Group Project Design Document

Table of contents

1 Project Summary . . . 4

1.1 Purpose . . . 4

2 Scope and Business model . . . 4

2.1 Scope . . . 4
2.2 Business Model . . . 4

3 Definition, Acronyms and Abbreviations . . . 4

3.1 Acronyms and Abbreviations . . . 4
3.2 Definitions . . . 4

4 Overall Description . . . 5

4.1 Product perspective . . . 5
4.2 Sample GUI . . . 5
4.3 User Requirement . . . 6
4.4 User Characteristics . . . 6
4.5 Constraints . . . 6
4.6 System Requirement . . . 6
4.7 Performence Requirement . . . 6
4.8 Security . . . 7
4.9 Portability . . . 7
4.10 Description . . . 7
4.11 Specifics . . . 7

5 State Diagram . . . 7

6 Sequence Diagram . . . 10

7 Object Diagram . . . 14

8 Use case . . . 15

9 Software Design Introduction . . . 15

9.1 Purpose . . . 15
9.2 Scope . . . 16
9.3 Definition . . . 16
9.3.1 Move . . . 16
9.3.2 System . . . 16
9.3.3 Places . . . 16
9.3.4 Help . . . 17

9.4 Reference . . . 17
9.4.1 The Requirement Documentation . . . 17
9.4.2 Software Development Life Cycle essay . . . 17
9.4.3 IEEE . . . 17
9.5 Overview . . . 17

10 Interface Description . . . 17

10.1 Black Market . . . 17
10.2 Buy in & Sell out . . . 18
10.3 Protagonist's Storehouse . . . 18
10.4 Protagonist's Status . . . 19
10.5 Places . . . 20
10.6 The City Map . . . 20
10.7 The Day Left in the Game . . . 21
10.8 System Menu . . . 21
10.9 Places Menu . . . 22
10.10 Help Menu . . . 22

11 System Architecture . . . 23

11.1 System Architecture Diagram . . . 23
11.1.1 Game Interface . . . 23
11.1.2 Bank . . . 23
11.1.3 Hospital . . . 23
11.1.4 Puchase . . . 23
11.1.5 Sell . . . 23
11.1.6 Day . . . 23
11.1.7 Player's status . . . 23
11.1.8 Map . . . 24

12 Detailed design . . . 24

12.1 Class diagram . . . 24
12.1.1 Main System . . . 25
12.1.2 Player . . . 25
12.1.3 Wallet . . . 26
12.1.4 Events . . . 27
12.1.5 Stock . . . 27
12.1.6 Goods . . . 28
12.1.7 Health . . . 28
12.1.8 eventA . . . 29
12.1.9 eventB . . . 29
12.1.10 reputation . . . 29
12.1.11 Location . . . 30
12.1.12 LeaderBoard . . . 30
12.1.13 Update . . . 31

13 Development techniques & supported scope . . . 31

13.1 Programming language & environment . . . 31
13.2 Supported platform . . . 31
13.2.1 Android platform . . . 31
13.2.2 Web-based application . . . 31
13.3 Graphical drawing software . . . 32
13.3.1 Draw.io . . . 32
13.3.2 Visual Paradigm . . . 32

14 Game software maintenance . . . 32

14.1 Bug reporting . . . 32
14.2 Futher improvement . . . 32
14.3 The larger scope of platforms & necessary update . . . 33

15 Testing . . . 33

16 Update Log . . . 36

1 Project Summary

1.1 Purpose

To make people happy and gain a better mathematical skill and financial skill. You will encounter various events in the game, making you feel alive and hard, the world is cold, and the absurdity of this era.

2 Scope and Business model

2.1 Scope

This document outlines the overall functionality, constraints and specifications and requirements (functional, non-functional, user and system requirements), use cases, and the first version of the informal spec. This game available on multiple platforms (Web and Android).

2.2 Business Model

This game will be free to use. There will be no cost for users downloading the game and there will be no advertisements in the game. The reason for allowing users to use the game for free is to we want to get experience on how to make games and gain a better understanding of how to make great games. Users will have ease and enjoyment while they play this game. This will create a network effect which will motivate others to download. Once enough exposure is reached, further product is coming soon.

3 Definition, Acronyms and Abbreviations

3.1 Acronyms and Abbreviations

API - Application Program Interface - External software

GUI - Graphical User Interface

Java SE - Java Platform, Standard Edition

3.2 Definitions

Devices: In our application, devices will consist of Android smartphones and website.

Health: The game will end if player's hp decreases to 0.

Reputation: At the beginning it is 100, it will only be lost, and buying and selling illegal goods will reduce its reputation. Lower reputation will lead to some bad random events and may affect the final game evaluation.

Warehouse: The place where the player can store the goods, the real estate store can increase the warehouse ceiling.

Realtor: The place to buy a warehouse.

Cash: The amount that players can control at any time, may be affected by random events.

Deposit: The player has a property of the bank that is not affected by random events and increases interest on a regular basis.

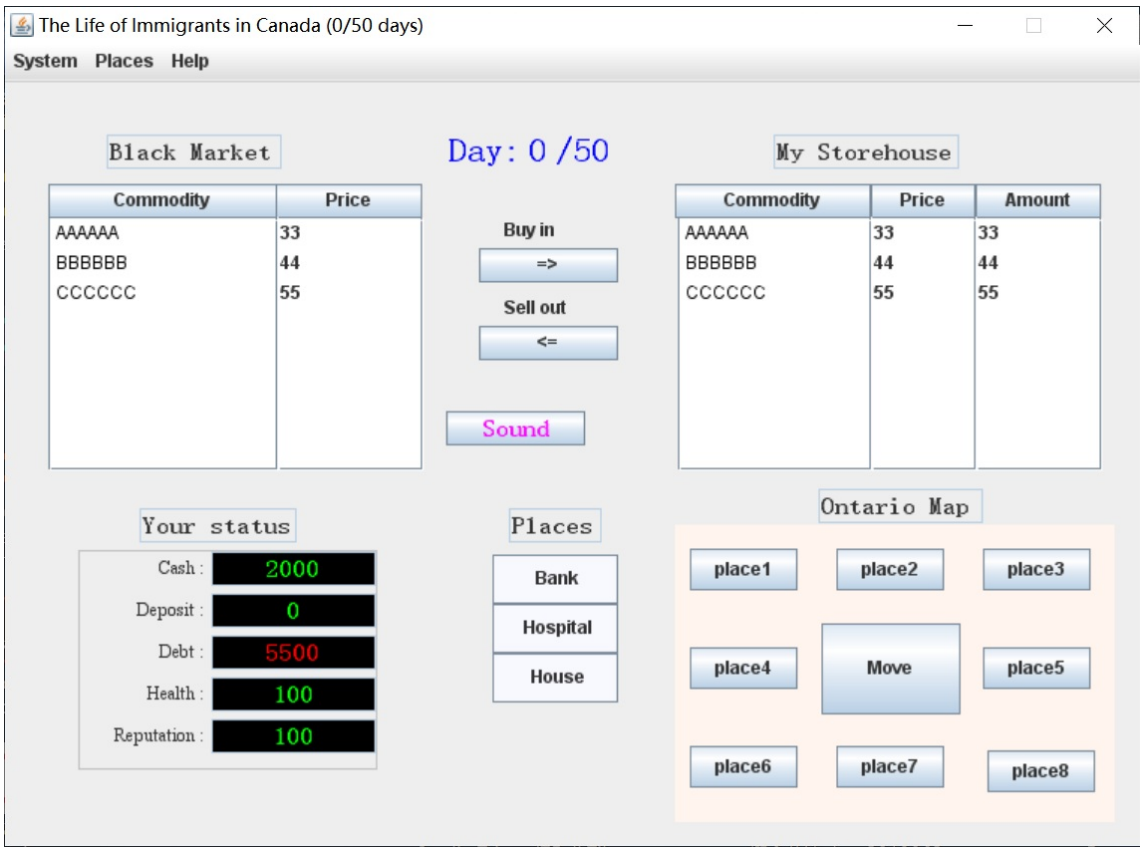
Arrears: A loan shark that will increase over time.

4 Overall Description

4.1 Product perspective

This is an RPG game. You will play a new Canadian immigrant in the game, walk around the streets, make money by buying and selling goods, pay off the road and live a better life. During this time, you will also encounter a series of interesting events to help (or prevent) your new life in Canada.

4.2 Sample GUI



4.3 User Requirement

The game will advance by time, and each move will advance one day for a total of 60 days.

You will owe a usury loan at the beginning because of the toll, and pay off the debt as early as possible to prevent snowballing.

You can make money by buying and selling goods in the market.

Every time there is a chance to happen, there are good and bad events, and the probability of receiving health, reputation, and arrears.

Banks can pay off their debts, save money, save money and generate interest, and prevent property damage caused by accidents.

The warehouse can be purchased in the real estate market, and the price and warehouse space are related to your total assets.

The hospital can recover its health by consuming money.

The new area will be opened on the 30th day (new goods will be available in the new area)

Reselling illegal goods can be highly profitable but will reduce reputation.

After the game is over, the score will be counted in the leaderboard.

4.4 User Characteristics

Anyone who wants to play real-life-simulation game.

4.5 Constraints

This is a stand-alone game that only offers 1 person to play.

Game updates require an internet connection, please ensure that the environment has fundamental internet connection.

This game is for entertainment only, please comply with laws and regulations, and do not imitate the behavior in the game.

4.6 System Requirement

The system will automatically check for game updates on the last day of each month.

The player's score is automatically saved.

The system automatically collects errors from the game crash and asks the user if they want to upload an error message.

4.7 Performence Requirement

This game only supports 1 player. The game is recommended to run on a dual-core processor with 1G of memory for a better gaming experience.

4.8 Security

The game does not send any information to the server without the user's permission.

4.9 Portability

This game maybe available in Android and iOS in the future.

4.10 Description

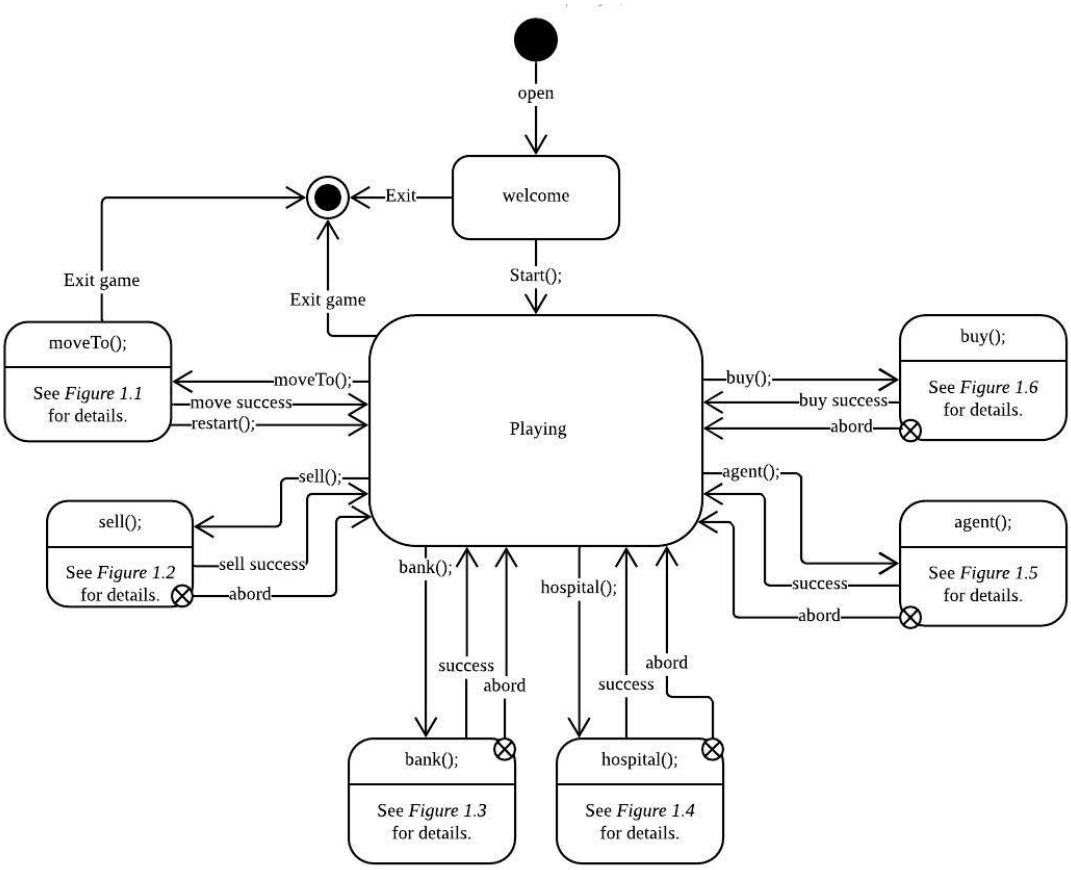
Due to the differing design restrictions and guidelines, the android and iOS versions will not be able to have the exact same interface.

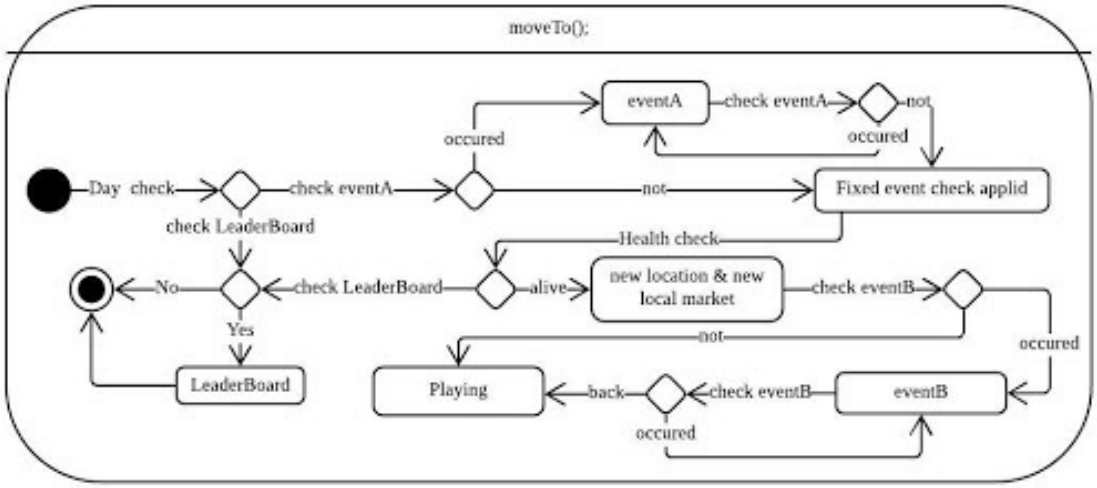
To get around some of these difficulties, the prototypes will be created using a platform called Figma which can be used to merge the designs to be as similar as possible.

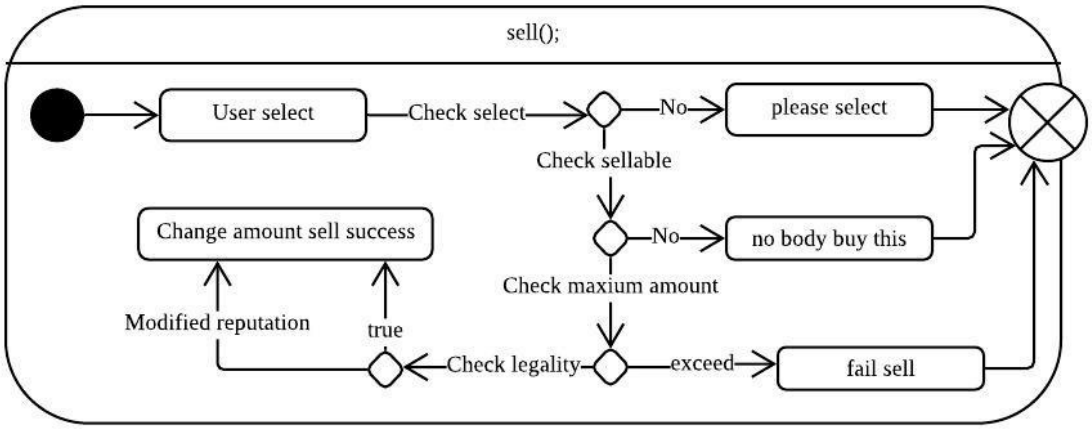
4.11 Specifics

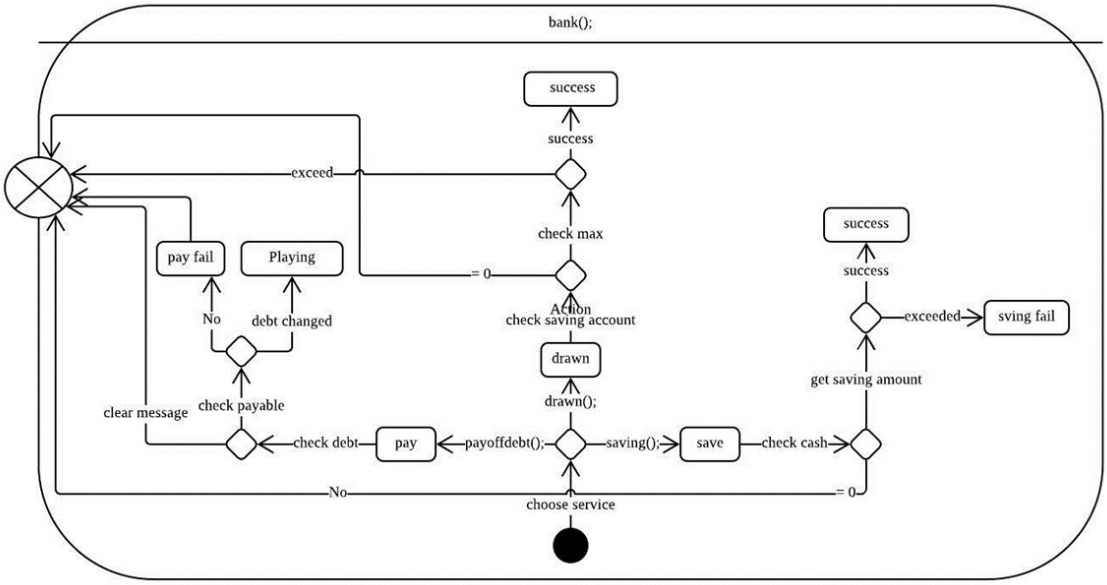
The specifics of the GUI and the languages used for iOS and Android are in the Design Specs Document.

5 State Diagram




moveTo();
eventA
check eventA
not
occured
occured
Day check
check eventA
not
Fixed event check applid
check LeaderBoard
Health check
No
check LeaderBoard
alive
new location & new local market
check eventB
Yes
not
occured
LeaderBoard
Playing
back
check eventB
eventB
occured


sell();
User select
Check select
No
please select
Check sellable
Change amount sell success
No
no body buy this
Check maxium amount
Modified reputation
true
Check legality
exceed
fail sell


bank();
success
success
exceed
check max
success
pay fail
Playing
= 0
Action
check saving account
success
exceeded
sving fail
No
debt changed
drawn
get saving amount
clear message
check payable
drawn();
check debt
pay
payoffdebt();
saving();
save
check cash
No
= 0
choose service

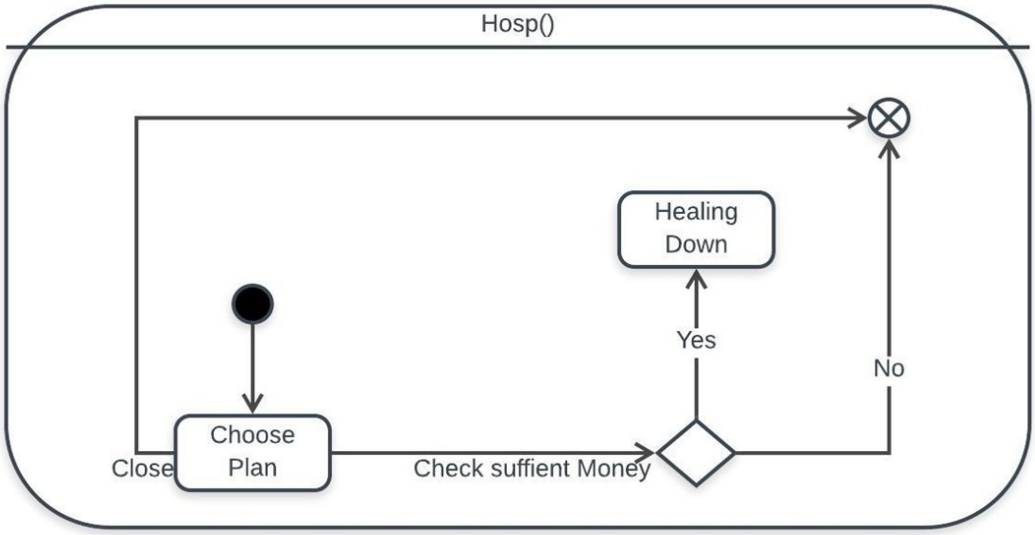


Figure 1.4

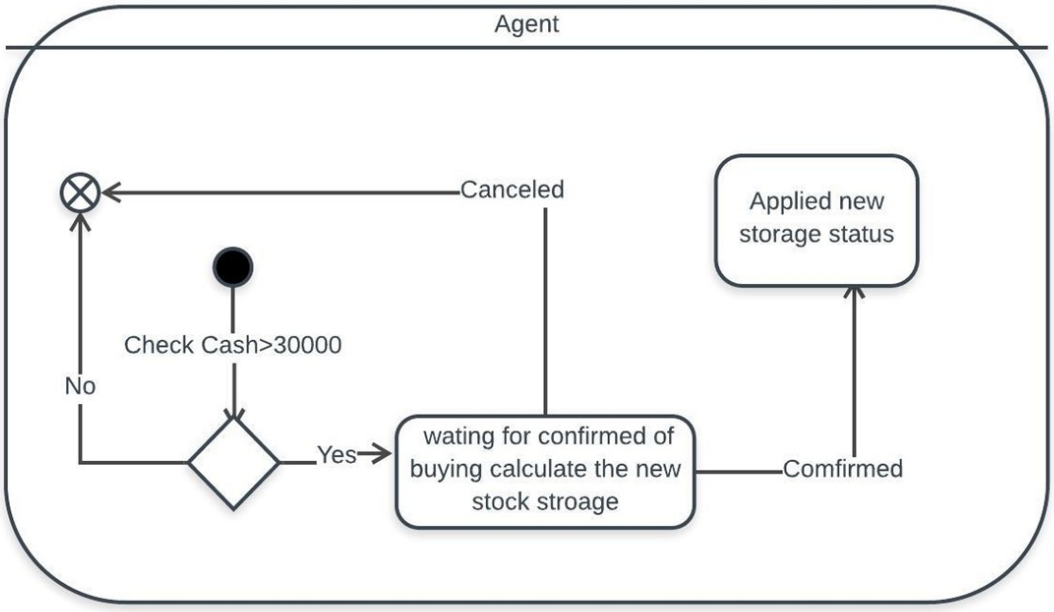
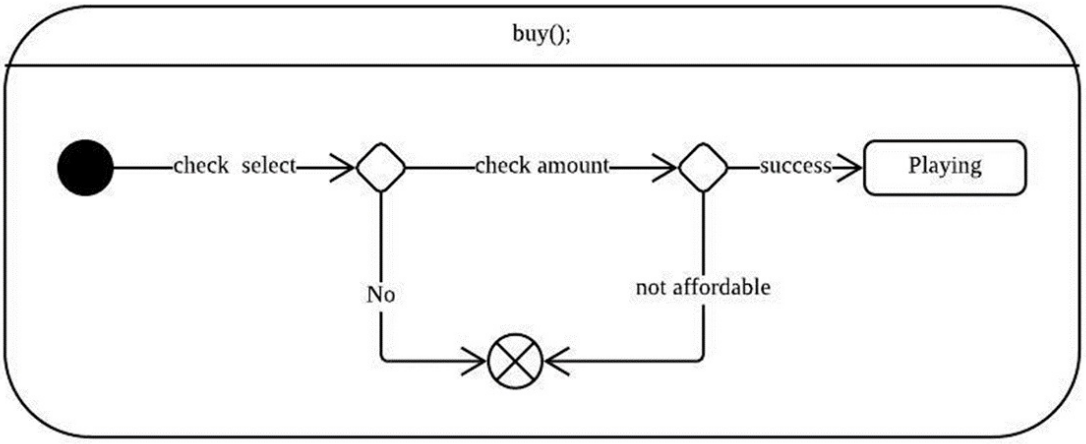
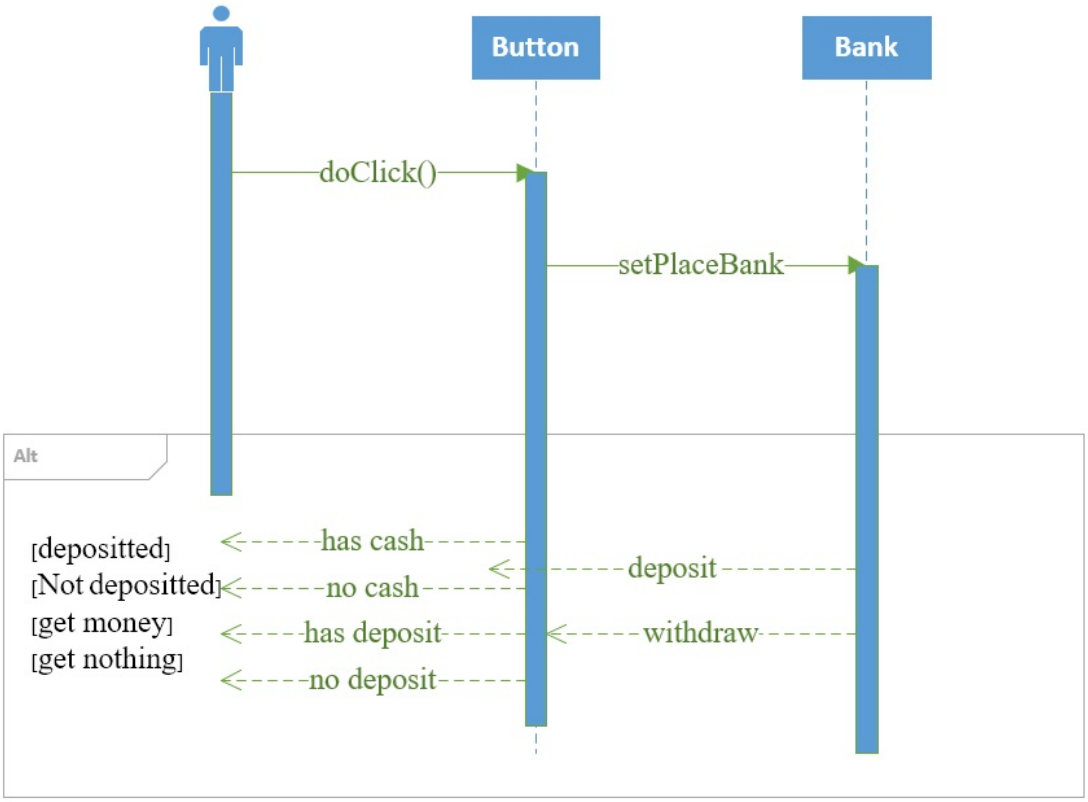
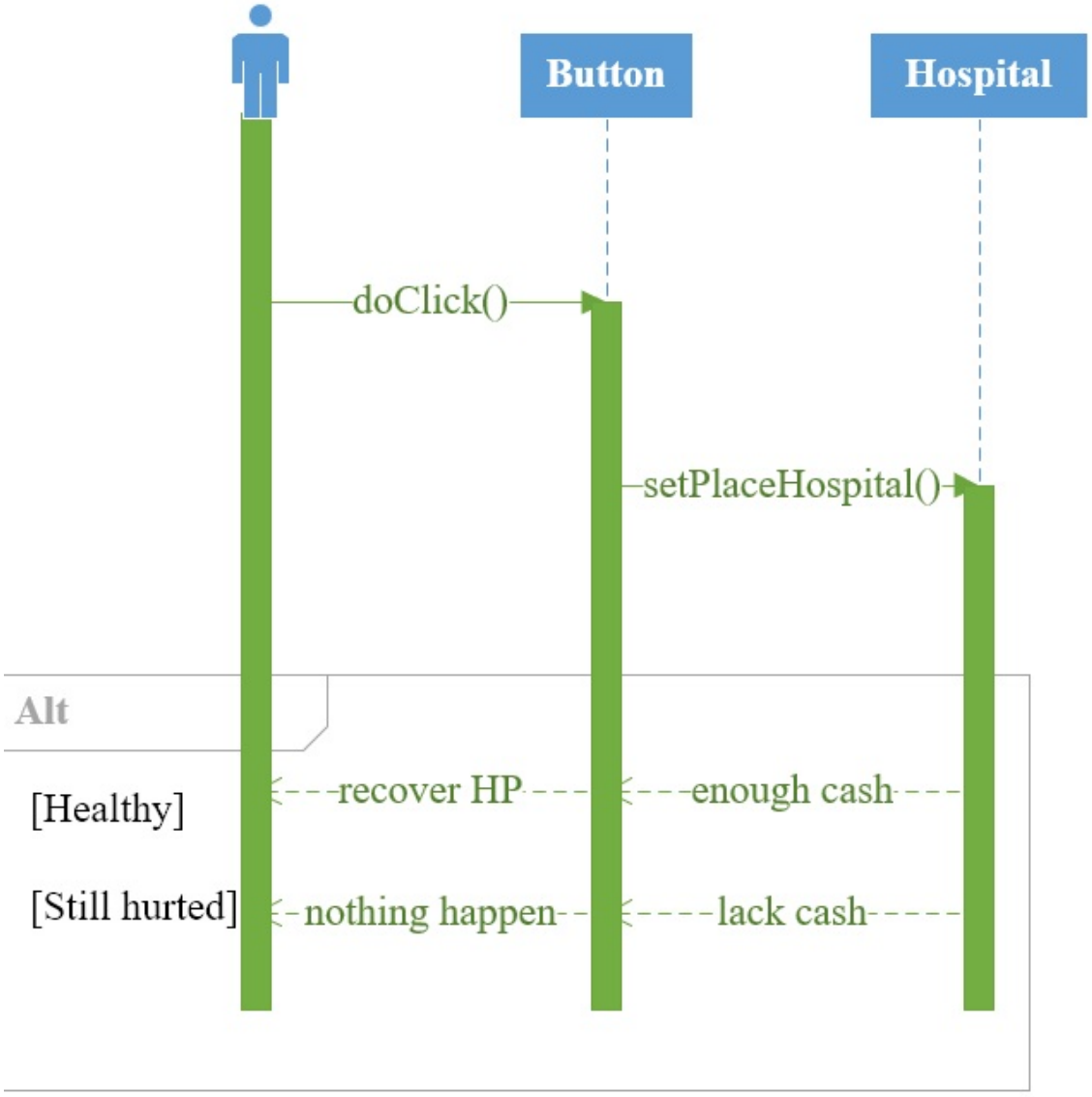


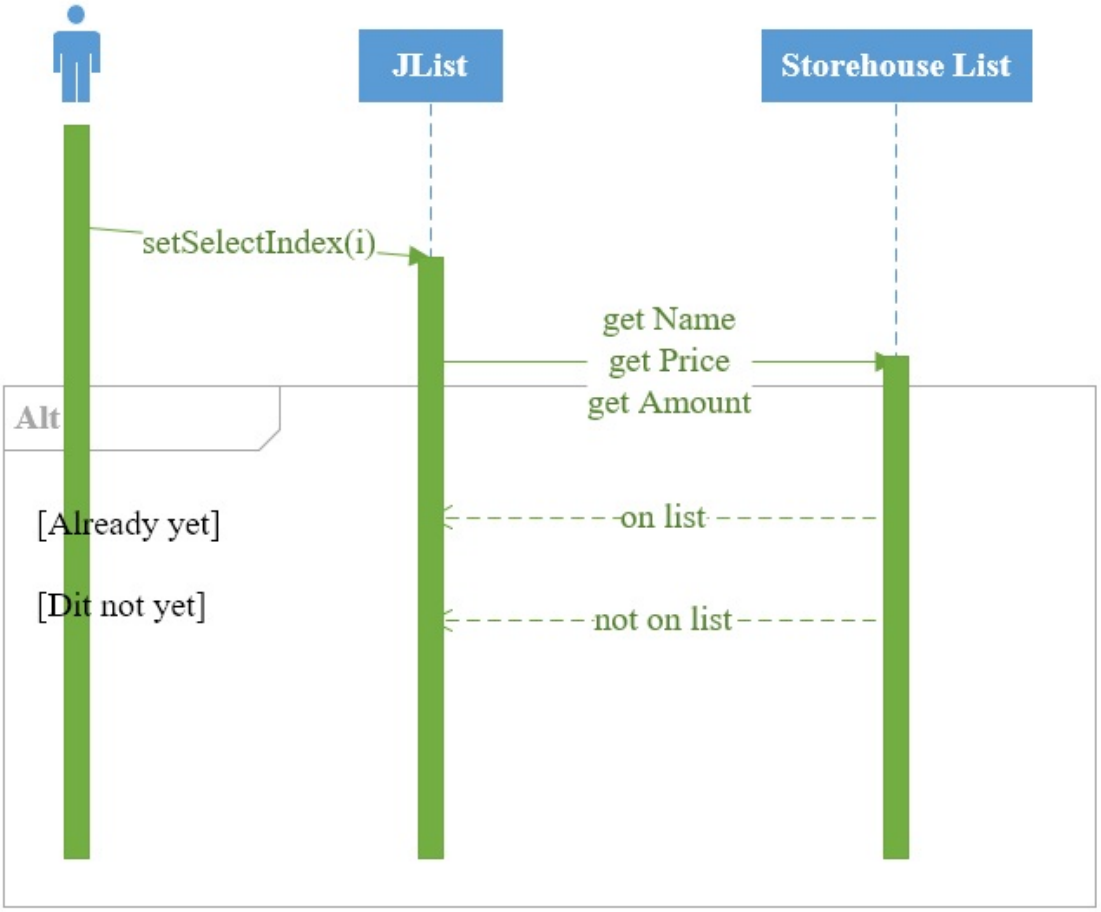
figure 1.5

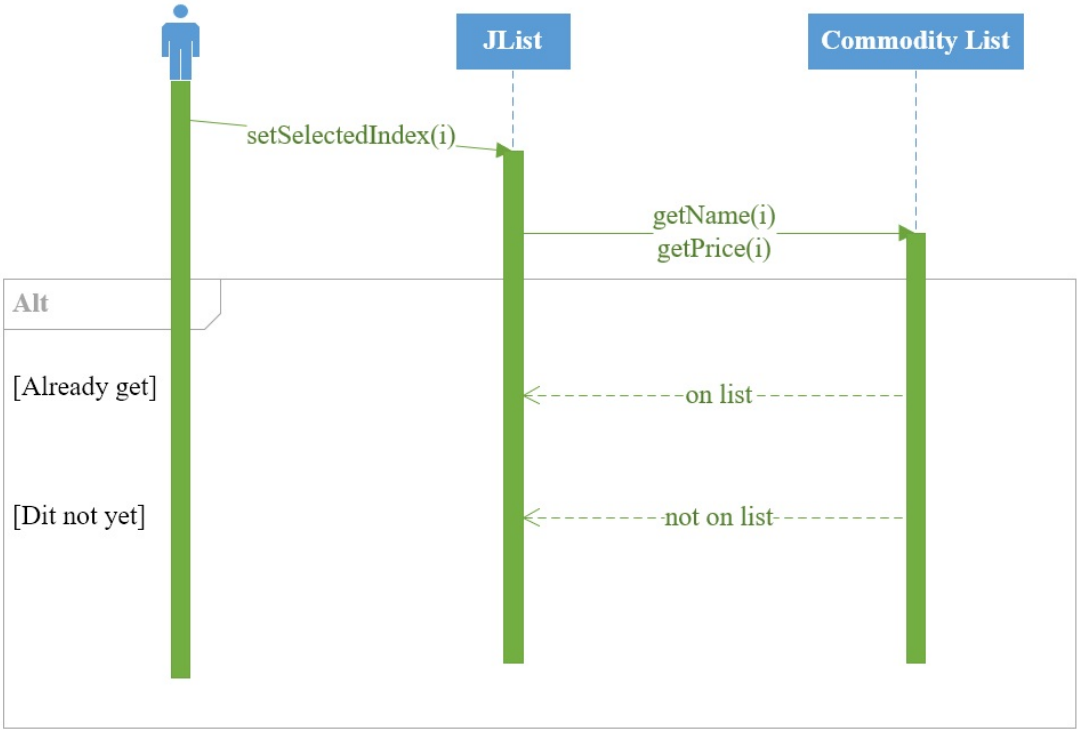


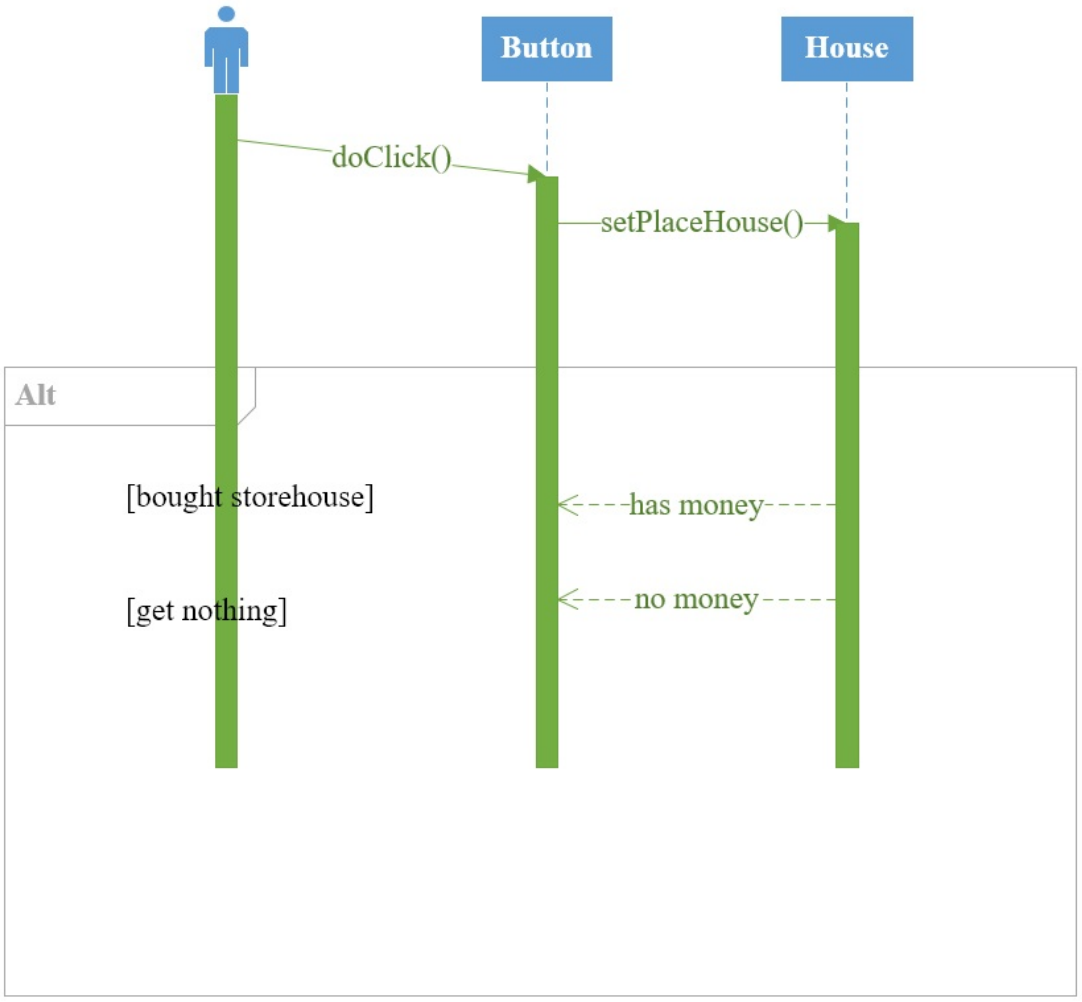
6 Sequence Diagram



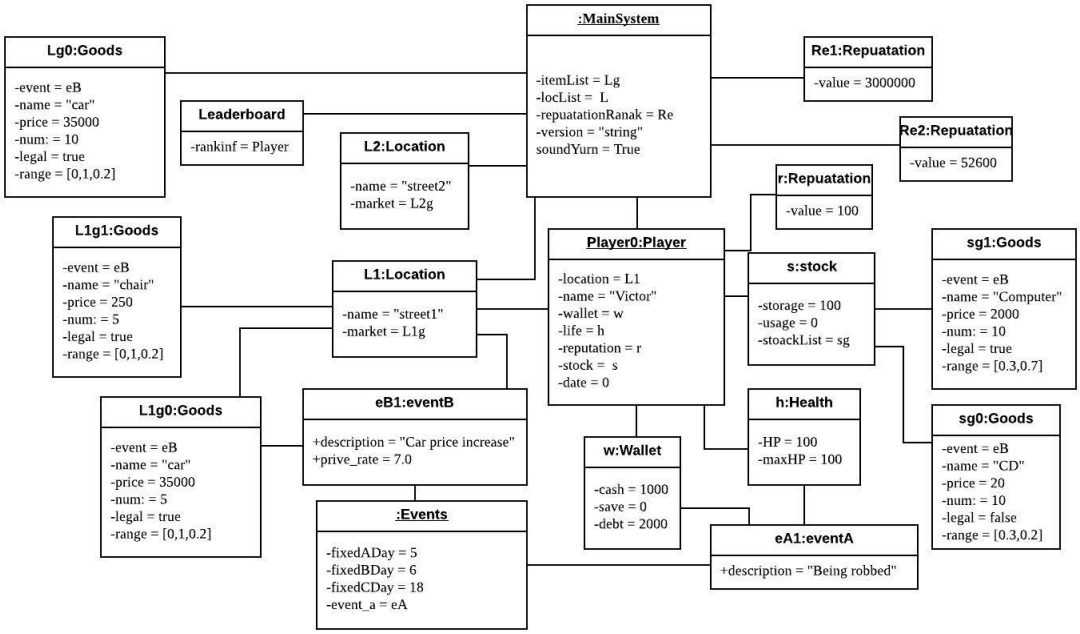

Button
Hospital
doClick()
setPlaceHospital()
Alt
[Healthy]
recover HP
enough cash
[Still hurted]
nothing happen
lack cash


JList
Storehouse List
setSelectIndex(i)
get Name
get Price
get Amount
Alt
[Already yet]
on list
[Dit not yet]
not on list


JList
Commodity List
setSelectedIndex(i)
getName(i)
getPrice(i)
Alt
[Already get]
on list
[Dit not yet]
not on list


Button
House
doClick()
setPlaceHouse()
Alt
[bought storehouse]
has money
[get nothing]
no money

7 Object Diagram



8 Use case

9 Software Design Introduction

9.1 Purpose

The game "Life of immigrants in Ontario" is designed for the customers, in other words, the players, who wants to play this game in order to have some funny moment and to release some of their stresses. This gaming software system designation doc-

umentation is aimed to verbally summarize our overall designing process, including the brief description of the each kind of interfaces of our game, the rough game system architecture as well as schematic representation of the details of the architecture, which using a architecture diagram to assist to representing the architecture of the system. Some of the initial details of the game will be discussed here to help the customers of our game to have the preliminary impression on the rough gaming experience, which will be illustrated by using a class diagram along with some of its methods and attributes, more specific technical details are likely to be added in the future version of the documentation for the users to read. There might be an activity diagram here which shows the end to end process of the gaming software system design or use case(s).

9.2 Scope

One of the possible intended reader of our gaming software system design documentation is each developing members in our development team, one of the usage of this document is to assist their designation by representing the brief designing process of the whole gaming software system. As well, the other possible audiences of this software design documentation is stakeholders who might be the future customers or users of the game, they are likely to review and provide some useful advices to help avoiding the possible technical mistakes that developers will make during developing the software, on the other hand, this will meet their requirements better.

9.3 Definition

9.3.1 Move

There are more than 10 virtually designed places in our game, it would be visually redundant if we put all of the places in the main interface of the game. It is better to limit the places in one page, however, there might be more places required to put into the game, so it is required to put a "move" button to switch the sub-interfaces of the places. This contributes, to a great extent, to the interface cleanliness of the game.

9.3.2 System

The menu bar which are put on the top of the gaming interface is to help provide the basic operations of the gaming interface such as restarting the game when players are judged to loss the game or not satisfied with the current status of the protagonist of the game, and some basic settings of the game like turning on and off the sound or allow some of the irregular events to happen. The ranking list helps to check the rank status of each player comparing to other players. Of course, when players are tired or some other reasons, they can choose to exit the game.

9.3.3 Places

The Places menu is aimed to help players to switch to the important virtual locations in the game such as the bank to depositing or withdrawing deposit; a hospital when the players wants to resume their HP value; a real estate agency to provide the warehouse...

9.3.4 Help

The help menu is to assist the new comer of this game to familiar with the game, there will be a brief introduction of the place where the game happens - the Ontario. And the background of the game to help the players to know what are they supposed to do in the game. Also, there will be a link to the documentations of the game.

9.4 Reference

9.4.1 The Requirement Documentation

The gaming software requirement documentation of "The Life of immigrants in Ontario"

9.4.2 Software Development Life Cycle essay

The survey paper on Software Development Life Cycle of "The Life of immigrants in Ontario"

9.4.3 IEEE

IEEE Std 1016-1998 IEEE Recommended Practice for Software Design Descriptions

9.5 Overview

This gaming software designation documentation is on the basis of the requirement documentation of the software system and its software development life cycle essay. As well, it is following the guideline of the IEEE Recommended Practice for Software Design description. In addition, there are some of the gaming elements added for designing the gaming software system.

10 Interface Description

10.1 Black Market

Interface elements	Type	Description/Use
Black Market	The label of the interface field	The market for players to purchase goods
Commodity	Item name	Name of the goods
Price	Item name	Prices of the goods

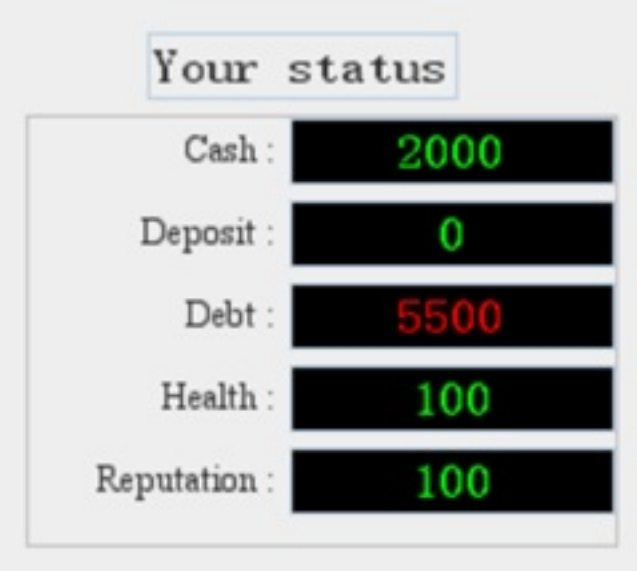
10.2 Buy in & Sell out

Interface elements	Type	Description/Use
Buy in	Button	User purchase goods from the market to his/her own warehouse
Sell out	Button	User sell his/her goods from his warehouse to the market

10.3 Protagonist's Storehouse

Interface elements	Type	Description/Use
Storehouse	The label of the interface field	The storehouse is used to store the goods that user bought from the market, might in the future
Commodity	Item name	The name of the goods bought
Price	Item name	The price of goods bought
Amount	Item name	The number of goods bought

10.4 Protagonist's Status



Interface elements	Type	Description/Use
Protagonist's status	The label of the interface field	Some fundamental status of a person(here the protagonist)
Cash	Item name & data area	The money that the protagonist own carried currently
Deposit	Item name & data area	The money that the protagonist own and are stored in the bank.
Debt	Item name & data area	The sum of money that the protagonist owes others
Health	Item name & data area	The Hit Point value of the protagonist
Reputation	Item name & data area	The repuation of the protagonist might some other aspects accidently

10.5 Places

Interface elements	Type	Description/Use
Bank	Button	Let the player go to the bank to deposit or withdraw the deposit from the bank account
Hospital	Button	Let the player go to the hospital to resume some of a certain amount of the hit point value
House	Button	Let the player go to the Real Estate Intermediary to buy a house to expand the warehouse

10.6 The City Map

Interface elements	Type	Description/Use
City map	The label of the interface field	The map gives the cities in Ontario, user can choose the city to go to
Places	Button	User click on the place button to go to the corresponding city to buy or sell commodities
Move	Button	User click on the "Move" button to switch the sub-interfaces of the places

10.7 The Day Left in the Game

This shows the "day" left in the game, to mention the player to pay the debt within these limited days, otherwise, the game will fail.

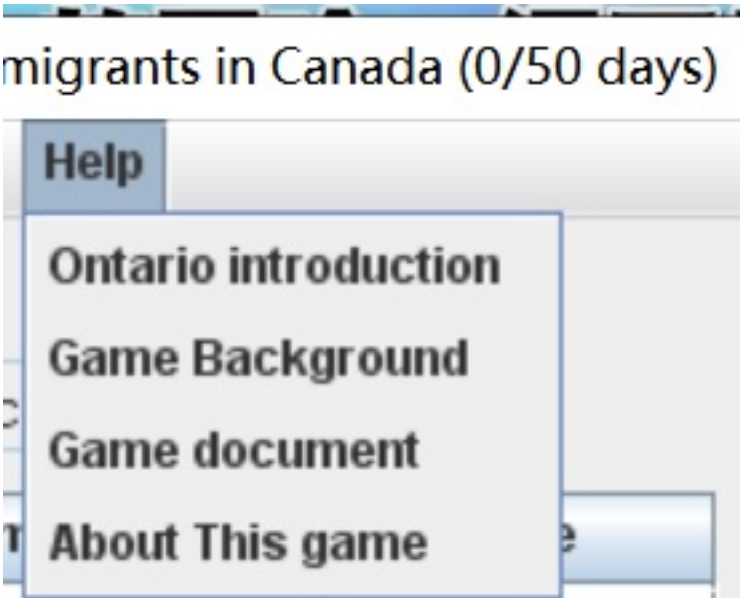
10.8 System Menu

Interface elements	Type	Description / Use
New game	Menu item	User click to restart *a* new game
Setting	Menu item	User clickes to the setting interface
Exit	Menu item	User click to exit the game

10.9 Places Menu

Similar functionality with "places" field of the game interface.

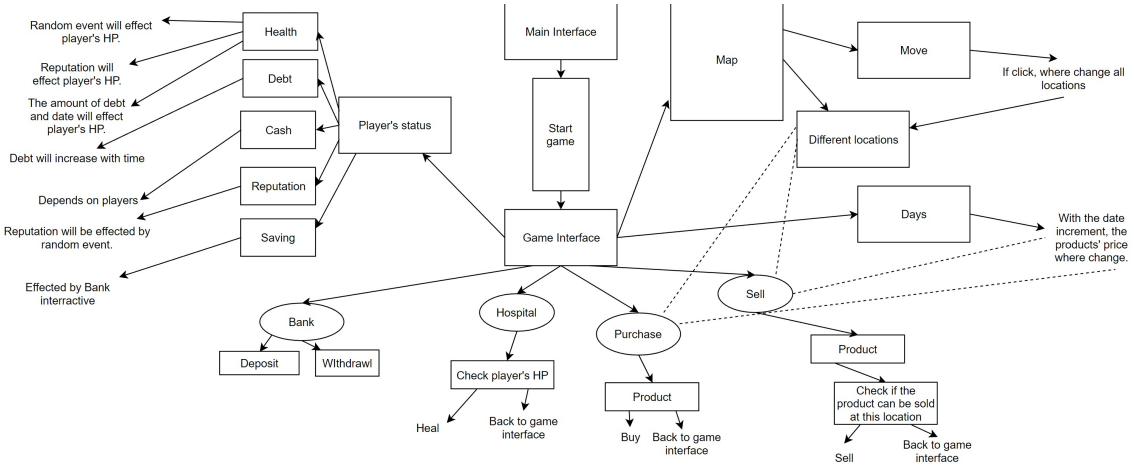
10.10 Help Menu



Interface elements	Type	Description/Use
Ontario introduction	Menu item	User click to see the brief introduction of the background city of this game - Ontario
Game background	Menu item	User click to see the background story of this game
Game documentation	Menu item	User click to see all of the documentation of this game
About the game	Menu item	User click to see some fundamental information about this game

11 System Architecture

11.1 System Architecture Diagram



11.1.1 Game Interface

This is the most important component in the game, it has 7 other sub components: bank, hospital, purchase, sell, day, player's status and map.

11.1.2 Bank

The bank can let user deposit and save money.

11.1.3 Hospital

The hospital can heal user's health if their health is not 100.

11.1.4 Puchase

Purchasing system can make players purchase the things which they want to sell.

11.1.5 Sell

The selling system can let players sell their owned goods.

11.1.6 Day

The day indicator displays the current days and total days.

11.1.7 Player's status

Cash is the money that players can buy products or repayment of arrears at any time.

Saving cannot affect by random events, it is the safest way to save player's money.

Reputation can be affected by random events, it has been connected by goods' price.

Repaying money is the goal of this game.

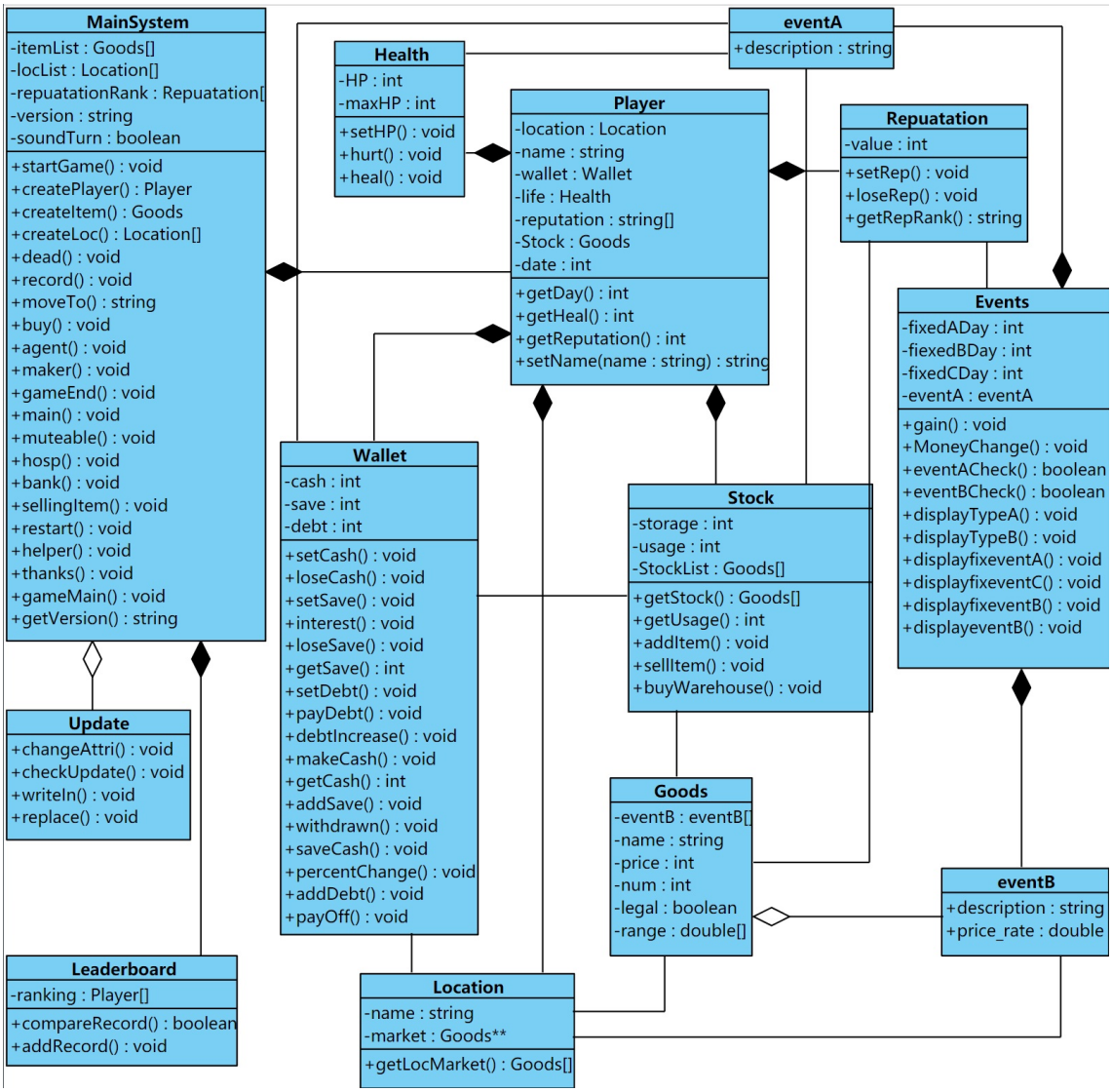
Health is the most basic value of this game, it determines whether the player can continue the game, it will be affected by random events.

11.1.8 Map

Map displays the locations which user can move. It has a "move" button, will change all locations randomly. Each time player press the "move" button, the number of days will increase by one day.

12 Detailed design

12.1 Class diagram



12.1.1 Main System

1. Class Description

   Class record basic information of game setting and version.

2. Attributes

   i. itemList: A list contained all object of good that will appear in the game.(List of Goods)

   ii. locList: A list contained all object of location that will appear in the game.(List of Location)

   iii. reputationRank: A list of String contained the representation words for reputation.(List of String)

   iv. SoundTurn: A switch for sound setting. ()

3. Methods

   i. Button-called methods

      These methods will be implemented to different button in the game view, It included: startGame(), record(), moveTo(), agent(), maker(),buy(), sellingItem(), hosp(), bank(), reastart(), muteable(), helper()

   ii. Methods for view

      These are the different view will be appeared in the program It included:

      main(), thanks(),gameMain()

   iii. Methods for crediting objects

      These methods are for creating certains object. It included:

      createPlayer(),createLoc(),createItem()

12.1.2 Player

1. Class Description

   A class of the player records the player status, when the game start, a player object

   has been created.

2. Attributes

    i. location: Record the current player location.(int)

    ii. name: Default name is “Nobody”. required enter name if enter leaderBoard.(String)

    iii. wallet: Record the current player financial status.(wallet)

    iv. life: Record the user health level. zero will lead to bad ending.(health)

    v. reputation: Record the current player reputation situation.(reputation)

    vi. warehouse: Record the current player warehouse status.(stock)

    vii. date: Record the current date as game progress notify.(int)

3. Methods

    getDay(); getHeal(); getReputation(); setName(name);

12.1.3 Wallet

1. Class Description

    A class record the financial status of the player, handle the financial transaction.

2. Attributes

    i. cash: Player current cash amount.(int)

    ii. save: Player current saving amount.(int)

    iii. debt: Player current debt need to payoff.(int)

3. Methods

    setCash(int); loseCash(int); setSave(int); loseSave(int); setDebt(int); payDebt(int);

    getCash(); getSave(); getDebt();

    addSave(int); makeCash(int); addDebt(int);

    These three mean increase the amount of target attributes.

withdrawn(); saveCash(); payOff();

These three are the operation in bank been performed.

interest(); debtIncrease();

These two are the natural increase of saving and debt.

precentChange()

This method handles the percentage changing in cash or saving. mostly occurred by TypeA random event.

12.1.4 Events

1. Class Description

   A class handles the fixed/random events happened.

2. Attributes

   i. fixedADay: The counter for fixed event A. Once every five days after certain conditions are met.(int)

   ii. fixedBDay: The counter for fixed event B. Once every five days after certain conditions are met.(int)

   iii. fixedCDay: The counter for fixed event C. Once every five days after certain conditions are met.(int)

   iv. eventA: A list of random TypeA event.(List of eventA)

3. Methods

   i. gain(): handled obtained goods event.

   ii. moneyChange(): handled lose or get money event.

   iii. eventACheck(); eventBCheck(); check the random event happened or not.

   display method still undetermined the usage. Need further testing.

12.1.5 Stock

1. Class description

   A class record the stock status of the player. handling the transaction of goods.

2. Attributes

    i. storage: The maximum size of stock.(int)

    ii. usage: Warehouse used capacity.(int)

    iii. stockList: A list contained Goods that player owned.(List of Goods)

3. Methods

   getWarehouse(); Get the content of stockList attribute.

   getUsage();

   addItem(); sellItem(); Modified the stock after transaction.

   buyStock(); Increase the storage of stock.

12.1.6 Goods

1. Class Description

   A class of single good information. uses to create Goods object.

2. Attributes

    i. name: The name of good.(string)

    ii. price: The price of good.(int)

    iii. legal: the legality of good.(boolean)

    iv. range: Two double number indicate the Lower limit and upper limit of Price floating range. (List of Double)

    v. num: The amount of good. (int)

    vi. eventB: List of TypeB random event which may happen affect good price.(List of eventB)

3. Methods

   None

12.1.7 Health

1. Class description

   A class records the health status of the player.

2. Attributes

    i. HP: The health point of player.(int)

    ii. maxHP: The maximum of HP could be.(int)

3. Methods

   setHP(int);

   hurt(int); HP decrease. mostly used by random typeA event.

   heal(double); HP recovery by percentage.mostly used by hosp();

12.1.8 eventA

1. Class description

   A class of single TypeA random event. TypeA random events included obtained good, lose health and lose or get money.

2. Attributes

   description: Textual description of the Type A random event.(string)

3. Methods

   createTypeA()

12.1.9 eventB

1. Class description

   A class of single TypeB random event. TypeB random events included the changing of goods price in markets.

2. Attributes

    i. description: Textual description of TypeB random event.(string)

    ii. price_rate: Fluctuation in the price of good.(double)

3. Methods

   createTypeB();

12.1.10 reputation

1. Class description

A Class record the player's reputation status.

2. Attributes

    i. value: the representation of reputation. (int)

3. Methods

setRep(); loseRep();

getRepRank()

The current reputation points will be compared with the reputationRank belongs to mainSystem class, and the evaluation words of the reputation will be returned.

12.1.11 Location

1. Class description

A class of single location object stored crucial information.

2. Attributes

    i. name: The name of location.(string)

    ii. market: A list of Goods for sale.(list of Goods)

3. Methods

getLocMarket(); Return the list of on-sale good in current location.

12.1.12 LeaderBoard

1. Class description

A class uses to process leaderBoard information.

2. Attributes

    i. ranking: A list of ten Players.(List of Player)

3. Methods

compareRecord(Player):

comparing the current player record with the LeaderBoard players, return the inserting index. -1 means did not enter leaderBoard.

addRecord(Player(player), index(int)):

inserting Player into certain index location and remove the last one player information.

12.1.13 Update

As the consequence of the time limitation, This functionality might not intended to deploy this feature now.

13 Development techniques & supported scope

13.1 Programming language & environment

We are using the Java programming language to write this gaming software system, in which the most commonly used techniques here, for this software in particular, is the Window Builder which is the additional plug-ins for the Eclipse Integrated Development Environment, the major usage of the Window Builder is to considerably accelerate the process of building the graphical user interface since there are many off-the-shelf java graphical user interface components built within the Window Builder plug-in.

13.2 Supported platform

13.2.1 Android platform

Our developing team is intended to develop a gaming software system on one of the most well-known platforms, which is the Android platform, in order to achieve this point, we have considered using a significantly powerful software framework, the Android Studio platform, which is the official integrated development environment for Google, designed to write the software for the Android operating system.

13.2.2 Web-based application

Another platform that we are trying to scheme a plan for is trying to develop a web-based application, the approach to accomplish this is to use a technique which is called the Java web start technique. The usage of java web start technique is that it allows the users to use a web browser to deploy the Java application software program directly on the internet. The java web start also supports the Java application software on the Eclipse integrated development environment, so here, in particular, using an Eclipse platform is sufficient for our developers in the application software development team.

13.3 Graphical drawing software

13.3.1 Draw.io

The *draw.io* here for our software development is used to draw the architecture diagram, this online diagram drawing software is aimed to draw flowcharts, process diagrams, org charts, UML, ER and network diagrams professionally, while there are no any additional charges.

13.3.2 Visual Paradigm

Visual Paradigm for our software development is used to draw the UML model. Its UML modelling tools allow users to model the classes, properties, and operations in the system in a UML class diagram, therefore it is particularly useful for our gaming software system. UML class diagrams are the blueprints needed to build a gaming software system since our developers need the help of class diagrams and class specifications.

14 Game software maintenance

14.1 Bug reporting

It is extremely likely that any software system would exist bug(s), and as a consequence, our gaming software system will take this into account and try our best to design a mechanism so that the players of our gaming software system are able to report any possible mistakes or bugs during the processing of the game, our developers will then modify the gaming software system in order to decrease the amount of the unbearable part of the game.

14.2 Futher improvement

Except for the obvious accidently-made bugs or mistakes, it is also considered possible that the user experience will be affected by the other looks-good part, such as the data balance of the entire game, or the rationality of the random event, or the overall interface's visual experience for the players. Those part might seem good for the developers, however, the ones that are most likely to discover the mistakes is the long-term experiencers of the gaming software system, which is our customers here in particular. Hence, our gaming software system's report mechanism must also consider this point to take their opinions into account and make some further improvements to the gaming software system.

14.3 The larger scope of platforms & necessary update

One of the regrets of our gaming software system is that we did not design for the IOS platform, we only support the Android application software operating system and the web-based platform. The reason for this is that our time is finite. As a consequence, if time allows in the future, we are likely to design the gaming software system for the IOS platform. Moreover, the platform that we have already considered will also have the updated version, the "Life of immigrants in Ontario" gaming software system will have to be updated as necessary in order to adopt those changes.

15 Testing

Testing is splited by seven parts:

1. sell
2. buy
3. mainUI
4. moveTo
5. agency
6. bank
7. hosp

The following content is seven corresponding testing table.

<table>
<tr><th>Test ID</th><th>Purpose</th><th>Steps</th><th>Input</th><th>Expected Output</th><th>Actual Output</th></tr>
<tr><td>sell-A</td><td>Test whether the product can be sold successfully</td><td rowspan="2">Click the sell button</td><td rowspan="5">The number of product</td><td>Can be sold and change amount sell success</td><td>Sold successfully and change amount sell success</td></tr>
<tr><td>sell-B</td><td>Test whether the product cannot be sold</td><td>Cannot sold and exit</td><td>Exit successfully</td></tr>
<tr><td>sell-C</td><td rowspan="2">Test the amount of product that will be sold</td><td>Click the sell button and input a number that greater than the maximum amount of the product</td><td>Will be sold</td><td>Sold successfully</td></tr>
<tr><td>sell-D</td><td>Cannot be sold</td><td rowspan="2">Sold failed</td></tr>
<tr><td>sell-E</td><td>Check the legality</td><td>Click the sell button</td><td>Sold failed</td></tr>
</table>

Test ID	Purpose	Steps	Input	Expected Output	Actual Output
buy-A	Test whether the product can be purchased successfully	When the product can be purchased, click the buy	The number of product	Successfully	Successfully
buy-B		When the product cannot be purchased, click the buy		Cannot be purchased	Cannot be purchased
buy-C	Check if the player has enough money to purchase the product	When the player has enough money, click the buy button		Buy successfully	Buy successfully
buy-D		When the player does not hava enough money, click the buy button		Buy failed	Buy failed
buy-E	Test the amount of product that will be sold	Click the sell button and input a number that greater than the maximum amount the product		Buy successfully	Buy successfully
buy-F				Buy failed	Buy failed

Test ID	Purpose	Steps	Input	Expected Output	Actual Output
mainUI-A	Test all the game button can work successfully	Click all the game buttons		Successfully get correspond functionality	Successfully get into the game
mainUI-B	Test all the top button can work successfully	Click help, system and important place		Display the corresponding message	Display the corresponding message successfully
mainUI-C	Test the game can exit successfully	Click the exit button of window		The game process will be terminated	The game process will be terminated

Test ID	Purpose	Steps	Input	Expected Output	Actual Output
moveTo-A	Test the player can move to the destination	Click the destination		Applied events if event happened and update new location and new market	Applied events and update new location and new market successfully
moveTo-B				Health eqals to zero then game over	Game over
moveTo-C	Check if the remaining days is equivalent to zero			End game	End game successfully
moveTo-D				Go to the leaderboard	Updated the leaderboard

Test ID	Purpose	Steps	Input	Expected Output	Actual Output
agency-A	Test housing agency	1. Click the housing agency button 2. Click transaction button 3. Display the result message		If user has enough money then update the storage status	Update the storage status successfully
agency-B				If user does not enough money then does not update the storage status exit	Does not update the storage status and exit the window successfully
agency-C		1. Click the housing agency button 2. Click the cancel button 3. Exit the agency window		Exit the windows	Exit the windows successfully

Test ID	Purpose	Steps	Input	Expected Output	Actual Output
bank-A	Test whethe it is drawn money successfully	Click the drawn button		Drawn successfully	Drawn successfully
bank-B	Test whether the max in saving account			Check the money max	Does not exceed max and success
bank-C	Test whether saving account is zero			Drawn zero and failed	Drawn zero and failed
bank-D	Test whether it saves money successfully			Save successfully	Save successfully
bank-E	Test whether it exceed the max			Check the money max	Does not exceed max and success
bank-F	Test whether it is zero in cash			Cash is zero and failed	Cash is zero and failed
bank-G	Test whether it clear message	Click the debt button		Clear message	Clear message
bank-H	Test whether it is payable			Pay successfully	Pay successfully
bank-I	Test whether it pay failed			paid failed	paid failed

Test ID	Purpose	Steps	Input	Expected Output	Actual Output
hosp-A	Test whether it choose a plan	Click the choose plan button		Do not choose and fail	Choose one, having sufficient money and healing down
hosp-B	Test whether it healing down			Having sufficient money and healing down	Having sufficient money and healing down
hosp-C	Test whether it is sufficient money			Do not have sufficient money and failed	Do not have sufficient money and failed

16 Update Log

- 1.0.0

  Setup the basic main interface & menu bar of the GUI

- 1.0.1

Allow to add day by 1 by switching places (show on JFrame title)

- 1.0.2

  Implemented top menu bar "help" item initially

  Added "Ontario introduction"

  Added "Game background"

  Added "About this game"

- 1.0.3

  Allow to add day by 1 by switching places (more showy, show on interface)

- 1.0.4

  Aetup the "move" button on the "map panel" initially

  (Each map has 8 places, click "move" to switch map1 / map2)

  (But click "move" button, interface will flash)

- 1.0.5

  Aetup the JList of the "market" panel initially

  (User can click on the items to select them)

- 1.0.6

  Added sound effect, click some of the buttons can active sounds

- 1.0.7

  Setup "generating commodity list" machanism

  (Each place generate 5-7 items randomly)

- 1.0.8

  Added mouse listener on the JButton

  Mouse enter / exit button: change color / background

- 1.0.9

  Added background icon to the JButton (bank,hospital,post office,house)

- 1.1.0

  Found a way to compile eclipse project to exe file

- 1.1.1

Setup buy-in and sell-out buttons

- 1.1.2

  Setup bank mechanism initially (listener inside bank button)

- 1.1.3

  Setup hospital mechanism initially (listener inside hospital button)

- 1.1.4

  Setup post office mechanism initially (listener inside post office button)

- 1.1.5

  Setup cheat & test mechanism

  Cheat: good things to player

  Test: bad things to player

- 1.1.6

  Setup house mechanism initially (listener inside house button)

- 1.1.7

  Setup some basic logic

  50 days game over

  10% debt everyday

- 1.1.8

  Improve "map panel" problem

  Click "move" button, interface won't flash anymore

- 1.1.9

  Wrote "commodity" class to represent "commodity" objects

- 1.2.0

  Added some basic logics

  Bank interest add 1% everyday

  Some basic random events

- 1.2.1

  Added sequence for commodities

- 1.2.2

  Added "game ranking" to the "system" menu

- 1.2.3

  Improved random event mechanism

  Added "random event A" & "random event B" mechanism

- 1.2.4

  Added "game initializing"

- 1.2.5

  Coordinate scroll bar of "market" panel & "storehouse" panel

- 1.2.6

  Added discard button

  Player can discard items when they are not allowed to sell

- 1.2.7

  Added reputation logic

  Too low: get hit

  High enough: low price buying, higher price selling

  Fix reputation bug

  Limit buy & sell in same area

- 1.2.8

  Added dialogs for each random events

  Setup the probability for each random events

- 1.2.9

  Added progress bar when initializing

- 1.3.0

  Added news sliding at buttom of the interface

- 1.3.1

  Added "nuclear" background to the initializing phase

- 1.3.2

Added the update function to the game

"help" - "about this game" - "check for update"

- 1.3.3

  Reduce JRE size

  200+MB

  ⇓

  110MB

  ⇓

  83MB