Table Of Contents

Abstract

The aim of this project is to design a game of monopoly using an Object-Oriented approach and following the Agile Scrum framework. This paper will outline the research required for the design of monopoly and will begin with an analysis of the requirements, use cases, and class descriptions needed for the design of the game. The product will be designed with the use of object diagrams and user interface mockups as well as state machines. Classes will be modelled using a collection of diagrams to portray the structure of the project, these will include collaboration and sequence diagrams. Finally, the source code will be written and provided to the reader based on the designs.

Introduction

This paper will seek to analyse, design and structure a game of monopoly with an Object Oriented Approach. The project is split into stages which include Analysis, Product Design, Class Design and finally the implementation. Source code for the game of monopoly will be provided and this will incorporate the designs and structure as outlined in the different stages.

Analysis

The first stage of this project is the Analysis Stage. Here, the requirements will be defined, the scenarios will be described, a primary class list determined, class diagrams and use case diagrams will be created and the results of a structured walkthrough will be provided.

Requirements

These requirements outline what the game will and will not do. They will highlight the scope of the project and give light to how the game will be developed. Below is the requirements specification of the game of monopoly.

Requirements Specification

1. Allowing 2-8 users to join a game session.
2. Permitting each player to pick a game piece from a choice of 8.
3. Allocating €1500 to players at the start of the game.

4. Enabling the player to navigate around the board by rolling two dice and moving the game piece.
5. Giving €200 to a player when they pass the ‘Go’ square.
6. Allowing player to take another turn if a double is rolled.
7. Permitting players to purchase property if it is unowned.
8. Validating that property is unowned.
9. Placing Title Deed Cards in players’ inventories when they have purchased a property.
10. Requiring a player to pay rent to another player if they land on their unmortgaged property.
11. Doubling the rent price for a set of properties once the complete set of properties is owned by one player.
12. Raising the rent price for a property when the property is improved with houses or hotels.
13. Auctioning a property if a property landed on by a player is unwanted and unowned.
14. Permitting players to mortgage property according to the property’s worth as specified on the Title Deed Card.
15. Enforcing that a property has no improvements before being mortgaged.
16. Receive payment for a mortgaged property with 10% interest when a player wishes to unmortgage it.
17. Preventing the player from collecting rent on a mortgaged property.
18. Allowing players to purchase up to four houses on a property when they own the full set of properties of the same colour.
19. Validating that houses are being built evenly on a property grouping.
20. Allowing the player to purchase hotels once four houses have been purchased on all properties in a grouping.
21. Enforcing that only one hotel may be built on a given property.
22. Preventing players from building improvements on a property if any of the properties in the grouping are mortgaged.
23. Auctioning buildings in the case of a building shortage when more than one player wishes to purchase the same building.
24. Allowing the player to receive a chance or community chest card when landing on the relevant space and permitting the player to follow the instruction given.
25. Putting a player in jail when the player lands on ‘Go To Jail’, rolls three doubles or receives a card instructing them to do so.
26. Enforcing a player in jail to stay in jail unless €50 is paid, the player waits three turns and pays €50, rolls a double on one of their three turns or the player has a ‘Get Out Of Jail Free’ card and wishes to use it.
27. Collecting taxes and fines from players.
28. Buying properties and buildings back from the player when the player wishes to sell.
29. Validating that a player’s buildings are sold back evenly.
30. Allowing players to make deals with each other which can include trading money, property, or ‘Get Out Of Jail Free’ cards.
31. Eliminating a player from the game when they can no longer pay what they owe and declare bankruptcy.

32. Enforcing a bankrupt player to give their mortgaged properties and ‘Get Out Of Jail Free’ cards to the other player that they are in debt to.
33. Requiring that the player who receives assets from a bankrupt player to pay 10% interest on any received mortgaged properties at time of receiving.
34. Auctioning a player’s mortgaged properties at full (unmortgaged price) and returning ‘Get Out Of Jail Free’ cards to the relevant decks when that user is bankrupt and in debt to the bank.
35. Allowing the last player left in the game to win.

Scenarios

Below is a brief list of scenarios describing a sample monopoly game play using players A,B,C and D.

1. Users A, B, C and D join a game together. They each choose their pieces, colours and names. The system sets up a wallet for each player and deposits €1500 in each wallet.
2. Each player clicks to roll the dice. Player A rolls the highest sum so Player A has the first go, clicks to roll again and moves their piece by the amount shown on the dice.
3. Player A lands on a property and the system gives the player the option to buy or auction the property as the property is unowned. The player clicks to purchase the property. The system deducts the cost of the property from the player’s wallet and deposits the money in the bank. The system places the title deed card in Player A’s inventory.
4. Player B clicks to roll the dice and lands on a property. The property is unowned so the System gives the player the option to purchase or auction the property. Player B decides to auction the property. Players A,C and D click to bid. Player A is the first to bid and enters €200 as the initial bid. Players C and D enter a bidding price greater than the current one. This continues until Players A and C click to exit the bid. Player D is the last bidder and purchases the property for €310. The System deducts the money from Player D’s wallet and deposits it in the bank, the title deed card is placed in Player D’s inventory.
5. Player C clicks to roll the dice and lands on a property. The property is owned by Player A. The System prompts player C to pay the rent specified on the title deed card owned by Player A. The rent is deducted from player C’s wallet and placed in Player A’s wallet.
6. Player D clicks to roll the dice and they land on a Community Chest square. Player D opens the card menu and selects a random card from the deck. The System turns the selected card over and the player reads the card and it says it is a ‘Get Out Of Jail Free’ card. The card is placed in Player D’s inventory.
7. Player A clicks to roll and lands on a tax square. The system deducts the money specified on the square from Player A’s wallet and places it in the bank.
8. Player B rolls the dice and lands on the ‘Go To Jail’ square. The System moves their piece to the jail square. The System gives the player the option to pay €50 to get out of jail, to roll the dice to try and get a double on their next turn or to use a ‘Get Out Of Jail Free’ card if they have one. Player B decides to pay €50. Their game piece is

moved to ‘Just Visiting’ and €50 is deducted from their wallet and placed into the bank.

9. Player C has two properties of the same grouping. The player clicks to roll the dice and lands on the third property in that grouping. The property is unowned so the player decides to purchase it. The money is deducted from their wallet and placed in the bank. Player C now has all three properties from the same grouping so can collect more rent as specified on the title deed card from other players who land on the properties. Now Player C can also access the housing menu.
10. On their next turn, Player C decides to buy a house for their property group. They click to open the house menu and click to purchase a house. They System shows the player the available places to build and Player C chooses where to place the house. The cost of the house as shown on the title deed card is deducted from their wallet and moved into the bank.
11. On Player D’s turn, the player clicks to roll the dice. They move their piece the amount specified by the dice roll and they pass Go. The System adds €200 to their wallet and deducts €200 from the bank.
12. It is Player C’s turn. They have four houses on each property and the system will not allow the player to buy another house for those properties. The System now makes hotels available to the player and Player C chooses to buy a hotel. The System shows the player the available spaces to build the hotel and Player C clicks on the space to build. The System replaces the houses on that space with a hotel. Player C is deducted the cost of the hotel as shown on the title deed card from their wallet and the money is put in the bank.
13. Player A lands on Player C’s property and does not have enough money in their wallet to pay the rent. Player A checks the value of their assets but still does not have enough to cover the debt. Player A clicks to declare bankruptcy and the System gives their mortgaged properties and ‘Get Out Of Jail Free’ cards to Player C. Player C pays 10% interest on the mortgaged properties to the bank. The System transfers Player A’s money back to the bank. The System offers Player A the option to quit the game or to watch. Player A clicks to watch.
14. Player D lands on a tax square and owes the bank €100 but Player D does not have €100 in their wallet and has no unmortgaged properties to mortgage. Player D clicks to declare bankruptcy. The System begins to auction Player D’s mortgaged property at their unmortgaged value and returns player D’s ‘Get Out Of Jail Free’ card to the bottom of the card deck. Player D is offered the option to watch or quit the game and the player quits the game.
15. It is player B’s turn and they mortgage three of the properties they own. The System puts the money in Player B’s wallet with the amount based on that stated on the title deed card. Player B can no longer collect rent on these three properties and the cards are turned over in their card deck.
16. It’s Player C’s turn and they want to do a deal with Player B. Player C wants to buy a property from Player B. Player B has two houses on this property so before Player B can sell the property they need to sell the two houses back to the bank. Once player B has done this then Player C enters their proposal to buy the property for one ‘Get Out Of Jail Free’ card, a Utility and €60. The System gives player B the option to click

Accept, Decline or Counter Offer and Player B accepts. The assets are transferred to each respective inventory and wallet.

17. It is Player B's turn and they wish to purchase a property from Player C. The property Player B wishes to buy is mortgaged and so once the two players agreed on a deal, Player B is given the option to pay the price of unmortgaging the property now or pay 10% of the list price on the card and keep the property mortgaged. Player B chooses to repay the bank the mortgaged price and the System deducts the money from Player B's wallet and puts it in the Bank.
18. Player C lands on Player B's property which has a hotel on it. Player C checks the value of their assets but cannot pay off the rent. Player C clicks to declare bankruptcy. Player B is the only player left so the game ends and Player B wins
19. A success screen appears with Player B as the winner. The game quits and returns to the main menu.

Use Case Descriptions

Use Case 1		Buying property
Goal In Context		Purchasing a property from a player or the bank
Scope and Level		System Scope, User Goal Level
Preconditions		Property is unowned, player has adequate funds to purchase property.
Success End Condition		Player purchases property.
Failed End Condition		Player does not purchase property.
Primary, **Secondary Actors**		Player Bank, Other Players
Trigger		Player lands on a property.
Description	**Step**	**Action**
	1	Player lands on a property.
	2	Player checks that the property is unowned.
	3	Player decides if they want to purchase the property.
	4	Player checks if they have adequate funds to purchase property.
	5	System takes money from players wallet and deposits the money in the bank.
	6	The player receives the title deed card.
Extensions	**Step**	**Branching Action**
	2a	If property is owned, player pays rent on property.
	3a	Player doesn't wish to purchase property so the

	property goes up for auction. **4a** Player cannot afford to purchase the property and property goes up for auction. **6a** If the player now owns all properties in a grouping, they can begin purchasing houses for those properties.
Variations	**Branching Action**
1	Player purchases property through auction or a deal with another player.

Use Case 2	Going to Jail
Goal In Context	Player gets locked in jail.
Scope and Level	System Scope, Summary Level
Preconditions	Player lands on ‘Go To Jail’ square.
Success End Condition	Player goes to jail.
Failed End Condition	Player does not go to jail.
Primary, **Secondary Actors**	Player ‘Go To Jail’ Square
Trigger	Player lands on the ‘Go To Jail’ Square.
Description **Step**	**Action**
1 **2** **3**	Player lands on ‘Go To Jail’ Square. Player navigates to the jail. Player remains in jail for three turns and pays €50 after their third turn.
Extensions **Step**	**Branching Action**
2a	Player cannot collect €200 if they pass Go.
Variations	**Branching Action**
1a **1b** **3a**	Player gets a chance or community chest card which orders them to go to jail. Player rolls three doubles in a row which means they must go to jail immediately. Player uses a ‘Get Out Of Jail Free’ card and moves into ‘Just Visiting’.

3b	Player rolls a double on their next turn and moves into 'Just Visiting'.
3c	Player pays €50 on their next turn and moves into 'Just Visiting'.

Use Case 3	Player declares bankruptcy.
Goal In Context	Player declares bankruptcy when they no longer have adequate funds to pay off debt.
Scope and Level	System Scope, User Goal Level
Preconditions	Player is in debt and does not have adequate funds.
Success End Condition	Player declares bankruptcy.
Failed End Condition	Player does not declare bankruptcy.
Primary, **Secondary Actors**	Player Other players, Bank
Trigger	Player owes money and cannot pay off the debt.
Description **Step**	**Action**
1	Player owes money.
2	Player pays some of the debt with their leftover money.
3	Player checks the value of their assets and it is not sufficient.
4	Player declares bankruptcy.
5	Player must give away their assets.
6	Player is eliminated from the game.
Extensions **Step**	**Branching Action**
1a	Player can owe money to the bank or to another player.
4a	Player declares bankruptcy when the value of their assets is not sufficient to pay off the debt.
5a	Assets include properties and 'Get Out Of Jail Free' cards.
5b	If the player is in debt to the bank, the player's properties go for auction unmortgaged and 'Get Out Of Jail Free' cards are returned to the card decks
5c	If the player is in debt to another player, the bankrupt player must give the other player all their properties and ''Get Out Of Jail Free' cards. The player receiving any mortgaged properties must pay 10% interest on them.
6a	The player can choose to quit the game or to watch the

	rest of the game.
Variations	**Branching Action**

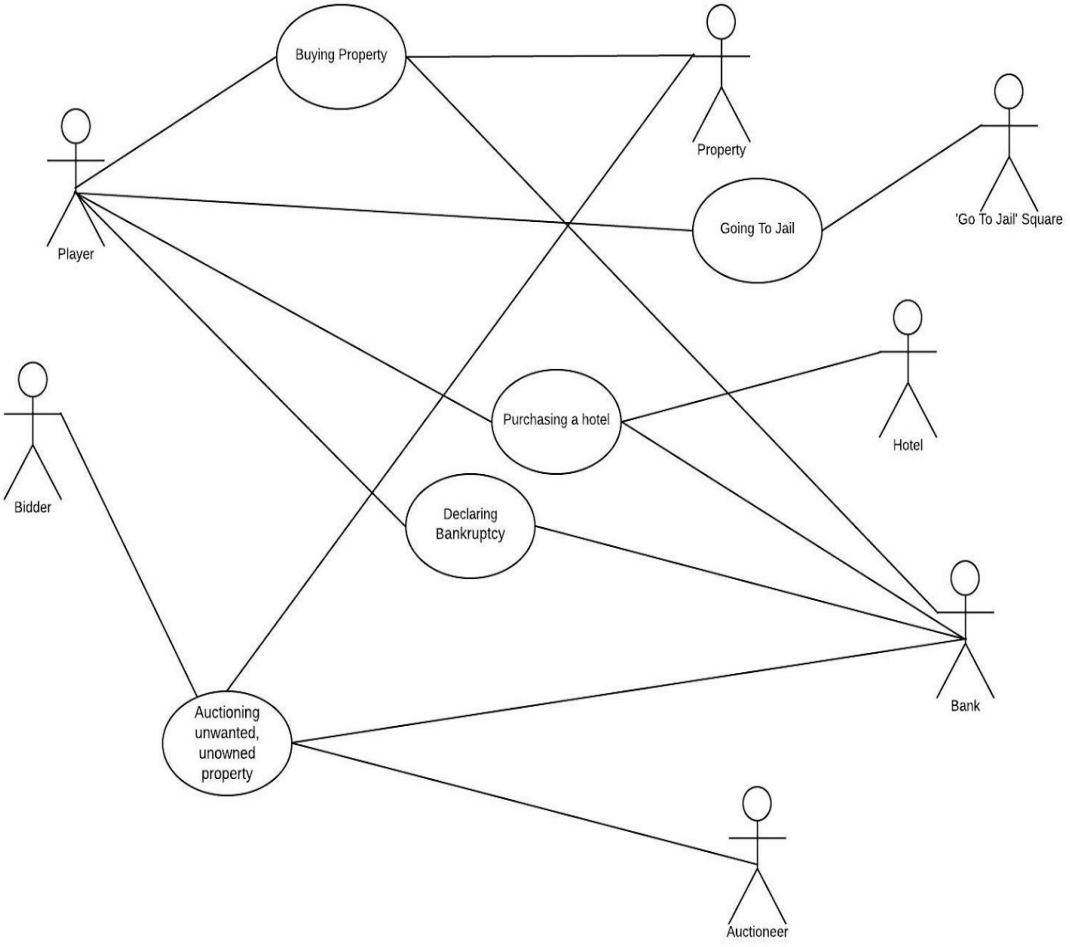
Use Case 4	Property Auction for unwanted, unowned property
Goal In Context	Unwanted, unowned property is being auctioned and the player with the highest bid can purchase the property.
Scope and Level	System Scope, Summary Level
Preconditions	Player has sufficient funds for each of their bids. A player landed on the property being auctioned. The property being auctioned is unowned.
Success End Condition	Property up for auction is purchased by highest bidder.
Failed End Condition	Property being auctioned is not purchased as no bids are made.
Primary, **Secondary Actors**	Bidders, Property Auctioneer
Trigger	Player lands on an unowned property and does not wish to purchase it.
Description **Step**	**Action**
1	Player lands on property.
2	Player checks if property is owned.
3	Player decides not to purchase property.
4	Property goes up for auction.
5	First player to bid decides the initial bidding price.
6	Other players bid on the property.
7	The highest bidder purchases the property.
Extensions **Step**	**Branching Action**
4a	The player who lands on the property cannot participate in the bidding process.
5a	The initial bidder can make the initial bid for as low as €1.
7a	The highest bidder must purchase the property at their last bid value.

Variations	Branching Action
1a	Properties can go up for auction when a player declares bankruptcy and is in debt to the bank.

Use Case 5	Purchasing a hotel
Goal In Context	Player improves on their property by building a hotel on it.
Scope and Level	System Scope, User Goal Level
Preconditions	Player has sufficient funds to purchase the hotel. The player has all the properties in the grouping they are building on. The player has four houses on the property. Property must be unmortgaged. Hotels are available.
Success End Condition	Hotel is purchased.
Failed End Condition	Hotel is not purchased.
Primary, **Secondary Actors**	Player, Hotel Bank
Trigger	It is the player's turn and they wish to build a hotel on their property.
Description **Step**	**Action**
1	Player decides to build a hotel on a property.
2	System takes money from player's wallet and deposits it in the bank.
3	The system replaces the houses on the property with a hotel.
Extensions **Step**	**Branching Action**
1a	Player must have four houses on each property of the same grouping before they can purchase a hotel.
1b	Player can only have one hotel per site.
1c	The property must be unmortgaged to build a hotel on it.

1d **2a** **3a**	If no hotels are available, the player must wait until one becomes available. The cost of the hotel is specified on the title deed card for the property. The property is now worth the amount as specified on the title deed card.
Variations	**Branching Action**
1a	Players can purchase hotels in an auction in the occasion of a building shortage.

Use Case Diagram



Primary Class List

Below is a list of our primary domain-based classes.

1. Player
2. Bank
3. Board
4. Card
5. Property
6. Game
7. Square
8. Piece
9. Die
10. Inventory

CRC Descriptions

Class Name: Player	**ID:** 1
Description: Represents a user playing the game	**Associated Use Cases:** Player joins a game session. Player buys property. Player buys buildings. Player sells property. Player sells buildings. Player bids in an auction. Player goes to jail. Player declares bankruptcy. Player does a deal. Player picks up a card. Player pays rent.
Responsibilities	**Collaborators**
Maintains data about the player	Game Piece
Keeps track of the player's inventory	Property, Card
Retains the player's location on the board	Board, Square
Attributes	
Name	
Colour	
Piece	
Inventory	
Position	
Relationships	

Generalisation (a-kind-of)	
Aggregation (has-parts)	
GamePiece	
Inventory	
Other Associations	**Collaborators**
‘Is on’	Square
‘Buys’, ‘Sells’, ‘Bids’	Property, Card, Bank
‘Joins’	Game
‘Goes to’	Jail
‘Picks Up’, ‘Keeps’	Card

Class Name: Bank	**ID:** 2
Description: Holds all the money, properties and buildings	**Associated Use Cases:** Player is allocated money. Player buys property. Player buys buildings. Player sells properties. Player sells buildings. Player mortgages property. Player unmortgages property. Property is up for Auction.
Responsibilities	**Collaborators**
Maintains money	
Maintains properties	Property, Cards
Maintains buildings	Property
Gives out money	Player
Sells property	Property, Player
Sells buildings	Property, Player

Buys property	Property, Player
Buys buildings	Property, Player
Gives out mortgages	Property, Player
Holds Auctions	Property, Player
Attributes	
Money	
Houses	
Hotels	
Properties	
Relationships	
Generalisation (a-kind-of)	
Inventory	
Aggregation (has-parts)	
Property	
Other Associations	**Collaborators**
‘Sells to’, ‘Buys from’, ‘Allocates to’, ‘Gives mortgage to’	Player
‘Sells’, ‘Buys’, ‘Auctions’	Property

Class Name: Board	**ID:** 3	
Description: Maintains information about the game board and is what players will navigate around in the game play.		**Associated Use Cases:** Player moves around the board
Responsibilities	**Collaborators**	
Maintain information about squares on the board	Square	
Maintains information about the dimensions of the board		
Attributes		
Dimensions		

Num_squares
Squares
Relationships
Generalisation (a-kind-of)
Aggregation (has-parts)
Square

Other Associations	**Collaborators**

Class Name: Card	**ID:** 4	
Description: This represents a game card and can be a chance, community chest or title deed card.		**Associated Use Cases:** Player picks up a card. Player buys a property. Player sells a property. Player does a deal. Player declares bankruptcy.

Responsibilities	**Collaborators**
Maintains information about each card	
Gives information about properties	Property
Orders a player to go to a square	Square

Attributes
Type
Description
Keep
Relationships
Generalisation (a-kind-of)
Aggregation (has-parts)

Other Associations	**Collaborators**
‘Is linked to’	Property
‘Sold to’, ‘Bought by’, ‘Picked up by’, ‘Orders’	Player
‘Is kept in’	Inventory

Class Name: Property	**ID:** 5	
Description: Represents a property that can be owned by a player.		**Associated Use Cases:** Player purchases a property. Player sells a property. Property goes up for auction. Player pays rent.
Responsibilities	**Collaborators**	
Maintains information about the cost of each property	Card	
Holds information about the property grouping		
Maintains the number of houses on the property		
Maintains the number of hotels on the property		
Holds information about the location of the property	Square	
Maintains data about the owner of the property	Player	
Attributes		
title_deed_card		
cost		
mortgage_cost		
grouping		
name		

location	
colour	
houses	
hotel	
owned	
owner	
Relationships	
Generalisation (a-kind-of)	
Aggregation (has-parts)	
Card	
Square	
Player	
Other Associations	**Collaborators**
‘Is bought by’, ‘Is sold by’, ‘Is bid on by’, ‘Collects rent on’	Player
‘Is auctioned by’	Player, Bank
‘Is landed on by’	Piece
‘Is described by’	Card

Class Name: Game	**ID:** 6
Description: This represents the actual game. It keeps the game running and keeps track of the players in the game.	**Associated Use Cases:** Player joins game session. Player wins game. Player loses game.
Responsibilities	**Collaborators**
Keeps track of players in the game	Player
Keeps the game running	
Initialises the board	Board

Initialises the cards	
Shows success screen for winner	Player
Shows the player menu	
Allows players to choose their game pieces and colours	Game Piece, Player
Ends the game	
Eliminates players from the game	Player
Determines the player who moves first	Player, Die
Determines the order of player turns	Player
Attributes	
Players	
Winner	
Running	
Relationships	
Generalisation (a-kind-of)	
Aggregation (has-parts)	
Player	
Other Associations	**Collaborators**
'Initialises'	Board, Card, Bank

Class Name: Square	**ID:** 7	
Description: Is a space on the board that can hold properties, jail, 'Go', tax, utilities, cards and stations.		**Associated Use Cases:** Player lands on a property. Player lands on a card space. Player passes Go. Player goes to Jail. Player lands on tax square.
Responsibilities	**Collaborators**	
Holds information about each space	Card, Property, Board	

Attributes	
Location	
Type	
Name	
Relationships	
Generalisation (a-kind-of)	
Aggregation (has-parts)	
Property	
Other Associations	**Collaborators**
‘Landed on by’	GamePiece
‘Is part of’	Board

Class Name: GamePiece	**ID:** 8	
Description: Represents a game piece which the player uses to navigate around the board		**Associated Use Cases:** Player makes a move Player lands on square
Responsibilities	**Collaborators**	
Keeps track of player’s location on the board	Player, Board, Square	
Attributes		
Name		
Position		
Image		
Relationships		
Generalisation (a-kind-of)		
Aggregation (has-parts)		

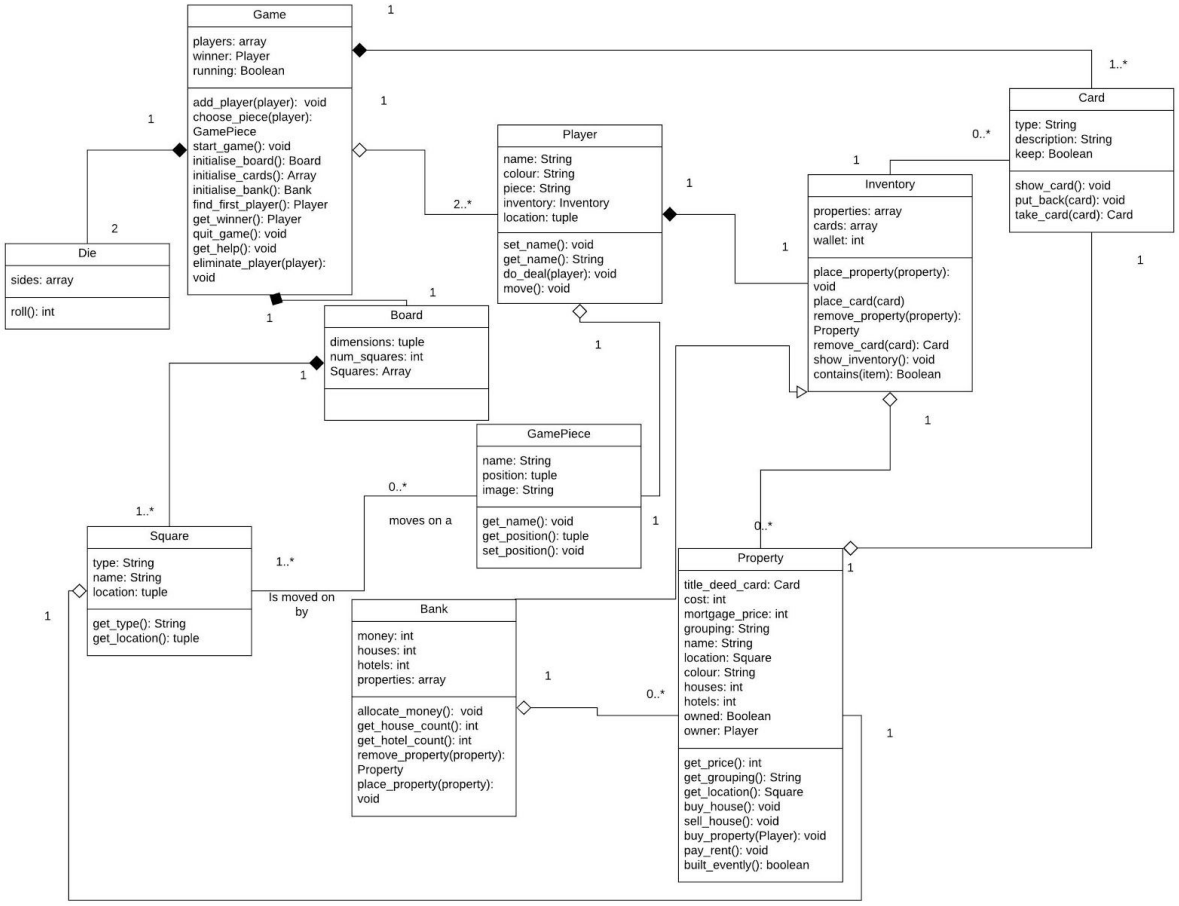
Other Associations	**Collaborators**
‘Is moved by’	Player, Die
‘Moves around’	Board
‘Lands on’	Square

Class Name: Die	**ID:** 9
Description: Represents a die that is used to determine the number of spaces the player navigates around the board.	**Associated Use Cases:** Player rolls the dice
Responsibilities	**Collaborators**
Determines the amount of spaces a player moves around the board	Player, Square
Attributes	
sides	
Relationships	
Generalisation (a-kind-of)	
Aggregation (has-parts)	
Other Associations	**Collaborators**
‘Is rolled by’	Player

Class Name: Inventory	**ID:** 10
Description: Is a collection of properties, cards and the wallet that a player owns.	**Associated Use Cases:** Player purchases property. Player sells property. Player gets a ‘Get Out Of Jail’ card. Player does a deal. Player pays rent. Player declares bankruptcy.
Responsibilities	**Collaborators**

Holds a player's property, 'Get Out Of Jail Free' cards and wallet	Player, Property, Card
Maintains a player's money	Player
Attributes	
Property	
'Get Out Of Jail Free' cards	
Wallet	
Relationships	
Generalisation (a-kind-of)	
Aggregation (has-parts)	
Property	
Card	
Other Associations	**Collaborators**
'Owned by'	Player
'Items are placed in by'	Bank

Class Diagram



Structured Walkthrough

The Monopoly Game is an online game allowing 2-8 users to play. The Game Class will show a main menu allowing players to join a game session. The Game Class also keeps track of the players in the game. The Bank Class, Board Class and Card Class are initialised by the Game Class once the players have joined. An instance of the Player Class is created with each user that joins the game. This in turn initialises a player's inventory which is an attribute of the Player Class and is handled by the Inventory Class. Players are given the option to choose a game piece which is represented by the Game Piece Class and also a colour and a name which are stored as attributes of the Player Class. The player uses the game piece to move around the board. The Bank Class allocates €1500 to each player which is stored in the player's wallet in the Inventory. Each player then has the option to roll the dice which are managed by the Die Class. The player who rolls the highest sum gets to go first. The order of the turns from there is clockwise based on the order in which the player's joined. On each turn the player rolls the dice and the Player Class moves their game piece by the sum of the two die.

After rolling the dice, a player will land on a square which is represented by the Square Class. A square of type 'Property' is managed by the Property Class. The Player Class will check if the property is owned by checking if the “owned” attribute is True or False. If the property is owned, the player is obliged to pay rent on the property; the Property Class will deduct the rent from the player’s wallet and transfer it to the wallet of the owner of the property. The player can attempt to buy it from the player who owns the property by making a deal through the Player Class. A player clicks to do a deal and selects the player they wish to do the deal with. The player can then choose from the other player’s inventory what they would like to purchase and choose from their own what they will exchange for it. The other player will be given the option by the Game Class to accept, refuse or counter the offer. If the offer is accepted, the Bank Class will transfer the items to the respective inventories. If the offer is rejected, the player is not allowed to continue the deal and the Game Class will enforce that they continue with their turn. In the case of a counter offer, the other player is permitted to choose the items of the deal from their and the other player’s inventory. The player can then accept, reject or counter the deal. This loop will continue from within the Player Class until a deal is rejected or accepted.

If the “owned” attribute is False, the player can decide if they want to buy the property from the bank, the money will be taken from the player’s wallet and deposited in the bank. The Title Deed card will be placed in the player’s inventory. Once the player owns all properties in a colour grouping, the Game Class will allow the player to access the housing menu and can start purchasing houses which are managed by the Bank Class. The houses can be bought for a property from the bank through the Property Class. This class will also ensure the houses are built evenly. The class keeps count of the number of houses a player has on each property. When a player has four houses on each property in a grouping, the Game Class will permit the player to have access to the hotel option. The Property Class will allow the player to purchase one hotel for each property in the grouping from the bank.

Another branch of this case is when a property goes up for auction. If a player does not have adequate funds in their wallet to pay for the property or they choose not to buy it, then the property is auctioned by the Bank Class. The player can now choose the price for the initial bid and the other players make bids for the property. The Bank Class will ensure the bids are all greater than the previous bid and that the player has the funds to make the bid. At any time during the bid, a player can decide to exit the bid until there is one player left in the auction. The highest bidder must purchase the property.

A player may land on many other different types of squares which are all instances of the Square Class. If the player lands on a Community Chess square or a Chance square, they must pick up a card which is part of the Card Class. The Game Class will open the card menu and the player will select a random card. The Game Class will turn the selected card over so the player can read the card and follow the instructions. For example, the player may be instructed to move to a certain square in which case the Player Class will move their piece to that square. If the player passes the “Go!” square during this process, they are permitted to collect €200 from the bank. This is added to the player’s wallet via the Bank Class. The player may also land on a tax square. If this use case occurs, the money

specified on the tax square is deducted from the players wallet and deposited in the bank by the Bank Class.

The player may also land on the “Go To Jail” square, which is an instance of the Square Class. Subsequently, the game piece will be moved to the jail square by the player Class and it will remain there for the next three turns. In this circumstance, the extension is that the player cannot collect €200 from the bank if they pass the ‘Go’ instance of Square. Another variation of ending up in jail is if the player gets a chance or community chest card managed by the Card Class which orders them to go to jail and the Player Class will move the game piece to the jail square. The player could also roll three doubles in a row which would cause them to be moved to jail immediately. The failed end condition is that the player does not end up in jail and this can happen in many different ways, if the player has a ‘Get Out Of Jail Free’ card in their inventory, this allows the player to move to the ‘Just Visiting’ square. The card will be removed from the inventory and placed back in the card deck by the Square Class. The player could also roll a double on their next turn and move to “Just Visiting”. The player can pay €50 to the bank on their next turn and move into “Just Visiting”.

If there is a scarcity of buildings in the bank and a player wishes to buy one, the player must wait until other players sell their buildings back to the bank. The Property Class is responsible for selling the buildings back to the Bank Class. If there are limited buildings in the Bank and more than one player wishes to purchase a building, then the Bank Class auctions the building to the highest bidder. The bank gives the player the available options of where to place it and the player picks the property to build on. The Bank deducts the money from the player’s wallet and places the money in the bank.

When a player is running out of money, they have the option to mortgage property. If the player has buildings on the property, they will need to sell these back to the bank. The Property Class will remove the houses from the property and the Bank will deposit the money in the player’s wallet. The houses will be returned to the Bank. When the player clicks to mortgage the property, a function from the Property Class is called and this changes the state of the property from ‘active’ to ‘mortgaged’. The function tells the Bank Class to deposit the loan in the player’s wallet. The card will be turned over and be displayed as red in the player’s inventory. The Property Class will no longer enable the player to receive rent on this property until it is unmortgaged. When a player clicks to unmortgage a property, the Property Class will take the mortgage value plus 10% interest off the property. The Title Deed Card will turn back over and return to it’s normal colour. The Property Class will now allow the player to receive rent on the property.

If a player is in debt then the player declares bankruptcy through the Player Class. This means the player gets eliminated from the game which is managed by the Game Class. This could happen if the following instances take place. A player declares bankruptcy when they are in debt and do not have the adequate funds. The player can owe money to the bank or to another player. The player pays whatever they can with their leftover money. The player then checks the value of their assets in their inventory. If the value of their assets are not sufficient to pay off the debt, the player can declare bankruptcy through the Player Class. The player must then give away their assets in their inventory which include their properties

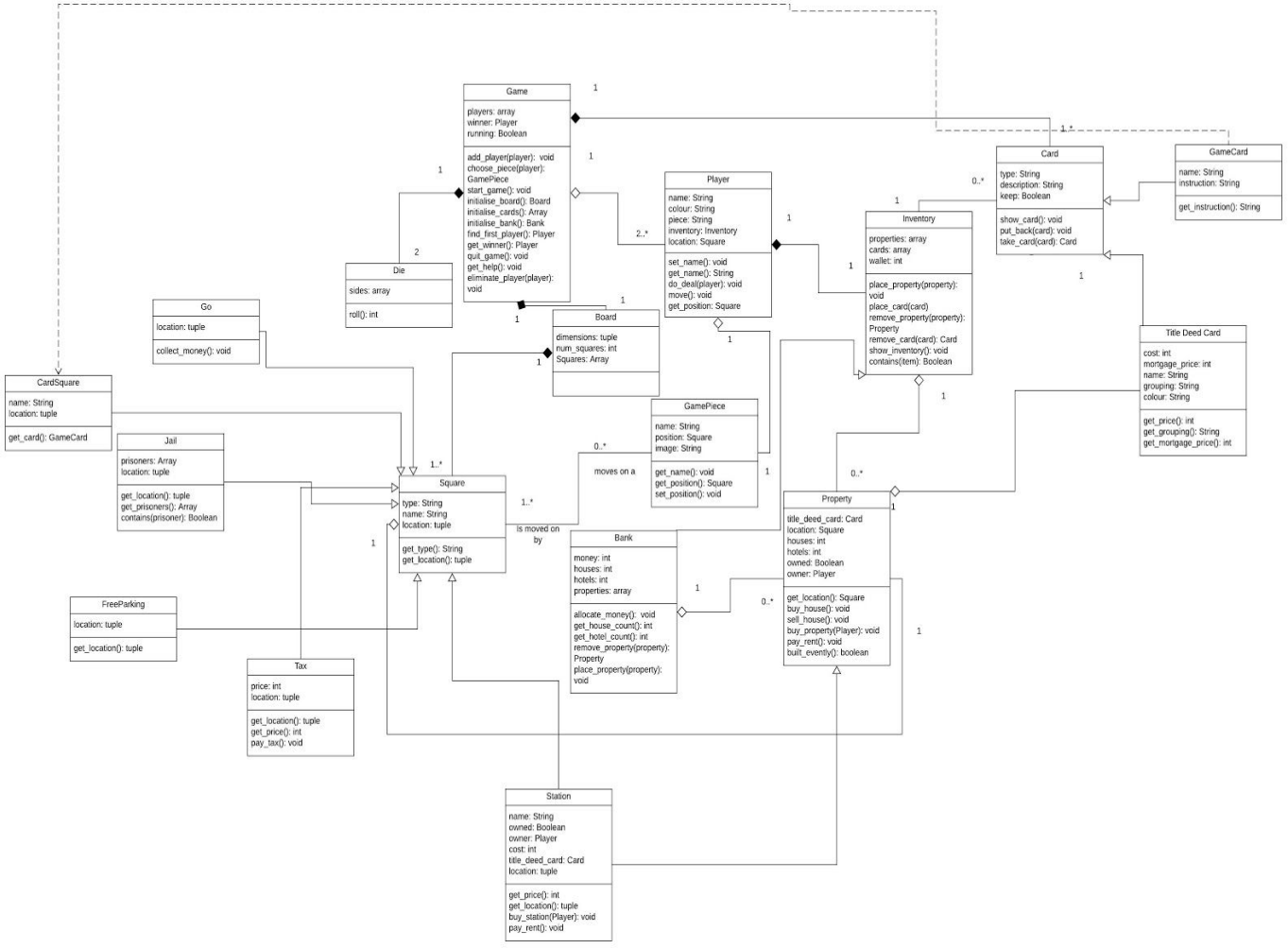
and “Get Out Of Jail Free” cards. If the player is in debt to the bank, the Bank Class puts the properties up for auction and returns the ‘Get Out Of Jail Free’ card to the card decks. If the player is in debt to another player, the Bank Class transfers the bankrupt player’s properties and ‘Get Out Of Jail Free’ cards to the other player’s inventory. The player receiving any mortgaged properties must pay 10% interest to the bank. The player is then eliminated from the game through the Game Class and this Class gives the player the option to quit the game or stay and watch the rest of the game. The Game Class keeps track of the players left in the game and when only one player is left active in the game, the Game Class assigns this last player as the winner and it will then show a success screen. This ends the game session and the game will quit back to the main menu.

As a result of the structured walkthrough, we have discovered some new classes and functions that we need to add in order to complete the flows. We realised that the Card Class encompasses too many different types of cards to be manageable, so we decided that the Card Class will have two children; Title Deed Card and the Game Card (which includes the chance and community chest cards). We are going to break down the Square Class to have children Classes for Jail, Tax, Free Parking, Utilities, Stations, Card Square and Go as there are too many responsibilities for this one Class. We also realised we need to figure out what Classes will handle the auctioning, to check if a user wishes to purchase a property and to check if a player has all properties in the same grouping.

Product Design

The second stage of the Project is the Product Design. This includes the UI designs, refinements on the classes, object and sequence diagrams. We will also be performing client server tests.

Refined Class Diagram



User Interface MockUps

Blah blah blah

Client Server Experiments

Blah blah blah

State Machines

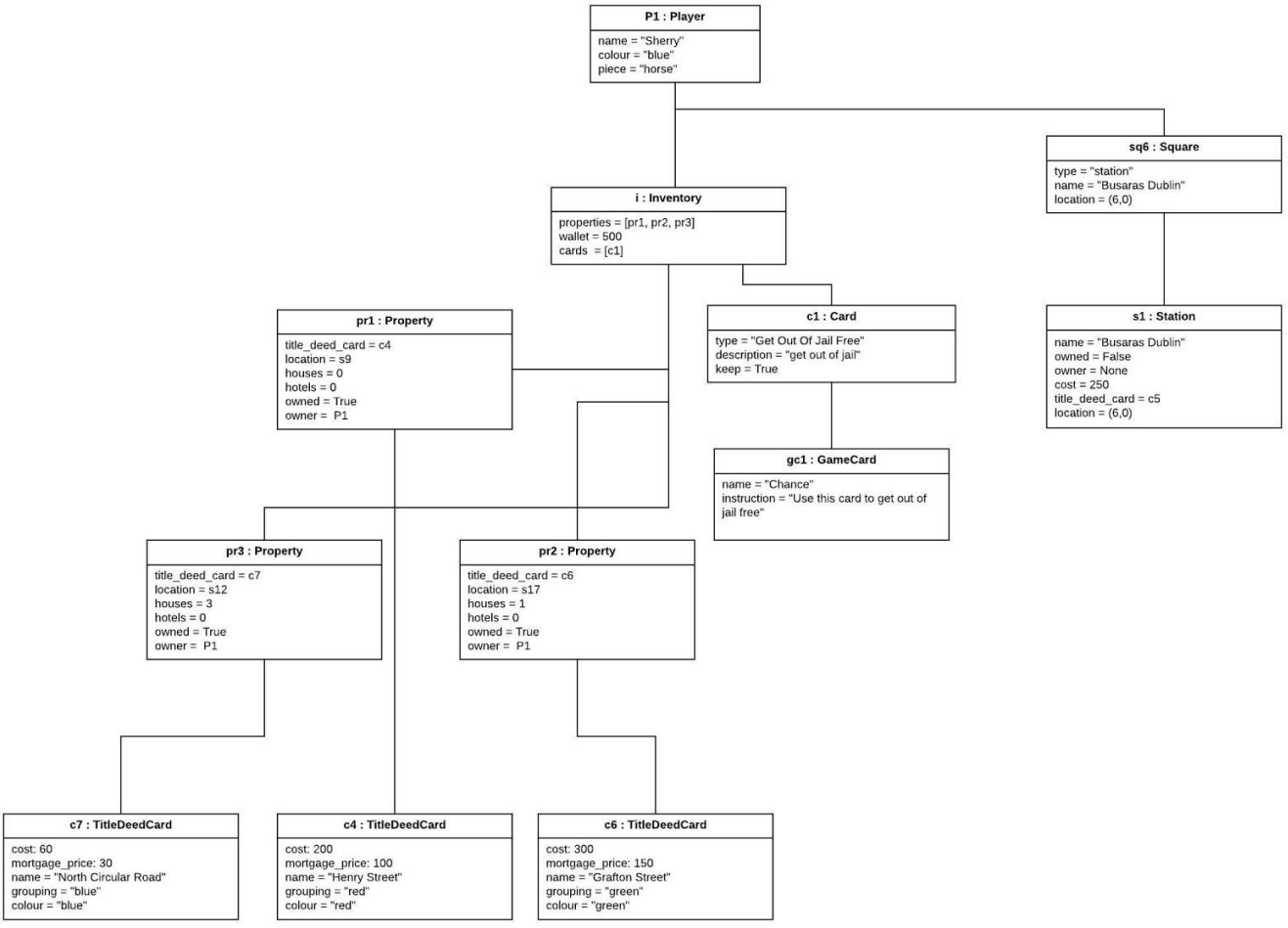
Blah blah blah

Sequence Diagrams

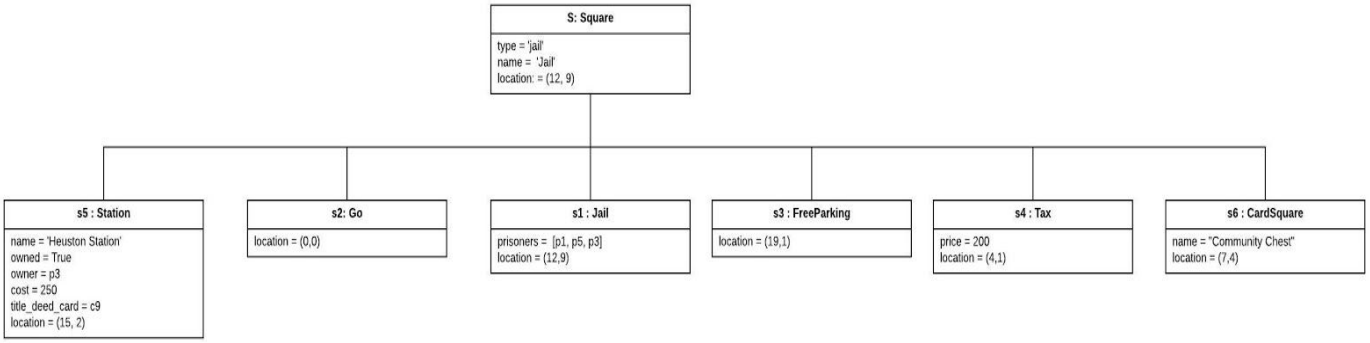
Blah blah blah

Object Diagrams

Player:



Square:



Communication Diagrams

Blah blah blah

Revised Object Diagrams

Blah blah blah

More Refined Class Diagrams

Blah blah blah

Class Skeletons

Blah blah blah

Appendix

Notes of Team Meetings

Meeting 1

Friday 27th September
Attendees: Aifric, Comfort, Jabeen, Rachel, Aine
Topic: Organisation and Method
Minute Taker: Aifric
Leader: Comfort
Deputy: Jabeen

- First group meeting. We set 20 minutes for this initial meeting. Aifric took notes.
- Discussed how we would plan the project and what tools we would use
- Decided we would use Github. Reason: We can create private repositories on it. There is a project management feature which will allow us to track our progress and we can set product boards to automate with each issue status. It allows for collaboration between different people. Enables us to keep all our files in one place.
- Concluded we would create a project board for each section/milestone of the assignment. Eg. Analysis, Product Design, Implementation. We will then open issues on each task within this. This method is suitable because we can create checklists within a task to break a task down (divide and conquer) and we would have to complete each subtask before we can set the status of an issue as 'Done'. It also allows us to see what we have to do, what is in progress and what is done very clearly so we can analyse our own methods and see what is working. This will help us analyse our own methods for sprint planning.
- We discussed how we will plan sprints with Github. We were aware github doesn't have any sprint planning feature but a service like jira will cost us €10 a month when one of us already has premium github. We decided we would create a label for each issue with a story point. We have decided on 2 week sprints as this will divide the semester up nicely. At the end of a sprint we will put the story points of the completed issues into an excel sheet and calculate how much we have completed. We will use milestones to mark our sprints and also to manage our deadlines. So anything that is not in the milestone Sprint A (for example) will be in the backlog and we can filter issues this way.
- Our methods of communication will be: Weekly meetings, Sprint planning meetings, sprint retro meetings and also an online group chat.

Meeting 2

3rd October
Attendees: Aifric, Comfort, Rachel, Aine
Topic: Sprint Planning and Requirements
Spec Minute Taker: Rachel
Leader: Aine
Deputy: Aifric

- We planned our first sprint and decided on sprint length. Our sprint lengths will be 2 weeks.
- Decided on the game play and created a list of all required pieces/cards.
- Decided on the format of the requirements specification. The decision was between numbered list format or table format. We chose to use the numbered list format.

Meeting 3

October 14th
Attendees: Aifric, Comfort, Jabeen, Rachel,
Aine Topic: Scenarios and class lists
Minute Taker: Comfort
Leader: Jabeen
Deputy: Rachel

- Made sure we were aware of the maximum amount of work we need to put into each section.
- We made a list of possible use cases.
- We looked at the burn down charts and use case template and decided to do the scenarios first and complete the use case diagram another day.
- We looked at our use cases and listed the possible scenarios that matches it.
- We looked at different factors that contributed to our scenario lists, factors such as housing, bank, rules, etc.
- We decided to use the official rules instead of creating our own house rules or using house rules.
- We talked about the various classes and discussed their attributes.
- We played the online monopoly game to refresh our memory and give us a solid foundation on the implementation of the game.

Meeting 4

October 17th
Attendees: Aifric, Comfort, Jabeen, Rachel,

Aine Topic: CRC descriptions and Use Case
Diagram
Minute Taker: Comfort
Leader: Aifric
Deputy: Aine

- We researched the CRC descriptions to find out what was needed for them.
- We completed the Use Case Descriptions.
- We decided we would use LucidChart for the diagrams as this has the UML notation and it is also online so we can all contribute.
- We created the Use Case Diagram.
- We also started on the CRC descriptions but decided we need to do some more research to fill in some of the fields and look at the relationships of the classes in more detail.

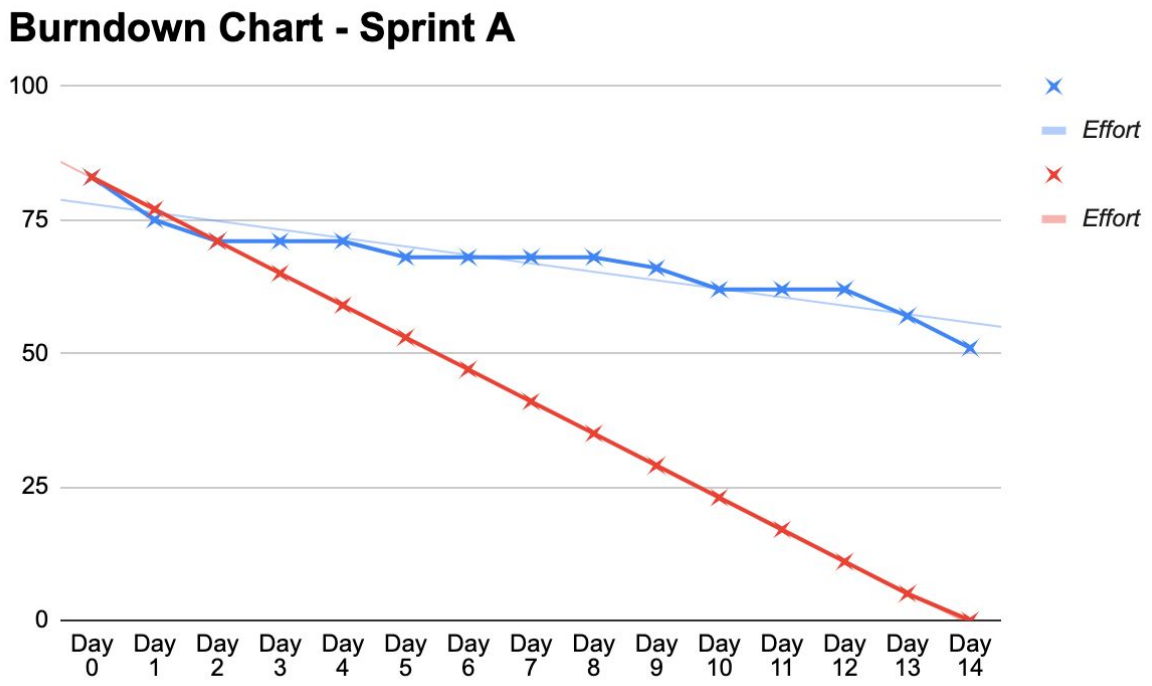
Meeting 5

October 18th
Attendees: Aifric, Comfort, Jabeen, Rachel,
Aine Topic: Sprint A Retrospective
Minute Taker: Aine
Leader: Aifric
Deputy: Comfort

- We reviewed our sprint by looking at the
  burn down chart.
- We overestimated how much time it would take to complete some tasks
- We also overestimated how much time the team could spend on the project on an average day which prevented the expected linear decrease in the chart.
- We did not complete two of the issues scheduled to be completed by the end of the sprint.
- Some good insights: We estimated the issues in the sprint would take 83 hours to do and we spent 40 hours on these issues with only one issue left to complete. This is nearly half of what we expected so we will take this into account for the next sprint. We also tend to work in batches of every three days, working a lot on the third day and none at all on the other, This will go into our sprint planning for Sprint B.
- We realised that a lot of time is spent on researching the requirements of the issue rather than solving the issue so we need to allow time for this in the sprint.
- We liked the clarity and frequency of our communication.
- Group chat is very effective for keeping everyone in the loop.
- Github Project Boards are useful for keeping track of the issues we have to do.
- We want to keep up the regular meetings.
- We want to start delegating tasks better.
- We will add the two issues we did not complete into the next sprint.
- Overall, was a successful sprint with more things we want to start doing than stop

doing.

- We also planned our next sprint (Sprint B) which will start on the 21st October. It has a velocity of 17 and the goal of it is to complete the Product Design project board. We created a burndown chart to track our progress and a table to input our hours.
- We used the miscalculation of time tasks will take to estimate these new tasks.



Meeting 6

October 21st
Attendees: Aifric, Comfort, Jabeen, Rachel,
Aine Topic: Class Diagrams
Minute Taker: Aifric
Leader: Rachel
Deputy: Jabeen

- We started on the class diagrams.
- We conducted some research into the different kinds of relationships classes can have and how they interact with each other.
- We debated on some of the relationships, especially with composition vs aggregation.
- We had a difficult time keeping the classes down to 10 as the more we went through the class diagrams, the more classes became apparent to us.
- We looked at some of the relationships and were confused over dependency relationships vs. aggregation relationships between classes. We don't think we have enough information on the system yet to be able to classify a relationship as dependant and so decided we would include this in the refined class diagram.

- We also researched the structured walkthrough and tried to get an understanding of what was expected.
- We also assigned out tasks to each member of the team for the remaining few and decided Aifric will finish the class diagram, Comfort will help fill in the CRC diagrams and keep them up to date with any changes in the class diagram and Jabeen will start on the structured walkthrough.
- We concluded that we would meet again on the 23rd October to look over and edit the structured walkthrough together as well as to proof read all our work and make edits if necessary.
- We realise the difficulty will be ensuring everything is consistent as we have made a few changes over the course of the work as we learned about new aspects of the product.

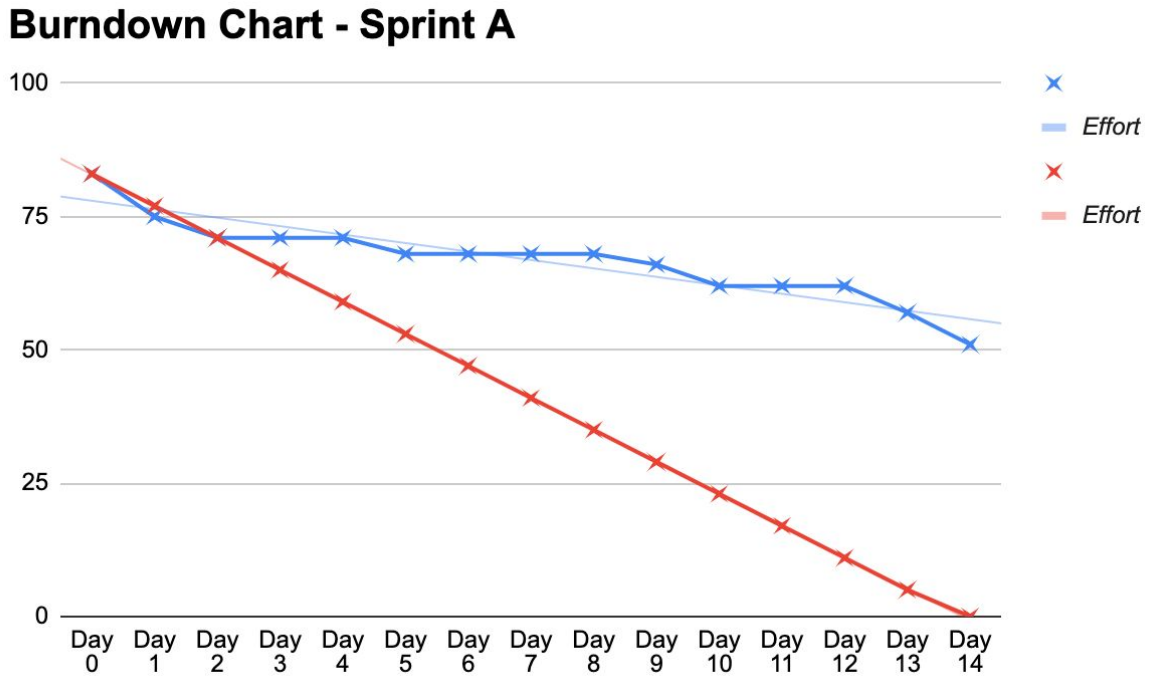
Meeting 7

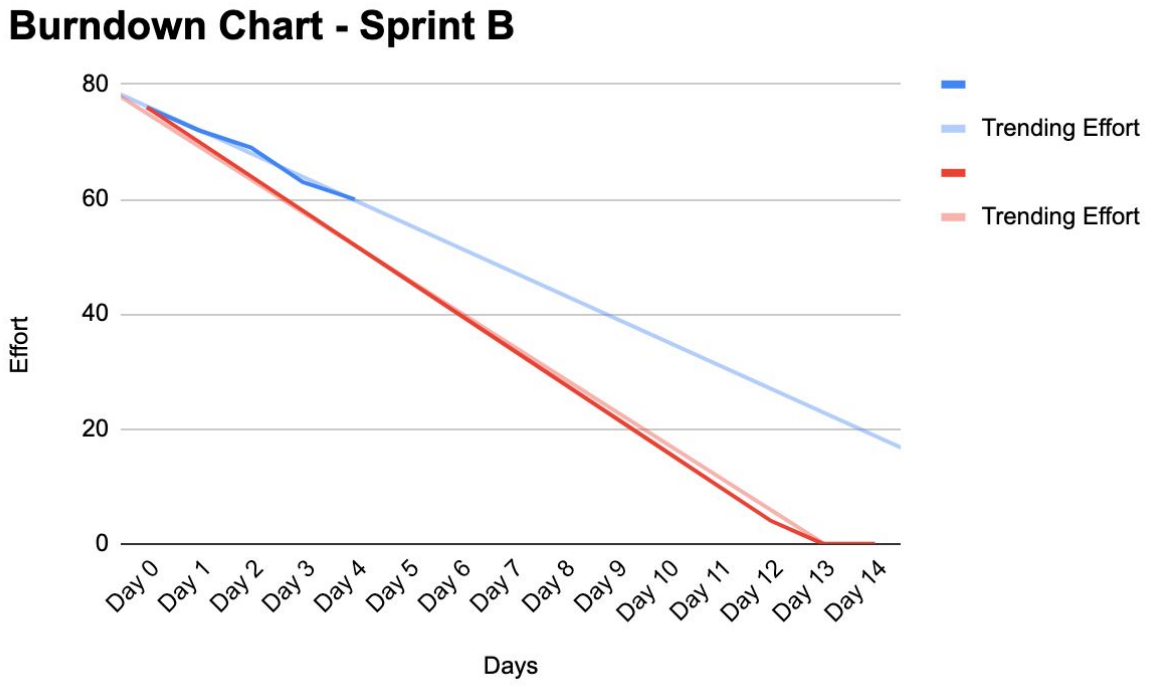
October 23rd
Attendees: Aifric, Comfort, Jabeen, Rachel,
Aine Topic: Structured Walkthrough
Minute Taker: Aifric
Leader: Jabeen
Deputy: Comfort

- We looked at the completed class diagrams and discussed if any edits needed to be made.
- We looked at the draft of the structured walkthrough and decided we needed to make it more technical
- We realised there are a few broken flows in our game and some extra responsibilities may need to be added to the classes.
- We looked over the CRC and class diagrams to make sure they are consistent.

Sprint Burndowns

Colour	**Meaning**
Blue	Actual Effort
Red	Estimated Effort





References

Class Relationships: https://en.wikipedia.org/wiki/Class_diagram#Relationships
Class Diagrams: https://creately.com/blog/diagrams/class-diagram-relationships/
Diagram Tool: www.lucidchart.com
Use Case Diagram: https://www.tutorialspoint.com/uml/uml_use_case_diagram.htm

Scope and Level:
https://www.oracle.com/technetwork/testcontent/gettingstartedwithusecasemodeling-133857.pdf, pg. 8. https://www.infor.uva.es/~mlaguna/is1/materiales/BookDraft1.pdf