# Req

This project is about making a virtual board game, similar to the popular game Monopoly. The focus is on the steps of figuring out what the game needs, creating a plan, making the game, and testing it to make sure it works properly. The plan will be shown using a type of picture language called UML. The game will have a unique theme based on the world of computers and information technology, instead of buying properties and charging rent, players will control different areas of technology and allocate resources. The game will be played using a simple text interface through a program like Eclipse, rather than having a fancy graphical interface. The game will communicate with players using English text

A use case is a set of steps that an actor takes to achieve a desired outcome. Each use case is represented by an ellipse in a diagram. The ellipse contains the name of the use case and the corresponding description explains the normal flow of actions and alternative flows under certain circumstances. Several use cases can appear in a single use case diagram and each use case must have a description. The relationships between use cases in the diagram can be <<extend>> or <<include>> and they show how one use case is affected by another as actions take place. The use case description also mentions the preconditions needed for a use case to occur. A game like monapoly app starts with a set of use cases and unfolds as the actor takes actions to achieve the desired outcomes.

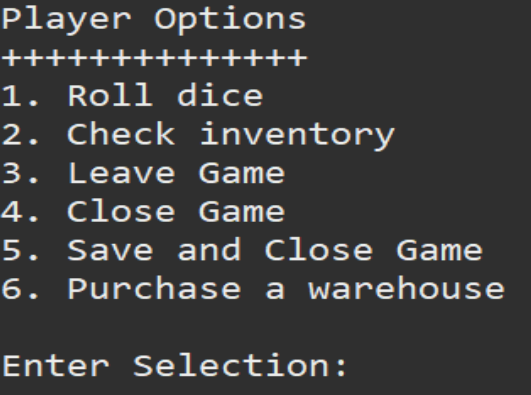
monapoly app is a board game that involves up to 8 players who take turns rolling virtual dice and landing on squares. The players have to enter their names. They are then told what to do or what their obligations are when they land on a square. If a player's resources change, the system shows the reason and announces their new balance. There is a start square where players pick up resources and other squares where things happen. The game has fields, which are like color groups in Monopoly, and they are made up of different areas. Players must own a whole field before they can develop an area within it. To develop an area, players build houses, and when they have three houses, they can build a hotel. If a player lands on an area they don't own, they have to give up resources. If a player runs out of resources or stops playing, the game ends, and the final state of play is shown. The outcome of the game depends on how it is set up.

# Games description

In the game, all players start on the "Start of Quarter" square at the bottom right of the board, similar to "GO" in Monopoly. Players take turns rolling the dice and move the number they roll. They can land on squares for products or events. When a player lands on an unowned product square, they can buy it. If they land on a "Product Bidding" square, their products are put up for auction and other players can bid to buy them. If someone else lands on a player's owned square, they have to pay to use the product. Players can build warehouses to increase the price others have to pay to use the product. When a player has warehouses on all squares of a field, they can select a region to expand their business, but it will cost them. When a player passes the "Start of Quarter" square 4 times, the income event happens and if their income is too low, they are eliminated. There are "Stock Exchange" squares where players can take a gamble to either double or lose money. There's also a "Tax" square which charges the player 10% of their value and a "Bank Holiday" square which does nothing. Each product has a unique value that changes when upgraded. The last player left is the winner.

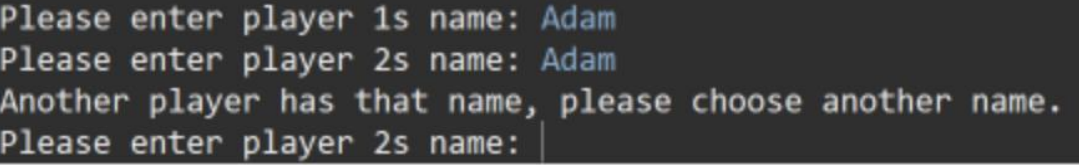
User interface

The player options menu is easy to use and makes the game experience better. The menu box is clearly written so that everyone can understand it. Players choose what they want by entering a number. Some options may not be available if the player hasn't met certain requirements. For example, you can only buy a warehouse if you own all the products of a company. This makes sure players can't cheat. The same goes for requesting a warehouse.

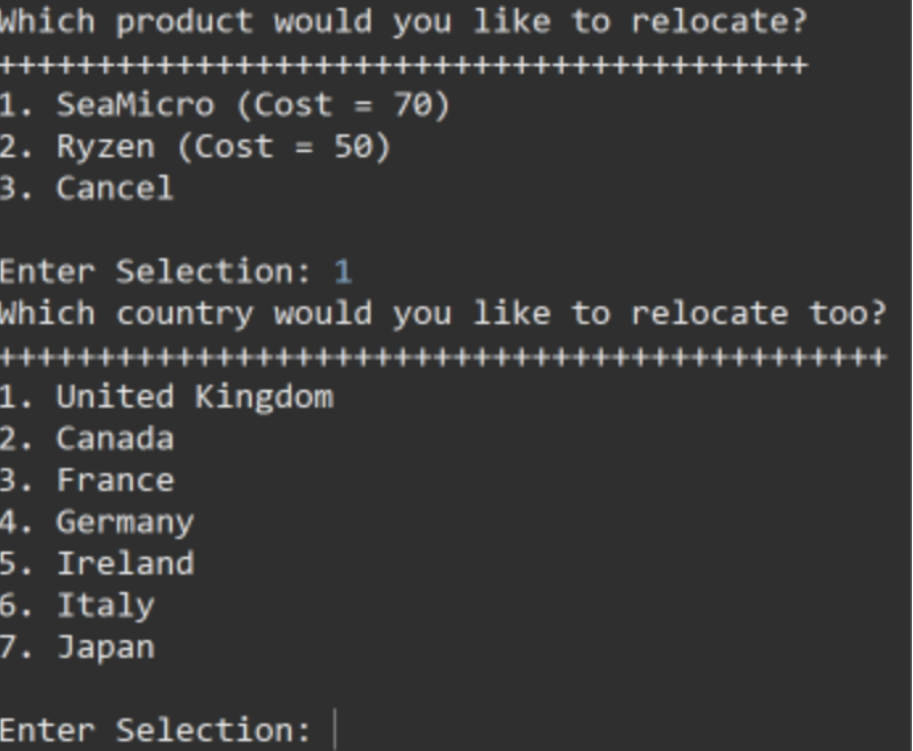


Player name

The player names section allows players to easily enter their names. It's simple and straightforward, making it easy for players to understand. The names are checked to make sure no two players have the same name to avoid confusion. If a name is already taken, the player will have to enter a different name to move on with the game.



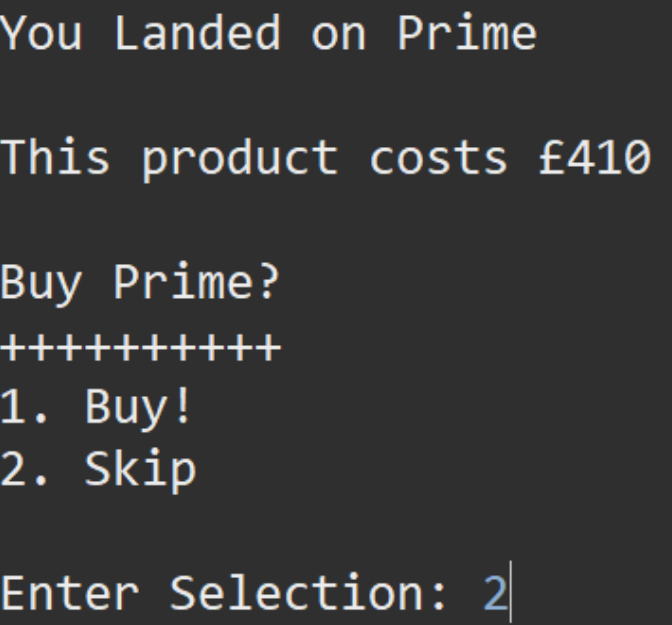
Purchase a country



The "Purchase a country" menu is simple and easy for players to use. They just pick a number for the option they want and the game moves forward. Only products the player owns will be shown at the top of the menu. This is to make sure the player doesn't accidentally break the game by choosing the wrong product. There's also a validation check to make sure the player can't pick the same country twice within the same company. If they already bought a country, like "France", that option will be removed from the list for other products in the company. This is to prevent the player from using the same country again.

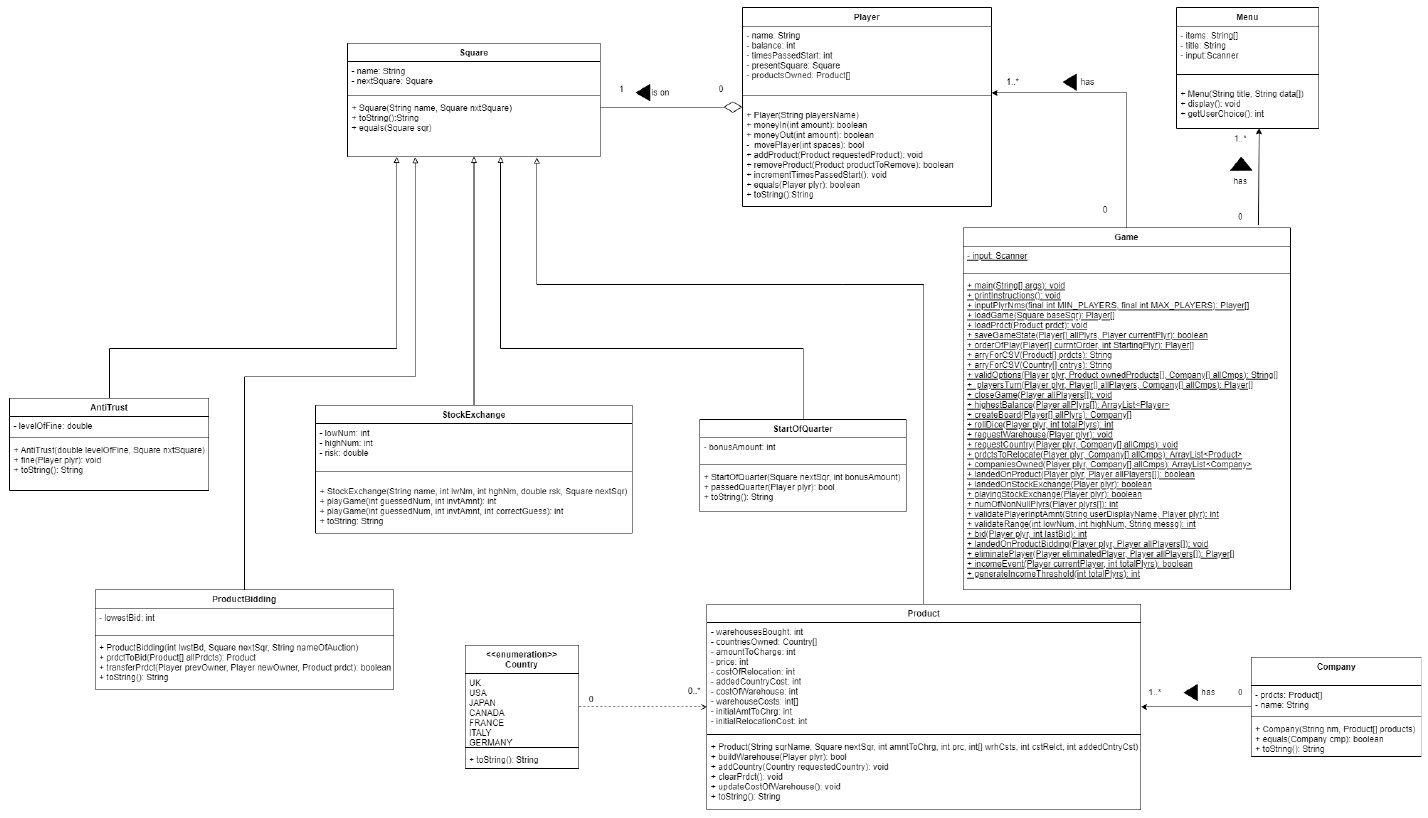
Buying a product

In simple words: When buying products, the layout is simple and easy to understand. The system shows the price of the product and lets the player choose if they want to buy it or not. If the player doesn't have enough money, they get an error message. This is because of validation, which helps the game work properly and stops the player from making mistakes that could end the game. To make the game more challenging, the details of the product are hidden.



Design Documentation

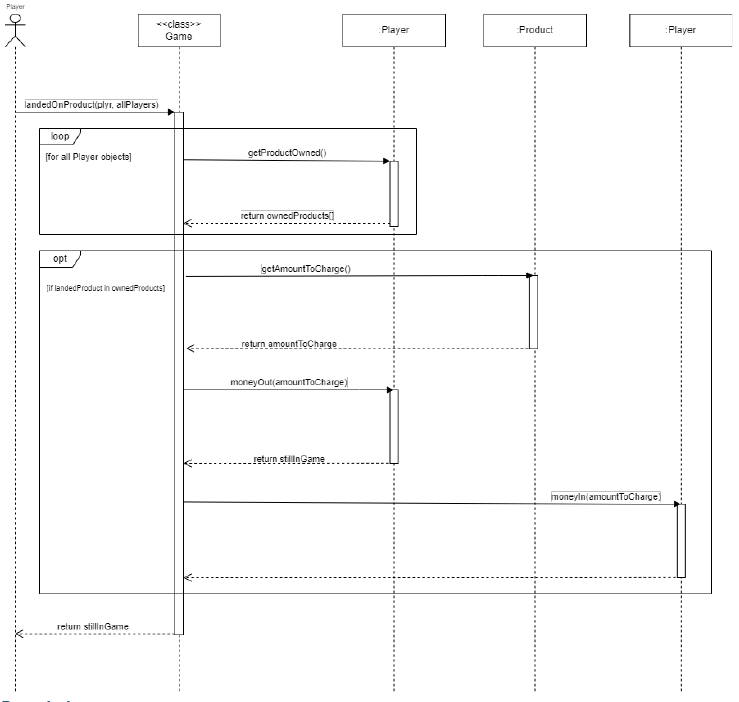
The diagram shows the different classes in the system and how they connect with each other. The classes have both things that get information (getters) and things that set information (setters), but they are not shown in the diagram. The main way the player interacts with the game is through the Game class and the Menu class makes it easy to create a menu for the player. The Menu class was taken from a class the team took in level 1. The diagram changed from the first report because the "Company" section is now its own class, which makes it easier to check if the player owns all the products in that company. The diagram also includes new information and changes that have been added during the project. The relationships between the classes have not changed.



User case

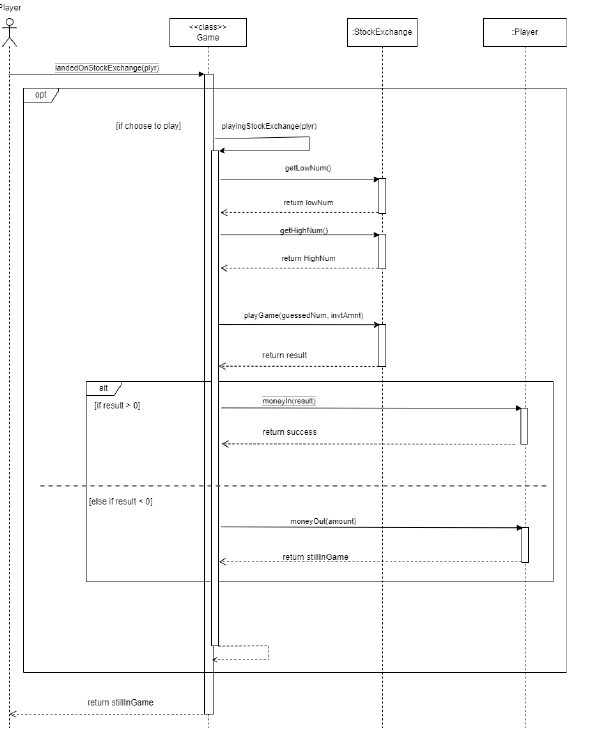
Sequence

In simple words, this is a diagram that shows what happens when a player lands on a product in the game that is owned by another player. The game checks if the player has enough money to pay for using the product. If the player does, the payment is made and the player who owns the product gets the money. If the player does not have enough money, it returns false.

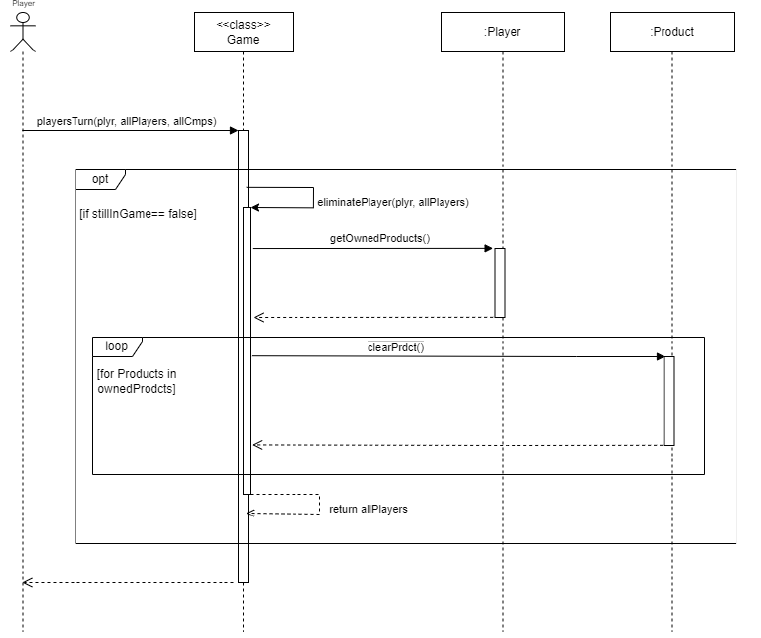


Stock exchange

This diagram shows how the game "Choice to Play Game," "Stock Exchange Game," "Money In," "Lost Stock Exchange," and "Money Out" work. It was in the first report, but has been updated. The "Player Eliminated" use case was removed because it got complicated. A new method called "playingStockExchange()" was added to make the code easier to read and maintain. The conditions in the if and else if statements changed and the calls to "moneyIn()" and "moneyOut()" were moved to "playingStockExchange()".

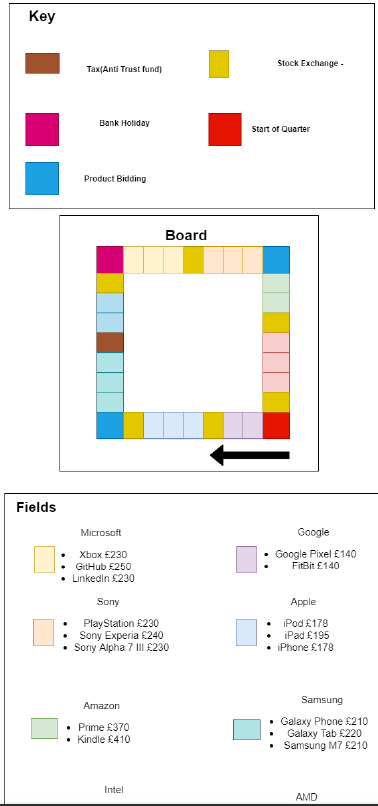


Eliminate Player



This diagram shows how the game "Choice to Play Game," "Stock Exchange Game," "Money In," "Lost Stock Exchange," and "Money Out" work. It was in the first report, but has been updated. The "Player Eliminated" use case was removed because it got complicated. A new method called "playingStockExchange()" was added to make the code easier to read and maintain. The conditions in the if and else if statements changed and the calls to "moneyIn()" and "moneyOut()" were moved to "playingStockExchange()".

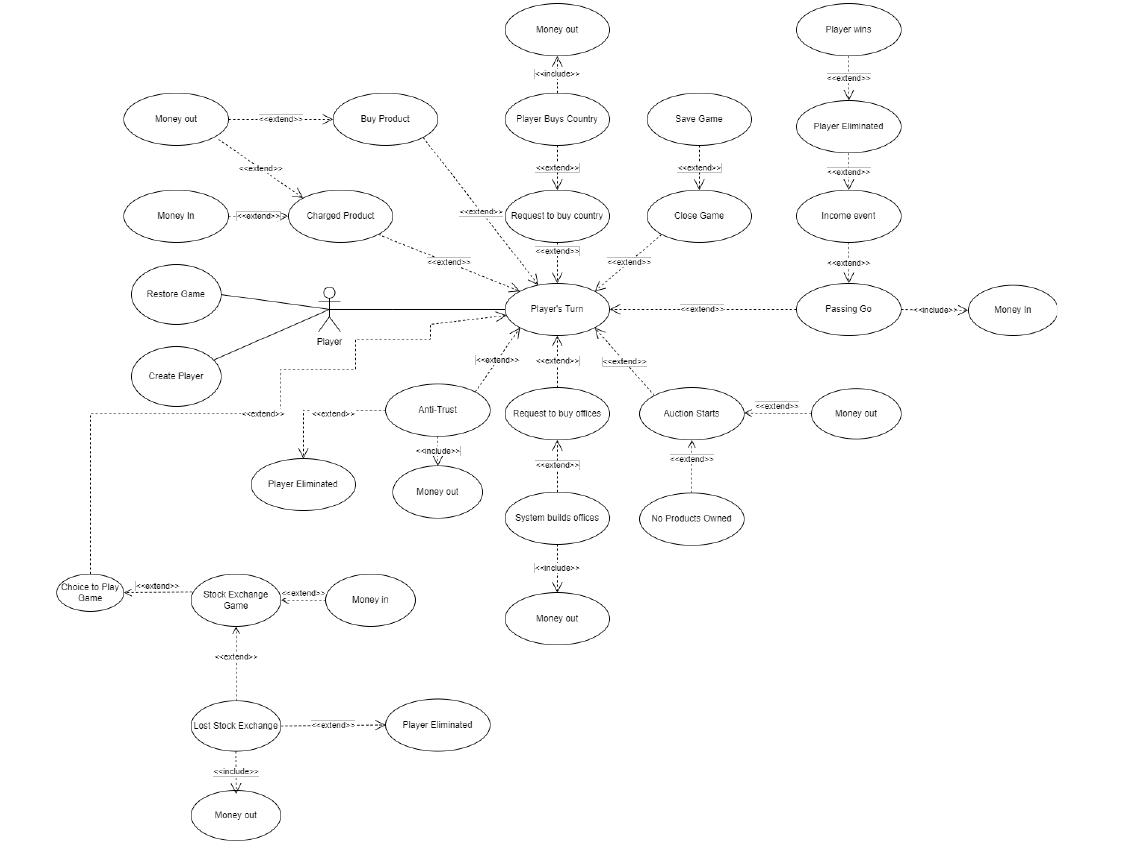
Board



Products pricing

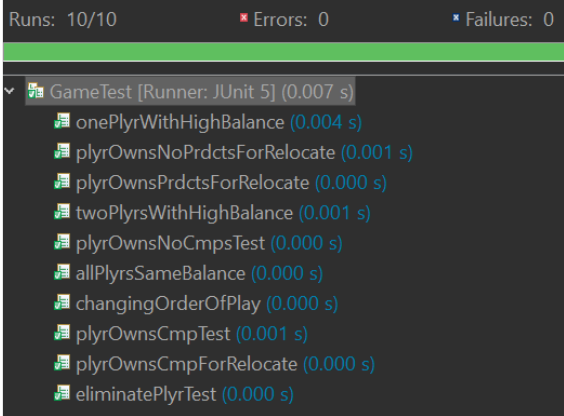


Use case

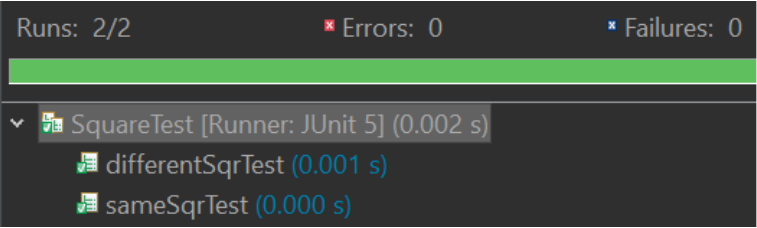


Junit Test Results

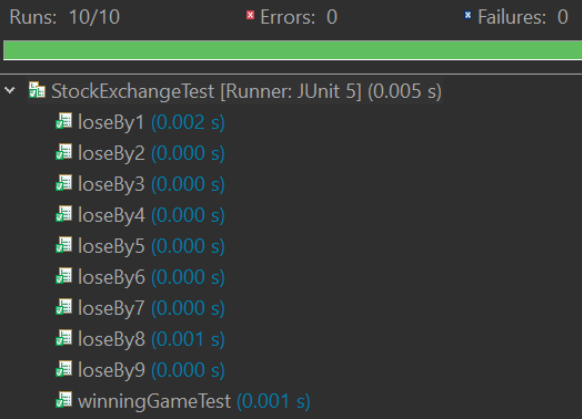
Game test



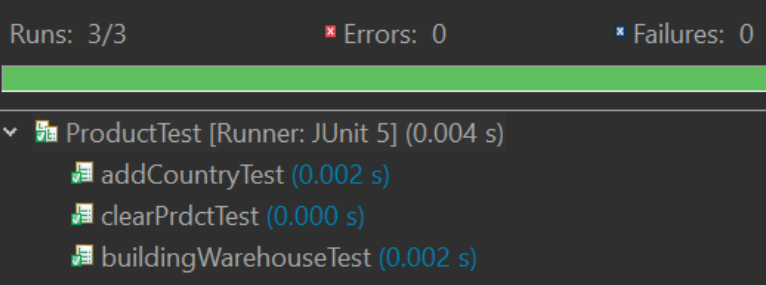
Square Test



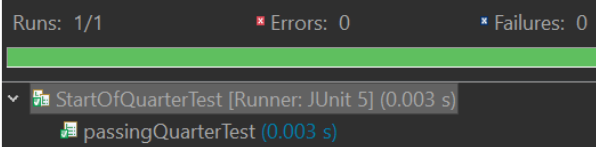
Stock ex



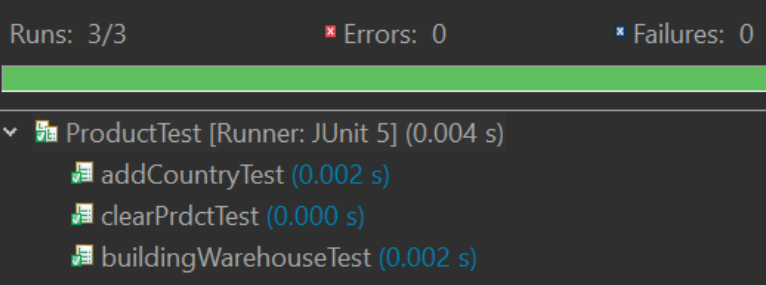
ProductBidding Test



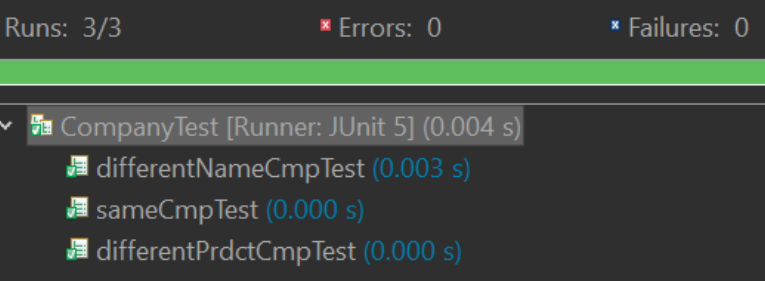
StartOfQuarter Test



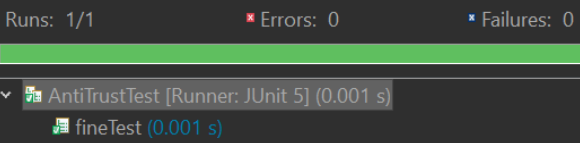
Product Test



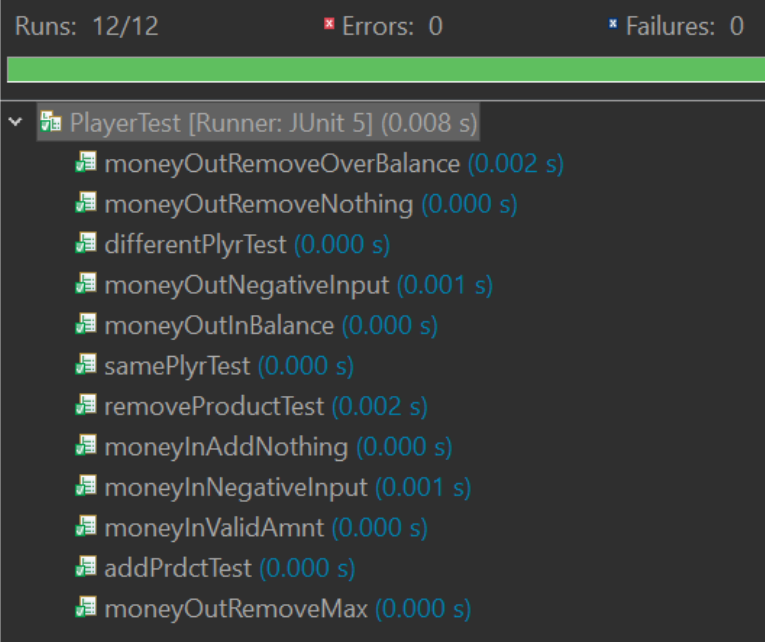
Company test



Antitrust test



Players test



# monapoly

A feasibility study on a Monopoly game application, with a proposed idea and based on IEEE 830 (or a similar standard), would involve the following steps:

Market demand analysis: Evaluate the demand for mobile gaming and board games, specifically Monopoly, to determine if there is a market for the game application.

Technical requirements: Define the technical requirements for the development of the Monopoly game application, including platform compatibility, graphics, and user interface, in accordance with IEEE 830.

Software Requirement Specification (SRS): Develop an SRS that outlines the functional and non-functional requirements for the Monopoly game application, based on market demand and technical requirements, in accordance with IEEE 830.

Development costs: Estimate the costs of developing, testing, and launching the Monopoly game application, including the costs of hiring developers, designers, and testers, and the cost of any necessary software development tools.

Revenue potential: Forecast the potential revenue from sales and in-app purchases, based on market demand and the features offered in the application.

Competition: Analyze the competition in the market, including other Monopoly game applications, to determine the potential for success.

Risk analysis: Identify and assess the risks associated with developing and launching the Monopoly game application, such as changes in consumer preferences, technical challenges, and market competition, in accordance with IEEE 830.

Based on the results of this analysis, a conclusion can be drawn regarding the feasibility of developing and launching the Monopoly game application, including recommendations for improving its viability, in accordance with IEEE 830.

# risk

A feasibility study on a risk game application, with a proposed idea and based on IEEE 830 (or a similar standard), would involve evaluating the following aspects to determine its viability and potential for success:

Market demand: Evaluate the demand for mobile gaming and strategy games, specifically risk games, to determine if there is a market for the game application. Conduct market research to gather data and insights on the size and characteristics of the target audience, their preferences and behaviors, and the trends in the mobile gaming industry. This analysis should include a review of similar risk games and their popularity, as well as an assessment of any unmet needs or gaps in the market that the proposed game application can fill.

Technical requirements: Determine the technical requirements for the development of the risk game application, including the platform it will be compatible with, the graphics and user interface required, and any other technical specifications. These should be based on the SRS (Software Requirement Specification) as outlined in IEEE 830 or a similar standard. The technical analysis should consider the feasibility of implementing the proposed features, the cost and availability of necessary development tools and resources, and the compatibility with existing systems and technologies.

Development costs: Estimate the costs of developing, testing, and launching the risk game application, including the costs of hiring developers, designers, and testers, the cost of any necessary software development tools, and any other expenses related to the development process. This analysis should include a breakdown of costs by phase, as well as a realistic timeline for the development process. The budget should be adequate to support the development of the game application and provide a margin for contingencies.

Revenue potential: Forecast the potential revenue from sales and in-app purchases, based on market demand, the features offered in the application, and the pricing strategies adopted. Consider different pricing models, such as a one-time purchase or a subscription-based model, to maximize revenue potential. The revenue analysis should consider the costs of marketing and promoting the game application, as well as any potential barriers to adoption or competition in the market.

Competition: Analyze the competition in the market, including other risk game applications, to determine the potential for success. Consider factors such as the features and functionality offered by competing games, their popularity, and user reviews. This analysis should identify areas where the proposed risk game application can differentiate itself from the competition and provide a competitive advantage.

Risk analysis: Identify and assess the risks associated with developing and launching the risk game application, such as changes in consumer preferences, technical challenges, and market competition. Develop mitigation strategies to address the identified risks and minimize the potential impact on the viability and success of the game application. The risk analysis should consider the potential consequences of not launching the game application, as well as the impact of external factors, such as economic conditions, regulatory environment, and technological advancements.

Conclusion and recommendations: Based on the results of the analysis, draw a conclusion regarding the feasibility of developing and launching the risk game application. This conclusion should summarize the findings and provide recommendations for improving the viability and success of the game application. The recommendations should address the strengths and weaknesses of the proposal, and suggest alternative solutions or options if necessary.

The feasibility study should provide a comprehensive and detailed assessment of the proposed risk game application, allowing stakeholders to make informed decisions about its development and launch. The study should be regularly reviewed and updated to ensure that it remains relevant and reflects any changes in the market, technology, or other factors that may impact the viability of the game application.

monapoly

A feasibility study is a way to figure out if it's a good idea to create a monopoly game application. To do this, we will look at different parts of the project to see if it is possible and if it will be successful.

Market demand: We will check if people are interested in playing a monopoly game on their phones by looking at how popular mobile games and board games, especially monopoly games, are. We will also look at what kind of people would play this game and what they want in a game like this.

Technical requirements: We will make a list of what the game needs in order to work well, such as what kind of phone it will work on and what it will look like. This list will follow a standard called IEEE 830 or something similar.

Development costs: We will figure out how much money it will cost to make the game, including hiring people to make it and buying the things we need.

Revenue potential: We will try to predict how much money the game will make by looking at how much people are willing to pay for it and what they will buy within the game.

Competition: We will see what other monopoly games are out there and what makes our game different and better.

Risk analysis: We will think about any problems that could happen while making the game and try to stop them from happening.

Conclusion and recommendations: After looking at all these parts, we will decide if it is a good idea to make the game and what we can do to make it even better.

The feasibility study will give us a good idea of what we need to do to make the monopoly game application successful. We will update the study as things change so we always have the most up-to-date information.

# requirements traceability matrix (RTM)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Use Case | Description of Test | Test Initialisation | Test Inputs | Test Procedure | Expected Results | Passed? | Result of Test | Reason for failure | JUnit Test | Tester |
| 1 | Money In | Testing if a positive value is added to the user's balance and if the function will return true | The user's balance is 1000 | 100 | Pass '100' into the method 'moneyIn (amount)' | The user's balance should be 1100 and the 'moneyIn' method should return true |  | The amount was successfully added to the user's balance and the method returned 'True' | N/A | moneyInValidAmnt() | Adam Logan [A.L] |
| 2 | Money In | Testing if a negative value is not added to the player's balance and if the function will return false | The user's balance is 1000 | -100 | Pass '-100' into the method 'moneyIn(amount)' | The 'moneyIn' method should return false and the user's balance should remain unchanged |  | The user's balance did not change and the method returned 'False' | N/A | moneyInNegativeInput() | Adam Logan [A.L] |
| 3 | Money In | Testing if nothing will be added to the user's balance if 0 is the value entered into 'moneyIn' | The user's balance is 1000 | 0 | Pass '0' into the method 'moneyIn (amount)' | Nothing should be added to the player's balance |  | The user's balance did not change | N/A | moneyInAddNothing() | Adam Logan [A.L] |
| 4 | Money Out | Testing if a positive value is subtracted from the user's balance and if the function will return true | The user's balance is 1000 | 100 | Pass '100' into the method 'moneyOut (amount)' | The user's balance should be 900 and the 'moneyOut' method should return true |  | The user's balance was 900 and the method returned 'True' | N/A | moneyOutInBalance() | Adam Logan [A.L] |
| 5 | Money Out | Testing if a negative value is not subtracted from the player's balance and if the function will return false | The user's balance is 1000 | -100 | Pass '-100' into the method 'moneyOut (amount)' | The 'moneyOut' method should return true and the user's balance should be 1000 |  | The user's balance was 1100 and not 1000 | The original check in the method simply checked if the balance was greater or equal to the amount and therefore the negative number was subtracted (which added the value to the balance as a '-, - = +') from the balance (successful test ID = 6) | moneyOutNegativeInput() | Adam Logan [A.L] |
| 6 | Money Out | Testing if a negative value is not subtracted from the player's balance and if the function will return false | The user's balance is 1000 | -100 | Pass '-100' into the method 'moneyOut (amount)' | The 'moneyOut' method should return false |  | The user's balance was 1000 | N/A | moneyOutNegativeInput() | Adam Logan [A.L] |
| 7 | Money Out | Testing if nothing will be subtracted to the user's balance if 0 is the value entered into 'moneyOut' | The user's balance is 1000 | 0 | Pass '0' into the method 'moneyOut (amount)' | Nothing should be subtracted from the player's balance |  | The user's balance did not change | N/A | moneyOutRemoveNothing() | Adam Logan [A.L] |
| 8 | Money Out | Testing if the user can lose all the money in their balance and if the function will return true | The user's balance is 1000 | 1000 | Pass '1000' into the method 'moneyOut (amount)' | The user's balance should be 0 and the 'moneyOut' method should return true |  | The user's balance was 0 and the method returned 'True' | N/A | moneyOutRemoveMax() | Adam Logan [A.L] |
| 9 | Money Out | Testing if the 'moneyOut' function will return false if the user loses more money than there is in their account | The user's balance is 1000 | 1001 | Pass '1001' into the method 'moneyOut (amount)' | The user's balance should be -1 and the 'moneyOut' method should return false |  | The user's balance is -1 and the method returned 'False' | N/A | moneyOutRemoveOverBala nce() | Adam Logan [A.L] |
| 10 | Anti-Trust | Testing if the correct persentage is removed from the user's balance when landing on the Anti- Trust square | The user's balance is 1000 and their current square is an Anti-Trust Square | Anti-Trust Square set with a value of 10% | A Player class is created and is passed into the 'fine' method of the AntiTrust class | The user's new balance should be 900 |  | The user's balance is 900 | N/A | fineTest() | Adam Logan [A.L] |
| 11 | Choice to Play Game | Testing if the player can choose not to play the stock exchange game | The player has landed on the stock exchange square | 2 (this is the option to skip the game) | Enter the option 2 | Message displayed informing the player that they have chosen not to play the game |  |  | N/A | N/A | Adam Logan [A.L] |
| 12 | Choice to Play Game | Testing if the player can choose to play the stock exchange game | The player has landed on the stock exchange square | 1 (this is the option to play the game) | Enter the option 1 | The Stock Exchange Game will be played |  |  | N/A | N/A | Adam Logan [A.L] |
| 13 | Stock Exchange Game | Testing if the user can enter a valid amount of money in the stock exchange game | The player has landed on the stock exchange square, the player has choosen to play and they have more than £20 in their balance | 20 | Enter option 1 then enter 20 | The Stock Exchange Game will continue and will prompt the player for a random number |  |  | N/A | N/A | Adam Logan [A.L] |
| 14 | Stock Exchange Game | Testing if the user can enter a valid amount of money in the stock exchange game | The player has landed on the stock exchange square, the player has choosen to play and they have exactly £1000 in their balance | 1000 | Enter option 1 then enter 1000 | The Stock Exchange Game will continue and will prompt the player for a random number |  |  | N/A | N/A | Adam Logan [A.L] |
| 15 | Stock Exchange Game | Testing if the user can enter an invalid amount of money in the stock exchange game and for an error message to be displayed | The player has landed on the stock exchange square, the player has choosen to play and they have exactly £1000 in their balance | 2000 | Enter option 1 then enter 2000 | An error message will be displayed and the player will be prompted to enter another amount |  |  | N/A | N/A | Adam Logan [A.L] |
| 16 | Stock Exchange Game | Testing if the user can enter an invalid amount of money in the stock exchange game and for an error message to be displayed | The player has landed on the stock exchange square, the player has choosen to play and they have exactly £1000 in their balance | 1001 | Enter option 1 then enter 1001 | An error message will be displayed and the player will be prompted to enter another amount |  |  | N/A | N/A | Adam Logan [A.L] |
| 17 | Stock Exchange Game | Testing if the user can enter any character in the stock exchange game and for an error message to be displayed | The player has landed on the stock exchange square, the player has choosen to play | win! | Enter option 1 then enter win! | An error message will be displayed and the player will be prompted to enter an integer value |  |  | N/A | N/A | Adam Logan [A.L] |
| 18 | Stock Exchange Game | Testing if the user can enter any character and some numbers in the stock exchange game and for an error message to be displayed | The player has landed on the stock exchange square, the player has choosen to play | win12! | Enter option 1 then enter win12! | An error message will be displayed and the player will be prompted to enter an integer value |  |  | N/A | N/A | Adam Logan [A.L] |
| 19 | Stock Exchange Game | Testing if the user can enter a valid amount of money in the stock exchange game | The player has landed on the stock exchange square, the player has choosen to play and they have more than £20 in their balance | 20.0 | Enter option 1 then enter 20.0 | The Stock Exchange Game will continue and will prompt the player for a random number |  |  | Test failed due to when a string is being converted into an integer it does not recognise the number as an integer even though they have the same value (successful test ID = 20) | N/A | Adam Logan [A.L] |
| 20 | Stock Exchange Game | Testing if the user can enter a valid amount of money in the stock exchange game | The player has landed on the stock exchange square, the player has choosen to play and they have more than £20 in their balance | 20.1 | Enter option 1 then enter 20.1 | The Stock Exchange Game will continue and will prompt the player for a random number |  |  | N/A | N/A | Adam Logan [A.L] |
| 21 | Stock Exchange Game | Testing if the user can enter a float into the stock exchange game | The player has landed on the stock exchange square, the player has choosen to play and they have more than £20 in their balance | 20.3 | Enter option 1 then enter 20.3 | An error message will be displayed and the player will be prompted to enter an integer value |  |  | N/A | N/A | Adam Logan [A.L] |
| 22 | Stock Exchange Game | Testing if the user can enter a value within the range of the stock exchange's defined numbers | The player has landed on a stock exchange with the range set to 1 - 10 | 5 | Enter option 1, enter 20 and then enter 5 | The apprpraite message will be displayed (this depends if the value entered is the number generated by the system) |  |  | N/A | N/A | Adam Logan [A.L] |
| 23 | Stock Exchange Game | Testing if the user can enter a value within the range of the stock exchange's defined numbers | The player has landed on a stock exchange with the range set to 1 - 10 | 10 | Enter option 1, enter 20 and then enter 10 | The apprpraite message will be displayed (this depends if the value entered is the number generated by the system) |  |  | N/A | N/A | Adam Logan [A.L] |
| 24 | Stock Exchange Game | Testing if the user can enter a value that is not within the pre- defined range | The player has landed on a stock exchange with the range set to 1 - 10 | 11 | Enter option 1, enter 20 and then enter 11 | An error message will be displayed and the player will be prompted to enter another value |  |  | N/A | N/A | Adam Logan [A.L] |
| 25 | Stock Exchange Game | Testing if the user can enter characters when asked to choose a number and for an error message to be displayed | The player has landed on a stock exchange with the range set to 1 - 10 | win! | Enter option 1, enter 20 and then enter win! | An error message will be displayed and the player will be prompted to enter an integer value |  |  | N/A | N/A | Adam Logan [A.L] |
| 26 | Stock Exchange Game | Testing if the user can enter any characters and some numbers in the when asked to choose a number and for an error message to be displayed | The player has landed on a stock exchange with the range set to 1 - 10 | win12! | Enter option 1, enter 20 and then enter win12! | An error message will be displayed and the player will be prompted to enter an integer value |  |  | N/A | N/A | Adam Logan [A.L] |
| 27 | Stock Exchange Game | Testing if the user can enter a float into the stock exchange game when asked to guess a random number | The player has landed on a stock exchange with the range set to 1 - 10 | 5.0 | Enter option 1, enter 20 and then enter 5.0 | The apprpraite message will be displayed (this depends if the value entered is the number generated by the system) |  |  | Test failed due to when a string is being converted into an integer it does not recognise the number as an integer even though they have the same value (successful test ID = 28) | N/A | Adam Logan [A.L] |
| 28 | Stock Exchange Game | Testing if the user can enter a float into the stock exchange game when asked to guess a random number | The player has landed on a stock exchange with the range set to 1 - 11 | 5.1 | Enter option 1, enter 20 and then enter 5.1 | The apprpraite message will be displayed (this depends if the value entered is the number generated by the system) |  |  | N/A | N/A | Adam Logan [A.L] |
| 29 | Stock Exchange Game | Testing if the user can enter a float into the stock exchange game when asked to guess a random number | The player has landed on a stock exchange with the range set to 1 - 10 | 5.3 | Enter option 1, enter 20 and then enter 5.3 | An error message will be displayed and the player will be prompted to enter an integer value |  |  | N/A | N/A | Adam Logan [A.L] |
| 30 | Stock Exchange Game | Testing if the correct number is returned when the user correctly guesses the number | The stock exchange object has been set with the range 1 - 10 and a risk of 0.25. The correct guess is 5 and the investement is 100. | 5 | pass 5 as they guessedNum, 100 as the invtAmnt and 5 as the correctGuess into the 'playGame()' method | The returned value should be 125 |  | Returned 125 | N/A | winningGmaeTest | Adam Logan [A.L] |
| 31 | Lost Stock Exchange | Testing if the correct amount is returned when the user incorreclty guesses the number | The stock exchange object has been set with the range 1 - 10 and a risk of 0.25. The correct guess is 5 and the investement is 100. | 1 | pass 1 as the guessedNum, 100 as the invtAmnt and 5 as the correctGuess into the 'playGame()' method | The returned value should be -40 |  | Returned -40 | N/A | loseBy4() | Adam Logan [A.L] |
| 32 | Lost Stock Exchange | Testing if the correct amount is returned when the user incorreclty guesses the number | The stock exchange object has been set with the range 1 - 10 and a risk of 0.25. The correct guess is 5 and the investement is 100. | 2 | pass 2 as the guessedNum, 100 as the invtAmnt and 5 as the correctGuess into the 'playGame()' method | The returned value should be -30 |  | Returned -30 | N/A | loseBy3() | Adam Logan [A.L] |
| 33 | Lost Stock Exchange | Testing if the correct amount is returned when the user incorreclty guesses the number | The stock exchange object has been set with the range 1 - 10 and a risk of 0.25. The correct guess is 5 and the investement is 100. | 3 | pass 3 as the guessedNum, 100 as the invtAmnt and 5 as the correctGuess into the 'playGame()' method | The returned value should be -20 |  | Returned -20 | N/A | loseBy2() | Adam Logan [A.L] |
| 34 | Lost Stock Exchange | Testing if the correct amount is returned when the user incorreclty guesses the number | The stock exchange object has been set with the range 1 - 10 and a risk of 0.25. The correct guess is 5 and the investement is 100. | 4 | pass 4 as the guessedNum, 100 as the invtAmnt and 5 as the correctGuess into the 'playGame()' method | The returned value should be -10 |  | Returned -10 | N/A | loseBy1() | Adam Logan [A.L] |
| 35 | Lost Stock Exchange | Testing if the correct amount is returned when the user incorreclty guesses the number | The stock exchange object has been set with the range 1 - 10 and a risk of 0.25. The correct guess is 5 and the investement is 100. | 6 | pass 6 as the guessedNum, 100 as the invtAmnt and 1 as the correctGuess into the 'playGame()' method | The returned value should be -50 |  | Returned -50 | N/A | loseBy5() | Adam Logan [A.L] |
| 36 | Lost Stock Exchange | Testing if the correct amount is returned when the user incorreclty guesses the number | The stock exchange object has been set with the range 1 - 10 and a risk of 0.25. The correct guess is 5 and the investement is 100. | 7 | pass 7 as the guessedNum, 100 as the invtAmnt and 1 as the correctGuess into the 'playGame()' method | The returned value should be -60 |  | Returned -60 | N/A | loseBy6() | Adam Logan [A.L] |
| 37 | Lost Stock Exchange | Testing if the correct amount is returned when the user incorreclty guesses the number | The stock exchange object has been set with the range 1 - 10 and a risk of 0.25. The correct guess is 5 and the investement is 100. | 8 | pass 8 as the guessedNum, 100 as the invtAmnt and 1 as the correctGuess into the 'playGame()' method | The returned value should be -70 |  | Returned -70 | N/A | loseBy7() | Adam Logan [A.L] |
| 38 | Lost Stock Exchange | Testing if the correct amount is returned when the user incorreclty guesses the number | The stock exchange object has been set with the range 1 - 10 and a risk of 0.25. The correct guess is 5 and the investement is 100. | 9 | pass 9 as the guessedNum, 100 as the invtAmnt and 1 as the correctGuess into the 'playGame()' method | The returned value should be -80 |  | Returned -80 | N/A | loseBy8() | Adam Logan [A.L] |
| 39 | Lost Stock Exchange | Testing if the correct amount is returned when the user incorreclty guesses the number | The stock exchange object has been set with the range 1 - 10 and a risk of 0.25. The correct guess is 5 and the investement is 100. | 10 | pass 10 as the guessedNum, 100 as the invtAmnt and 1 as the correctGuess into the 'playGame()' method | The returned value should be -90 |  | Returned -90 | N/A | loseBy9() | Adam Logan [A.L] |
| 40 | Auction starts | Testing if the system can choose a field to put up for auction | The player has landed on the Company Bidding field, and the system will select a field from the resulting player | N/A | System will randomly select a field from the resulting player to auction to the other players | The system will pick a field at random and the auction will begin |  | One of the products was chosen | N/A | getValidPrdctTest() | XXXXX [X.X] |
| 41 | No Products Owned | Testing if the System will display an appropriate error message when the player who landed on the product bidding square has no products | Player lands on the product bidding square, but player does not own any products to bid | N/A | System tries to randomly select a field from resulting player | The Sytem will display an error message, stating the resulting player does not own any products. game will procced as normal |  |  | N/A | N/A | XXXXX [X.X] |
| 42 | Auction starts | Testing if the current player can place their bid | Player 2 will enter their bid for Player 1s feild | 500 | When prompted, enter 500 | Player 2 will be able to successfully enter their bid for player 1s field |  |  | N/A | N/A | XXXXX [X.X] |
| 43 | Auction starts | Testing what happens when the current player, enters a lower bidding amount than the previous player | The current player will enter thier bid for the field. | 250 | When prompted, enter 250 | The System will display an error message stating the current players bis is lower than the previous players amount, and ask them to re-enter their bid |  |  | N/A | N/A | XXXXX [X.X] |
| 44 | Auction starts | Testing to see if the system will correctly select the player with the highest bid | The system will look at all the bids and select the highest | 500 | System will select 500 (as it is the largest) | The system will select the correct player and this player will recieve Player 1s field |  |  | N/A | N/A | XXXXX [X.X] |
| 45 | Create Player | Testing if the user is able to enter valid charactors | The system will ask "what is your name", the Player will enter a name | "Jake" | Player will enter "Jake" when prompted by System | The system will allow this as it contains valid charactors. the Game will move onto the next players name |  |  | N/A | N/A | XXXXX [X.X] |
| 46 | Create Player | Testing if the user can enter a non-valid charactor | System will ask player to enter their name | "Jak£" | Player will enter "Jak£" when prompted by System | The system will display an error message back to the player and say that they are not allowed to enter non-letter charactors |  |  | There is no check for special characters or numbers (successful test ID = 47) | N/A | XXXXX [X.X] |
| 47 | Create Player | Testing if the user can enter a non-valid charactor | System will ask player to enter their name | "Jak£" | Player will enter "Jak£" when prompted by System | The system will display an error message back to the player and say that they are not allowed to enter non-letter charactors |  |  | N/A | N/A | XXXXX [X.X] |
| 48 | Create Player | Testing if the game will proced once everyone has entered thier names | The system will ask each player to enter their names, in turns | "Jake", "Adam", "Joe", "ShaunChathal" | Players will enter thier own names, in turns. system will finish once everyone has entered their names. | Once the players have entered therir names, the system willl proced onto the game as normal, starting with the first players' turn |  |  | N/A | N/A | XXXXX [X.X] |
| 49 | Create Player | Testing to see if an Error message will appear when 2 players enter the same name as each other | The system will ask each player to enter their names, in turns | "Adam" "Adam" | Players will enter thier own names, in turns. system will finish once everyone has entered their names. | when a player enters the same name as another player, the system will display an error message to the user stating they cant have the same name as another player |  |  | N/A | N/A | XXXXX [X.X] |
| 50 | Income Event | Testing to the player is kept in the game, if they have more money than the threshold | when the player passes go, the threshold is £500, and the player has £1000 in their balance | N/A | player lands / passes go with £1000 in their balance, and is above the threshold | When the player passes go for the 4th time in a row, the player will be kept in the game as they have more funds in their balance, than the threshold |  |  | N/A | N/A | XXXXX [X.X] |
| 51 | Income event | Testing to the player is kept in the game, if they have more money than the threshold | when the player passes go, the threshold is £500, and the player has £450 in their balance | N/A | player lands / passes go with £450 in their balance, and is below the threshold | When the player passes go for the 4th time in a row, the player will be kept eliminated from the game |  |  | N/A | N/A | XXXXX [X.X] |
| 52 | Request to buy country | Testing to see if the player is able to purchase an country for a product they own | The player will select the 'request to buy country' option and will own all the products the company 'AMD' and 4 warehouses are dedicated to each of the products. | Selects to buy | The player will relocate the product SeaMicro to the United States. | The option to request a country is shown and the amount to charge for the product SeMicro has increased by 150. The player's balance will decrease by 45. |  | The amount to charge of SeaMicro was increased by 150 and the player was charged 45. | N/A | N/A | XXXXX [X.X] |
| 53 | Request to buy country | Testing to see if the option to purchase a country is not shown, if they dont own all the products within the company | The player will select the 'request to buy country' option and will own 'ryzen' with all 4 warehouses, within the company 'AMD' | Selects to buy | the player wont get the option to relocate their product as they dont own all products within the company | The option to request a country will not be shown as the player does not own all the products within 'AMD' |  |  | N/A | N/A | XXXXX [X.X] |
| 54 | Request to buy country | Testing to see if the player can see the option to request to purchase a country, if they dont have all 4 warehouses on the products | The player will select the 'request to buy country' option and will own all then products within the company 'AMD', but doesnt own all 4 warehouses on all the products | Selects to buy | The player wont be given the option to purchase a country when they dont own all the warehouses on the products | The option to request a country will not be shown as the player does not own all the warehouses, on their products, within 'AMD' |  |  | N/A | N/A | XXXXX [X.X] |
| 55 | Request to buy country | Testing to see if, when the player tries to purchase another country, the same option will not appear twice | The player will select 'request to purchase a country' and wont be given the option 'United States', as the player already has a product with this county | Selects to buy | the player will select the 'request to buy office' option. The system will check to see if the player already owns any other countries on other products. | Since the player already owns 'United States' on another product, the system wont display this option again to the user, making them choose a different option. the relocation fee has also increased by £25 |  |  | N/A | N/A | XXXXX [X.X] |
| 56 | Charged Product | Testing to see if player is charged money when they land on owned product | Both users balance start with 1000 | 1 (Roll dice) | Enter 1 (Land on an owned product) | The player landing on the owned product will have their balance reduced, the player owning product will receive their balance |  |  | N/A | N/A | XXXXX [X.X] |
| 57 | Charged Product | Testing if player will be charged more if product has warehouse | Player with product must also have a warehouse | 1 (Roll dice) | Enter 1 (Land on an owned product) | Since player 1 has added a warehouse to their product, it should charge the next player who lands on it |  |  | N/A | N/A | XXXXX [X.X] |
| 58 | Charged Product | Testing if product with a country will charge more | Player with product must own a country | 1 (Roll dice) | Enter 1 (Land on an owned product) | When player that owns a product upgrades and gets a country, it should charge the other players landing on it |  |  | N/A | N/A | XXXXX [X.X] |
| 59 | Request Warehouse | Testing if warehouse will be built on owned product | Player must own a product | 7 (Purchase Warehouse) | Choose option 'Purchase Warehouse' | When choosing to purchase a in the menu it should subtract the money from users balance and build warehouse |  | 17 was subtracted from the player's balance and the new amount to charge a player for landing on the product has been updated | N/A | N/A | XXXXX [X.X] |
| 60 | Request Warehouse | When choosing a product to purchase a warehouse for enter string instead of int | Player must own a product | f | Choose option 'Purchase Warehouse' then enter "f" for answer | The system should reject the input and return an error message and allow them to re enter a number. |  |  | N/A | N/A | XXXXX [X.X] |
| 61 | Request Warehouse | Continue entering incorrect values when purchasing warehouse | Must be purchasing a warehouse | 5 | Choose the option to purchase the warehouse enter integer out of range, letter and larger integers. | The system should display an error message |  |  | N/A | N/A | XXXXX [X.X] |
| 62 | Request Warehouse | Continue entering incorrect values when purchasing warehouse | Must be purchasing a warehouse | 55 | Choose the option to purchase the warehouse enter integer out of range, letter and larger integers. | The system should display an error message |  |  | N/A | N/A | XXXXX [X.X] |
| 63 | Request Warehouse | Continue entering incorrect values when purchasing warehouse | Must be purchasing a warehouse | 1000000 | Choose the option to purchase the warehouse enter integer out of range, letter and larger integers. | The system should display an error message |  |  | N/A | N/A | XXXXX [X.X] |
| 64 | Request Warehouse | Continue entering incorrect values when purchasing warehouse | Must be purchasing a warehouse | 100000000000000000000000 0000000000000000000 | Choose the option to purchase the warehouse enter integer out of range, letter and larger integers. | The system should display an error message |  |  | N/A | N/A | XXXXX [X.X] |
| 65 | Request Warehouse | When buying a new product will it be added to the list of available products to build warehouses for | Purchase an available product | Enter 7 on menu | Enter 1 (Land on an available product), purchase product. Purchase warehouse on menu. | The system should add the purchased product to the list of options for a warehouse |  |  | N/A | N/A | XXXXX [X.X] |
| 66 | Request Warehouse | Try to build a warehouse when not owning any products | N/A | Choose to buy warehouse on menu | Choose the option to purchase the warehouse | The system should hide the ability to do anything if they player cannot do it, so the option to build warehouse should be hidden |  |  | N/A | N/A | XXXXX [X.X] |
| 67 | Request Warehouse | Building multiple warehouses an the one product | N/A | Choose to buy warehouse on menu | Choose the option to purchase the warehouse | The system should increase the price when landing on product |  | The resource use when landing on the product will increase, the money will also be took from the players balance | N/A | N/A | XXXXX [X.X] |
| 68 | Request Warehouse | Test system to see if it stops player from purchasing more than 4 warehouses for 1 product | Purchase 4 warehouses for a single product | 7 on menu multiple times | Purchase 4 warehouses for one product, after 4 request to buy warehouse | Any product that reaches the limit of 4 warehouses, will be removed from the list of available products |  |  | N/A | N/A | XXXXX [X.X] |
| 69 | Player's Turn | Testing if the turn switches between four players. | It is the first player's turn. | Enter | Press 'Enter' to proceed through the game to the next player's turn. | The turn will change between the four players. |  | The turn switches between all four players. | N/A | N/A | XXXXX [X.X] |
| 70 | Player's Turn | Testing if the player can choose other game options, i.e. purchasing a product, before ending their turn. | It is the first player's turn. | 5 | Pass '5' into the menu to check the player's inventory. | The player will be able to choose view their inventory before ending their turn. |  | The player is able to choose to view their inventory before ending their turn. | N/A | N/A | XXXXX [X.X] |
| 71 | Player's Turn | Testing if the system lets eliminated players have a turn. | A player must be eliminated. | Enter | Press 'Enter' to proceed through the game to the next player's turn until it gets back to the turn of the player it started on. | The game will cycle through the turns, skipping eliminated players. |  | The game cycles through the turns, skipping eliminated players. | N/A | N/A | XXXXX [X.X] |
| 72 | Player's Turn | Testing if the turn changes in a consistent order. | It is the first player's turn. | Enter | Press 'Enter' to proceed through the game to the next person's turn 12 times. | The turn will change in a consistent order between all players. |  | The turn switches in a consistent order. | N/A | N/A | XXXXX [X.X] |
| 73 | Passing Go | Testing if the player receives money when they land on/pass the starting position. | The player is one move away from landing on or passing the starting position. | 0 | Pass '0' into the menu to roll the dice. | The player will receive money when they land on or pass the starting position. |  | The player receives money when they land on or pass the starting position. | N/A | N/A | XXXXX [X.X] |
| 74 | Passing Go | Testing if the player receives the correct amount of money when they land on/pass the starting position. | The player's balance is 1000. | 0 | Pass '0' into the menu to roll the dice. | The player will have recieved £150 when passing the start of quarter. |  | The player receives £150. | N/A | N/A | XXXXX [X.X] |
| 75 | Buying Product | Testing if the system offers the player the option to buy a product. | The product must not be owned by any players. | N/A | Roll the dice until the player lands on a product which is known not to be owned by another player | The player will be prompted with a message asking if they want to purchase the product |  | The player is presented with the option to buy the product. | N/A | N/A | XXXXX [X.X] |
| 76 | Buying Product | Testing if the player can purchase a product with sufficient funds. | The player must have sufficient funds and the product must not be owned by a player. | N/A | Roll the dice until the player lands on a product which is known not to be owned by another player | The player will be able to purchase the product. |  | The player is able to purchase the product. | N/A | N/A | XXXXX [X.X] |
| 77 | Buying Product | Testing if the player can purchase a product without having sufficient funds. | The player must not have sufficient funds to purchase the product. | N/A | Roll the dice until the player lands on a product which is known not to be owned by another player | The player won't be able to purchase the product. |  | The player is not able to purchase the product. | N/A | N/A | XXXXX [X.X] |
| 78 | Buying Product | Testing if a player can purchase a procuct which is owned by someone else. | The product must be owned by someone else. | N/A | Roll the dice until the player lands on a product which is known to be owned by another player | The player won't be able to purchase the product. |  | The player is not able to purchase the product. | N/A | N/A | XXXXX [X.X] |
| 79 | Buying Product | Testing if a player can purhcase a product that they already own. | The player must already own the product. | N/A | Roll the dice until the player lands on a product which is known to be owned by the current player | The player won't be able to purchase the product again. |  | The player is not able to purchase the product again. | N/A | N/A | XXXXX [X.X] |
| 80 | Buying Product | Testing if the system removes the correct amount from the player's balance for the purchase. | The player must land on Galaxy Tab with a balance of 1000. | N/A | Roll the dice until the player lands on a product which is known to be owned by the current player | 220 will be deducted from the player's balance leaving them with a balance of 780. |  | The correct amount is deducted from the player's balance. | N/A | N/A | XXXXX [X.X] |
| 81 | Buying Product | Testing if the system changes the players turn once they select refuse to buy the product. | The player must be given the option to buy a product. | N/A | Pass 'N' into the landedOnProduct method. | The turn will move onto the next player. |  | The turn moves onto the next player. | N/A | N/A | XXXXX [X.X] |
| 82 | Buying Product | Testing if when a player buys a product, it adds the product to their inventory. | The first player must be given the option to buy a product. | N/A | Check the player's balance | The product will be added to the player's inventory and will be listed when the player views their inventory. |  | The product is added to the player's inventory. | N/A | N/A | XXXXX [X.X] |
| 83 | Eliminate Player | Testing if the correct player is eliminated. | One player must be called "a". | a | Pass "a" into the eliminatePlayers method. | The specified player will be eliminated. |  | The correct player is eliminated. | N/A | N/A | XXXXX [X.X] |
| 84 | Eliminate Player | Testing if the player is eliminated once their balance reaches 0. | Player's balance must be at 5 when they land on FitBit. | Y | Pass 'Y' into the landedOnProduct method. | The player will be eliminated after they purchase the product. |  | The player is eliminated after they purchase the product. | N/A | N/A | XXXXX [X.X] |
| 85 | Eliminate Player | Testing if the products owned by the player are returned to the system once the player is eliminated. | Player must own at least one product when they are eliminated | 2 | Pass '2' into the menu to eliminate the player. | The products owned by the player will be returned to the system. |  | The products owned by the player are returned to the system. | N/A | N/A | XXXXX [X.X] |
| 86 | Winning | Testing if the win message displays correctly when a player wins. | There must be two players left. | 2 | Pass '2' into the menu to eliminate a player. | The win message will display correctly. |  | The win message displays correctly. | N/A | N/A | XXXXX [X.X] |
| 87 | Winning | Testing if a player wins once they are the last player left. | There must be two players left. | 2 | Pass '2' into the menu to eliminate a player. | The player will win when they are the last one left. |  | The player wins when they're the last one left alive. | N/A | N/A | XXXXX [X.X] |
| 88 | Winning | Testing if the game ends once a player wins. | A player must have won. | Enter | When the win message is displayed, input 'Enter'. | The game will end and the application will close. |  | The game ends when a player wins. | N/A | N/A | XXXXX [X.X] |
| 89 | Save Game | Testing if the correct details of the players are saved into a csv file correclty | The game is started with 3 players Adam, Joe and Jake | Adam (balance = 980, present square = Anti Trust), Joe (balance = 1000, present square = Sotheby's Auction) and Jake (balance = 711, present square = Kindle). None of the players have passed the start. Adam ows ipad, ipod and iphone while Jake owns Galaxy Phone. | To save and close the game on Jake's turn | A csv file will be created with the correct inputs |  |  | N/A | N/A | Adam Logan [A.L] |
| 90 | Save Game | Testing if the correct deails of the products are saved into a csv file correclty | The products Galaxy Phone, iPad, iPod and iPhone should be owned | Galxy Phone (amount to charge = 10, cost of relocation = 145), iPad (amount to charge = 9, cost of relocation = 25), iPod (amount to charge = 7, cost of relocation = 20), iPhone (amount to charge = 57, cost of relocation = 45). The products iPad, iPodm and iPhone will have 4 warehouses bought and Galaxy Phone will not have any bought. The only product to have a countyr owned is iPhone and the country should be the United States. | To save and close the game on Jake's turn | A csv file will be created with the correct inputs |  |  | N/A | N/A | Adam Logan [A.L] |
| 91 | Restore Game | Testing if the correct details of the players are loaded from the csv file correclty, into the game | For a Game to be saved with the details described in the 'Test Inputs' column | Adam (balance = 980, present square = Anti Trust), Joe (balance = 1000, present square = Sotheby's Auction) and Jake (balance = 711, present square = Kindle). None of the players have passed the start. Adam ows ipad, ipod and iphone while Jake owns Galaxy Phone. | To load the saved game | For the correct details to be loaded into the game, and for Jake to be the first player to have their turn |  | All player's present square was the one which was saved in the csv file and Jake was the first player to have their turn | N/A | N/A | Adam Logan [A.L] |
| 92 | Restore Game | Testing if the correct details of the products are loaded from the csv file correclty, into the game | For a Game to be saved with the details described in the 'Test Inputs' column | Galxy Phone (amount to charge = 10, cost of relocation = 145), iPad (amount to charge = 9, cost of relocation = 25), iPod (amount to charge = 7, cost of relocation = 20), iPhone (amount to charge = 57, cost of relocation = 45). The products iPad, iPodm and iPhone will have 4 warehouses bought and Galaxy Phone will not have any bought. The only product to have a countyr owned is iPhone and the country should be the United States. | To load the saved game | For the correct details to be loaded into the game |  |  | N/A | N/A | Adam Logan [A.L] |
| 93 | Close Game | Testing if all players that have the same balance are returned | For an array of three players to be created | All players in the array to have 1000 in their balance | Pass an array of the players into the 'highestBalance()' method | For the returned arraylist to contain all three players |  | Returned an arraylist of all three players | N/A | allPlyrsSameBalance() | Adam Logan [A.L] |
| 94 | Close Game | Testing if the two players that have the highest balance are returned | For an array of three players to be created | For two players in the array to have 1001 in their balance | Pass an array of the players into the 'highestBalance()' method | For the returned arraylist to contain the two players with the highest balance |  | Returned an arraylist of all two players | N/A | twoPlyrsWithHighBalance() | Adam Logan [A.L] |
| 95 | Close Game | Testing if the player with the highest balance is returned | For an array of three players to be created | For one player in the array to have 1001 in their balance | Pass an array of the players into the 'highestBalance()' method | For the returned arraylist to contain the one player with the highest balance |  | Returned an arraylist with just the one player with the highest balance | N/A | onePlyrWithHighBalance() | Adam Logan [A.L] |