longer wants to play, the game is over.	R32	High	Models of the System/Risk	
T_CON33	To show that when the game is over the amount of money each player has will be displayed.	R33	High	Models of the System/Risk	
T_CON34	To show that when the game is over the player with the most money will be declared the winner.	R34	High	Models of the System/Risk	

## Test Cases

JC, ND, NC, TM, AW

Test case ID	Objective	Preconditions	Test data	Expected Result	Test condition(s)	Priority	Test completion date	Test Status	Tester	Defect ID	Defect Status	Defect Severity	Open Date	Close Date	Description / Comment Priority	Priority
TCase_1	Check that the game can be played by 2 - 4 players	In Technopoly package, running the gameManager class	Name of players: 1: TestName1; 2: TestName2; 3: TestName3; 4: TestName4;	The Technopoly game accepts the number of players as valid, then prompts for names	TCON_1 TCON_2	High	06/03/2019	Passed	James Campbell							
TCase_2	Check that the game cannot be played by less than 2 players or more than 4	In Technopoly package, running the gameManager class	Number of players: 1, 5	The Technopoly game does not accept number of players as valid and prompts for a valid value	TCON_1	High	06/03/2019	Passed	James Campbell							
TCase_3	Check that player enters a valid name	In Technopoly package, running the gameManager class	Name: TestName1	The Technopoly game accepts the input as valid and proceeds with game	TCON_2	High	06/03/2019	Passed	James Campbell							
TCase_4	Check that the player cannot enter an invalid name	In Technopoly package, running the gameManager class	Name: " "	The Technopoly game does not accept the input as valid and does not move on until valid input entered	TCON_2	High	06/03/2019	Passed	James Campbell							
TCase_5	Check that a player is prevented from entering a name that has already been entered	In Technopoly package, running the gameManager class	Name of players: 1: TestName1; 2: TestName1;	Only the first entry of name will be accepted, second player will be asked to enter again	TCON_3	High	06/03/2019	Failed	James Campbell	DEF_1	Closed	Moderate	06/03/19	06/03/19	Unclear which player should reenter their name	High
TCase_6	To show that the order of players is randomised	In Technopoly package, running the gameManager class Player details have been entered		Players are randomly ordered and game proceeds with first turn	TCON_4	Medium	06/03/2019	Passed	James Campbell							
TCase_7	Check that the 'Go' square exists and that players start there	In Technopoly package, running the gameManager class		The Player is prompted with 'Your current area is: Go' prior to taking first turn	TCON_5 TCON_7	High	06/03/2019	Passed	James Campbell							
TCase_8	Check that the 'Vacation' square exists	In Technopoly package, running the gameManager class		If the Player rolls a dice number which results in them landing on the 'Vacation' square. The system will inform them of their new position	TCON_6 TCON_7	High	06/03/2019	Passed	Neill Calvert							
TCase_9	Check that the most expensive area on the board is Neural Networks in London	In Technopoly package, running the gameManager class		When the Player lands on Neural Networks, the cost of the property (as well as fees) will be greater than those when the Player lands on any area in Beijing, San Francisco or New York	TCON_7 TCON_8	High	06/03/2019	Passed	Neill Calvert							

TCase_10	Check that the least expensive areas are in the Beijing field	In Technopoly package, running the gameManager class		When the Player lands on either Drones or Sensors, the cost of the property (as well as fees) will be less than those when the Player lands on any area in San Francisco, New York or London	TCON_7 TCON_9	High	06/03/2019	Passed	Neill Calvert								
TCase_11	Check that players will take turns in the order that they were shuffled into at the beginning of the game	In Technopoly package, running the gameManager class		Order of players from the randomised start order is maintained with each player taking their turn in the specified order. E.g. Player One offered a turn, once finished, Player Two will be offered a turn, etc.	TCON_4 TCON_10	High	06/03/2019	Passed	Neill Calvert								
TCase_12	Check that all players will start on the same 'Go' square with the same initial amount of resources	In Technopoly package, running the gameManager class		After player names have been entered, the system will inform them that 'Your current area is: Go' and 'Your current balance is: £200'	TCON_5 TCON_11	High	06/03/2019	Passed	Neill Calvert								
TCase_13	Check that that players can throw 2 virtual dice	In Technopoly package, running the gameManager class		System will generate a random number between 2-12 then the Player will be informed of that number	TCON_12	High	06/03/2019	Failed	Neill Calvert	DEF_2	Closed	Moderate	06/03/19	06/03/19	Numbers from 1-12 were being generated	High	
TCase_14	Check that the player will move position by the number given by the dice	In Technopoly package, running the gameManager class		System will move the Player by the number of squares they rolled and inform them of their new position.	TCON_13	High	06/03/2019	Passed	Neill Calvert								
TCase_15	Check that if a player passes through 'Go' area during their position movement, they will receive £200	In Technopoly package, running the gameManager class		If the Player rolls a dice number which will result in them moving/landing on or passing the 'Go' area, the system will inform them of this and their available balance will be credited with £200	TCON_5 TCON_14	High	06/03/2019	Passed	Neill Calvert								
TCase_16	Check that a player will be informed about which area they have landed on and it's status	In Technopoly package, running the gameManager class		The Player is informed of what area they have landed on, what field it belongs to, if it has an owner, how much it costs to acquire (if it has no owner), if the Player is required to pay a fee	TCON_15	High	06/03/2019	Passed	Nial Daly								
TCase_17	Check that if a player lands on an area that is available to buy, the player will be offered the chance to purchase the area to become it's owner	In Technopoly package, running the gameManager class Area must not be owned by another player		The system will ask if the Player wishes to purchase the square, and will be asked to confirm via Input Player's available resources will be updated and displayed	TCON_15 TCON_16 TCON_27 TCON_28 TCON_29	High	06/03/2019	Passed	Nial Daly								

TCase_18	Check that if a player buys an area, the correct desk price is withdrawn from their balance	In Technopoly package, running the gameManager class Area must not be owned by another player		The system will deduct the desk price from the players balance when they buy an area	TCON_15 TCON_16 TCON_27 TCON_28 TCON_29	High	06/03/2019	Failed	Nial Daly	DEF_3	Closed	Severe	06/03/19	06/03/19	Desk fee was deducted instead of desk price	High
TCase_19	Check that if a player chooses to buy a property that costs more than their current resources they are prevented from buying the desk	In Technopoly package, running the gameManager class Area must not be owned by another player		The system will prevent the purchase of areas that cost more than the current players resources	TCON_15 TCON_16 TCON_27 TCON_28 TCON_29	High	06/03/2019	Failed	Nial Daly	DEF_4	Closed	Severe	06/03/19	06/03/19	Defect dependant on DEF_3. Desk price was incorrect	High
TCase_20	Check that if a player lands on an area that is owned by another player; they must pay the area's owner a fee.	In Technopoly package, running the gameManager class Area must be owned by another player		The system will inform the Player that they are required to pay a fee to the area's owner. The Player will be informed of the amount and then their updated balance	TCON_15 TCON_17	High	06/03/2019	Passed	Nial Daly							
TCase_21	Check that if a player lands on an area which is owned by themselves no further action is needed	In Technopoly package, running the gameManager class Area must be owned by current player		The system will inform the Player that they own the square and no further action is needed	TCON_15 TCON_18	High	06/03/2019	Passed	Nial Daly							
TCase_22	Check that if a player lands on an area which is a neutral area, no further action is needed	In Technopoly package, running the gameManager class Area is either 'Go' or 'Vacation'		The system will inform the Player that they have landed on a neutral area and no further action is needed	TCON_5 TCON_6 TCON_15 TCON_18	High	06/03/2019	Passed	Tom Mills							
TCase_23	Check that after a player moves position, they will have the opportunity to invest their resources in one of their areas	In Technopoly package, running the gameManager class Player must own an area		The system will display all areas owned by the Player. The player will be given the option to select an area by entering its corresponding number.	TCON_15 TCON_19 TCON_20 TCON_27 TCON_28 TCON_29	High	06/03/2019	Passed	Tom Mills							
TCase_24	To show that a player must own all areas within the same field for them to develop an area	In Technopoly package, running the gameManager class Player must own all areas in a particular field		The Player will be able to select and invest in the area.	TCON_15 TCON_20	High	06/03/2019	Passed	Tom Mills							
TCase_25	To show that a player cannot develop an area unless they own all areas within that field	In Technopoly package, running the gameManager class Player must not own all areas in a particular field		The Player will attempt to select an area to invest in, but they system will inform them that they cannot invest until they own all areas in that particular field	TCON_15 TCON_20	High	06/03/2019	Passed	Tom Mills							
TCase_26	Check that after a player moves position, they have the opportunity to manage their resources	In Technopoly package, running the gameManager class		After being informed of their new position, the Player will be asked "Would you like to manage resources (y/n)?"	TCON_21	High	06/03/2019	Passed	Tom Mills							
TCase_27	Check that a player must have the appropriate resources to invest in chosen area and develops by purchasing a floor	In Technopoly package, running the gameManager class Player must not own all areas in a particular field		After selecting the area to develop, the Player will be informed of the new development level, the updated number of	TCON_22 TCON_23 TCON_24 TCON_27 TCON_28 TCON_29	High	06/03/2019	Passed	Tom Mills							

		Player must possess more resources than the investment fee for the seleted area		floors and their updated available resources													
TCase_28	Check that a player must have the appropriate resources to invest in chosen area	In Technopoly package, running the gameManager class Player must own all areas in a particular field Player must possess less resources than the investment fee for the seleted area		After selecting the area to develop, the Player will be informed that they do not possess enough resources to invest in that area. No change to development level	TCON_22 TCON_24	High	06/03/2019	Passed	Andrew Wilson								
TCase_29	To show that three floors are needed to invest in an office	In Technopoly package, running the gameManager during player manage resources turn. Player balance is higher than investment price		After selecting to invest in the area, the Player is informed that they now have an office Player balance and area development level updated and displayed	TCON_24 TCON_25 TCON_27 TCON_28	High	06/03/2019	Passed	Andrew Wilson								
TCase_30	To show that investment level is capped at an office	In Technopoly package, running the gameManager during player manage resources turn. Player balance is higher than investment price		The Player chooses to develop an area where they own an office. The system informs the Player no further development can occur. Player balance and area development level are not changed	TCON_24 TCON_26	High	06/03/2019	Passed	Andrew Wilson								
TCase_31	Check that when the current player lands on an area owned by another player they must pay the appropriate fee and both accounts will be appropriately updated	In Technopoly package, running the gameManager class Current player lands on an area owned by another player		Current player is informed of the ownership status of the area they have landed on. Current player is informed of the fee they are required to pay Current player's balance is debited and owner's current balance is credited	TCON_17 TCON_29 TCON_30	High	06/03/2019	Passed	Andrew Wilson								
TCase_32	Check that when a player is asked to pay a fee which exceeds their available resources, the game will end	In Technopoly package, running the gameManager class Current player lands on an area owned by another player, where the fee exceeds the current players resources		Current player is informed of the fee they are required to pay The fee they owe exceeds their available balance The current player is removed from the game, and the game ends. The remaining players are ordered according to their available resources at that moment in time and the winner is declared.	TCON_17 TCON_29 TCON_30 TCON_31 TCON_33 TCON_34	High	06/03/2019	Failed	Andrew Wilson	DEF_5	Closed	Severe	06/03/19	06/03/19	Current player removed from game but game continued with other player (s)	High	
TCase_33	Check that when a player decides to exit the game at the start of their turn, the game is over	In Technopoly package, running the gameManager class		The Player is prompted with "Would you like to continue (y/n)?" The Player enters 'n' The system prompts a confirmation	TCON_32 TCON_33 TCON_34	High	06/03/2019	Passed	Andrew Wilson								



				question 'Are you sure (y/n)?' The Player enters 'y' The Player is removed from the game. The remaining players are ranked according to available resources and a winner declared												
TCASE_34	Check that when a player indicates that they wish to exit the game, they are asked to confirm before exiting.	In Technopoly package, running the gameManager class		The Player is prompted with "Would you like to continue (y/n)?" The Player enters 'n' The system prompts a confirmation question 'Are you sure (y/n)?' The Player enters 'n' The game return's to the Player's turn.	TCON_21 TCON_32	High	06/03/2019	Passed	Andrew Wilson							

## Test Procedures

JC, ND, NC, TM, AW

Test Procedure ID	Test Case ID	Description	Comments
TProc_1	TCase_1 TCase_3 TCase_4	Open Eclipse, then the technopoly package and run the gameManager class. Enter a valid number of players. Visually note if game requests players to enter names. Visually check if system accepts valid names. Check system rejects invalid names then asks player to enter name again.	Visual Inspection
TProc_2	TCase_1 TCase_2	Open Eclipse, then the technopoly package and run the gameManager class. Enter an invalid number of players (outside of range). Visually check that system requests a valid number of players is entered and proceeds with game when valid value is entered.	Visual Inspection
TProc_3	TCase_3 TCase_5	Open Eclipse, then the technopoly package and run the gameManager class. Enter duplicate names. Check that the system recognises the same name has been entered twice and prompts the user to enter a different name.	Visual Inspection
TProc_4	TCase_6 TCase_7 TCase_11 TCase_12	After following the system instructions for entering player's names. Visually note if the order is randomised and different from the order in which the names were entered. Note where each player begins prior to their first roll along with their available resources. Compare to ensure the positions and balances are the same.	Visual Inspection

TProc_5	TCase_7 TCase_8 TCase_11 TCase_13 TCase_14 TCase_15	Follow the system prompts to navigate around the board a number of times. Note the order of turns and compare it to the randomised order initially set out. Note all the areas that are encountered (particularly a 'Vacation' or 'Go' area). Note any resources that are credited to a players account when they pass or land on the 'Go' area. Compare these amounts between turns and between players. Take note of the number of squares that a player is moved each turn and whether this is in line with the number they rolled.	Visual Inspection
TProc_6	TCase_9 TCase_10 TCase_16	Follow the system prompts to navigate around the board a number of times. Take note of all areas encountered and the information associated with each area (name, field, cost, fee). Compare this information to the board and that the ordering of squares is correct.	Visual Inspection
TProc_7	TCase_14 TCase_16 TCase_17 TCase_18 TCase_19 TCase_21 TCase_22 TCase_23 TCase_24 TCase_25 TCase_26 TCase_27	Follow the system prompts to navigate around the board a number of times. Invest in property when presented with the opportunity. Take note of the area information (e.g. ownership) and the impact on their available resources. Compare amount deducted from available resources to investment cost. Develop property when presented with the opportunity. Note if development level and floors/offices are added. Attempt to invest in an area which exceeds their available balance. Note impacts (if any) on available resources.	Visual Inspection














TProc_8	TCase_20 TCase_31 TCase_32	Follow the system prompts to navigate around the board a number of times. Navigate around the board a number of times, checking if the current player's available balance is decremented when they land on an area owned by another player. Note the fee charged compare it to the amount deducted (if any) from the current player's available resources. Continue game until one player does not have the required resources to pay a fee owned to another player. Visually check which player is removed, whether resources of the remaining players are displayed and which player is declared the winner.	Visual Inspection
TProc_9	TCase_33 TCase_34	Follow system prompts and choose to terminate the game if presented with the option. Visually check which player is removed, whether resources of the remaining players are displayed and which player is declared the winner.	Visual Inspection
TProc_10	TCase_33 TCase_34	Follow system prompts and choose to exit the game when presented with the option but opt out of the decision. Note whether game play resumes appropriately.	Visual Inspection

## JUnit Testing

JC, ND, NC, TM, AW

NeutralArea class, Player class and PropertyArea class tests focus on business rules, getters, setters and constructors that can be viewed in the source folder 'test' in the project submission.

## Appendix B – Project Delivery Programme

ID	Task Name	Start	End	14-Jan-19					21-Jan-19					28-Jan-19					04-Feb-19					11-Feb-19					18-Feb-19					25-Feb-19					04-Mar-19					11-Mar-19				
				M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S			
1	Project Release	Monday 14/01/19	Monday 14/01/19																																													
2	Project Kick Off Meeting	Monday 21/01/19	Monday 21/01/19																																													
3	Requirements Analysis	Tuesday 22/01/19	Monday 28/01/19																																													
4	Game Board Design	Friday 25/01/19	Monday 28/01/19																																													
5	UML Use Case Diagram Design & Description	Tuesday 22/01/19	Friday 01/02/19																																													
6	UML Sequence Diagrams Design	Monday 11/02/19	Friday 22/02/19																																													
7	UML Class Diagrams Design	Tuesday 19/02/19	Monday 25/02/19																																													
8	Individual Programming	Monday 04/02/19	Tuesday 05/03/19																																													
9	Collation of Programming Contributions	Tuesday 05/03/19	Tuesday 05/03/19																																													
10	Software Testing	Monday 04/02/19	Tuesday 12/03/19																																													
11	Report Development	Friday 25/01/19	Tuesday 12/03/19																																													
12	Recording Demonstration Video	Wednesday 13/03/19	Wednesday 13/03/19																																													
20	Project Submission	Thursday 14/03/19	Thursday 14/03/19																																													

## Appendix C – Project Values and Rent Costs

Field	Area	Price (£)		Fee (£)				
		Buy Desk	Investment Price	Desk	1 floor	2 floors	3 floors	Office
Neutral	Go							
Beijing	Drones	60	50	2	10	30	90	250
Beijing	Sensors	60	50	4	20	60	180	350
San Francisco	Hardware Development	160	100	12	50	150	450	900
San Francisco	App Development	180	100	14	60	180	540	950
San Francisco	Web Development	200	100	16	70	210	630	950
Neutral	Vacation (Free Parking)							
New York	Data Cleaning	160	100	12	50	150	450	900
New York	Data Analytics	180	100	14	60	180	540	950
New York	Data Mining	200	100	16	70	210	630	950
London	Machine Learning	300	200	26	100	300	900	1250
London	Neural Networks	320	200	28	120	360	1080	1400






## **Appendix D – Meeting Minutes**



## Meeting 1

Minutes for Group 11, Week commencing 21/01/19 (Week 2), Date of this minute 21/01/19

The following team members were present

Name (printed/typed)	Signature
Nial Daly	
James Campbell	
Neill Calvert	
Tom Mills	
Andrew Wilson	

**No tasks to report**

### **Actions Planned:**

Name & Role (1): Nial Daly, Team Member

- Establish Slack workspace which will act as the method of communication between team members.
- Make a Gantt chart framework.
- Read and review Technopoly document.
- Ensure that the team has familiarised themselves with the game of Monopoly.

Name & Role (2): James Campbell, Team Member

- Establish a Google Drive folder which will act as the central hub for all documentation.
- Give other team members permission to edit and upload documents.
- Read and review Technopoly document.
- Ensure that the team has familiarised themselves with the game of Monopoly.

Name & Role (3): Neil Calvert, Team Member

- Read and review Technopoly document.
- Ensure that the team has familiarised themselves with the game of Monopoly.

Name & Role (4): Tom Mills, Team Member

- Create a mock up board (using Lucid chart) to assist other team members of the team visualise the board squares.
- Read and review Technopoly document.
- Ensure that the team has familiarised themselves with the game of Monopoly.
- Consider any additional requirements that may need to be added to the 'core requirements section'.






Name & Role (5): Andrew Wilson, Team Member

- Read and review Technopoly document.
- Ensure that the team has familiarised themselves with the game of Monopoly.
- Consider any additional requirements that may need to be added to the 'core requirements section'.

## Meeting 2

Minutes for Group 11, Week commencing 21/01/19 (Week 2), Date of this minute 22/01/19

The following team members were present

Name (printed/typed)	Signature
Nial Daly	
James Campbell	
Neill Calvert	
Tom Mills	
Andrew Wilson	

### Task Reporting

Name & Role (1): Nial Daly, Team Member

- Confirmed each team member has access to Slack.
- Produced Gantt chart framework.
- Requirements analysis

Name & Role (2): James Campbell, Team Member

- Confirmed each team member has access to the Google Drive folder.
- Requirements analysis

Name & Role (3): Neill Calvert, Team Member

- Additional requirement to be added to the team's requirements; include the need for at least two players for a game to commence.
- Requirements analysis

Name & Role (4): Tom Mills, Team Member

- Investigated the use of Lucid Chart for developing UML Use Case diagram and descriptions.
- Produced Technopoly board.
- Nial Daly suggested limiting the number of squares to 12 (the minimum number required) to reduce complexity.
- Requirements analysis

Name & Role (5): Andrew Wilson, Team Member

- Additional requirement to be added to the team's requirements; include the need to randomise rolls of the dice.
- Requirements analysis

\*Printouts giving an overview of interim deliverables may be added as a supplement to these minutes.

### **Actions Planned (Briefly list the actions required of each team member for the next week.)**

It was agreed that each team member would be allocated responsibility for a number of use cases:

Name & Role (1): Neill Calvert, Team Member

- Use Case: Create players
- Use Case: Change player

Name & Role (2): Andrew Wilson, Team Member

- Use Case: Buy desk
- Use Case: Pay Fee

Name & Role (3): Nial Daly, Team Member

- Use Case: Move position
- Use Case: Take turn – manage resources (Use case responsibility to be shared with James)

Name & Role (4): James Campbell, Team Member

- Use Case: Take turn – manage resources (Use case responsibility to be shared with Nial)
- Use Case: Invest in area






Name & Role (5): Tom Mills, Team Member

- Use Case: Quit game
- Use Case: End game

### Meeting 3

Minutes for Group 11, Week commencing 28/01/19 (Week 3), Date of this minute 28/01/19

The following team members were present

Name (printed/typed)	Signature
Nial Daly	
James Campbell	
Neill Calvert	
Tom Mills	
Andrew Wilson	

#### Task Reporting

Name & Role (1): Neill Calvert, Team Member

- Reports Use Cases to group: Create players, Change player

Name & Role (2): Andrew Wilson, Team Member

- Reports Use Case to group: Buy desk, Pay Fee

Name & Role (3): Nial Daly, Team Member

- Reports Use Cases to group: Move position, Take Turn – Manage Resources

Name & Role (4): James Campbell, Team Member

- Reports Use Cases to group: Take Turn – Manage Resources, Invest in Area

Name & Role (5): Tom Mills, Team Member

- Reports Use Case to group: Quit Game, End game

**Actions Planned (Briefly list the actions required of each team member for the next week.)**

Name & Role (1): Nial Daly, Team Member

- Review Use Case documents.
- Produce table with property areas and fee values for the respective areas.

Name & Role (2): James Campbell, Team Member

- Review Use Case documents.
- Ensure consistency of language tense and terms.

Name & Role (3): Neill Calvert, Team Member

- Make a requirements table.
- Develop short synopsis of game concept.

Name & Role (4): Tom Mills, Team Member

- Reformat board to incorporate agreed areas/fields.
- Write a short synopsis on the format of the board.
- Make initial use case diagram (with actor and use cases).






Name & Role (5): Andrew Wilson, Team Member

- Examine possible tools (such as draw.io / Lucid) to make a sequence diagram for use cases.
- Investigate possible templates for the final report.

## Meeting 4

Minutes for Group 11, Week commencing 28/01/19 (Week 3), Date of this minute 01/02/19

The following team members were present

Name (printed/typed)	Signature
Nial Daly	
James Campbell	
Neill Calvert	
Tom Mills	
Andrew Wilson	

Task Reporting (Briefly list the progress for each team member in the last week.\*)

Name & Role (1): Nial Daly, Team Member

- Table with property areas and rent values for the respective areas/fields completed.

Name & Role (2): James Campbell, Team Member

- Amended use cases.
- Combined a number of use cases to reduce complexity of use case design.

Name & Role (3): Neill Calvert, Team Member

- Requirements table created.
- Produced short synopsis of game concept.

Name & Role (4): Tom Mills, Team Member

- Reformatted board to incorporate fields/areas.
- Produced short synopsis on the format of the board.
- Made initial use case diagram (will be evolved at a future date).

Name & Role (5): Andrew Wilson, Team Member

- Examine possible tools (such as draw.io / Lucid) to make a sequence diagram for use cases.
- Became familiar with Lucid chart.

- Generated template for sequence diagrams for use cases.
- Created template for final report. Uploaded for Google Drive folder.

\*Printouts giving an overview of interim deliverables may be added as a supplement to these minutes.

### **Actions Planned (Briefly list the actions required of each team member for the next week.)**

Name & Role (1): Nial Daly, Team Member

- Set up GitLab.
- Consider factors that will impact upon the layout of class diagram.
- Explore possible classes and objects.

Name & Role (2): James Campbell, Team Member

- Further refine Use Cases.
- Consider factors that will impact upon the layout of class diagram.
- Explore possible classes and objects.

Name & Role (3): Neill Calvert, Team Member

- Consider factors that will impact upon the layout of class diagram.
- Explore possible classes and objects.

Name & Role (4): Tom Mills, Team Member

- Consider factors that will impact upon the layout of class diagram.
- Explore possible classes and objects.

Name & Role (5): Andrew Wilson, Team Member






- Look at evolving sequence diagrams.
- Consider factors that will impact upon the layout of class diagram.
- Explore possible classes and objects.



## Meeting 5

Minutes for Group 11, Week commencing 04/02/19 (Week 4), Date of this meeting 04/02/19

The following team members were present

Name (printed/typed)	Signature
Nial Daly	
James Campbell	
Neill Calvert	
Tom Mills	
Andrew Wilson	

Task Reporting (Briefly list the progress for each team member in the last week.\*)

Name & Role (1): Nial Daly, Team Member

- Ensured each member of the team had access to GitLab.

Name & Role (2): James Campbell, Team Member

- Ensured group was content with changes made to Use Cases.

Name & Role (3): Neill Calvert, Team Member

- Further developed UML Sequence Diagrams.

Name & Role (4): Tom Mills, Team Member

- Further developed UML Sequence Diagrams.

Name & Role (5): Andrew Wilson, Team Member

- Amended sequence diagram template.

\*Printouts giving an overview of interim deliverables may be added as a supplement to these minutes.

Actions Planned (Briefly list the actions required of each team member for the next week.)

Name & Role (1): Nial Daly, Team Member

- Conduct initial design of code base.
- Review of sequence diagrams for game manager methods.

Name & Role (2): James Campbell, Team Member

- Conduct initial design of code base.
- Review of sequence diagrams for game manager methods.

Name & Role (3): Neill Calvert, Team Member

- Conduct initial design of code base.
- Review of sequence diagrams for game manager methods.

Name & Role (4): Tom Mills, Team Member

- Conduct initial design of code base.
- Review of sequence diagrams for game manager methods.
- Complete class diagram on Lucid chart using the general overview agreed upon by group.






Name & Role (5): Andrew Wilson, Team Member

- Conduct initial design of code base.
- Review of sequence diagrams for game manager methods.

## Meeting 6

Minutes for Group 11, Week commencing 11/02/19 (Week 5), Date of this minute 11/02/19

The following team members were present

Name (printed/typed)	Signature
Nial Daly	
James Campbell	
Neill Calvert	
Tom Mills	
Andrew Wilson	

### Task Reporting (Briefly list the progress for each team member in the last week.\*)

Name & Role (1): Nial Daly, Team Member

- Investigated potential data structures for board.
- Established initial data structure as ArrayList for board.

Name & Role (2): James Campbell, Team Member

- Initial development of dice roll code.
- Initial development of Player class.

Name & Role (3): Neill Calvert, Team Member

- Initial development of Create Player functionality.
- Validation rules of Create Player

Name & Role (4): Tom Mills, Team Member

- Updated UML Diagram to reflect agreed changes.
- Develop UML Class diagram.

Name & Role (5): Andrew Wilson, Team Member

- Further work on sequence diagrams focusing on Player opportunities/responsibilities when on a PropertyArea.

\*Printouts giving an overview of interim deliverables may be added as a supplement to these minutes.

**Actions Planned (Briefly list the actions required of each team member for the next week.)**

Name & Role (1): Nial Daly, Team Member

- Develop Area arrays (PropertyArea and Area).

Name & Role (2): James Campbell, Team Member

- Develop move player functionality.

Name & Role (3): Neill Calvert, Team Member

- Link create players array to movement.
- Develop UML Sequence diagrams.
- Develop UML Class diagram

Name & Role (4): Tom Mills, Team Member

- Set fields to Enums
- Set areas to Enums
- Develop UML Sequence diagrams.
- Develop UML Class diagram






Name & Role (5): Andrew Wilson, Team Member

- Develop check area functionality.

## Meeting 7

Minutes for Group 11, Week commencing 18/02/19 (Week 6), Date of this minute 19/02/19

The following team members were present

Name (printed/typed)	Signature
Nial Daly	
James Campbell	
Neill Calvert	
Tom Mills	
Andrew Wilson	

### Task Reporting (Briefly list the progress for each team member in the last week.\*)

Name & Role (1): Nial Daly, Team Member

- Developed Area arrays (PropertyArea and Area).

Name & Role (2): James Campbell, Team Member

- Developed move player functionality.

Name & Role (3): Neill Calvert, Team Member

- Linked create players array to movement.

Name & Role (4): Tom Mills, Team Member

- Set fields to Enums
- Set areas to Enums (after discussion with group it was agreed that areas would not be held as enums)

Name & Role (5): Andrew Wilson, Team Member

- Developed check area functionality.

\*Printouts giving an overview of interim deliverables may be added as a supplement to these minutes.

**Actions Planned (Briefly list the actions required of each team member for the next week.)**

Name & Role (1): Nial Daly, Team Member

- Prevent duplicate players being entered.
- Delete dryRun.

Name & Role (2): James Campbell, Team Member

- Further work on movement of players.
- Develop game board structure.

Name & Role (3): Neill Calvert, Team Member

- Multiple player functionality to be developed.
- Initial design of test cases.
- Established test procedure google sheet
- Collate tests into test plan

Name & Role (4): Tom Mills, Team Member

- Add fields as enums.
- Add gameManager class
- Initial design of test cases
- Established test cases google sheet
- Collate tests into test plan






Name & Role (5): Andrew Wilson, Team Member

- Initial design of test cases
- Established test procedures google sheet
- Collate tests into test plan

## Meeting 8

Minutes for Group 11, Week commencing 25/02/19 (Week 7), Date of this minute 25/02/19

The following team members were present

Name (printed/typed)	Signature
Nial Daly	
James Campbell	
Neill Calvert	
Tom Mills	
Andrew Wilson	

### Task Reporting (Briefly list the progress for each team member in the last week.\*)

Name & Role (1): Nial Daly, Team Member

- Prevented duplicate players being entered.
- Deleted dryRun.

Name & Role (2): James Campbell, Team Member

- Game board structure developed.
- Further work on movement of players completed.

Name & Role (3): Neill Calvert, Team Member

- Presented test conditions sheet.
- Collated tests into test plan.
- Developed UML Sequence diagrams.
- Developed UML Class diagram.

Name & Role (4): Tom Mills, Team Member

- Removed areas as enums.
- Added gameManager class.
- Presented test cases sheet.

- Collated tests into test plan.
- Developed UML Sequence diagrams.
- Developed UML Class diagram.

Name & Role (5): Andrew Wilson, Team Member

- Presented test procedures sheet.
- Collated tests into test plan.

\*Printouts giving an overview of interim deliverables may be added as a supplement to these minutes.

### **Actions Planned (Briefly list the actions required of each team member for the next week.)**

Name & Role (1): Nial Daly, Team Member

- Develop user input validation for manageResource, quitGameStartTurn, quitGameEndTurn
- Develop Board class
- Further commenting
- Create buyArea method

Name & Role (2): James Campbell, Team Member

- Develop Board class
- Develop manageResources method
- Alter updateFee method
- Demo run through of game

Name & Role (3): Neill Calvert, Team Member

- Develop user input validation for manageResource, quitGameStartTurn, quitGameEndTurn
- Develop Board class
- Develop displayDevelopmentLevel method

Name & Role (4): Tom Mills, Team Member

- Edit diceRoll method
- Develop manageResources method
- Tidy up code
- Restructure Package

Name & Role (5): Andrew Wilson, Team Member






- Develop user input validation for manageResource, quitGameStartTurn, quitGameEndTurn
- Develop displayDevelopmentLevel method



## Meeting 9

Minutes for Group 11, Week commencing 04/03/19 (Week 8), Date of this minute 05/03/19

The following team members were present

Name (printed/typed)	Signature
Nial Daly	
James Campbell	
Neill Calvert	
Tom Mills	
Andrew Wilson	

### Task Reporting (Briefly list the progress for each team member in the last week.\*)

Name & Role (1): Nial Daly, Team Member

- User input validation developed for manageResource, quitGameStartTurn, quitGameEndTurn
- Board class developed
- Further commenting
- buyArea method created

Name & Role (2): James Campbell, Team Member

- Board class developed
- manageResources method developed
- updateFee method altered
- Demo run through of game

Name & Role (3): Neill Calvert, Team Member

- User input validation developed for manageResource, quitGameStartTurn, quitGameEndTurn
- Board class developed
- displayDevelopmentLevel method developed

Name & Role (4): Tom Mills, Team Member

- diceRoll method edited
- manageResources method developed
- Check
- Tidy up of code
- Package restructuring

Name & Role (5): Andrew Wilson, Team Member

- User input validation developed for manageResource, quitGameStartTurn, quitGameEndTurn
- displayDevelopmentLevel method developed

\*Printouts giving an overview of interim deliverables may be added as a supplement to these minutes.

**Actions Planned (Briefly list the actions required of each team member for the next week.)**

Name & Role (1): Nial Daly, Team Member

- Agreed to conduct JUnit tests
- buyArea method refactored
- Further commenting

Name & Role (2): James Campbell, Team Member

- Agreed to conduct JUnit tests
- Create PlayGame class

Name & Role (3): Neill Calvert, Team Member

- Review of test conditions after game had been further developed

Name & Role (4): Tom Mills, Team Member

- Review of test conditions after game had been further developed






Name & Role (5): Andrew Wilson, Team Member

- Review of test conditions after game had been further developed
- Further commenting

## Meeting 10

Minutes for Group 11, Week commencing 11/03/19 (Week 9), Date of this minute 12/03/19

The following team members were present

Name (printed/typed)	Signature
Nial Daly	
James Campbell	
Neill Calvert	
Tom Mills	
Andrew Wilson	

### Task Reporting (Briefly list the progress for each team member in the last week.\*)

Name & Role (1): Nial Daly, Team Member

- Push JUnit tests to Git
- buyArea method refactored
- createPlayers method updated to deal with duplicate name input
- Further commenting
- Submit Report

Name & Role (2): James Campbell, Team Member

- Push JUnit tests to Git
- PlayGame class created
- Further commenting
- Submit Report

Name & Role (3): Neill Calvert, Team Member

- Tidy up report and Carry out final checks on own classes
- Further Commenting
- Finalise Gantt Chart
- Submit Report

Name & Role (4): Tom Mills, Team Member

- Tidy up report and Carry out final check on own classes
- Further Commenting
- Finalise Gantt Chart
- Submit Report

Name & Role (5): Andrew Wilson, Team Member

- Tidy up report and Carry out final check on own classes
- Further Commenting
- Submit Report