Team Redmond

MONTREALOPOLY

Project Analysis and Development Plan

Version 1.3

TEAM MEMBERS

Name	Student ID
Stefan Thibeault	4498852
Robert Hanna	4737997
Simon Lacasse	5946964
Alexandre Bosserelle	5253217
Eugena Zolorova	4349598
Zhi Zhang	4912047
Xin Xi	4634799
Patrice Michaud	4701445
Hu Shan Liu	4815386
Jens Witkowski	5253969

Revision History

Date	Version	Description	Author
20/sept/03	1.0	Document start	Stefan Thibeault, Robert Hanna, Eugena Zorolova, Alexandre Bosserelle
23/sept/03	1.1	Added User Interface graphics	Stefan Thibeault, Robert Hanna, Eugena Zorolova, Alexandre Bosserelle
28/sept/03	1.2	Combined all parts together	Stefan Thibeault, Robert Hanna, Eugena Zorolova, Alexandre Bosserelle, Zhi Zhang, Xin Xi
29/sept/03	1.3	Final document revision	Stefan Thibeault

Table of Contents

Project Analysis and Development Plan

1. Introduction

The purpose of this document is to collect, analyze, and define the high-level needs and features of Montrealopoly, a multi-player game being developed by Team Redmond. Montrealopoly is a 1 to 8 player game capable of having both human and computerized opponents and it is similar to Parker Brother's Monopoly. It will describe the product functions, graphical user interfaces, requirements and constraints of the game.

The details of how the Montrealopoly game fulfills these needs are detailed in the use case and more details will be available in the upcoming design phase.

1.1 Purpose

This Software Requirements Document (SRD) describes the specification of the Montrealopoly multi-player game, which is in partial fulfillment of the requirements of COMP 354. It will define the high-level requirements of the user interfaces, product functions, user descriptions, assumptions and dependencies, constraints, specific requirements and an analysis model. The analysis model will include use case diagrams, class diagrams, sequence diagrams and state transition diagrams. Furthermore, a detailed project plan will be provided, including the schedule of the upcoming phases. This document is intended primarily for the members of Team Redmond and the project co-ordinator, Dr. Joey Paquet, as it will serve as a basis for the upcoming phases of the project.

1.2 Scope

This document only addresses the high level requirements of Montrealopoly that will be used as a basis for the design phase. Screen shots of the user interfaces will give one an idea on how the game will look once it is completed. The product functions will include a prioritized list of which functions are to be included in the game. The user description will include the target audience and working environment of the game. Specifications for the game will be outlined in the specific requirements section and the analysis model will contain UML diagrams. The use case diagrams will give an overview of the functions of Montrealopoly and how the users will interact with the game. The class diagrams will show the inter-relationships between the different objects in the game and the sequence diagrams will model the flow of logic within the game.

The development plan will outline the three phases of the project with a Work Breakdown Structure (WBS) with a Gantt chart. It will include the amount of man-days to complete the project and is broken down into tasks. The number of resources required for the project are also listed in this section.

1.3 Definitions and Abbreviations

1.3.1 Definitions

Board

The main playing area is made up of a square of 40 cells with 10 cells per side. There are two main types of cells on the board. The first type is properties (or deeds) and can be described as squares that can be purchased by a player. The second type consists of squares that cannot be purchased by anyone. The cells that you can purchase consist of streets, metros and utilities. The rest of the cells have specific instructions that the player must follow when the player lands on it. The board also keeps track of where each player's token is.

Bank

The bank has an unlimited amount of money and gives each player $1,500 at the start of the game. The bank also has all of the unpurchased title deeds, which players can buy. The bank can also mortgage properties and hotels can be bought and sold back to the bank.

Token

A token is an object that is used in the game to represent the position of a player in the game. The following are examples of the tokens: Expo67, Hockey Stick, Stanley Cup, Bagel, Saxophone, Beaver, F1 car and football.

Dice Roll

A dice roll is the action of throwing two dice randomly. The value of the roll is the sum of the values displayed on the top-facing sides of both dice.

Double Roll

A double roll is the action of rolling doubles. This occurs when the values displayed on the top-facing sides of both dice are the same (e.g.: 1-1, 2-2, etc…).

Title Deeds

Title Deeds are a legal document proving a player's ownership of a property. Street title deeds contain the purchase price, mortgage amount and rent for vacant and non-vacant properties. Metro and utilities title deeds contain the purchase price and the mortgage amount.

District

A district is composed of two or three streets of the same colour.

Just For Laughs (JFL) Card

JFL cards contain instructions that a player must follow when they land on a JFL cell (e.g. "Pay 50$ to Bank" & "Advance to Green Ave.")

1.3.2 Abbreviations

AI: Artificiel Intelligence
IEEE: Institute of Electrical and Electronics Engineers, Inc
JFL: Just For Laughs
SRD: Software Requirements Document
WBS: Work Breakdown Structure
OS: Operating System
FIFO: First In First Out

1.4 References

Paula Bo Lu, "COMP 354 Tutorial 1", Paula Bo Lu
http://www.cs.concordia.ca/~grad/blu/comp354.ppt (Current September 20, 2003)

IBM, "Rational Rose Beginner's FAQ", IBM,
http://www.rational.com/products/rose/gstart/online.jtmpl (Current September 20, 2003)

Hasbro, "Monopoly Rules", Hasbro
http://www.hasbro.com/common/instruct/monins.pdf (Current September 15, 2003)

Pressman, Roger S. Software Engineering: A Practitioner's Approach. 5th ed. Toronto: McGraw-Hill, 2001.

1.5 Overview

The rest of this document outlines the problem description and the development plan.

The problem description describes the game's user interfaces, product functions, user descriptions, assumptions and dependencies, constraints, specification requirements and the analysis model.

The development plan includes the project estimates and project plan.

2. Problem description

2.1 Project Purpose, Scope, and Objectives

The objective of this project is to create an interactive multi-user game similar to Parker Brother`s Monopoly. The game that will be developed will be called Montrealopoly and will contain districts and street names from the city of Montreal. The rules of the game will be similar to the original Monopoly, with some variations. Montrealopoly will be multi-user game where up to eight players can play with a combination of human and computer controlled opponents.

As a result, this will allow Team Redmond to experience the software engineering process along with its challenges of group collaboration and project management.

The project will be delivered in three phases consisting of the requirements phase, the design phase and the implementation phase. With the completion of each phase, there will be a deliverable as follows:

Phase	Deliverables	Date
Requirements	Requirements Document	September 30, 2003
Design	Design Document	October 20, 2003
Implementation	Final Document Completed Game	November 20, 2003

Project Phases

2.1.1 User interfaces

In order to create an efficient interface there needs to be a good graphical design, otherwise the user will have difficulty using it. This will result in lost productivity, user confusion and misunderstanding.

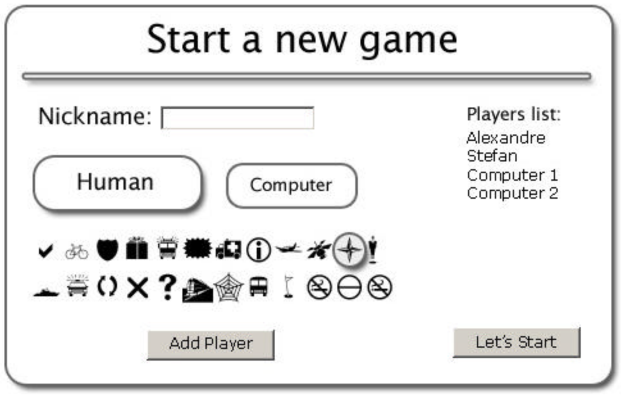
To avoid such fundamental problems, Team Redmond will design efficient and intuitive interfaces. This will include an intuitive interface that is uncluttered and easy to navigate. With the intuitive interface, the user's actions will be minimized, as the amount of mouse clicks required to perform an action will be kept to the minimum. Team Redmond will only use key or principle words for buttons and pop-up messages, which will be meaningful to the player.

As with any software product, user input validation is essential to prevent incorrect values from entering the system. All input entered into Montrealopoly will be fully validated and most data will be entered by clicking on an icon or the use of a drop down list. Only the user name and dollar amounts will be entered manually by the user through an input box, which will be validated by the system before being accepted.

Players will interact with the game through several interfaces, which will be detailed below. Through the use of intuitive interfaces, most game actions will be performed with the click of the mouse. The five main interfaces are:

- Game start & players configuration interface
- Game board
- Title Deed card
- JFL card
- Trading interface
- Winner's interface

2.1.1.1 Game Start & Players Configuration Interface



Players list:
Alexandre
Stefan
Computer 1
Computer 2

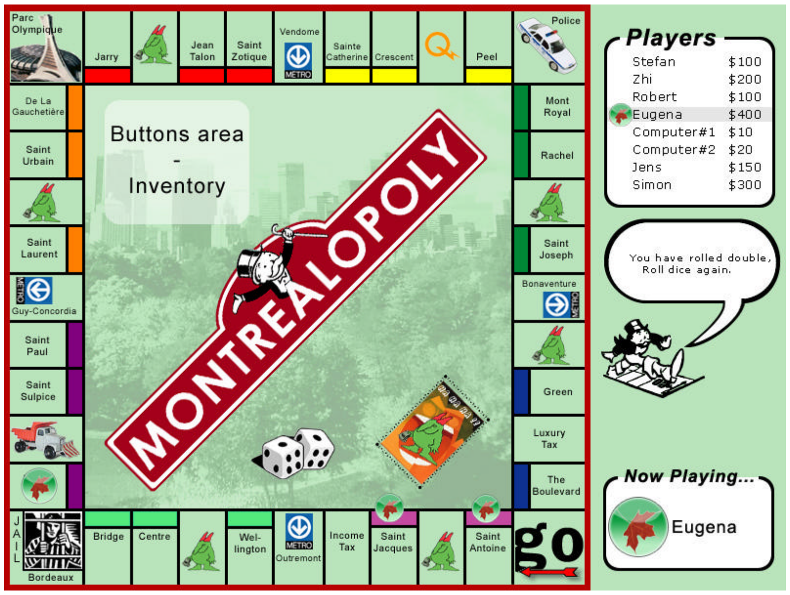
Game start and players configuration interface.

Above is the first interface, where the game is started and players enter their information. This interface decides the number of players with a minimum of two up to a maximum of eight.

For the first interface, each player enters his or her name for the game in the appropriate input box. Next, the player must choose whether the user will be a computer or human player. Then the user must pick a token that will use to represent them during the game. Finally, the "Add Player" button adds the current player to the game (he is added to the players list) and resets the form to enter a new player, and so on.

When all players have been entered, the game can start by clicking on the "Let's Start" button.

2.1.1.2 Game Board



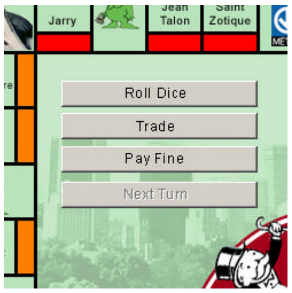
Stefan	$100
Zhi	$200
Robert	$100
Eugena	$400
Computer#1	$10
Computer#2	$20
Jens	$150
Simon	$300

You have rolled double, Roll dice again.

Board Game Interface.

The board is the main interface of the game and it will display the current state of the game. The information displayed will be fully dynamic and the following are the basic components of the game board:

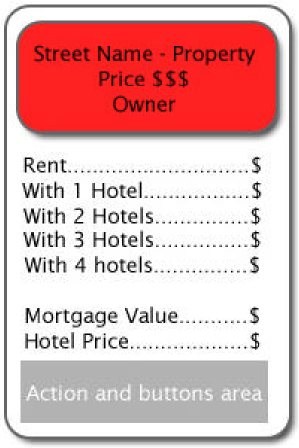
- The player's list shows the different players and the amount of money they have. Each player's token will be preceded by the user's name. The player currently at play, will be highlighted with a grey bar. The current user is also displayed in the portion of the board called "Now Playing." Both are for display purposes only.
- The thinking bubble below the player's list will display messages, created by the game's AI that will make comments in relation to the player's action. This thinking bubble will be for display purposes only.
- The board game itself is divided into three parts: 1) cells; 2) buttons area / inventory; and 3) dice area.
  - The dice area will display the result of a roll of dice.
  - Certain cells on the game board will be fully dynamic. Generally, they will react differently to a mouse click. Depending on the cell type, different actions can be performed. By clicking on a street cell, the title deed of this street will appear. By clicking on another cell nothing will happen.
  - The buttons area / inventory area is where the users will perform most of the games actions. Depending on the state of the game, different buttons will appear. When it will be the user's turn to roll the dice, a "Roll Dice" button will appear. If a player is in jail, a "Pay Fine" button will appear. There will also be a "Trade" button to trade properties and a "Next Turn" button to pass the play onto the next player. The "Next Turn" button will be disabled if a player is in debt and must pay off the debt in order to continue the game. If a user has a "Get Out of Jail" card in their possession, this card will be displayed here.



Buttons Area / Inventory.

All other actions by players such as token movement, JFL card drawing, rent and tax payments will be done automatically. Tokens will be automatically placed on the correct cell.

2.1.1.3 Title Deed Cards



Street Name - Property
Price $$$
Owner

Rent............................$
With 1 Hotel................$
With 2 Hotels...............$
With 3 Hotels...............$
With 4 hotels...............$

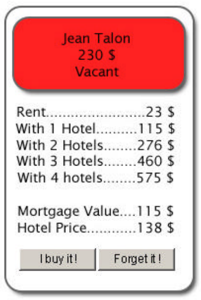
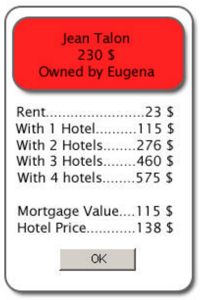
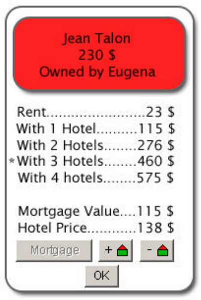
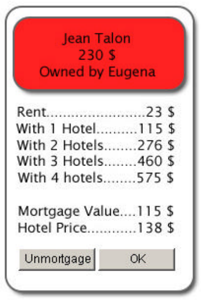
Mortgage Value..........$
Hotel Price...................$

Action and buttons area

Title Deed Interface.

Title Deed Cards will contain the purchase amount, mortgage and different rent values depending on the status of the property. These cards will have different action and button areas in relation to the current status of the game and the possibilities are:

- Landing on a vacant property will produce a Title Deed Card with two buttons that displaying two options. One button will say “I want it!” and by clicking on it will allow the player to purchase the property. The other button will say “Forget it!” and the player can choose not to purchase the property.
- At any time during the player’s turn, the player may view a Title Deed Card by clicking on its corresponding cell. The information pertaining to the property will appear and a click on the “OK” button will close the title deed.
- Once the player is the owner of a property and clicks on the corresponding cell, the action buttons displayed will allow the player to mortgage / unmortgage / buy / sell hotels. The details of the buttons are as follows:
    - o “Mortgage”: mortgage the property and will appear if the property is unimproved.
    - o “Unmortgage”: pay back the mortgage to the bank.
    - o “+Hotel”: to build a hotel and appears if there are sufficient funds.
    - o “-Hotel”: to sell a hotel back to the bank.
    - o “OK” to close the Title Deed Card
- If a property is mortgaged, the Title Deed Card will have action buttons to “Unmortgage” the property and the “OK” button to close the Title Deed Card.

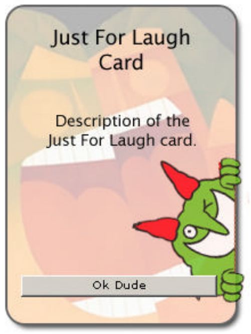
Different Types of Title Deed Cards			
 Jean Talon 230 $ Vacant Rent........................23 $ With 1 Hotel..........115 $ With 2 Hotels........276 $ With 3 Hotels........460 $ With 4 hotels........575 $ Mortgage Value....115 $ Hotel Price............138 $ I buy it! Forget it !	 Jean Talon 230 $ Owned by Eugena Rent........................23 $ With 1 Hotel..........115 $ With 2 Hotels........276 $ With 3 Hotels........460 $ With 4 hotels........575 $ Mortgage Value....115 $ Hotel Price............138 $ OK	 Jean Talon 230 $ Owned by Eugena Rent........................23 $ With 1 Hotel..........115 $ With 2 Hotels........276 $ With 3 Hotels........460 $ With 4 hotels........575 $ Mortgage Value....115 $ Hotel Price............138 $ Mortgage + - OK	 Jean Talon 230 $ Owned by Eugena Rent........................23 $ With 1 Hotel..........115 $ With 2 Hotels........276 $ With 3 Hotels........460 $ With 4 hotels........575 $ Mortgage Value....115 $ Hotel Price............138 $ Unmortgage OK
Vacant place	*Information about a title deed*	*Proprietary title deed*	*Mortgaged property*

Here are the prices and rents for the different title deeds for streets and properties in the game:

	Cost	Mortgage	Rent	Cost of a Hotel (unique, 60% of cost)	rent w/1 Hotel (50% of cost)	rent w/2 Hotels (120% of cost)	3 Hotels (200%)	4 Hotels (250%)
Burgundy								
St-Antoine	90	45	9	54	45	108	180	225
St-Jacques	100	50	10	60	50	120	200	250
Griffintown								
Wellington	130	65	13	78	65	156	260	325
Center	140	70	14	84	70	168	280	350
Bridge	150	75	15	90	75	180	300	375
Plateau								
Mont-Royal	170	85	17	102	85	204	340	425
Rachel	180	90	18	108	90	216	360	450
St-Joseph	190	95	19	114	95	228	380	475
Little Italy								
Jarry	210	105	21	126	105	252	420	525
St-Zotique	220	110	22	132	110	264	440	550
Jean-Talon	230	115	23	138	115	276	460	575
Chinatown								
St-Laurent	250	125	25	150	125	300	500	625
St-Urbain	260	130	26	156	130	312	520	650
De La Gauchet	270	135	27	162	135	324	540	675
Old Montreal								
Notre-Dame	290	145	29	174	145	348	580	725
St-Sulpice	300	150	30	180	150	360	600	750
St-Paul	310	155	31	186	155	372	620	775
Downtown								
St-Catherine	330	165	33	198	165	396	660	825
Crescent	340	170	34	204	170	408	680	850
Peel	350	175	35	210	175	420	700	875
Westmount								
Green	400	200	40	240	200	480	800	1000
The Boulevard	500	250	50	300	250	600	1000	1250
Bonavanture	300	150	30	-	-	-	-	-
Outremont	300	150	30	-	-	-	-	-
Guy-Concordia	300	150	30	-	-	-	-	-
Vendome	300	150	30	-	-	-	-	-
SnowDump	200	100	20	-	-	-	-	-
Hydro	200	100	20	-	-	-	-	-

Title Deed Prices and Rents.

2.1.1.4 JFL Cards



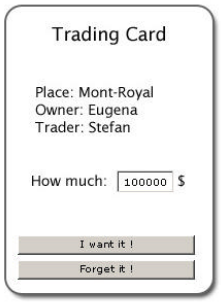
Description of the Just For Laugh card.

JFL Card Interface.

Once a player lands on a JFL cell, a JFL Card will pop-up on the game board. A JFL Card will give the player instructions that the player must follow. When the player clicks on the “OK Dude” button, the action based on the instructions will be performed. However, if the card is a “Get Out of Jail” Card, the player gets to keep the card for future use.

2.1.1.5 Trading Interface

If a player wants to purchase a property from another player, they must click on the trade button. The player will then select the property that is wanted and enter a dollar amount that the player wants to pay for it. By clicking on the “I Want It!” button will initiate the negotiations for the property. Clicking on “Forget It!” cancels the offer.



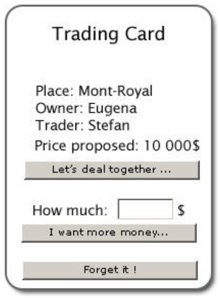
Place: Mont-Royal
Owner: Eugena
Trader: Stefan

How much: 100000 $

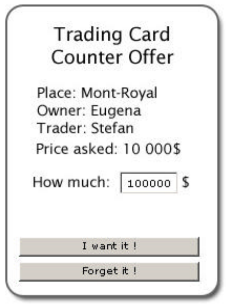
Trading Card Interface.

The owner of the property receives a pop-up with the proposal, after which the owner has several options. If the owner is interested in the trade he or she clicks on the “Let’s Deal Together” button. On the other hand, if the owner wants more money, he or she enters the amount desired in the “How much” box and then clicks on the “I Want More Money…” button. The offer can also be rejected by clicking on the “Forget about it!” button.

If a counter-offer is made the buyer will receive a pop-up with the counter-offer. The buyer then can accept or reject the offer.



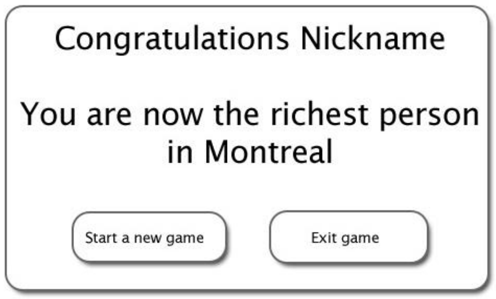
Trading Card Interface when replying.



Trading Card when counter-offering.

2.1.1.6 Winner Interface

When the game is won, the winner’s name will appear in the pop-up below. The player will then have the option of starting a new game or exiting the game.



Winner Interface.

2.2 Product Functions

Every Game function below has to support system functions, such as a click of a button, drawing objects, making sounds when necessary, producing popup screens and so on. The following functions will be part of Montrealopoly.

2.2.1 Introduction

An introductory or first function is to identify players. Thus, welcome screen sets the number of participating players and collects information on each of them. The sequence of actions is as follows:

Input

- Name of each player and the token chosen to visually represent moves of that player on the board
- Number of virtual players and corresponding tokens (or chosen randomly from the db of names and available tokens.

Action:

- An object, or an instant of each player is created, including the computer players.

Output:

- Total number of players are defined. A data structure containing all players together with their representing names and tokes is formed. Players' order is randomized. The system proceeds to the board.

Validity check:

- At least one human player has to be chosen. If no players are chosen, the information interface should appear inviting players to choose names.

- The systems should not allow blank names for any player. At least one letter/digit/ASCII character has to be entered to identify players throughout the game.

- No two players should have the same name. On accepting every player name, already accepted players are verified. If the name already exists, the player is notified with a pop-up screen and is offered to choose another name.

- No two players can choose the same token. As soon as the token is chosen, the system should disable the chosen token.

- Every player has to be either human or computerized. If the player attempts to skip this step, the system should warn the player that they must choose a type of player.

- No more than eight players in total can be chosen. If an attempt is made to choose the ninth player, the system should inform the player of the invalid action.

2.2.2 Board

The new board is generated every time a new game begins. The image of the board is displayed and all tokens are placed on the starting cell GO. The game then begins.

Input:

A collection of players together with names and tokens, each identified as human or computer.

Action:

Players and deeds are assigned to be either constant -C or variable - V (changing throughout the game) properties. JFL Cards are shuffled in random order (see JFL cards).

Every cell on the board has a corresponding number along with a deed for identification throughout the game. A deed contains information about the cell, including:

- C district
- C cost
- V owner (Bank if none)
- C rent (calculated as 10% of the cost)
- C cost of each hotel (-"- 60% of the cost)
- C rent w/1 hotel (50% of the cost)
- C rent w/2 hotels (120% of the cost)
- C rent w/3 hotels (200% of the cost)
- C rent w/4 hotels (250% of the cost)
- V mortgaged /un-mortgaged
- V number of hotels built

Each player assigned a set of properties, either constant or variable.
As the board is set, all the variable properties are set to initialized as follows:

Property	**Constant/Variable**	**Initial value**
Current position	V	(0)
Cash on hand	V	$2,000
Property owned	V	-
Rolled double	V	0
Amount of debt	V	0
In jail	V	N
In jail card	V	N

Output:

Properties set and game begins.

Validity check:

There is no human input at this point. Thus, the system has to verify its own actions:

- Players are assigned equal and valid attributes
- Tokens are placed properly
- Property are assigned their values
- The board /dice images are placed correctly
- JFL Cards have been randomly shuffled
- Order of game play stored

2.2.3 Game

Since the set of functions for every player is similar, the game mainly repeats the same path for each player in turn:

- Get player data
- Pre-roll actions
- Roll dice/move
- Post-roll actions
- Transition to next player

Input:

Player's settings: ID, current position, status, owned property and cash on hand.

Action:

Human or computer players choose further steps to perform, which can be classified as follows:

1. Roll the dice and move around the board
2. Pre-roll / Post-roll:
   Ignore
   Buy hotel(s)
   Buy property
   Sell (destroy) hotel(s)
   Sell (trade) property
   Pay rent
   Pay tax
   Take JFL Card
   Go to jail / Get out of jail
   Mortgage property/ Lift mortgage
   Check for doubles
   Collect $200 for passing through GO.

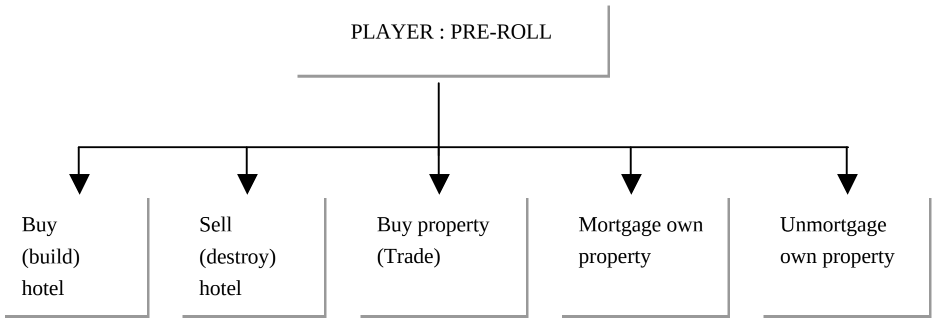
Computer player's choice defined by AI out of the above functions.

Output: some of the player's attributes have changed through the steps performed.

Validity check:

Sequence of the players is to be followed according to the order set earlier. Whenever the player finishes their turn, the next player becomes active.

2.2.4 Pre-roll



Validity check: System has to make sure that other buttons are disabled, so no improper choice is allowed. The player has to click on a button to continue the game.

For the following functions, the input contains player's attributes.

2.2.4.1 Roll Dice

This function is the same regardless of the nature of the player either human or computer.

The function consists of two sub-functions: Roll and Move, in that order. The output of Roll is a component of the input of Move.

Roll:

Action:

The system generates a sequence of images of two dice in changing positions, imitating 'rolling' and producing two pseudo-random numbers from 1 to 6. These numbers are drawn on the upper surface of each dice, representing the outcome. The sum of the numbers is calculated.

If both generated numbers are the same, the player has rolled a double. The system makes a note of that by changing the property of the player's "rolled double" variable to 1. If the player rolls three doubles in a row the GO TO JAIL method is called.

Outcome: a number representing the end result of throwing two dice is calculated and is passed to the next function.

Validity check and errors:

“Rolled double” variable has to be traced carefully. The number rolled should be checked and stored to use it in the next function before proceeding.

2.2.4.2 Move

The function simulates the movement of the player’s token on the board, one cell at a time, until the destination cell is reached.

Input: a number from 2 to 12; the player’s current position.

Action:

The total produced by rolling is added to the player’s current position, represented by the board cell’s number, and the result is assigned as a player’s new current position on the board. The system removes the player’s token from the cell where it is presently situated and moves it to the destination cell.

Output:

Player’s token is shifted on the board. This shift corresponds to the number that has been rolled.

Validity check:

The number of cells the token is moved should be the same as the number that was rolled. If doubles were rolled once, the “doubles rolled” variable should contain the amount of successive rolled doubles. On the third doubles roll, the player is placed in jail and the status of the player should be IN JAIL.

2.2.5 Check if passed go

Before proceeding, the system checks if the player has passed the GO cell during the last move. This is done by checking if the current position is less than the previous position and if the player passed the cell zero. In this case, the system displays a notification and $200 is added to the player’s account. Otherwise, the numeric value of current position is greater than that of the previous one and no actions are taken.

Input: player’s previous and current positions.

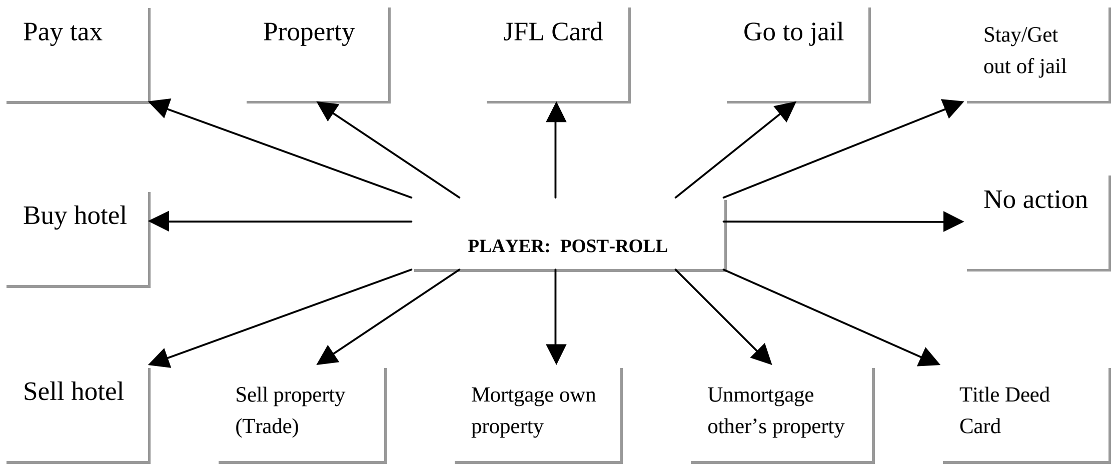
Action: has cell zero been passed? If yes, add $200 to player’s cash. If no, proceed.

Output: Player receives $200 if they pass or land on the GO cell.

Validity check:

Software has to verify if the advance through GO was not invokes by JFL “Return” card. In that case, nothing is added to player’s account.

Post-roll actions
According to the properties of the cell, the player has a following choice of actions:



2.2.6 Tax

Player has to pay a certain amount a tax.

Input: Player's property, cell property.

Action:
Income tax:

The player has an option of either paying $200 or 10% of the total worth. The smallest amount is calculated automatically. Costs of all properties that have players as an owner are added up, 10% of the sum is added and the amount is compared to $200. The smaller sum is subtracted from the player's account.

Output: player's cash is automatically subtracted from the player's account.

Validity check:

The software has to verify if all the properties costs were added (and 10% is calculated). There is no user input at this point and no error should occur. However, if the amount of a player's cash is unable to cover the tax payment, the player is given a choice of either selling hotels, mortgaging their property or selling property. A computerized player makes the same choice.

2.2.7 Luxury Tax

Input: Player's amount of cash, cell's property

Action: player's account is reduced by $75

If the player has not enough cash to cover the tax, he or she is given the choice of selling hotels, mortgaging property or selling property.

At this point the game makes a distinction between a computer and human player. While a real player chooses next action according to his or her understanding of the game, the AI makes the decision for computer opponents.

2.2.8 Selling hotels

If a user wants to sell a hotel, he or she has to choose a property that has hotels. By clicking on the desired property, the player obtains a list of hotels on it. The system checks if the choice is valid and notifies the player of the price (which is calculated as ½ of the regular price of the hotel on that property). The computer player's AI will sell hotels that are valued the least first.

Validity check and errors:

Before proceeding, the game has to check:

- If the property belongs to the player
- If the property is mortgaged
- If the property is improved (if there are hotels on that property to destroy)
- If other properties have more hotels (the player has to sell hotels evenly, one from the property)

If any of the conditions are not satisfied, the system should notify the player with a pop-up message indicating the error.

For the computer player, the system carries on verifications before making a choice.

2.2.9 Buying properties

A player can attempt to buy a property from another player. The buyer chooses a trading partner, the property to buy and makes an offer.

The seller can either accept the offer, refuse it or make a counter-bid. The buyer can then accept or refuse the amount and make a counter-offer.

COMPUTER PLAYER

The computer player can buy property when it already has one or two properties in the same district.

Validity check:

Before allowing the sell option, system checks:

- If the property belongs to the seller
- If the property is improved (contains hotels)

In the case where a validity check is not satisfied, the player is notified with a pop-up message.

While trade is in progress, the game has to make sure that all trades amount to something positive, as zero is invalid and all amounts are integers not exceeding a logical limit of 100*price. If an error occurs, a descriptive message is produced and the last erroneous action is restarted.

2.2.10 Mortgage property

In a situation where available funds are not sufficient, a player can mortgage any of his or her unimproved properties.

Input: the player's attributes property to mortgage.

Action: the bank finances the player for ½ of the cost of the property.

A player chooses the street, clicks on the "mortgage" button and the bank accepts the conditions of payback – full cost plus 10%. From this point on, the property is considered mortgaged and no hotels can be built on it. Moreover, no rent can be charged from the other players until the street is unmortgaged.

Output: the player's account is increased ½ of the cost of the property. Hotels may not be built and rent cannot be collected until it unmortgaged.

Validity check:

Property has to belong to the owner and be unimproved.

2.2.11 Unmortgage

Input: property to unmortgage, player's attributes.

Action: When the player retains necessary funds, he or she can unmortgage their properties at 110% of full cost. The player can now collect rent and buy hotels for the property.

Output: 110% of mortgaged property cost is subtracted from the player's account.

Validity check:

To unmortgage a property, the player has to be the owner of the property and have enough available money. If these conditions are not met, a pop-up message will notify the player of the error.

2.2.12 Jfl card

If the player lands on one of the JFL cells, he or she is obliged to take a JFL card which instructs the player to perform the action on the card.

The system stores cards in pseudo-random order, but retrieves them one by one, by the FIFO principle. This is done to ensure that no card is given to two players consecutively.

There are 5 possible sets of actions to perform:

-Pay a certain amount to the bank or other player(s)
-Receive a certain amount from the bank or other player(s)
-Advance or return to a particular cell on the Board
-"Get Out of Jail free"
-Keep the "Get Out of Jail" Card

Input: player's attributes, card attributes.

The instructions on the card carried out.

Output: according to the indicated action:

- Player's account is increased by the amount indicated
- Player's account is reduced by the corresponding amount and other players' accounts are increased by the same amount
- Player's current position is changed to the one indicated on the card

2.2.13 Buying hotels

This function gives an opportunity to build. No hotels can be built on utility squares or metro stations, even if the player owns the whole set.

As soon as the player acquires all streets of the same district, the game inquires if he or she wants to build a hotel on one of the streets of that district. Once a player has confirmed a street to construct a hotel on, paid the amount, the hotels are built sequentially on each street with a maximum of four hotels per street.

Input: player's attributes, street attributes.

Action: player confirms the decision to build a hotel by clicking on BUILD HOTEL button.

Output: hotel cost is deducted from player's account, an image of a hotel is drawn on the street cell on the Board and rent property is raised according to the values presented in section 2.1.1.

Validity check and errors:

To construct a hotel on a street, the player has to own a full district where that street belongs. The game should check if the property is a street or if it belongs to the owner or if other properties of the same district or no more than four hotels, and if the player has available money to acquire a hotel. The player is notified with a pop-up message if the conditions are not satisfied.

2.2.14 Pay rent

When a player finds his or her token on a street, utility or metro station that has an opponent as an owner, the player has to pay rent to the owner. Rent is calculated as a percentage of the total street (utility, metro) cost and is payable immediately.

Input: player's and street's attributes.

Action: Rent is deducted automatically from the player. If the player lacks money to cover the rent, he or she has to sell their own hotels or mortgage property.

If the property is mortgaged, no rent is paid.

Output: amount of the rent is subtracted from current player's cash and added to that of the property's owner.

Validity check:

Before notifying the player of the amount of rent to pay, the system verifies:

- If the property is owned
- If the property belongs to another player
- If it is mortgaged

2.2.15 Ignore

At times no actions are required of the player, depending if they are on the OLYMPIC STADIUM cell, JAIL cell or any of the player's own property squares.

2.2.16 Go to jail

If a player's token ends up on GO TO JAIL cell, rolls 3 consecutive doubles or gets a JFL Card instructing the player to "Go To Jail" he or she is forced to reposition their token on the JAIL cell.

Output: The play is passed onto the next player.

2.2.17 Declare bankruptcy

The player clicks on the 'declare bankruptcy' button at any time of the game, if he or she does not want to continue or thinks that he or she does not have enough money. Bankruptcy is declared automatically when the player cannot cover their debt. The player is then withdrawn from the game.

Input: Player's attributes

Action: All hotels the player owns are destroyed and the player no longer owns his properties.

Output:

Player has zero possessions and is blocked from further game.

2.2.18 End game

If there is only one player left and the rest have declared bankruptcy the remaining player is declared the winner.

During the game, any player can end the game by clicking on the "End Game" button.

Input: 'end game' button' is clicked.

Action: total worth of all players' possessions are calculated and the player with greatest worth is identified as the winner.

Output: the game produces an interface with the name of the winner. The player will then have the option of starting a new game or exiting the game.

2.3 User Description

2.3.1 User Environment

<u>Game Environment</u>

We have already seen how users will interact with the interfaces of the game, and now, the game's task will be examined. During a player's turn, several tasks can be performed of which some have immediate feedback. Non-immediate tasks require interaction from other users in the game and are dependent on them in order to complete the current task.

Examples of the two types of tasks are detailed below:

Immediate Tasks (immediate feedback)	Non-Immediate Tasks
• Dice throwing • Token moving • Property buying • Hotel building • Property mortgage • Property unmortgage • JFL Card picking • Tax payment (income tax, property tax, etc) if player has enough money	• Initialization of the game. Each player chooses a name, a status (human or computer), a token and then adds the player to the players list • Property trading: depending on the two players. It takes at least two operations in the best case: to make a proposal and accept/reject it. If an owner asks for more money, then the task will take longer as it will be resubmitted to the buyer for a new proposal, and so on. Two players are involved in this task. • Debt payment. If a player has not enough money to pay a tax, then they will have to sell hotels, properties or mortgage them. Duration time will depend on the amount of money he has to collect.

<u>Users System Environment</u>

The game will run on a PC with Microsoft Windows 9X, 2000 and XP.

The game will include an install program that will install all the required components of the game (including graphics, DLLs, etc.). No other special tools will be required to run the game (for instance Flash Player or Java Virtual Machine) as it will be programmed in Visual Basic.

<u>For further details, see the topic 2.6.2.2 Software Requirements</u>

<u>Users Hardware Environment</u>

<u>See the topic 2.6.2.1 Hardware Requirements</u>

2.3.2 User Profiles

In order to play Montrealopoly, knowledge of using Microsoft Windows is all that is required. The user interfaces are intuitive and online help is readily available and the game board's thinking bubble will indicate what actions are to be performed.

The objective of the Montrealopoly game is to make the game accessible to people of all cultures, creeds and both sexes. However, the minimal age for the game is slated for 10 years of age and older. By this age, the simple concepts of buying and selling should be understood and this basic understanding is required.

2.4 Assumptions and Dependencies

There are several assumptions for Montrealopoly.

1 The Operation system: Windows 9X, 2000, and XP.
2 The user is able to use the Windows operating system.
3 The user possesses a working knowledge of computers.
4 The bank in the game has an unlimited amount of money and cannot become bankrupt.
5 The game will support up to 8 players simultaneously.

2.5 Constraints

Considering that the majority of PC's of players run the Windows OS, the Montrealopoly must be written in a language that supports this platform and must be able to implement a GUI. Visual Basic has been chosen as the programming language.

2.6 Specific Requirements

2.6.1 Applicable Standards

Variable name should follow this rule:

Attributes name: each word other than the first word should be capitalized. For example: numberOfDices.
Method name: should contain a verb and non-verb and each word other than the first word should be capitalized. For example, isWinner().
Class name: noun or verb.

2.6.2 System Requirements

2.6.2.1 Hardware requirements

<u>Minimum Requirements:</u>

- PC with at least 300 MHz Processors
- Mouse
- Sound card
- 128 MB RAM
- Since the systems will use images as background and wav file as background music, storage requirements would be at least 10MB and not more than 100MB
- No Internet connection required, as it is not an online game

<u>Optimal Performance:</u>
For optimal performance, a Pentium III processor running at 800 MHz with 256 MB RAM with a minimum of 500 MB of free disk space is recommended.

2.6.2.2 Software Requirements

Several types of software are required for the Montrealopoly game. Before installation, the game must be extracted from a zip file. In order to do this, PKZIP or Winzip is required. For the game's music, the sound card must be able to playback MIDI files and the proper sound card drive must be installed.

2.6.3 Performance Requirements

Reliability – It is crucial that Montrealopoly game is error free throughout the game and this requires proper exception-handling mechanisms for all possible errors.

Efficiency – In order not to impede on the game flow, game response time must take less than a second. Otherwise game play will lag, which may result in disinterest among players. Therefore, the graphical user interface must not be bloated, so that each pop-up interface is instantaneous.

User Friendliness – The graphical user interface must be pleasing to the eye, uncluttered and intuitive to use. Along with simple game rules and quick response times, will make the game user friendly. Moreover, game installation should not take more than a few mouse clicks.

Accuracy - The game will accurately calculate the money of all the players and keep track of who owns which properties along with their status.

2.6.4 Environmental Requirements

Security – The installation of Montrealopoly will not compromise the existing security of one's personal computer. Only essential system updates will be made during the installation of the game, which will be reversed once uninstalled.

Maintainability – The design of Montrealopoly will be in a modular form allowing many developers to work on different modules concurrently. Documentation will be extensive in order to facilitate future changes to the code. All methods should have comments that describe the main function of the method, its input, its output and any relevant methods. Every programmer working on the source code will have to include their version number of the code, date of modification and what modifications were made in the header of the source code.

Portability - Montrealopoly will be installed on many versions of operating systems. However one of the design constraints is that the host systems will need to be running a 32-bit version of Windows. Thus, Montrealopoly will be portable only in terms of running on Windows-compatible machines. This does not include Windows NT.

Traceability - Each method will have an ID number associated with it. This feature can also be seen in the diagram in the following section.

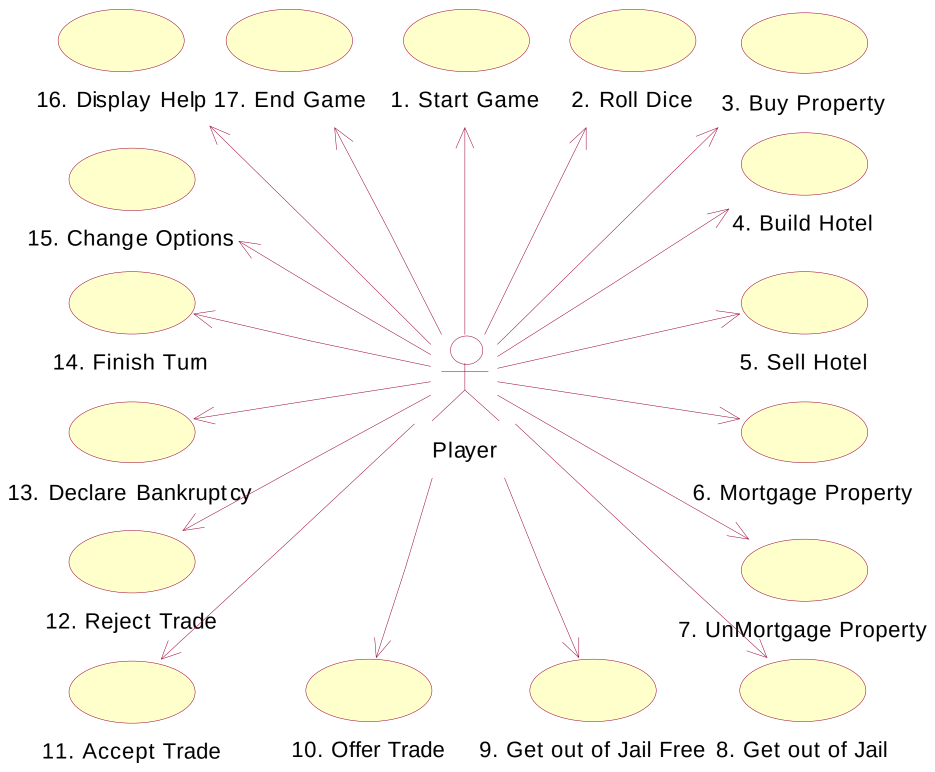
2.7 Analysis Models

2.7.1 Use Case Diagrams

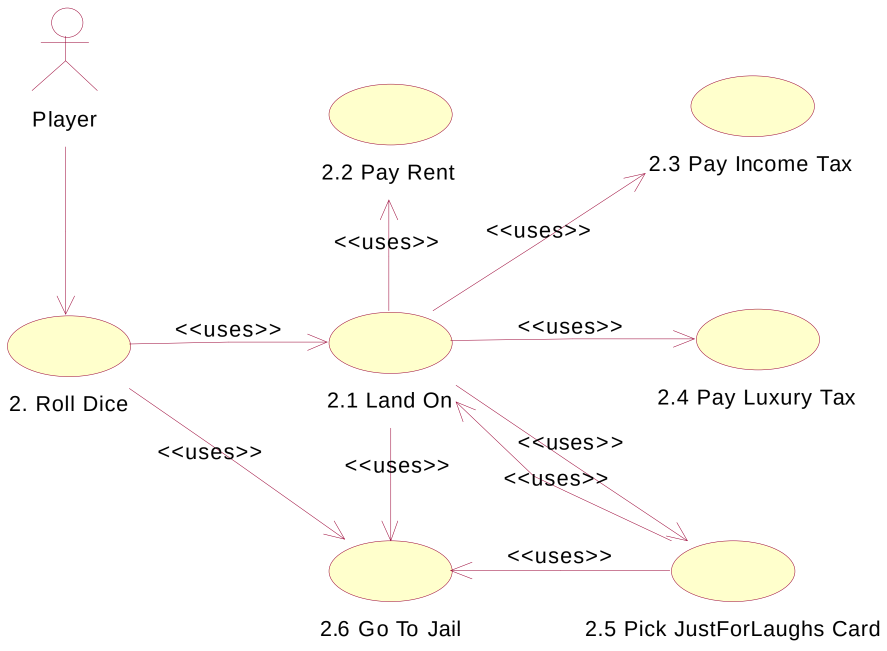
The following diagrams will help provide an overview of the functions of the game. They describe the actions that a player can perform, as well as the interaction between some of the system functions, which are not directly controlled by the player. For example, the player can choose whether or not to buy an un-owned deed he or she lands on. However, the player cannot choose whether or not to pay rent if he or she lands on a deed that is owned by another player.

Despite the simplicity of these use case diagrams, they have been included for their importance in defining the user-software interactions and the requirements and scope of the system.

2.7.1.1 The Player Use Cases



2.7.1.2 Roll Dice Use Case



2.7.2 Use Case Details

2.7.2.1 Use Case 1: Start Game

Description	The user is allowed to enter the parameters of the game.
Actors	Player
Pre-Conditions	None
Flow of Events	
Basic Path	1. The user selects the option “Start Game”. 2. The user enters the number of players in the game. 3. For each player, the user selects the following information: • Player’s name • Player’s token • Player type (computer or human). 4. The board is initialized. 5. The JustForLaughs cards are shuffled. 6. The player’s receive 1500$ each. 7. The players’ order is randomized. 8. The turn is given to the first player.
Alternative Paths	None
Post-Conditions	• The board has been initialized. • The first player can play his turn.
Related Use Cases	
Used Use Cases	None
Extending Use Cases	None

2.7.2.2 Use Case 2: Roll Dice

Description	Allows the player to roll the dice.
Actors	Player
Pre-Conditions	• It is the player's turn to play. • The player has not rolled the dice yet, or has rolled doubles only once or twice in a row since the beginning of his turn.
Flow of Events	
Basic Path	1. The player selects the option to roll the dice. 2. Two dice are rolled. 3. The player's token is moved the number of spaces indicated by the dice. The player lands on a square, as described by Use Case "Land On".
Alternative Paths	Alternative 1: • If the player is in jail and rolls doubles, then the player gets out of jail, and his token is moved accordingly. • Even though he rolled doubles, the player does not roll again until next turn. Alternative 2: • If the player is in jail and has failed to roll doubles for 3 turns, then the player must pay a 50$ fine to the bank, and gets out of jail. • The player's token is moved the number of squares indicated by the last dice roll. • If the player's balance becomes negative (i.e. the player did not have enough money to pay the fine), the player must find some means of paying his debt. • The game will not proceed until one of the following has occurred: the player has paid his debt (balance >= 0) or the player declares bankruptcy. • In the latter case, all of the bankrupt player's assets are transferred to the bank. Any properties he owned become un-owned properties. Any hotels on those properties are destroyed. Alternative 3: • After step 2, if the player has rolled doubles three times in a row, the player goes to jail. His token is moved accordingly. This is described in Use Case "Go To Jail". Alternative 4: • After step 2, if the player rolls doubles for the first or second times during his turn, he is allowed to roll the dice again before the end of his turn.
Post-Conditions	• The dice has been rolled.
Related Use Cases	
Used Use Cases	Land On, Go To Jail
Extending Use Cases	None

2.7.2.2.1 Use Case 2.1: Land On

Description	The player lands on a square.
Actors	None
Pre-Conditions	• It is the player's turn to play. • The player has rolled the dice. • The player's token has been moved.
Flow of Events	
Basic Path	1. If the player lands on a square owned by an opponent, the player pays rent. This is described in Use Case "Pay Rent". 2. If the player lands on an Income Tax square, the player pays income tax. This is described in Use Case "Pay Income Tax". 3. If the player lands on a Luxury Tax square, the player pays luxury tax. This is described in Use Case "Pay Luxury Tax". 4. If the player lands on a JustForLaughs square, the player picks a JustForLaughs card. This is described in Use Case "Pick JustForLaughs Card". 5. If the player lands on a Go To Jail square, the player goes to jail. This is described in Use Case "Go To Jail". 6. If the player has passed the Go square, the player collects 200$.
Alternative Paths	None
Post-Conditions	• The player has taken appropriate action depending on where he landed.
Related Use Cases	
Used Use Cases	Pay Rent, Pay Income Tax, Pay Luxury Tax, Pick JustForLaughs Card, Go To Jail
Extending Use Cases	None

2.7.2.2.2 Use Case 2.2: Pay Rent

Description	The player lands on an owned property and pays rent to the owner.
Actors	None
Pre-Conditions	• It is the player's turn to play. • The player has rolled the dice. • The player's token has landed on a property (deed) that is owned by another player (referred to as owner here) and is not mortgaged.
Flow of Events	
Basic Path	1. The player pays to the owner the rent amount indicated by the property's card. This amount also varies depending on the number of hotels build on the property.
Alternative Paths	Alternative 1: • After step 1, if the player's balance becomes negative (ie the player did not have enough money to pay the rent), the player must find some means of paying his debt. • The game will not proceed until one of the following has occurred: the player has paid his debt (balance >= 0) or the player declares bankruptcy. • In the latter case, all of the bankrupt player's assets are transferred to the owner (the player to whom he owes money).
Post-Conditions	• The player has paid the rent value to the owner. • The player's balance is positive or the player has declared bankruptcy.
Related Use Cases	
Used Use Cases	None
Extending Use Cases	None

2.7.2.2.3 Use Case 2.3: Pay Income Tax

Description	The player lands on an Income Tax square and pays income tax.
Actors	None
Pre-Conditions	• It is the player's turn to play. • The player has rolled the dice. • The player's token has landed on an Income Tax square.
Flow of Events	
Basic Path	1. The player pays to the bank the smallest of: • 200$ • 10% of all the player's current assets.
Alternative Paths	Alternative 1: • After step 1, if the player's balance becomes negative (ie the player did not have enough money to pay the tax), the player must find some means of paying his debt. • The game will not proceed until one of the following has occurred: the player has paid his debt (balance >= 0) or the player declares bankruptcy. • In the latter case, all of the bankrupt player's assets are transferred to the bank. Any properties he owned become un-owned properties. Any hotels on those properties are destroyed.
Post-Conditions	• The player has paid the value of the income tax.

	• The player's balance is positive or the player has declared bankruptcy.
Related Use Cases	
Used Use Cases	None
Extending Use Cases	None

2.7.2.2.4 Use Case 2.4: Pay Luxury Tax

Description	The player lands on a Luxury Tax square and pays luxury tax.
Actors	None
Pre-Conditions	• It is the player's turn to play. • The player has rolled the dice. • The player's token has landed on a Luxury Tax square.
Flow of Events	
Basic Path	1. The player pays to the bank the fixed amount of 75$.
Alternative Paths	Alternative 1: • After step 1, if the player's balance becomes negative (i.e. the player did not have enough money to pay the tax), the player must find some means of paying his debt. • The game will not proceed until one of the following has occurred: the player has paid his debt (balance >= 0) or the player declares bankruptcy. • In the latter case, all of the bankrupt player's assets are transferred to the bank. Any properties he owned become un-owned properties. Any hotels on those properties are destroyed.
Post-Conditions	• The player has paid the value of the luxury tax. • The player's balance is positive or the player has declared bankruptcy.
Related Use Cases	
Used Use Cases	None
Extending Use Cases	None

2.7.2.2.5 Use Case 2.5: Pick JustForLaughs Card

Description	The player picks a JustForLaughs card at random.
Actors	None
Pre-Conditions	• It is the player's turn to play. • The player has rolled the dice. • The player has landed on a JustForLaughs square.
Flow of Events	
Basic Path	1. The next JustForLaughs card (top card) from the JustForLaughs card deck is retrieved. 2. The actions described by the card are executed. 3. The card is returned to the bottom of the deck.
Alternative Paths	Alternative 1: • In step 2, if the card is a Pay card, the player pays the amount specified on the card to the bank. Alternative 2: • In step 2, if the card is a Collect card, the player collects the amount specified on the card from the bank. Alternative 3: • In step 2, if the card is an Advance To card, the player's token is moved to the square specified by the card. • The player lands on that square, as described in Use Case "Land On". Alternative 4: • In step 2, if the card is a Go Back card, the player's token is moved back the number of squares indicated by the card. • The player lands on that square, as described in Use Case "Land On". Alternative 5: • In step 2, if the card is a Get out of Jail Free card, the player keeps the card and may use it at a later time, as described in Use Case "Get out of Jail Free". • In step 3, the card is not returned to the deck. Alternative 6: • In step 2, if the card is a Go To Jail card, the player goes to jail, as is described in Use Case "Go To Jail".
Post-Conditions	• The player has executed the actions described by the JustForLaughs card. • The JustForLaughs card has been returned to the bottom of the deck or is kept with the player if it is a Get out of Jail Free card.
Related Use Cases	
Used Use Cases	Land On, Get out of Jail Free, Go To Jail
Extending Use Cases	None

2.7.2.2.6 Use Case 2.6: Go To Jail

Description	The player goes to jail.
Actors	None
Pre-Conditions	• It is the player's turn to play. • The player has rolled the dice. One of the following has occurred: • The player's token has landed on the "Go To Jail" square. • The player picked a "Go To Jail" JustForLaughs card. • The player has rolled doubles three times consecutively in the same turn.
Flow of Events	
Basic Path	1. The player goes to jail and is considered "in jail", as opposed to "just visiting". 2. The player's token is moved accordingly.
Alternative Paths	None
Post-Conditions	• The player is in Jail. • The player's token is "in" the Jail square.
Related Use Cases	
Used Use Cases	None
Extending Use Cases	None

2.7.2.3 Use Case 3: Buy Property

Description	Allows the player to buy an un-owned property.
Actors	Player
Pre-Conditions	• It is the player's turn to play. • The player has rolled the dice. • The player has landed on an un-owned property (deed). This can be a Street, Utility or Metro square.
Flow of Events	
Basic Path	1. The player selects the option to buy the property. 2. The player pays the price of the property to the bank. 3. The player becomes the owner of the property.
Alternative Paths	None
Post-Conditions	• The player has paid the price of the property to the bank. • The player is the owner of the property.
Related Use Cases	
Used Use Cases	None
Extending Use Cases	None

2.7.2.4 Use Case 4: Build Hotel

Description	Allows the player to build hotels on his streets.
Actors	Player
Pre-Conditions	• It is the player’s turn to play. • The player owns the district to which the street he wants to build on belongs.
Flow of Events	
Basic Path	1. The player selects the street on which he wants to build a hotel. 2. The player can add hotels, up to a maximum of 4 hotels per street, and as much as he can afford. 3. The player pays the price of the hotels to the bank.
Alternative Paths	None
Post-Conditions	• The specific number of hotels has been built on the street. • The player has paid the price of the hotel.
Related Use Cases	
Used Use Cases	None
Extending Use Cases	None

2.7.2.5 Use Case 5: Sell Hotel

Description	Allows the player to sell one or more hotels to the bank.
Actors	Player
Pre-Conditions	• It is the player’s turn to play. • The player has at least one hotel built on the street in question.
Flow of Events	
Basic Path	1. The player selects the street on which he wants to sell a hotel. 2. The player can sell all the hotels on the street. 3. The player is refunded half the price of each hotel he sells.
Alternative Paths	None
Post-Conditions	• The specific number of hotels has been sold to the bank. • The player has been refunded for the hotels.
Related Use Cases	
Used Use Cases	None
Extending Use Cases	None

2.7.2.6 Use Case 6: Mortgage Property

Description	Allows the player to mortgage a property he owns.
Actors	Player
Pre-Conditions	• It is the player’s turn to play. • The player owns the property in question. • All the properties in the district are un-improved (no hotels).
Flow of Events	
Basic Path	1. The player selects the property he wants to mortgage. 2. The player collects the mortgage value from the bank. 3. The property is now mortgaged, and the player cannot collect rent when opponents land on it.
Alternative Paths	None
Post-Conditions	• The property has been mortgaged. • The player has received the mortgage value of the property.
Related Use Cases	
Used Use Cases	None
Extending Use Cases	None

2.7.2.7 Use Case 7: Un-Mortgage Property

Description	Allows the player to un-mortgage a property he owns.
Actors	Player
Pre-Conditions	• It is the player’s turn to play. • The player has mortgaged the property in question.
Flow of Events	
Basic Path	1. The player selects the property he wants to un-mortgage. 2. The player pays the property’s mortgage value + 10% to the bank. 3. The property is now un-mortgaged, and the player can collect rent.
Alternative Paths	None
Post-Conditions	• The property has been un-mortgaged. • The player has paid back the mortgage to the bank.
Related Use Cases	
Used Use Cases	None
Extending Use Cases	None

2.7.2.8 Use Case 8: Get out of Jail

Description	Allows the player to pay $50 and get out of jail.
Actors	Player
Pre-Conditions	• It is the player's turn to play. • The player is in jail. • The player has not rolled the dice during this turn.
Flow of Events	
Basic Path	1. The player selects the option to use the Get out of Jail. 2. The player pays a $50 fine to the bank. 3. The player gets out of jail. His token is moved to the "Just Visiting" part of the Jail square. 4. The player rolls the dice; his token is moved the number of squares indicated by the dice. 5. The player lands on a square, as described in Use Case "Land On".
Alternative Paths	None
Post-Conditions	• The player is out of jail. • The player has paid the fine. • The player's token has been moved.
Related Use Cases	
Used Use Cases	Land On
Extending Use Cases	None

2.7.2.9 Use Case 9: Get out of Jail Free

Description	Allows the player to use his Get out of Jail Free card to get out of jail.
Actors	Player
Pre-Conditions	• It is the player's turn to play. • The player is in jail. • The player has rolled the dice and did not roll doubles. • The player has a Get out of Jail Free card.
Flow of Events	
Basic Path	1. The player selects the option to use the Get out of Jail Free card. 2. The Get out of Jail Free card is returned to the bottom of the JustForLaughs Card deck. 3. The player gets out of jail. His token is moved the number of squares indicated by the dice. 4. The player lands on a square, as described in Use Case "Land On".
Alternative Paths	None
Post-Conditions	• The Get out of Jail Free card has been returned to the JustForLaughs Card deck. • The player is out of jail. • The player's token has been moved.
Related Use Cases	
Used Use Cases	Land On
Extending Use Cases	None

2.7.2.10 Use Case 10: Offer Trade

Description	The player (initiator) makes a trade offer to another player (opponent).
Actors	Player
Pre-Conditions	• It is the initiator's turn to play.
Flow of Events	
Basic Path	1. The initiator selects the option to make a trade offer. 2. The initiator selects the properties that he wants to buy. These properties must be unimproved (no hotels). 3. The initiator enters the amount (if any) that he is willing to pay. 4. The initiator selects the option to send the trade offer to the other player.
Alternative Paths	In step 6, if the initiator enters an amount which exceeds his current balance, a message will indicate to the player that he is not allowed to do that.
Post-Conditions	• An option is displayed to allow the opponent to accept/reject the offer.
Related Use Cases	Accept Trade, Reject Trade
Used Use Cases	None
Extending Use Cases	None

2.7.2.11 Use Case 11: Accept Trade

Description	The player accepts a trade offer that has been made by an opponent (initiator).
Actors	Player
Pre-Conditions	• A trade offer has been made to the current player (by the initiator). • An option has been displayed to allow the current player to accept/reject the offer.
Flow of Events	
Basic Path	1. The player is prompted to accept or refuse the offer. 2. The player selects "Accept". 3. Ownership of the properties involved in the trade is transferred between the current player and the initiator of the trade. 4. The amounts involved in the trade are transferred between the current player and the initiator of the trade. 5. The opponent player (initiator of the trade) is allowed to continue to play.
Alternative Paths	
Post-Conditions	• The trade has been processed. The properties and money have been exchanged. • The control has been returned to the trade initiator.
Related Use Cases	Offer Trade
Used Use Cases	None
Extending Use Cases	None

2.7.2.12 Use Case 12: Reject Trade

Description	The player rejects a trade offer that has been made by an opponent (initiator).
Actors	Player
Pre-Conditions	• A trade offer has been made to the current player (by the initiator). • An option has been displayed to allow the current player to accept/reject the offer.
Flow of Events	
Basic Path	1. The player is prompted to accept or refuse the offer. 2. The player selects “Reject”. 3. The opponent player (initiator of the trade) is allowed to continue to play.
Alternative Paths	Alternative 1: • In step 2, the player can make a counter offer, and indicate the price he wants for the property. • The control passed back to the initiator, who can then accept/decline the price.
Post-Conditions	• The control has been returned to the trade initiator.
Related Use Cases	Offer Trade
Used Use Cases	None
Extending Use Cases	None

2.7.2.13 Use Case 13: Declare Bankruptcy

Description	The player declares bankruptcy and is withdrawn from the game.
Actors	Player
Pre-Conditions	• It is the player's turn to play. • The player owes money (balance < 0).
Flow of Events	
Basic Path	1. The player selects the option "Declare Bankruptcy". 2. The player's hotels (if any) are sold to the bank for half price. 3. The player's properties are mortgaged. 4. The player's properties are given to the opponent to whom the debt is owed. 5. The player's balance (even if balance < 0) is added to the opponent's balance. 6. The player is withdrawn from the game (does not get a turn). 7. The next player is allowed to play.
Alternative Paths	Alternative 1 • In step 2, if the player's debt is owed to the bank, then the bankrupt player's assets go to the bank. Any hotels on his properties are destroyed, and the properties he owns become un-owned. • The same applies to steps 4 and 5 (everything is given to the bank). Alternative 2 • After step 6, if there is only 1 player left in the game, then the game ends and the winner is that player.
Post-Conditions	• The turn has been given to the next player. or • A winner has been declared and the game ends.
Related Use Cases	
Used Use Cases	None
Extending Use Cases	None

2.7.2.14 Use Case 14: Finish Turn

Description	The player indicates that he has finished his turn.
Actors	Player
Pre-Conditions	• It is the player's turn to play. • The player has rolled the dice. • Either the player's last roll was not doubles or the player has rolled 3 doubles consecutively in the same turn.
Flow of Events	
Basic Path	1. The player selects the option "Finish Turn". 2. The next player is allowed to play.
Alternative Paths	After step 1, if the player's balance is negative, a message is displayed to the player indicating that he must pay his debts before proceeding, or he must declare bankruptcy.
Post-Conditions	• The turn has been given to the next player.
Related Use Cases	
Used Use Cases	None

Extending Use Cases	None

2.7.2.15 Use Case 15: Change Options

Description	The player is allowed to change the game options.
Actors	Player
Pre-Conditions	None
Flow of Events	
Basic Path	1. The player opens the Options window. 2. The player may disable/enable background music. 3. The player may disable/enable sound effects. 4. The player may disable/enable time delay (slows down the processing done by the computer player)
Alternative Paths	None
Post-Conditions	• The options changed by the user have been updated.
Related Use Cases	
Used Use Cases	None
Extending Use Cases	None

2.7.2.16 Use Case 16: Display Help

Description	Displays a help window to the player.
Actors	Player
Pre-Conditions	None
Flow of Events	
Basic Path	1. The player selects the "Help" option. 2. A window is displayed showing the user's manual for the game.
Alternative Paths	None
Post-Conditions	• The user's manual has been displayed.
Related Use Cases	
Used Use Cases	None
Extending Use Cases	None

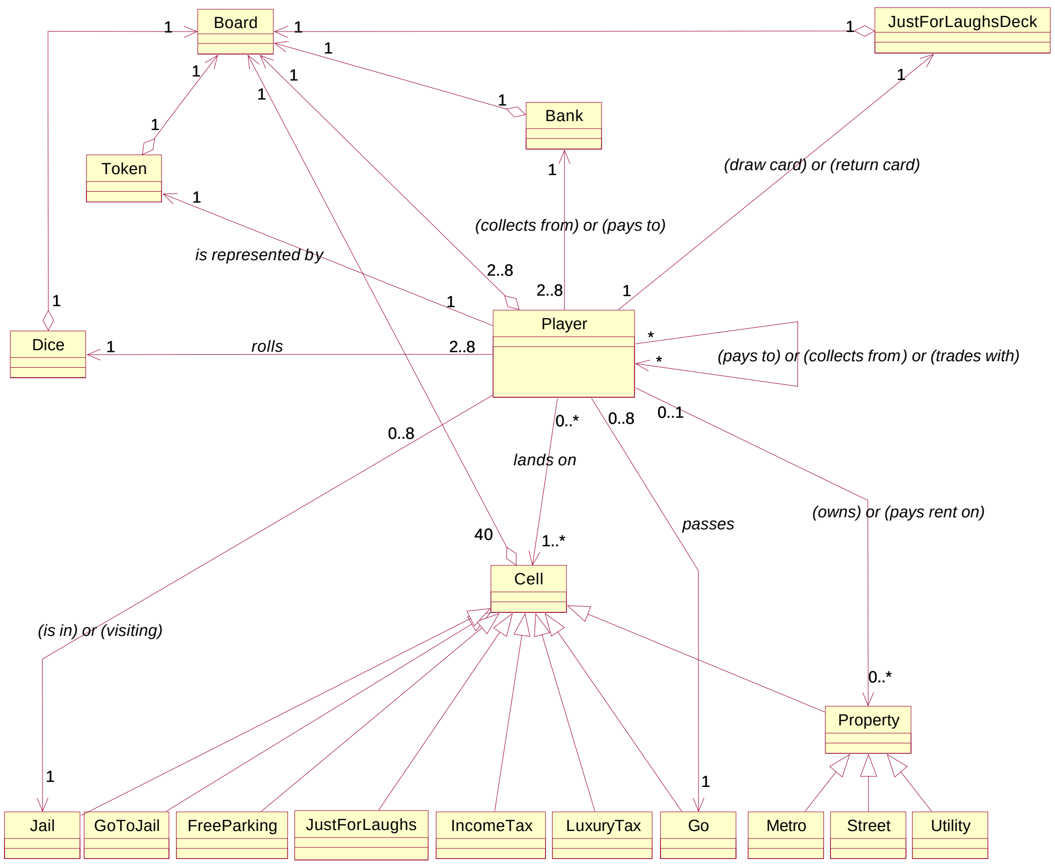
2.7.2.17 Use Case 17: End Game

Description	The player indicates that he wants to end the game.
Actors	Player
Pre-Conditions	None
Flow of Events	
Basic Path	1. The player selects the option “End Game”. 2. A confirmation dialog is shown to avoid an accidental click. 3. The user confirms that he wants to end the game. 4. For each player, his total assets are calculated. 5. The winner of the game is declared as the player with the highest value of assets.
Alternative Paths	Alternative 1: • In step 3, the user does not confirm, but selects the option “Cancel”. • Steps 4 and 5 are not performed, and the game proceeds as normal. Alternative 2: • In step 5, there is more than one player who has a highest asset value. • The game is declared as a tie between those players.
Post-Conditions	• The game has ended. • The winner(s) have been declared.
Related Use Cases	
Used Use Cases	None
Extending Use Cases	None

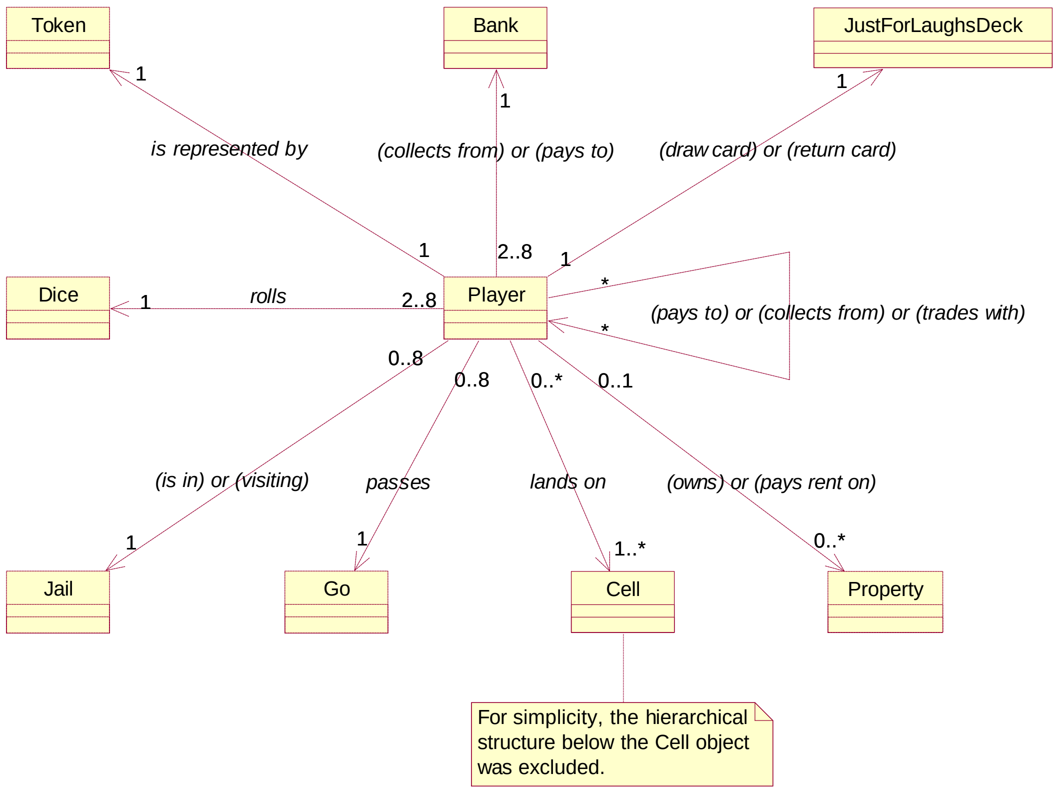
2.7.3 *Class Diagrams*

2.7.3.1 Full Class Diagram

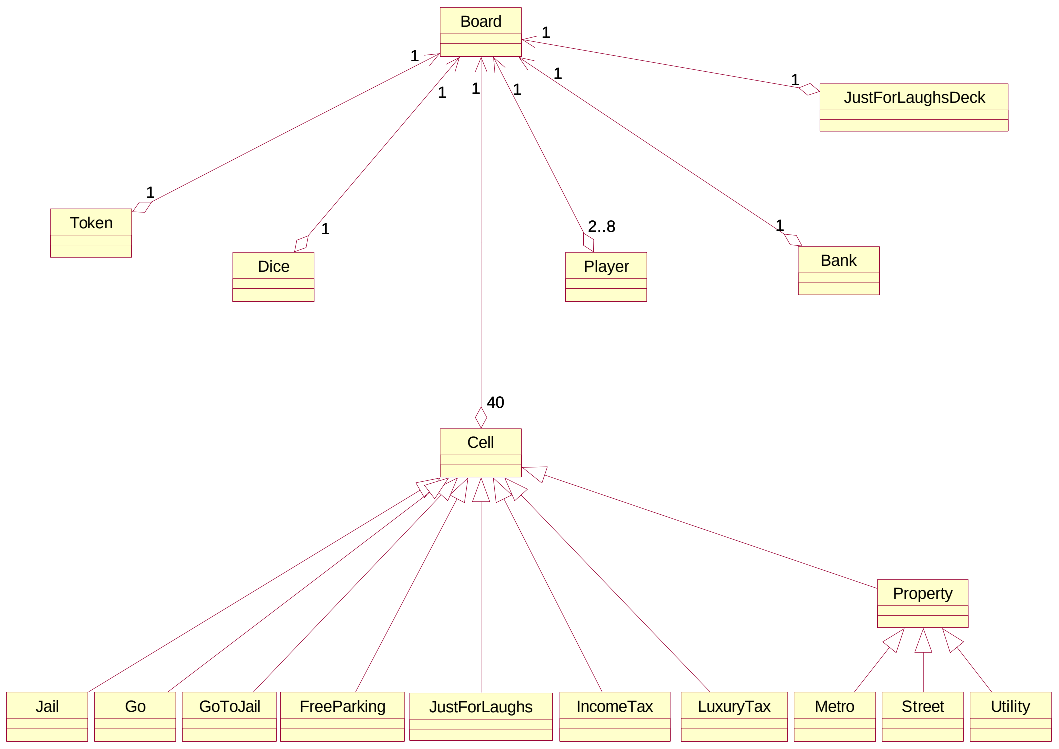
This diagram shows the different object identified during the analysis, and the relationships that exist between them. Due to the complex nature of this diagram, two different views of it will be explored in the next two sections. The first view displays the main relationships between the object, while excluding the hierarchical relationships. The second view displays only the hierarchical relationships.



2.7.3.2 Simplified View


Token
Bank
JustForLaughsDeck
1
1
1
is represented by
(collects from) or (pays to)
(draw card) or (return card)
1
2..8
1
Dice
1
rolls
2..8
Player
*
*
(pays to) or (collects from) or (trades with)
0..8
0..8
0..*
0..1
(is in) or (visiting)
passes
lands on
(owns) or (pays rent on)
1
1
1..*
0..*
Jail
Go
Cell
Property
For simplicity, the hierarchical structure below the Cell object was excluded.

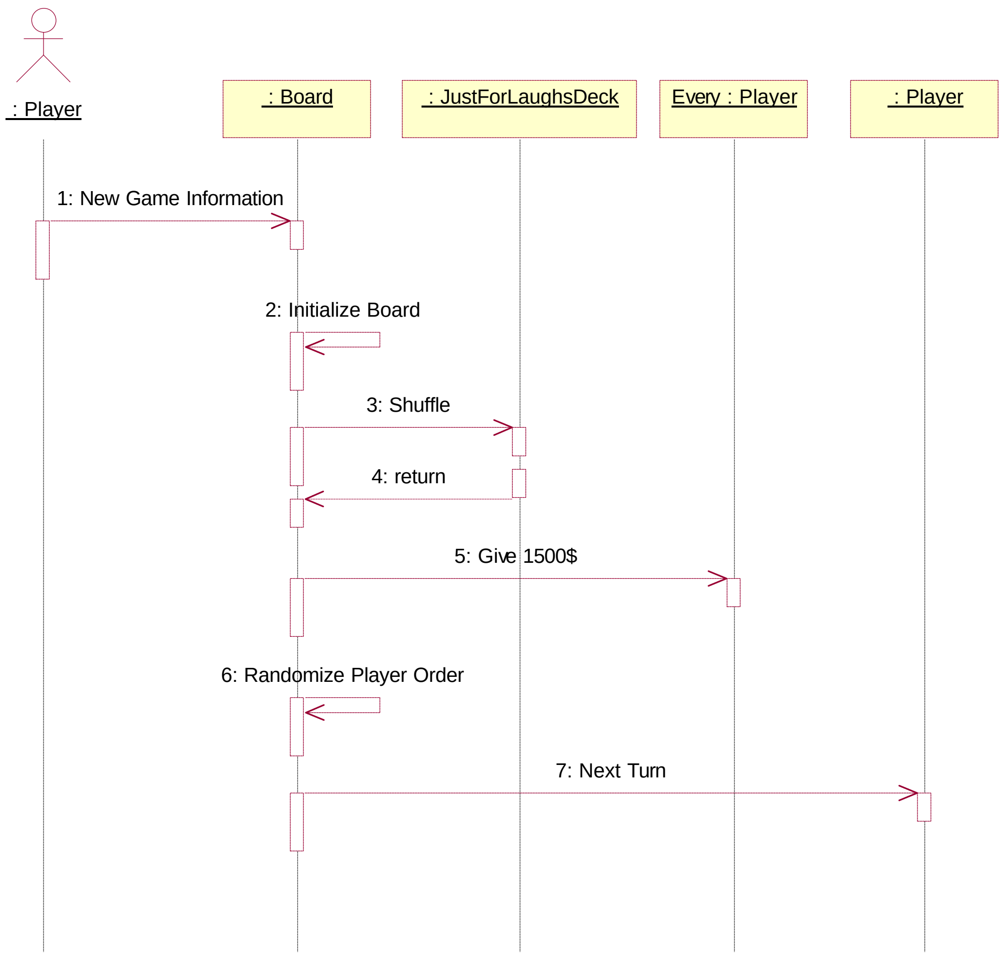
2.7.3.3 Hierarchical View



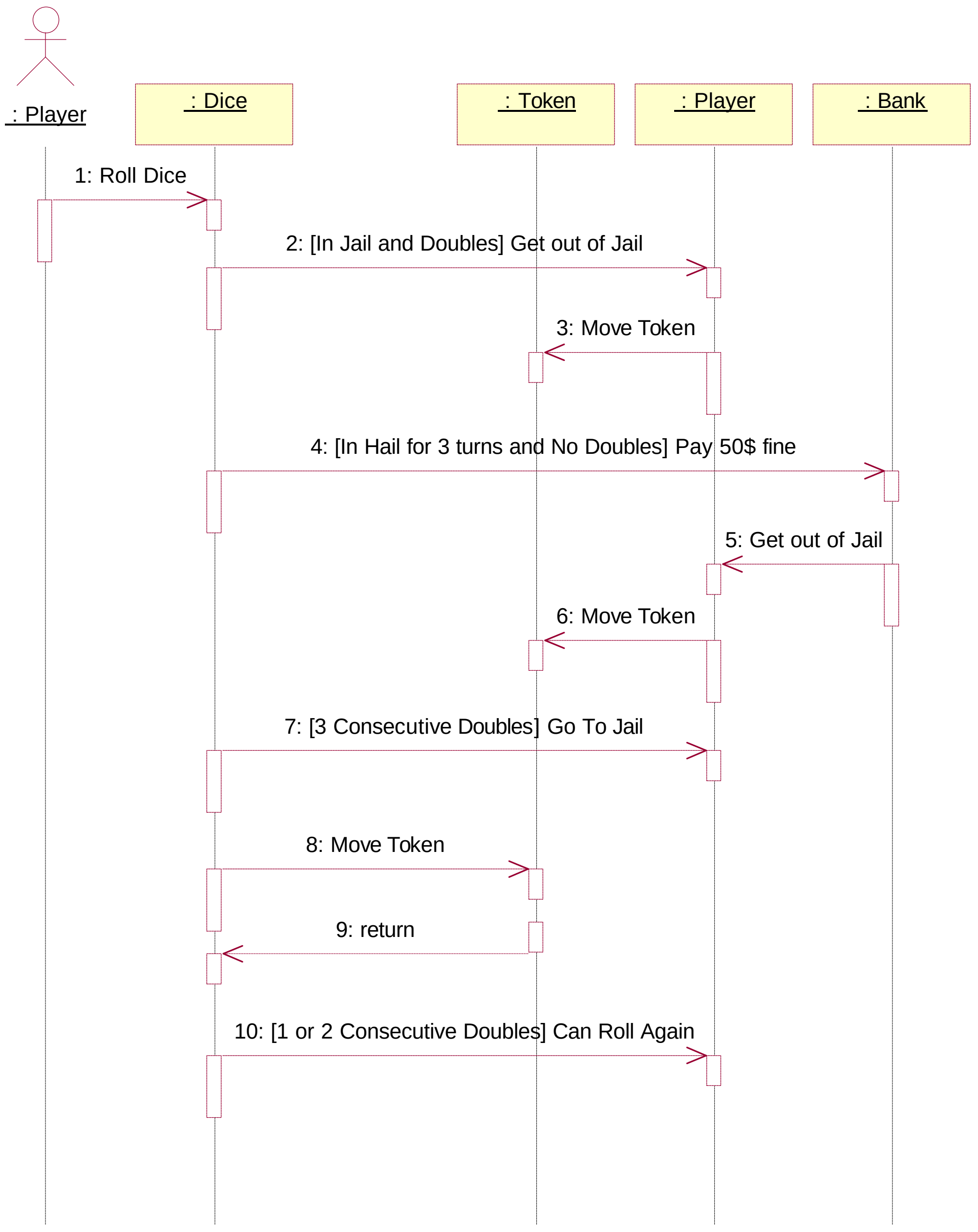
2.7.4 Sequence Diagrams

The following sequence diagrams describe in a visual, detailed manner, the use case scenarios that were mentioned in the previous sections. These sequence diagrams can be quite helpful in visualizing the sequence of actions required by the use cases.

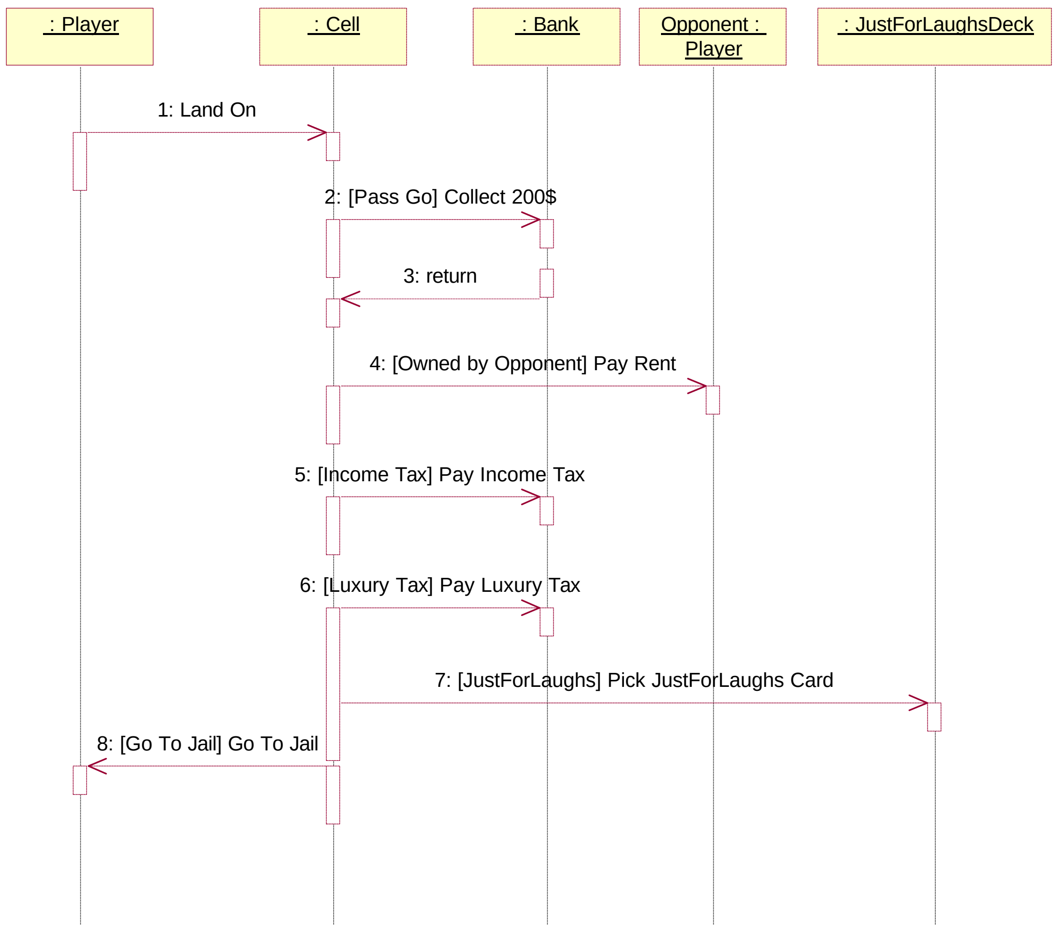
2.7.4.1 Start Game (Use Case 1)



2.7.4.2 Roll Dice (Use Case 2)

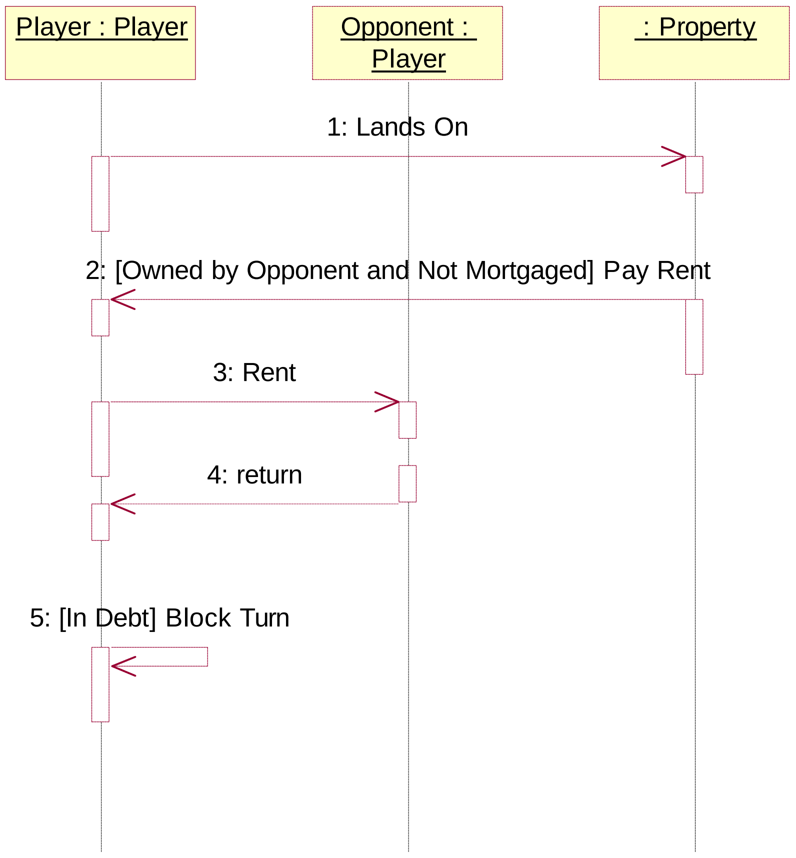

: Player
: Dice
: Token
: Player
: Bank
1: Roll Dice
2: [In Jail and Doubles] Get out of Jail
3: Move Token
4: [In Hail for 3 turns and No Doubles] Pay 50$ fine
5: Get out of Jail
6: Move Token
7: [3 Consecutive Doubles] Go To Jail
8: Move Token
9: return
10: [1 or 2 Consecutive Doubles] Can Roll Again

2.7.4.2.1 Land On (Use Case 2.1)

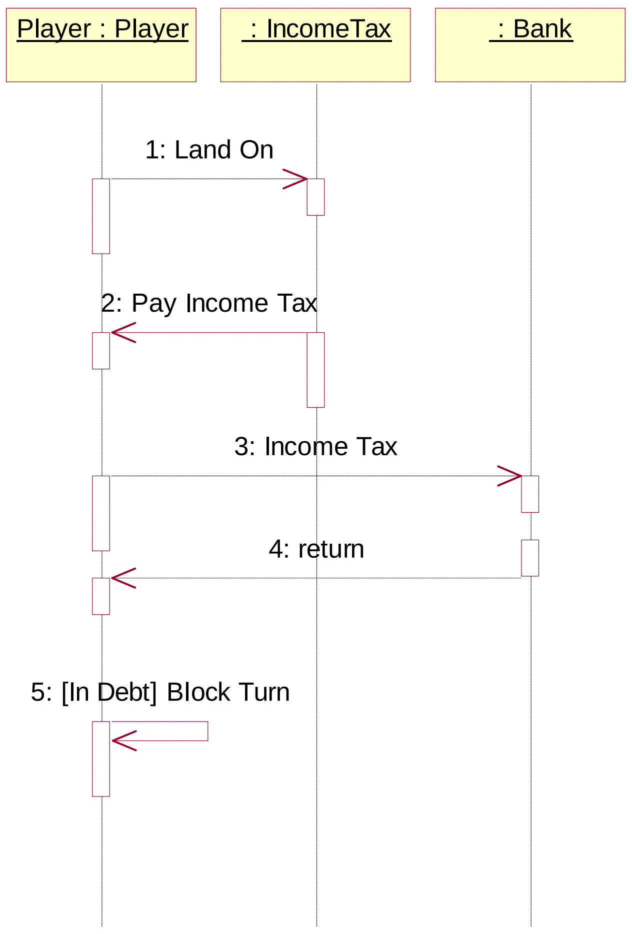


: Player
: Cell
: Bank
Opponent : Player
: JustForLaughsDeck
1: Land On
2: [Pass Go] Collect 200$
3: return
4: [Owned by Opponent] Pay Rent
5: [Income Tax] Pay Income Tax
6: [Luxury Tax] Pay Luxury Tax
7: [JustForLaughs] Pick JustForLaughs Card
8: [Go To Jail] Go To Jail

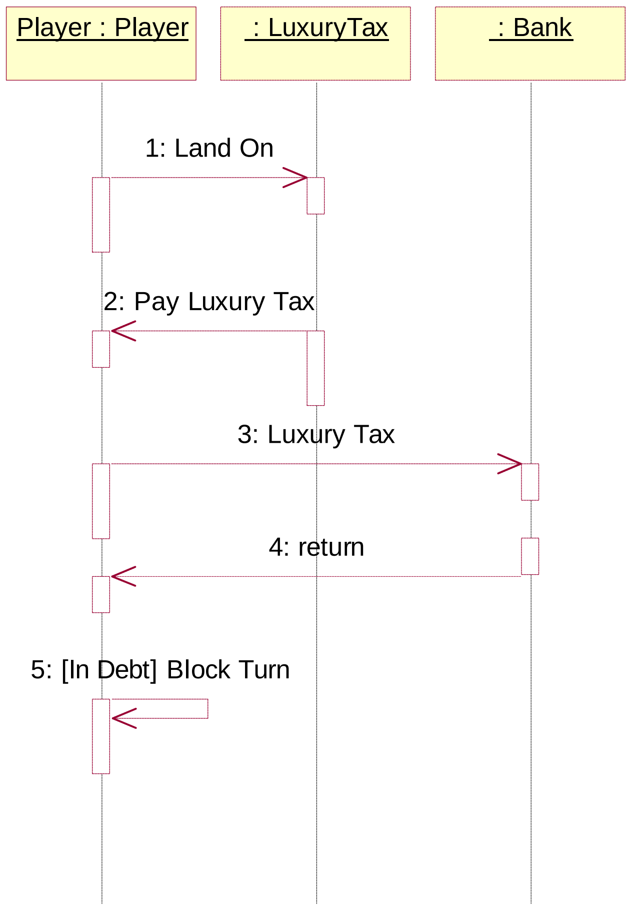
2.7.4.2.2 Pay Rent (Use Case 2.2)



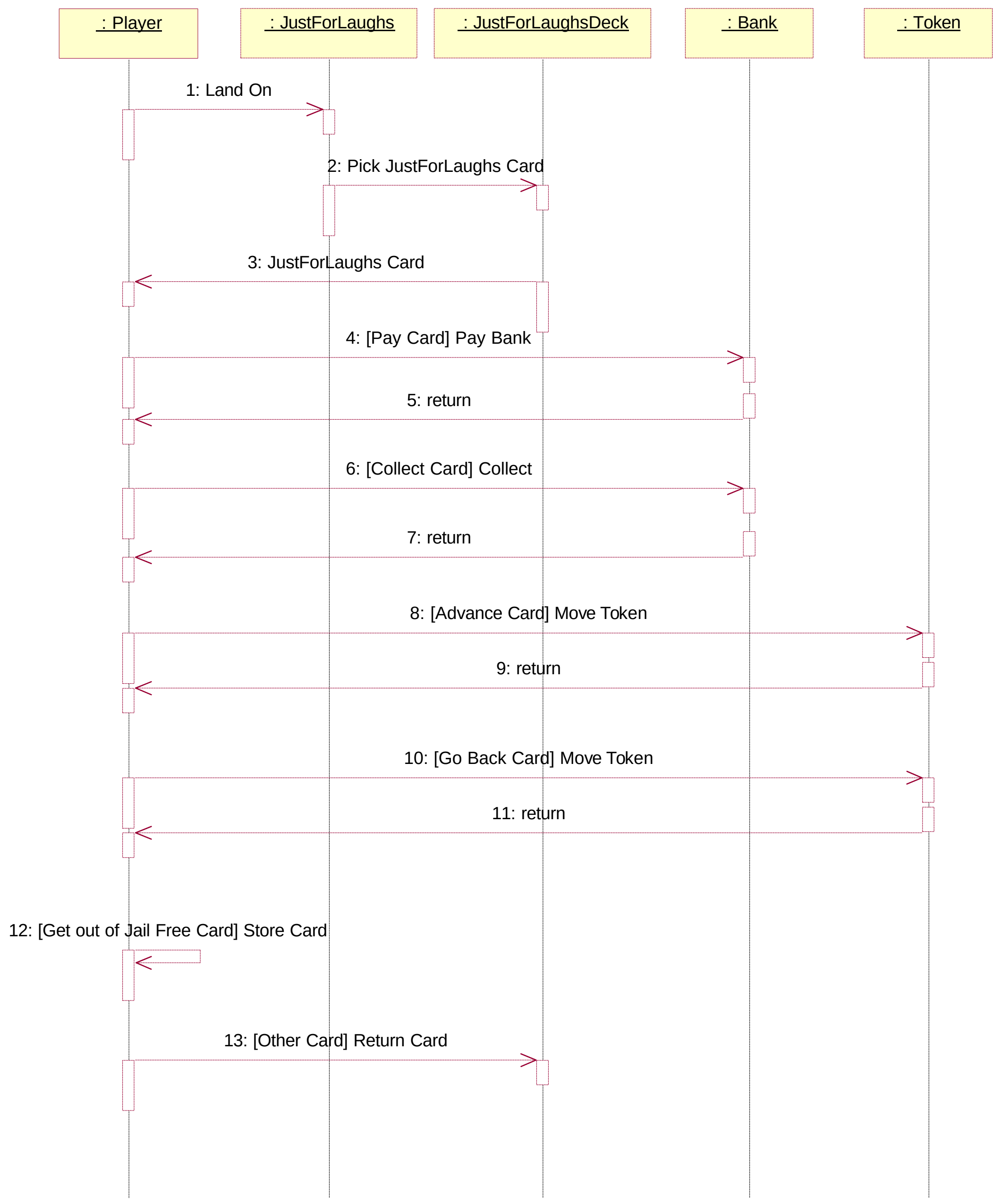
2.7.4.2.3 Pay Income Tax (Use Case 2.3)



2.7.4.2.4 Pay Luxury Tax (Use Case 2.4)

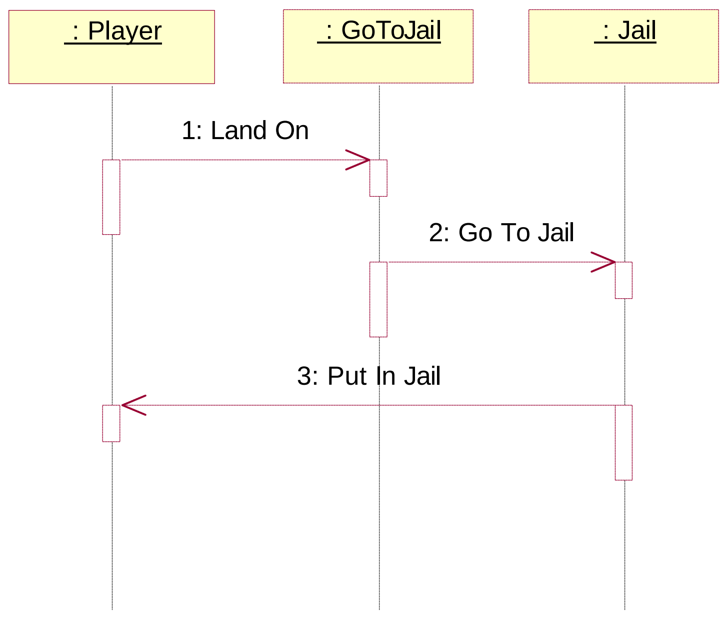


2.7.4.2.5 Pick JustForLaughs Card (Use Case 2.5)

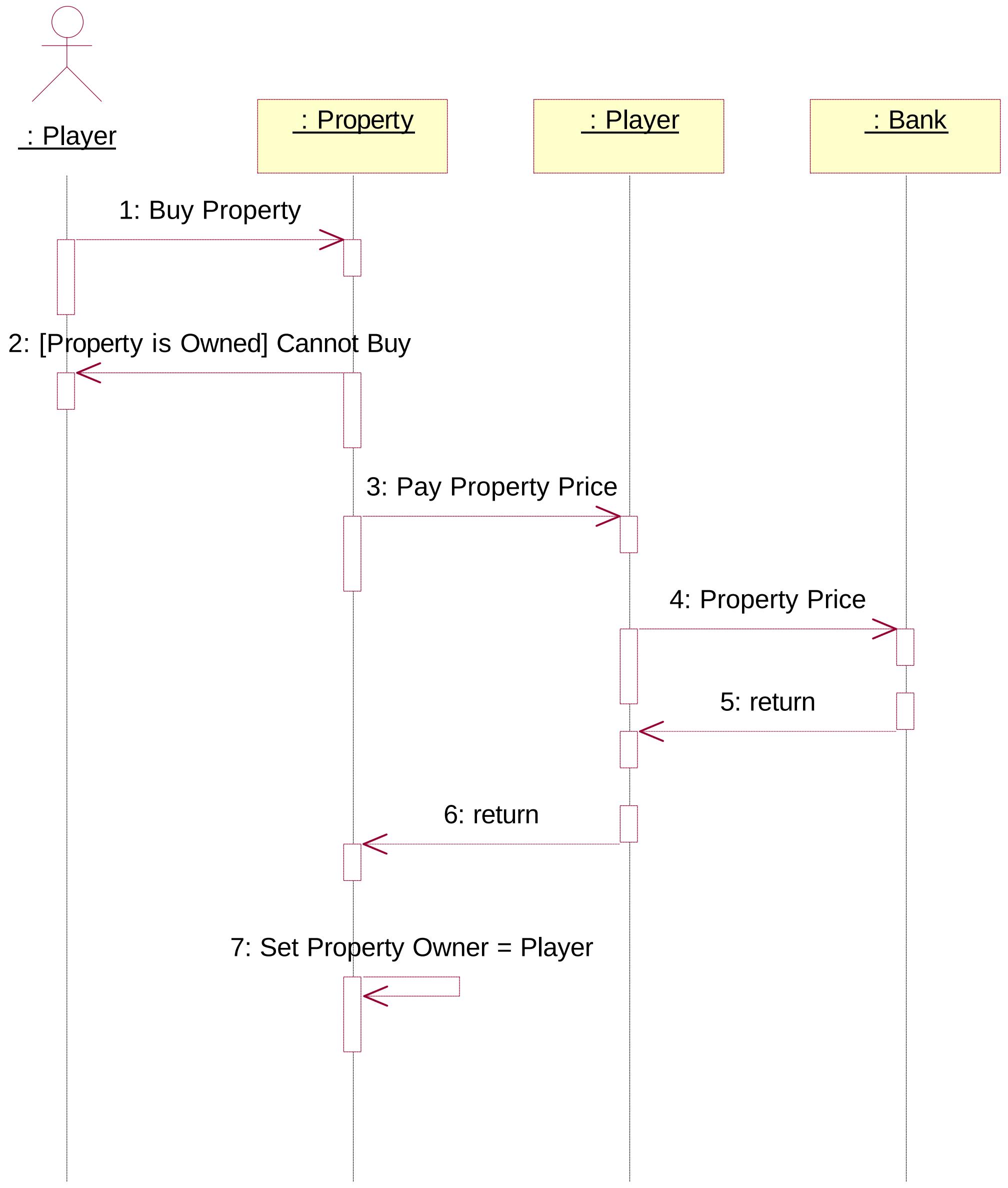


: Player
: JustForLaughs
: JustForLaughsDeck
: Bank
: Token
1: Land On
2: Pick JustForLaughs Card
3: JustForLaughs Card
4: [Pay Card] Pay Bank
5: return
6: [Collect Card] Collect
7: return
8: [Advance Card] Move Token
9: return
10: [Go Back Card] Move Token
11: return
12: [Get out of Jail Free Card] Store Card
13: [Other Card] Return Card

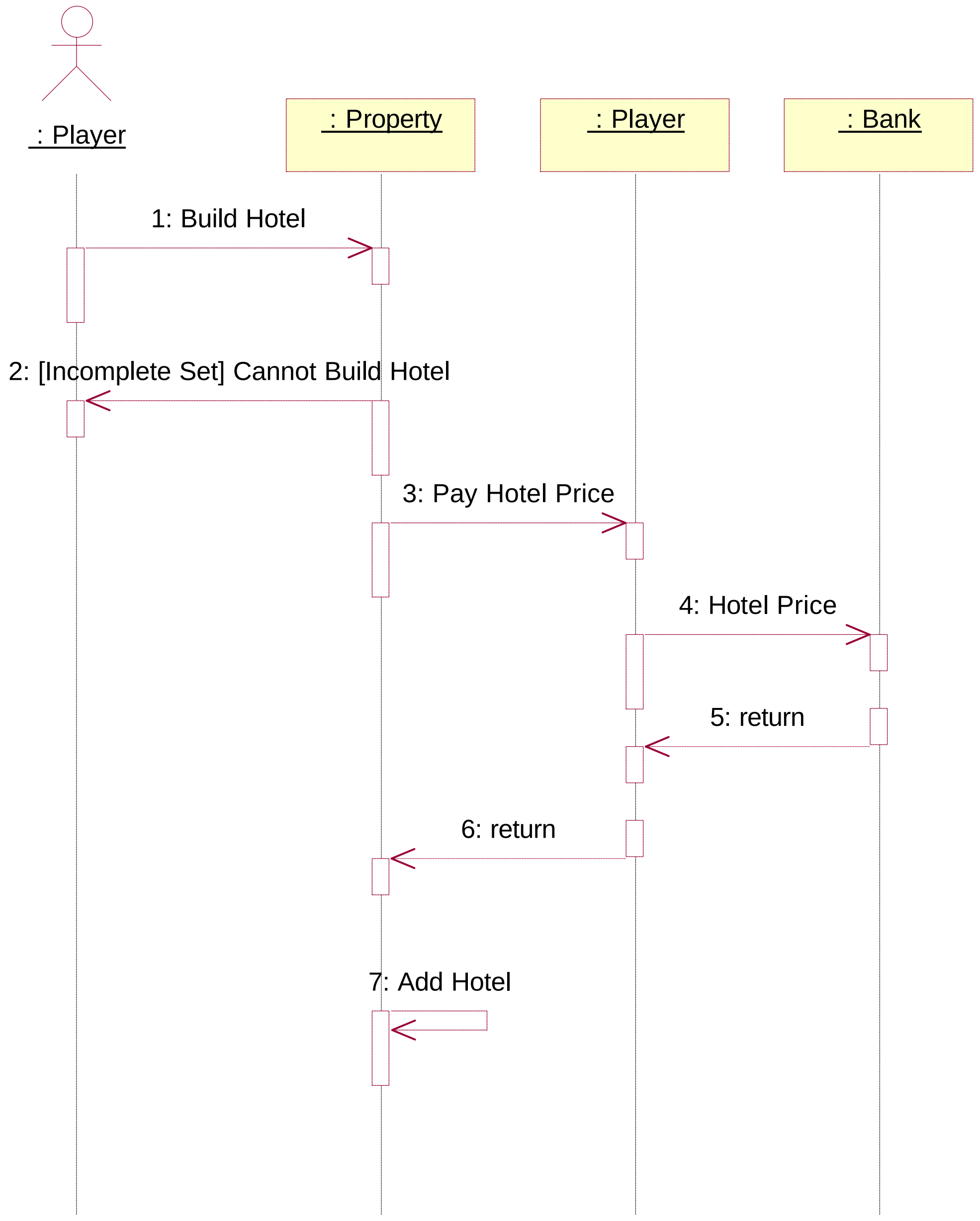
2.7.4.2.6 Go To Jail (Use Case 2.6)



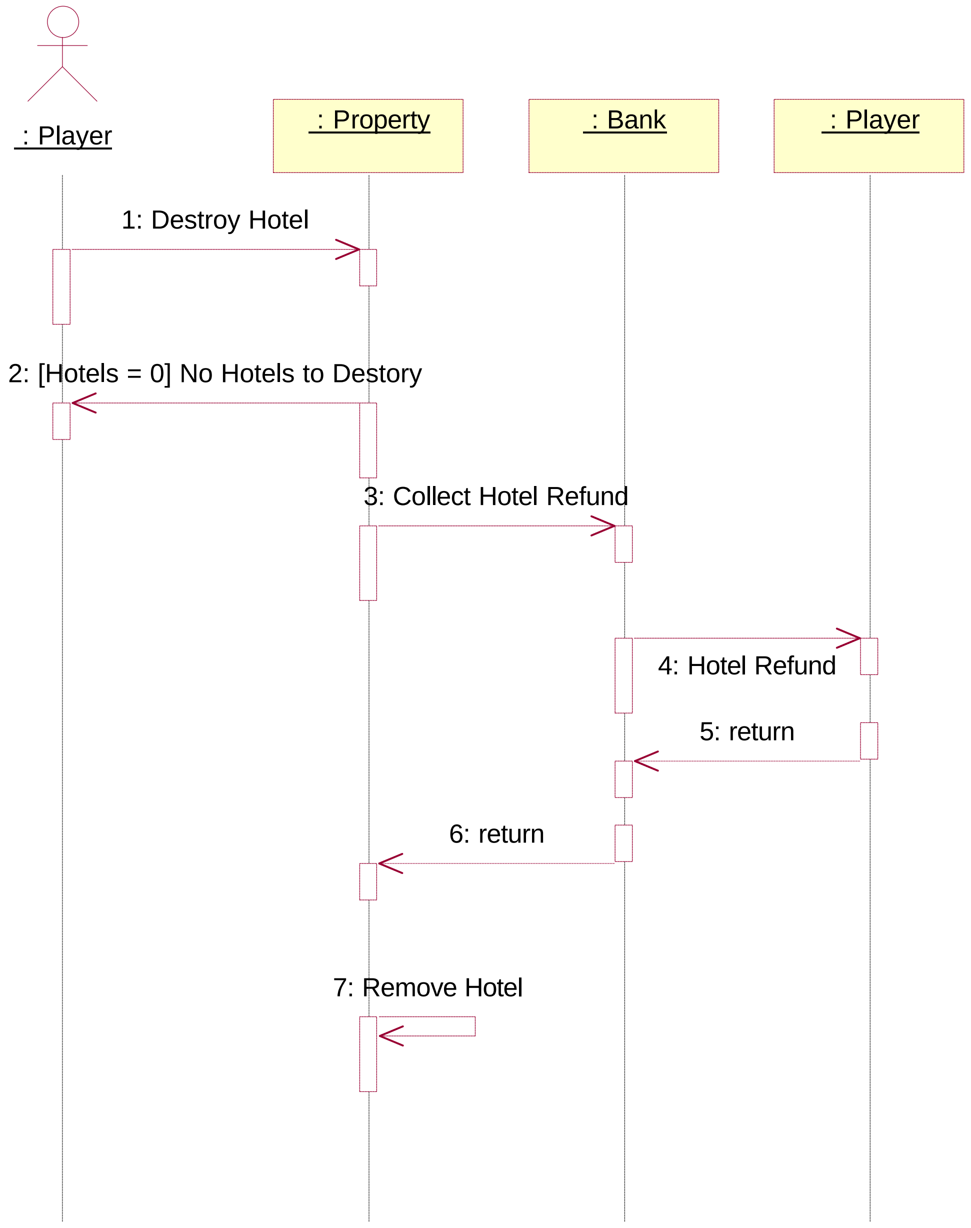
2.7.4.3 Buy Property (Use Case 3)



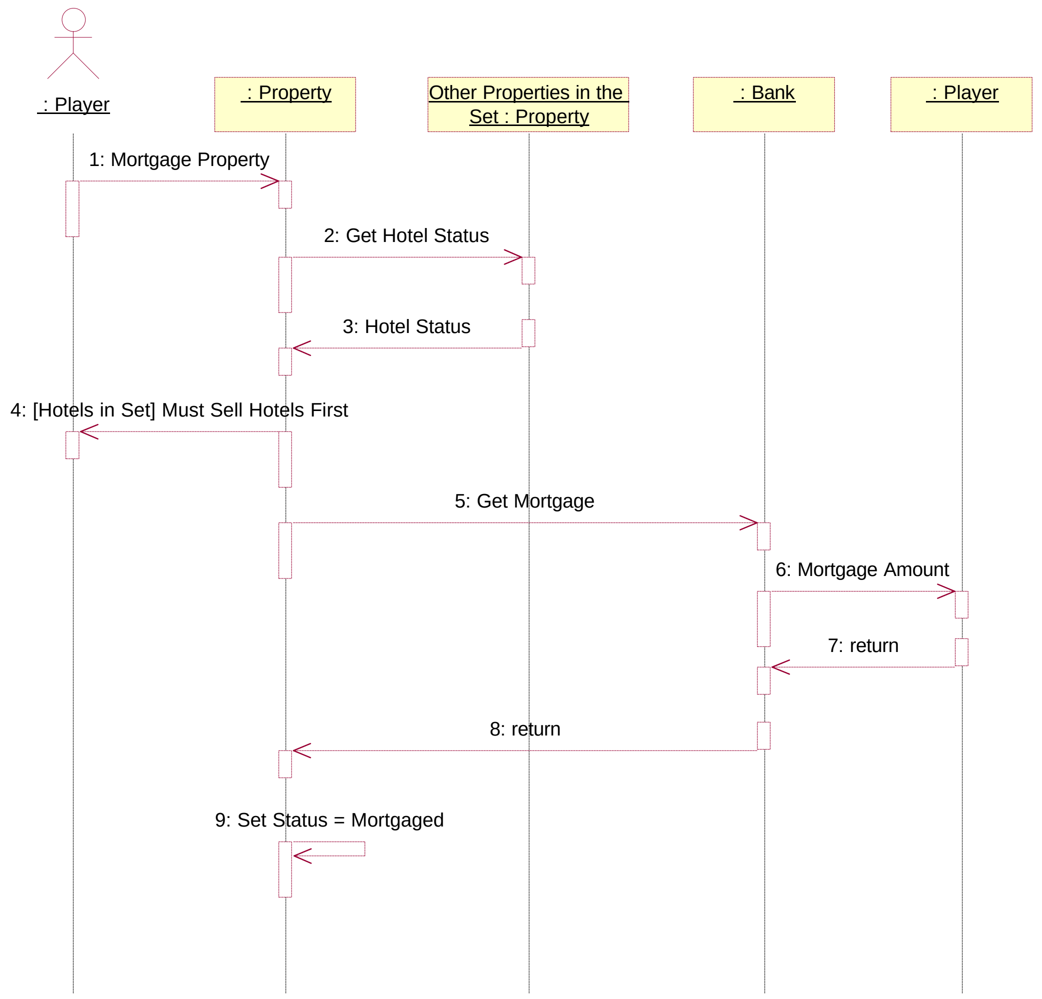
2.7.4.4 Build Hotel (Use Case 4)



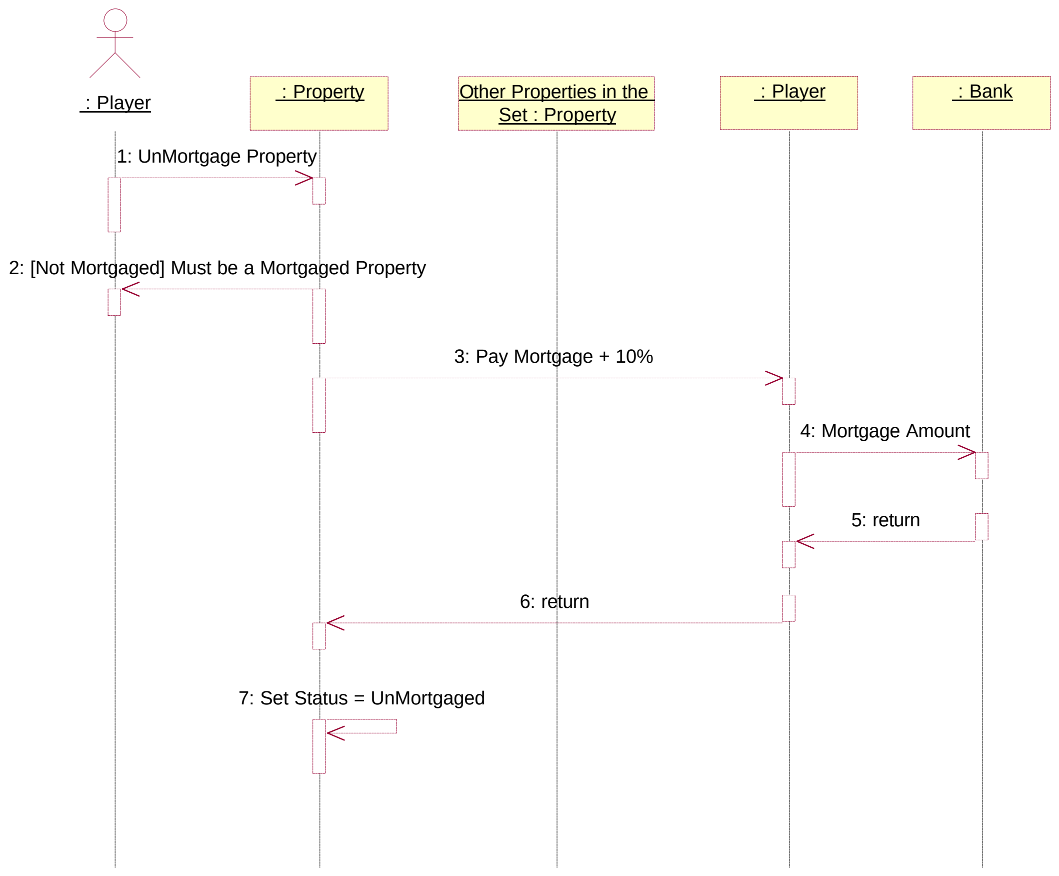
2.7.4.5 Destroy Hotel (Use Case 5)



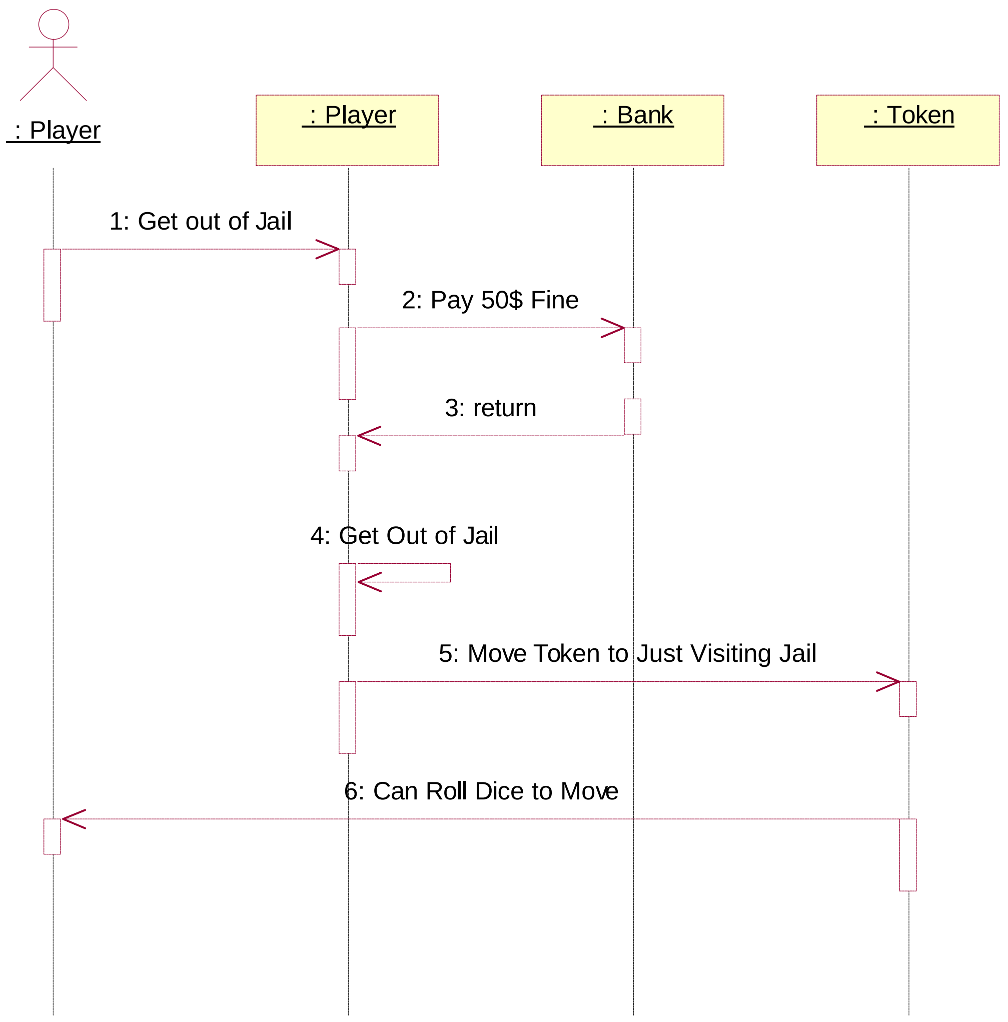
2.7.4.6 Mortgage Property (Use Case 6)



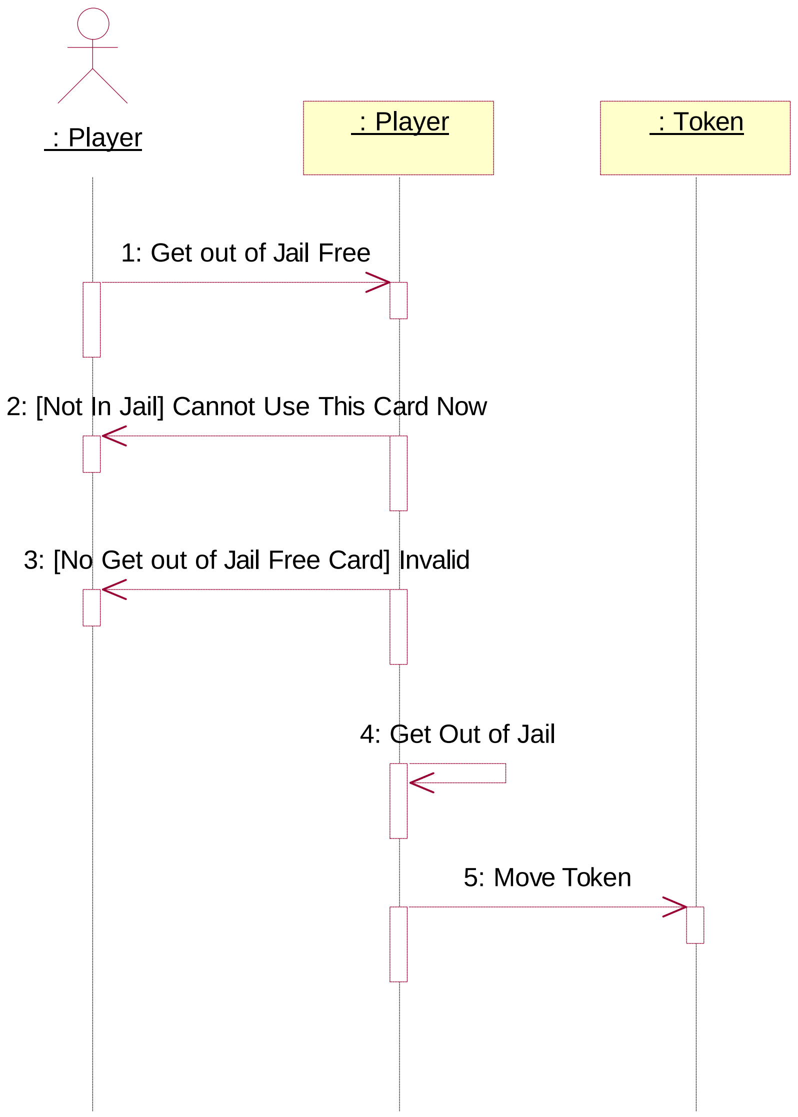
2.7.4.7 UnMortgage Property (Use Case 7)


: Player
: Property
Other Properties in the Set : Property
: Player
: Bank
1: UnMortgage Property
2: [Not Mortgaged] Must be a Mortgaged Property
3: Pay Mortgage + 10%
4: Mortgage Amount
5: return
6: return
7: Set Status = UnMortgaged

2.7.4.8 Get out of Jail (Use Case 8)

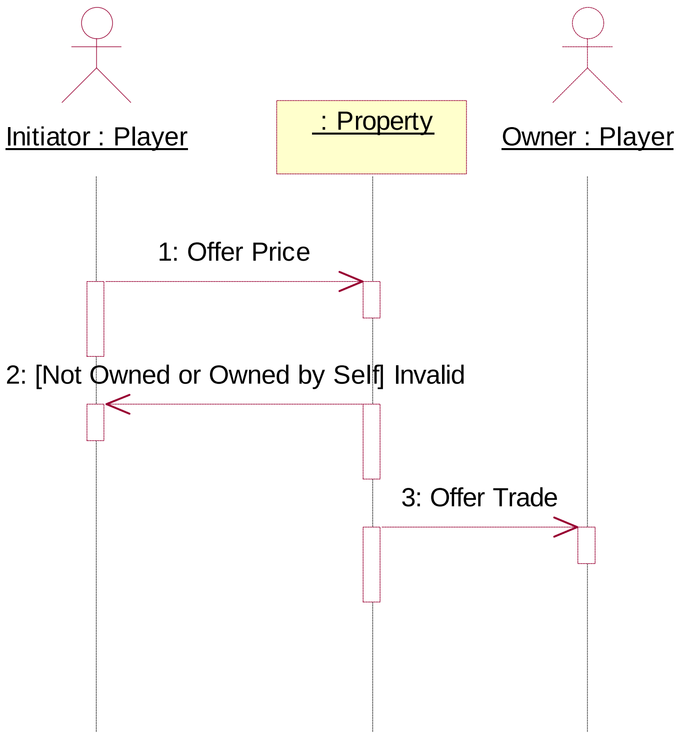


2.7.4.9 Get out of Jail Free (Use Case 9)

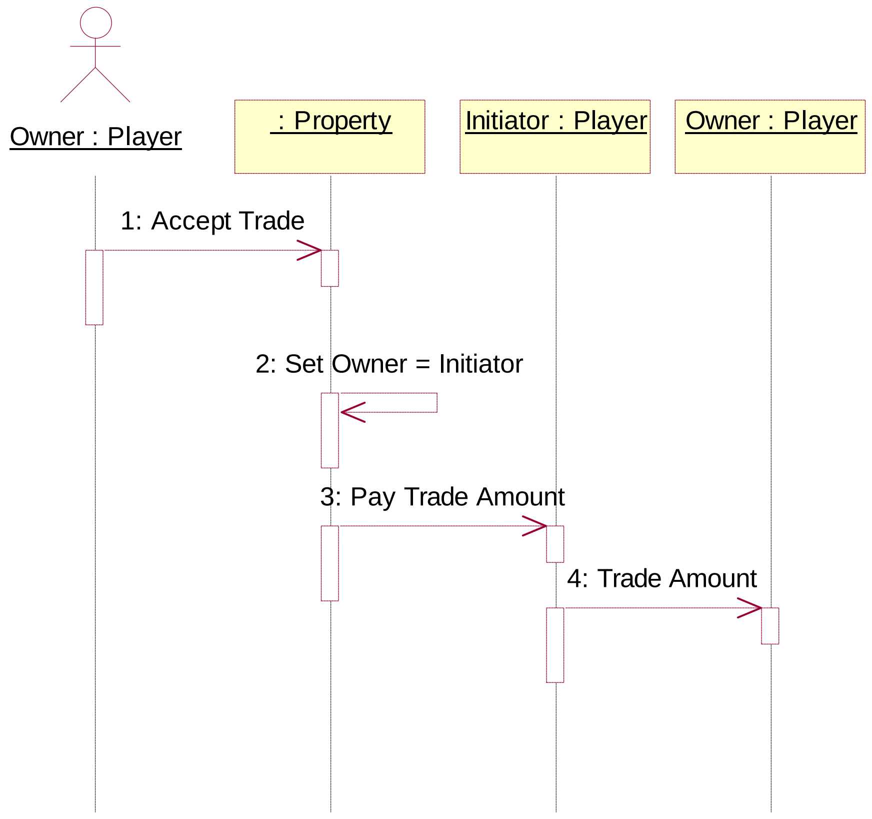


: Player
: Player
: Token
1: Get out of Jail Free
2: [Not In Jail] Cannot Use This Card Now
3: [No Get out of Jail Free Card] Invalid
4: Get Out of Jail
5: Move Token

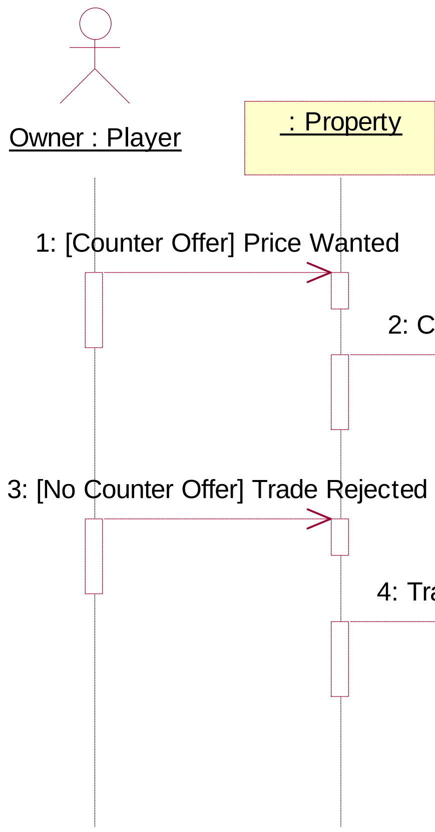
2.7.4.10 Offer Trade (Use Case 10)



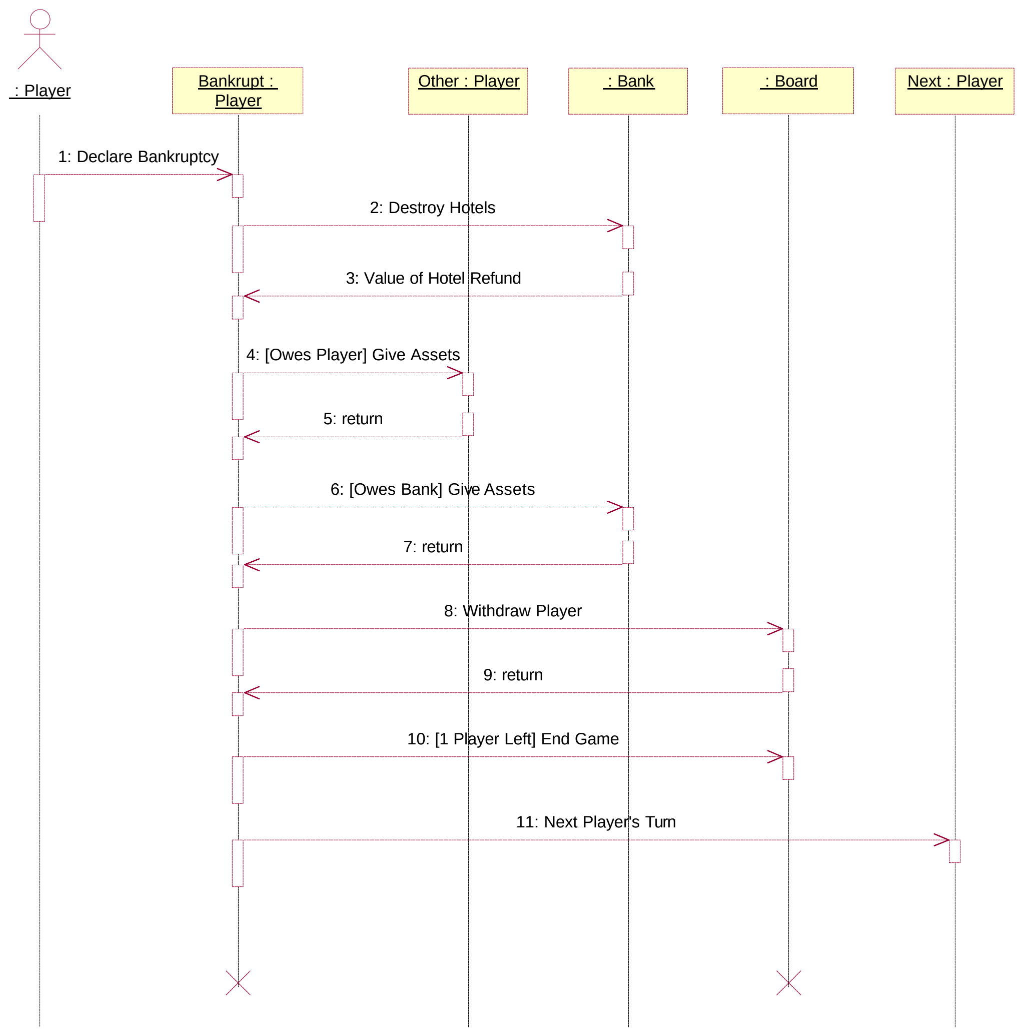
2.7.4.11 Accept Trade (Use Case 11)



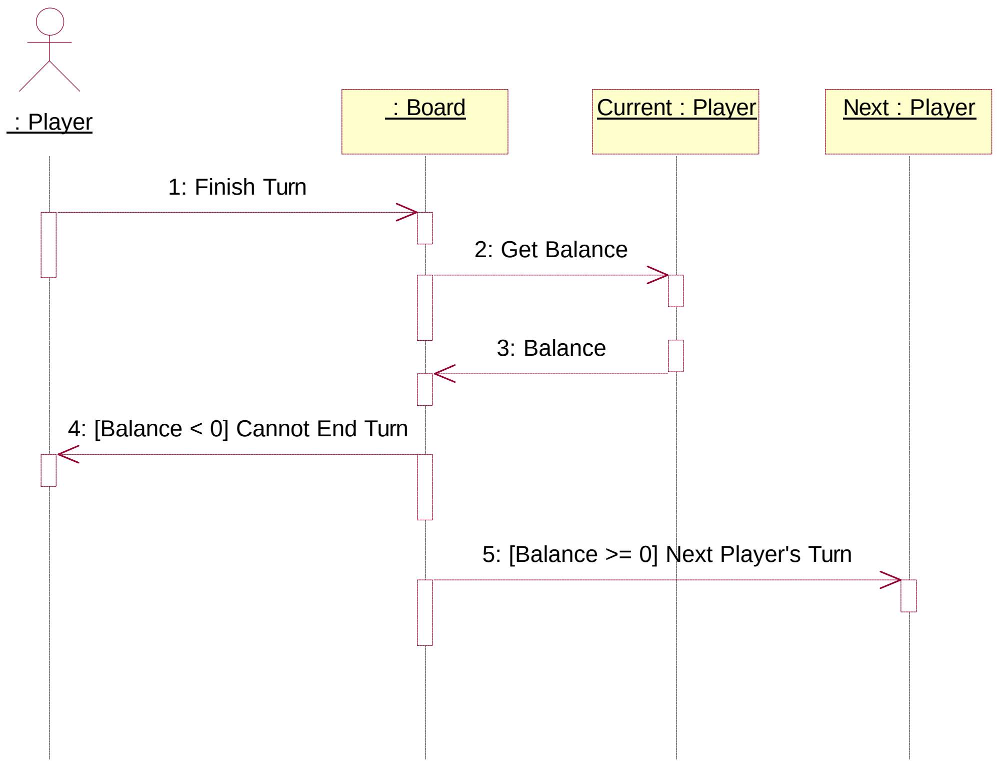
2.7.4.12 Reject Trade (Use Case 12)



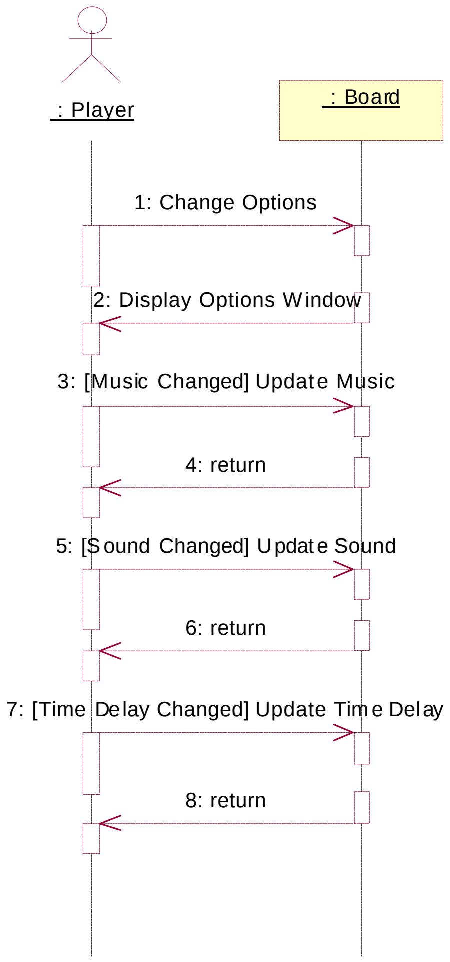
2.7.4.13 Declare Bankruptcy (Use Case 13)



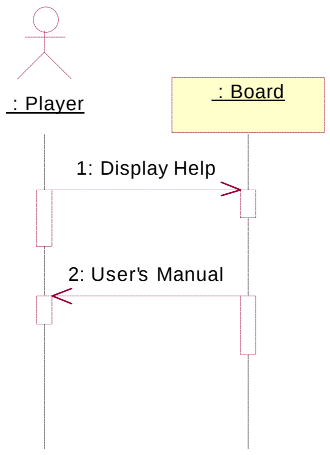
2.7.4.14 Finish Turn (Use Case 14)



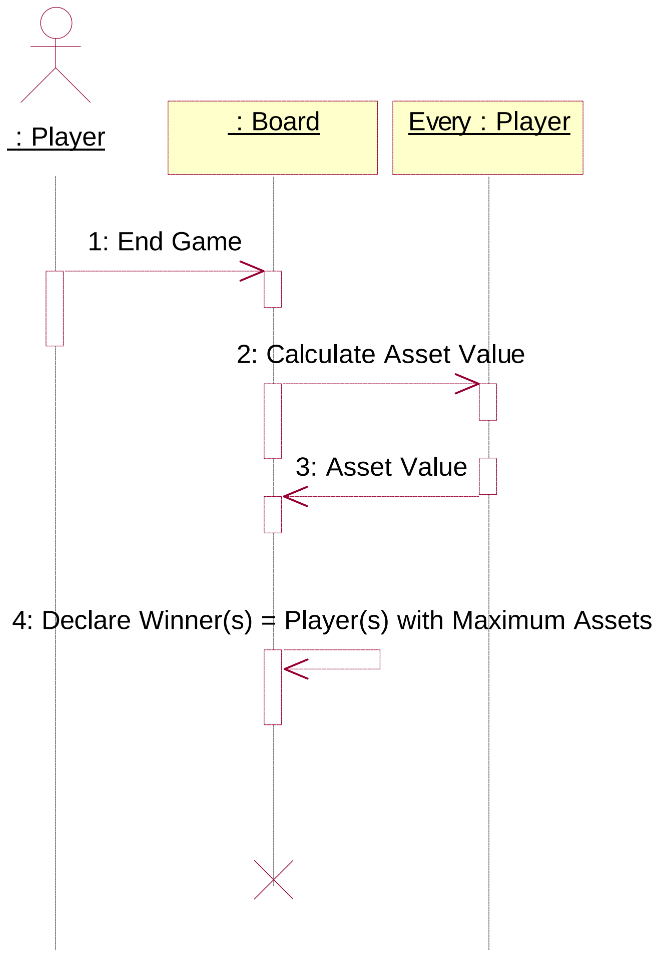
2.7.4.15 Change Options (Use Case 15)


: Player
: Board
1: Change Options
2: Display Options Window
3: [Music Changed] Update Music
4: return
5: [Sound Changed] Update Sound
6: return
7: [Time Delay Changed] Update Time Delay
8: return

2.7.4.16 Display Help (Use Case 16)



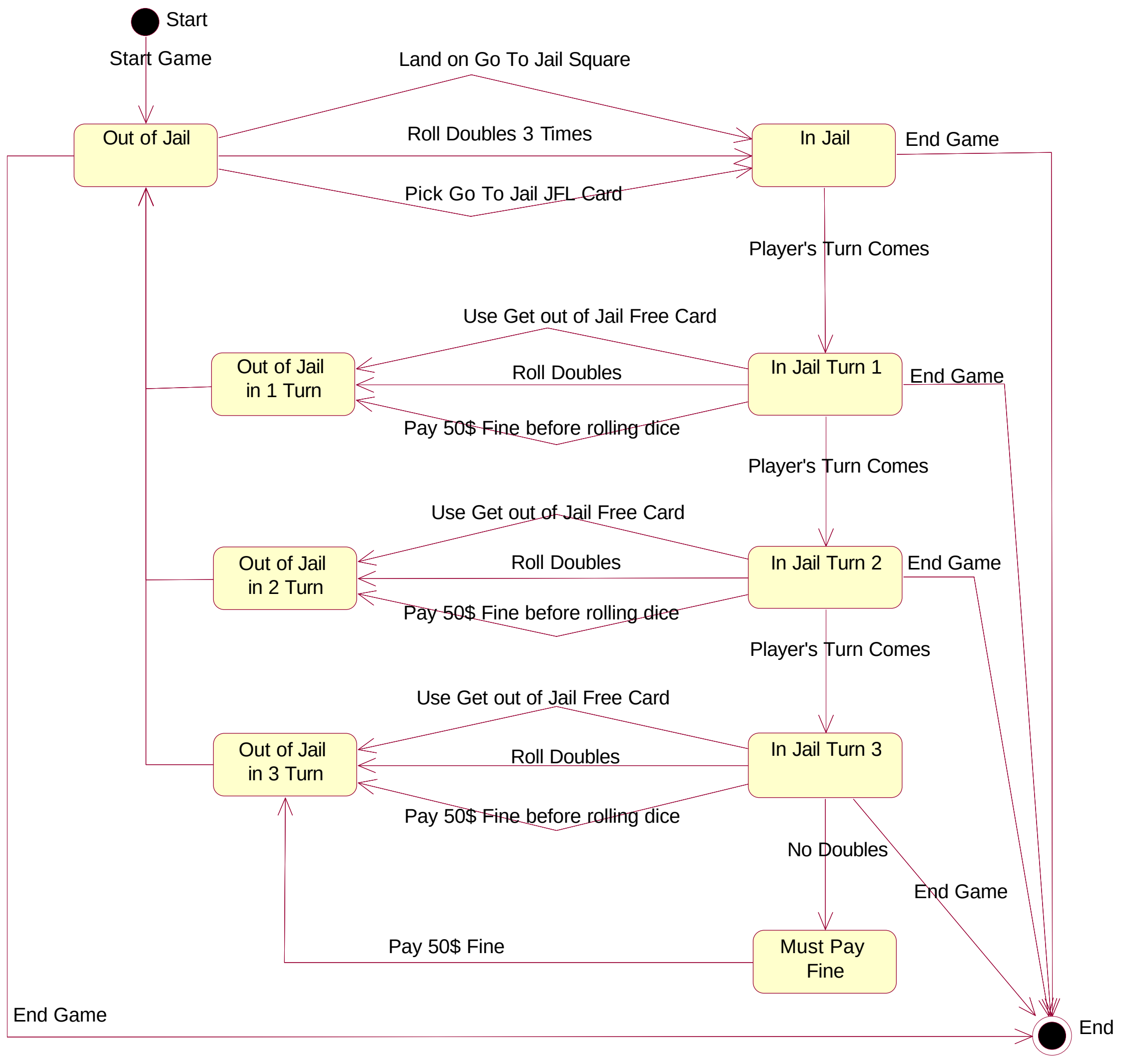
2.7.4.17 End Game (Use Case 17)



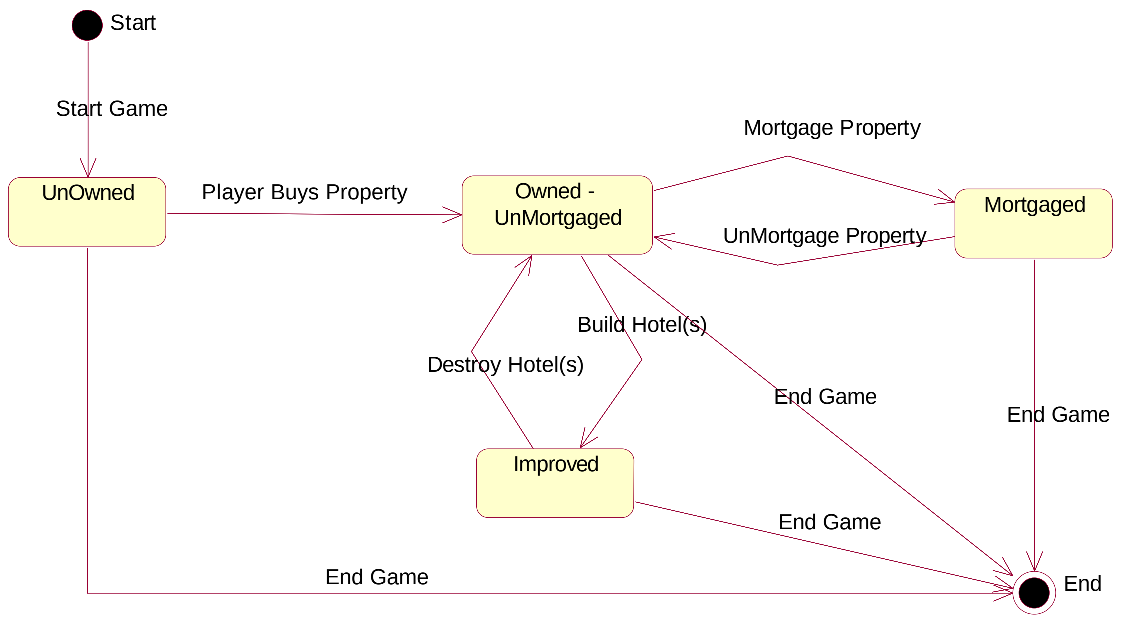
2.7.5 State Transition Diagrams

The following diagrams depicts the different states that some of the objects in the Montrealopoly game can be in. The first diagram describes the player’s many states concerning jail. This also shows the transitions (actions) that can cause a change in the state of a player. This helps visualize how a player can enter/exit jail. The second diagram depicts the different states a property can be in. A property can be owned or unowned, mortgaged or unmortgaged, improved or unimproved. It also shows some of the restrictions in the game. For example, a property cannot be mortgaged until all hotels have been destroyed.

2.7.5.1 Player Jail States



2.7.5.2 Property States



3 Development Plan

This section of the document contains the estimated cost and schedule for the project in the form of a phase plan and project schedule. The project schedule contains a WBS with a Gantt chart that contains a detailed view of how many man-days different components of the project will take. It also contains the minor and major milestones that are part of this project. The project plan contains the completion dates of the phase deliverables and milestone, such as prototypes and builds.

3.1 Project Estimates

Based on the WBS, this project will take a maximum of 85 man-days to complete. 1 Man-day is the equivalent of 8 hours of work, which can span over multiple days, weeks or months. With 10 members on the team, each member will contribute an average of 8.5 man-days, or 68 hours each on the project. Man-day estimates are based on previous tasks, the complexity of the task, and the amount of people working on it. The actual amount of time spent may be significantly less than the man-day estimate, as the man-day is the maximum amount of time that is estimated to complete the task.

The first phase took 15 man-days to complete, and was completed on September 29. This was based on the amount of time all the members had to complete the first phase.

The second phase has 26 man-days scheduled in order to complete it. This estimate includes the estimated time to complete the first prototype, which has been estimated to take 13 man-days. Once the first prototype has been completed, the amount of man-days to complete this phases will be re-estimated, as the prototype may have taken longer/less time to complete. A second prototype will then be developed based on the first one, but the amount of time it takes will be counted towards the total time to complete phase 3.

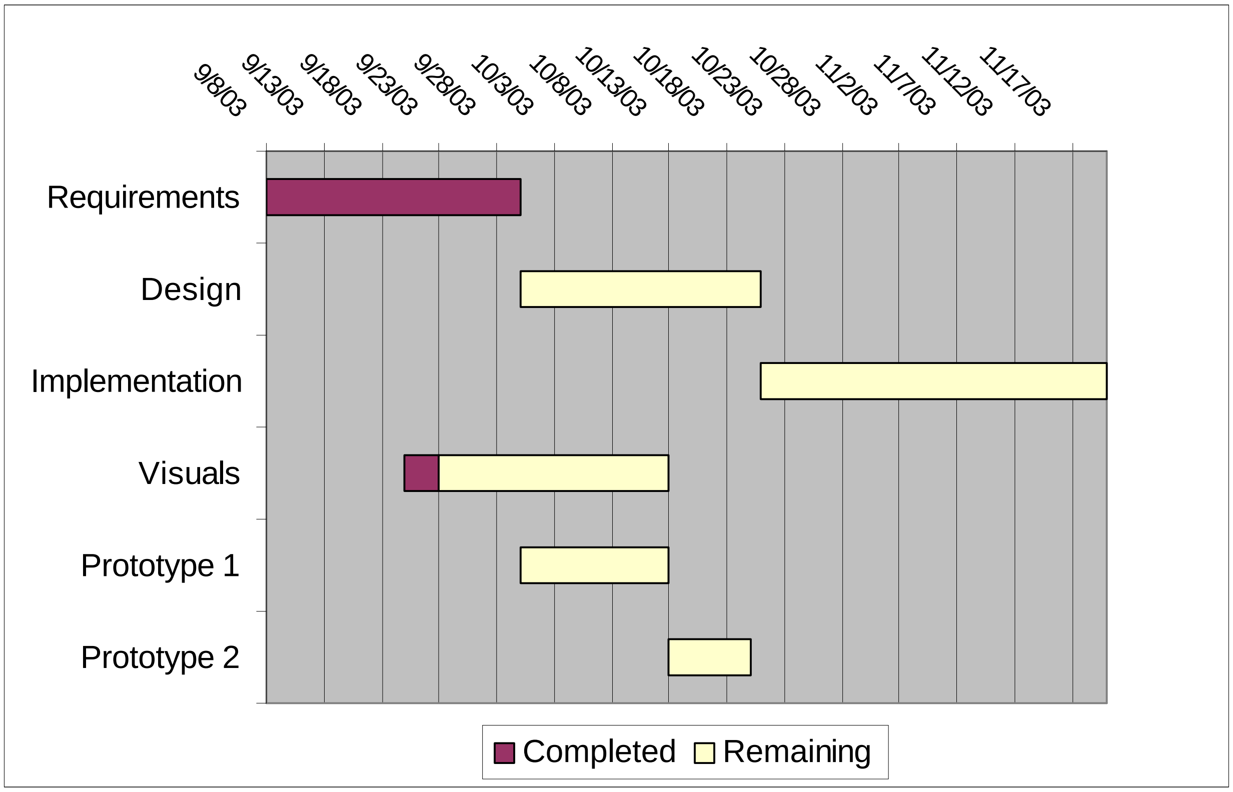
The third phase has been estimated to take 44 man-days to complete and it will be re-estimated once the second prototype has been completed. Once this prototype has been completed, the implementation team will be able to release their first build shortly after. After each build, adjustments will be made to the estimated amount of time required for this phase, based on the success of each build.

The visuals for this project will be used for the graphical user interface for the game. The main visuals will include the board game, player tokens, title deed cards and JFL cards. An estimated 11 man-days will be needed to complete all of the game's visuals.

The first prototype will commence once the first phase has been completed and will contain the core features introduced in this requirements document and use the visuals created for the graphical user interface. Feedback from this prototype will be used in the development of the second prototype.
The estimated time to complete the first prototype is 13 man-days.

The second prototype will be based on the feedback received from the first phase and will include some elements from the design phase. Even though the design phase and second prototype will be developed in parallel, the prototype will use some elements of the design phase that have been already completed. Once the second prototype has been developed, it will provide the basis for the first build of the game.
7 man-days have been allocated for the completion of the second prototype.

An overview of the major phases in the project are outlined below:



Project Gantt Chart

3.2 Project Plan

The project plan is made up of a phase plan and project schedule. The phase plan contains a WBS with a Gantt chart, which shows the amount of man-days each task will take. It also contains the minor and major milestones in the project.

3.2.1 Phase Plan

			September									October											November				
Task	Days*	Who	8	13	15	20	22	27	28	29	30	2	4	6	11	12	13	18	20	21	25	27	1	8	15	19	20
General																											
Group and team formation	2d	All	🟩	🟩																							
Discuss rules for Montrealopoly	1d	ST,AB,EZ,RH			🟩																						
Create deeds file	1d	EZ					■																				
Milestone: Agreed on Montrealopoly				^																							
Milestone: Agreed to use VB					^																						
Requirements																											
discuss requirements	1d	ST,AB,EZ,RH,XI				■	■																				
Phase 1 report	10d	ST,AB,EZ,RH,SL				■	■	■	■	■																	
finalise requirements	1d	ST,AB,EZ,RH,SL							■	■																	
Milestone: Phase 1 deliverable											@																
Visuals																											
Board	2d	AB				🟩	🟩																				
Deeds	4d	AB											▨	▨													
Dice	1d	AB						🟩																			
Tokens	4d	AB													▨	▨	▨										
Milestone: Completion of graphics																		@									
Design																											
Discuss design requirements	1d	RH,ZZ,XX,ST,SL										▨															
Phase 2 report	8d	RH,ZZ,XX,ST,SL												▨	▨	▨	▨	▨									
Design details	4d													▨	▨												
Milestone: Phase 2 deliverable																				@							
Prototypes																											
Prototype 1	13d	SL,PM,JW,HL									▨	▨	▨	▨	▨	▨											
Milestone: Prototype 1 complete																	@										
Prototype 2	7d	SL,PM,JW,HL															▨	▨	▨								
Milestone: Prototype 2 complete																				@							
Implementation																											
Discuss implementation requirements	1d	SL,PM,JW,HL,ST,RH																		▨							
Code	20d	SL,PM,JW,HL																		▨	▨	▨	▨	▨	▨	▨	
Test	5d	AB,EZ,ZZ,XX																							▨	▨	
Phase 3 Report	11d	SL,ST,RH																						▨	▨	▨	
Milestone: build 1																					@						
Milestone: build 2																							@				
Milestone: build 3																								@			
Milestone: Release Candidate																									@		
Milestone: Final Montrealopoly Game																										@	
Milestone: Phase 3 deliverable																											@

WBS and Gantt Chart

Legend			
■	Completed		
▨	To Complete		
^	Minor Milestone		
@	Major Milestone		
AB	Alexandre Bosserelle	RH	Robert Hanna
EZ	Eugena Zolorova	SL	Simon Lacasse
HL	Hu Shan Liu	ST	Stefan Thibeault
JW	Jens Witkowski	XX	Xin Xi
PM	Patrice Michaud	ZZ	Zhi Zhang

* Man days, not calendar days

The milestones and their significance to the project:

Minor Milestones	**Description**
Agreed on Montrealopoly	Several proposals were made before agreeing on Montrealopoly, including groupware, Risk and custom made game based on the rules to be created by the group
Agreed to use VB	There was a long debate between Java and Visual Basic, and VB was chosen as a prototype could be quickly built

Major Milestone	**Description**
Phase 1 deliverable	Signifies the completion of the first phase which is the ground work for the second phase
Completion of graphics	The first prototype can now be built using the graphics for the graphical user interface
Phase 2 deliverable	Provides the foundation for the final phase of the project
Prototype 1 complete	Will help validate the work done in the first phase and incorporate the visuals into the graphical user interface.
Prototype 2 complete	Incorporate any changes based on feedback form the first prototype
Build 1	The first build based on the first two phases of the project and contain the core features
Build 2	Addition of new features, code rewrites and bug fixes
Build 3	Incorporate features and improvements based on feedback from the previous build
Release Candidate	Feature freeze, fix any new bugs found
Final Release	Final product ready for release
Phase 3 deliverable	Completion of project

3.2.2 Project Schedule

The following table shows the completion dates for the project deliverables and major milestones.

Project Schedule	
Task Description	Completion Date
Phase 1 Deliverable	30-Sep
Milestone: Prototype 1	12-Oct
Milestone: Completion of Graphics	13-Oct
Phase 2 Deliverable	20-Oct
Milestone: Prototype 2	20-Oct
Milestone: build 1	25-Oct
Milestone: build 2	01-Nov
Milestone: build 3	08-Nov
Milestone: Release Candidate	15-Nov
Milestone: Final Release	19-Nov
Phase 2 Deliverable	20-Nov

3.2.3 Project Resourcing

This project will use a group of 10 members divided into three teams. Team 1, the requirements team, is made up of 3 members, Team 2, the design team, is made up of 3 members and team 3, the implementation team, is made up of 4 members. Each team will be responsible for one phase of the project, but will participate in the other phases as well. For example, team 1 was responsible for the first phase, and members from team 2 and 3 worked with team 1 to help produce the deliverable for the first phase.

Team composition for the Team Redmond group.

Member Name	**Team**
Stefan Thibeault ^*	Requirements
Alexandre Bosserelle	Requirements
Eugena Zolorova	Requirements
Robert Hanna *	Design
Zhi Zhang	Design
Xin Xi	Design
Simon Lacasse *	Implementation
Patrice Michaud	Implementation
Hu Shan Liu	Implementation
Jens Witkowski	Implementation

^ Group Leader
*Team Leader

The language chosen to write Montrealopoly was Visual Basic. Several members were already familiar with the language and opted to be part of the implementation team. They also provided tutorials for the rest of the group members in order to help them become familiar with Visual Basic.

4 Team Members Log Sheets

4.1 Stefan Thibeault

Date	Task	Duration
Sept. 8	Discussion on project choice	0.5 Hours
Sept. 9	Group meeting – decided on project choice	1 Hours
Sept. 11	Group meeting – finalized decision on project choice	0.5 Hours
Sept. 15	Team 1 meeting – game rules, planning	3 Hours
Sept. 16	Group meeting – assigned tasks	2 Hours
Sept. 20	Team 1 meeting – req. document, assigned tasks	8 Hours
Sept. 23	Group meeting – assigned tasks, set deadline	1.5 Hours
Sept. 25	Robert, Stefan – quick meeting with professor	1 Hour
Sept. 27	Individual – Introduction, Development Plan	8 Hours
Sept. 28	Team 1 meeting – synchronization, review of req. doc.	6 Hours
Sept. 28	Individual – Document Integration and revisions	4 Hours
Sept. 29	Individual -- Requirements Document revision	8 Hours
	Total :	43 Hours

4.2 Robert Hanna

Date	Task	Duration
Sept. 2	Group Formation	0.25 Hours
Sept. 5	Section change – new group	0.25 Hours
Sept. 8	Discussion on project choice	0.5 Hours
Sept. 9	Group meeting – decided on project choice	1 Hours
Sept. 11	Group meeting – finalized decision on project choice	0.5 Hours
Sept. 15	Team 1 meeting – game rules, planning	3 Hours
Sept. 16	Group meeting – assigned tasks	2 Hours
Sept. 20	Team 1 meeting – req. document, assigned tasks	8 Hours
Sept. 23	Group meeting – assigned tasks, set deadline	1.5 Hours
Sept. 24	Individual – UML diagrams, research for UML	4 Hours
Sept. 25	Individual – UML diagrams	5 Hours
Sept. 25	Robert, Stefan – quick meeting with professor	1 Hours
Sept. 26	Individual – UML diagrams	4 Hours
Sept. 27	Individual – UML diagrams	3 Hours
Sept. 28	Team 1 meeting – synchronization, review of req. doc.	6 Hours
	Total :	40 hours

4.3 Alexandre Bosserelle

Date	task	Duration
Sept. 8	Discussion on project choice	0.5 Hours
Sept. 9	Group meeting – decided on project choice	1 Hour
Sept. 11	Group meeting – finalized decision on project choice	0.5 Hours
Sept. 15	Team 1 meeting – game rules, planning	3 Hours
Sept. 16	Group meeting – assigned tasks	2 Hours
Sept. 20	Team 1 meeting – req. document, assigned tasks	5 Hours
Sept. 22	Design of the Montrealopoly board	2 Hours
Sept. 22	User interfaces	4 Hours
Sept. 23	User interfaces	2 Hours
Sept. 23	Group meeting – assigned tasks, set deadline	1.5 Hours
Sept. 26	Dice Design	3.5 Hours
Sept. 28	Team 1 meeting – synchronization, review of req. doc.	6 Hours
Sept. 28	Token Design – Documentation	2 Hours
	Total :	33 Hours

4.4 Zhi Zhang

Date	Task	Duration
Sept. 8	Discussion on project choice	0.5 Hours
Sept. 9	Group meeting – decided on project choice	1 Hour
Sept. 11	Group meeting – finalized decision on project choice	0.5 Hours
Sept. 15	Team 1 meeting – game rules, planning	3 Hours
Sept. 25	Individual – Wrote assumption, constraints and specifications requirements	5 Hours
Sept. 28	Team 1 meeting – synchronization, review of req. doc.	6 Hours
	Total :	16 Hours

4.5 Eugena Zolorova

Date	Task	Duration
Sept. 8	Discussion on project choice	0.5 hours
Sept. 9	Group meeting – decided on project choice	1 hour
Sept. 11	Group meeting – finalized decision on project choice	0.5 hours
Sept. 20	Team 1 meeting – req. document, assigned tasks	5 Hours
Sept. 28	Individual – Wrote product functions	6 Hours
	Total :	13 Hours

4.6 Xin Xi

Date	Task	Duration
Sept. 8	Discussion on project choice	0.5 Hours
Sept. 9	Group meeting – decided on project choice	1 Hours
Sept. 11	Group meeting – finalized decision on project choice	0.5 Hours
Sept. 16	Group meeting – assigned tasks	2 Hours
Sept. 28	Team 1 meeting – synchronization, review of req. doc.	6 Hours
	Total :	10 Hours

4.7 Simon Lacasse

Date	Task	Duration
Sept. 8	Discussion on project choice	0.5 Hours
Sept. 9	Group meeting – decided on project choice	1 Hour
Sept. 11	Group meeting – finalized decision on project choice	0.5 Hours
Sept. 20	Read First Requirement Phase Document	0.5 Hours
Sept. 23	Group meeting – Visual Basic/Java Decision	1 Hour
Sept. 28	Read and Analyzed Requirement Document	2 Hours
Sept. 28	Started Game Prototype Implementation	1.5 Hours
Sept. 29	Function Definitions	1 Hour
	Total :	8 Hours