Software Requirements Specification and Analysis

1) Introduction

This is digital recreation of the board game Monopoly. It is to be implemented in C++ using a using a console interface for gameplay . The object of the game is to by property and strategically manage finances. The object of the game is to bankrupt the other players.

Rules of the game:-

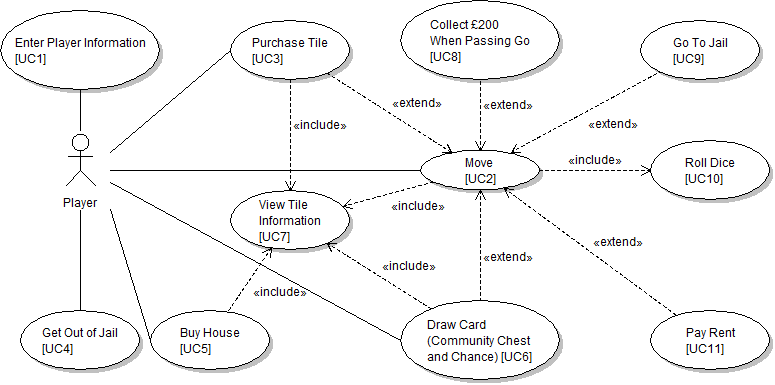
1. At the beginning of the game, each player rollers the dice. The player with the highest number goes first.
2. When a player moves to a property tile, player has the option to buy the property.
3. Once player owns a property tile, any other player who lands pays that player rent according to stated value.
4. Once a player owns a group of properties (indicated by colour), they can not only charge higher rent, but also are allowed to build houses and hotels on a property.
5. Each house increases rent.
6. A hotel can only be built ones 5 houses are built on ONE property.

Other Game Components include:-

1. The ‘Go’ tile is where each player starts from. Every time this tile is passed, each player receives £200.
2. ‘Community Chest’ and ‘Chance’ tiles are special cases. When a player lands on one of these, they are randomly assigned a task, which includes receiving or giving money, going to jail and being free from jail
3. The ‘Go To Jail’ tile penalises a player by putting them out of the game for up to 3 turns, unless the player pays a £50 fee.
4. ‘Water Works’ and ‘Electric Company’ are special tiles where the owner can charge rent to the target player according to the dice roll.
5. There are 4 Railway cards, which increase with rent depending on how many railways the player owns.
6. ‘Income Tax’ is a tile where the player is forced to pay £100 to the bank.

Use Case Diagram.

Below is a Use Case UML diagram showing the different actions a player takes throughout the process of the game. It is then followed by the flow descriptions detailing the different interactions.



**UC1:- Enter Player Information – Flow Events**

**Preconditions:** None

**Main Flows:** At the start of the game, console prompts for users to enter how many players there are. User then enters a value between 2 or 6. User then enters name for each of the players. Once this is done, each player roles the dice. The player with the highest number goes first.

**Subflows:** None

**Alternative Flows:**

* If the player does not enter an integer, or a number between and including 2 and 6, then the user is prompted to enter value again.
* If the player does not enter a string for the player name, they are prompted to enter again.

**UC2:- Move – Flow Events**

**Preconditions:** All player information has been accepted, and player order has been determined.

**Main Flows:** Each player is moved according to the value the dice returns after they roll it. Each player takes turns to roll the dice. Each player can also be moved depending on certain circumstances, such as landing on the ‘Go To Jail’ tile, or drawing a card from ‘Community Chest’ which commands the player to move to a specified tile, e.g. ‘Advance To Old Kent Road’.

Once the player has been moved to a tile, the Tile’s information is then displayed.

**Subflows:** Once player has been moved to a cell, an action or set of actions are taken depending on the type of cell:-

* Purchase Tile [UC3]
* Collect £200 when Passing Go [UC8]
* Draw Card(Community Chest and Chance) [UC6]
* Pay Rent [UC11]

**Alternative Flows:** None

**UC3:- Purchase Tile– Flow Events**

**Preconditions:** The player has rolled the dice and landed on a tile which is available for purchase

**Main Flows:** The information of the cell is displayed. The player is then prompted with a ‘yes or no’ question about whether they would like to purchase the property. If yes is selected, the player pays the amount of money indicated by the tile information to the bank, and the property belongs to that player. If they select no, the property remains available and the player’s turn is ended.

**Subflows:** None

**Alternative Flows:** If the player does not have enough money, a message stating so is presented on the console, and the turn is ended.

**UC4:- Get Out Of Jail – Flow Events**

**Preconditions:** The player must already be in jail before this event can be executed. A ‘Get Out Of Jail Free’ card can be acquired from one of the ‘Draw Card’ tiles.

**Main Flows:** The player can get out of jail either by using the ‘Get Out of Jail Free’ card, or they can pay £50 to get out.

**Subflows:** None

**Alternative Flows:** The player cannot afford the £50 and does not have a card to get out. Therefore they are out of the game.

**UC5:- Buy House– Flow Events**

**Preconditions:** The player must do this BEFORE they roll the dice. They can only purchase houses (and hotels) for properties for which they have all of the matching colour groups.

**Main Flows:** At the start of the players turn, if they have a group of properties, they are prompted as to whether they would like to build any houses. The player is able to build one house at a time on each of the properties. Each property has 5-house limit, where 5 houses becomes a hotel.

**Subflows:** None

**Alternative Flows:** If player does not have enough money, they are told so in a message, and nothing happens. The same is done for when the player already has 5 houses on a property.

**UC6:- Draw Card(Community Chest and Chance)– Flow Events**

**Preconditions:** The player has rolled the dice and has landed on either the ‘Community Chest’ or ‘Chance’ Tile.

**Main Flows:** When the player lands on one of the tiles, and ‘task card’ is drawn at random. This is presented in the form of instruction (e.g. You have been caught drink-driving, pay £150 fee) or a statement (You have prevented a Warp Core breech, have £10 Mr LaForge).

**Subflows:** None

**Alternative Flows:** None

**UC7:- View Tile Information– Flow Events**

**Preconditions:** Tile information can be viewed when a player lands on it, or when a player would like to build property on it.

**Main Flows:** Tile information is presented, displaying attributes such as name, rent price, who owns it (or if it is available) etc.

**Subflows:**  The property information is updated and re-presented when a player buys the property, puts a house on it, or when property is disowned by player due to bankruptcy.

**Alternative Flows:** None

**UC8:- Collect £200 When Passing Go– Flow Events**

**Preconditions:** The player passes the Go tile after dice roll.

**Main Flows:** Regardless of where the player lands afters passing the ‘Go’ tile, they receive £200 before any further transactions are made e.g. because they have landed on a card for which they must rent for.

**Subflows:** None

**Alternative Flows:** In some circumstances the player does not receive £200. These are:

* When player is sent to jail (by tile or by drawing card)
* When the player is asked to move back a certain amount of spaces

**UC9:- Go To Jail– Flow Events**

**Preconditions:** The player lands on the ‘Go To Jail’ tile, or the player draws a ‘Go To Jail’ card

**Main Flows:** Player is moved to the ‘Jail’ tile where they remain until the player pays £50, or they use a ‘Get Out Of Jail Free’ card.

**Subflows:** None

**Alternative Flows:** None

**UC10:-Roll Dice– Flow Events**

**Preconditions:** It’s the player’s turn.

**Main Flows:** The console prompts player to ‘Press Enter to Roll’. Random number is generated by the Dice. This number can be any integer between 2 and 12 (2 dice).

**Subflows:** None

**Alternative Flows:** None

**UC11:-Pay Rent– Flow Events**

**Preconditions:** Player rolls and lands on tile which is owned by another player

**Main Flows:** The player has to pay a certain amount of rent to the player who owns the property

**Subflows:** The amount of rent paid is dependent on the following factors:-

* The individual rent value of the property
* The effected rent value of a property when the owner has the whole set of properties
* The effected rent value of a property when houses (and hotel) are built on it
* If it is a utility (‘Water Works’ and/or ‘Electric company’) then the amount of rent to pay is dependent on the value of the dice roll.

**Alternative Flows:** If the player does not have enough money, they must declare bankruptcy.

**Non-Functional Requirements**

1. **Performance** 
   1. **Environment:-** The Monopoly game is designed to be executed as a text-based user-driven console application. Response time is expected to be minimal with a very short delay between user input and console output. Each ‘move’ is represented by text messages indicating each key even in the game, e.g. when a player moves to a new square, and has to pay rent to another player.
   2. **Game Updates:**- Due to the completely event-driven nature of the game, each update is expected to be almost immediate for each immediate.
2. **Usability**
   1. Each player is presented with the options as and when they are needed. Each option is decided with a keyboard input specified by the current instruction.
   2. It is all Keyboard interaction, where the amount of keys is kept to a minimum as much as possible. The most typing any player has to do is when entering their name.
   3. Other than that, it is mostly 1 or 2 character inputs
3. **Portability**
   1. **Implementation:-** The game is implemented in C++ on Unix based systems. It only utilises built-in libraries which would allow for greater cross compatibility. No graphics libraries are used as it is purely text-based.
   2. **Compilation:**- It is compiled using the GNU C Compiler
4. **Operating Constraints**
   1. Needs between 2 to 6 players to properly play it.
   2. Requires only once machine or terminal to play on. Cannot be played across multiple terminals (e.g. LAN).
   3. Need to have C++ (build essentials) installed on machine in order to play
5. **Modifiability**
   1. The **board** itself can be modified by editing the ‘Board’ File. All but the ‘Go’, ‘Go To Jail’, ‘Jail’, ‘Community Chest’ and ‘Chance’ tiles can be modified. Within the ‘Board’ file, each line is written with a particular format depending on the type of tile it is. As long as the user confirms to the customisation guidelines (see user manual), then they can make their own properties with different rent values and names etc. Generally, the format is POSITION NUMBER, TYPE FLAG, PROPERTY VALUE, RENT VALUES, GROUP, NAME.
   2. **Community Chest** and  **Chance**  cards can also be customised in the same way as the board. The lines in both the ‘Community Chest’ and ‘Chance’ file follow a simpler structure. The format goes FLAG, 0-to-2 PARAMETERS, INSTRUCTION. See user manual for proper formatting.
   3. **Modifying Card Manager:-**  The Card Manager can be modified to accept different/new types of cards. For example, if someone wanted to a make a card which causes the player to ‘Go To Jail’ and lose £300 at the same time, they would have to create their own flag. In order for card manager to recognise this, they would have to include it in the ‘if’ statement which checks for each flag.

The user might want this flag to be ‘jl’. They would have to also create a class to deal with this operation.

* 1. Number of players can also be modified from 6 to e.g. 8 players, as long as there are at least 2 players.

1. **Security Considerations**
   1. When it is a player’s turn, that player cannot in anyway directly access the statistics or attributes of any other player.
   2. An Object Oriented approach allows for abstract separation and integrations of key functionality without comprising any security. It is ensured that any methods or functions exclusive to an object or class remain private.
2. **Error Handling**
   1. If the player enters an invalid value, the system waits until a valid input has been entered.
   2. Invalid entries in the Board, Community Chest or Chance file, are flagged up as the game is loading, at which point the game will exit. So if someone for example uses the ‘g’ flag (for when a player receives money) in the Chance file, but they don’t have an integer parameter afterwards, this will be flagged up
   3. If a player enters an empty string when entering player name, they will be prompted to enter a name.