Business Startup Game

Prepared by
Krzysztof Para,
Mallika Patil,
Shehab Zalloum,
Manuel Martinez

November 2020

Table of Contents

List of Figures

List of Tables

I Project Description

1 Project Overview

The Business Startup Game is a map-based game where a player will be able to start their own food business. In this interactive game, a player will get the opportunity to develop and learn about the business world. One of the features includes being able to expand to your business in a certain part of the world. Another feature allows the user to manage their business from hiring employees to paying utilities to shopping for inventory. A player will be in charge of managing their money efficiently so they do not go bankrupt. The player's job is to make money while expanding their businesses. The player's success in the game is based on how effectively they can manage the food business in the real world.

2 Project Domain

The targeted audience for our project was general video game consumers.

3 Relationship to Other Documents

Most ideas for this project were taken from ProjectDescriptionGroup15PARTS123.pdf file. This document was produced by the Group 15 in the Spring 2019 semester.

4 Naming Conventions and Definitions

Information provided in the original document.

II Project Deliverables

1 First Release

The first release of the project took place on October 9, 2020. The main functionality that was implemented during the first release was the basic setup of the game. The key elements of the first release were the map, map markers, as well as the basic user interface. The map consisted of a Google Maps API which was integrated into the game, which placed the user in a certain city based on the location that the user had chosen. Once the location was chosen, the map would load the country. The user interface consisted of a header that always displayed the following options: restart, help, and exit. Every screen was designed to present and collect information from the user in an efficient manner. The diagram below was used to determine the course of the first release. The game was mainly set up during the first release and all user selections were collected and stored. The details of the first release are also highlighted in Group27BuildBusinessScenerio.pdf.

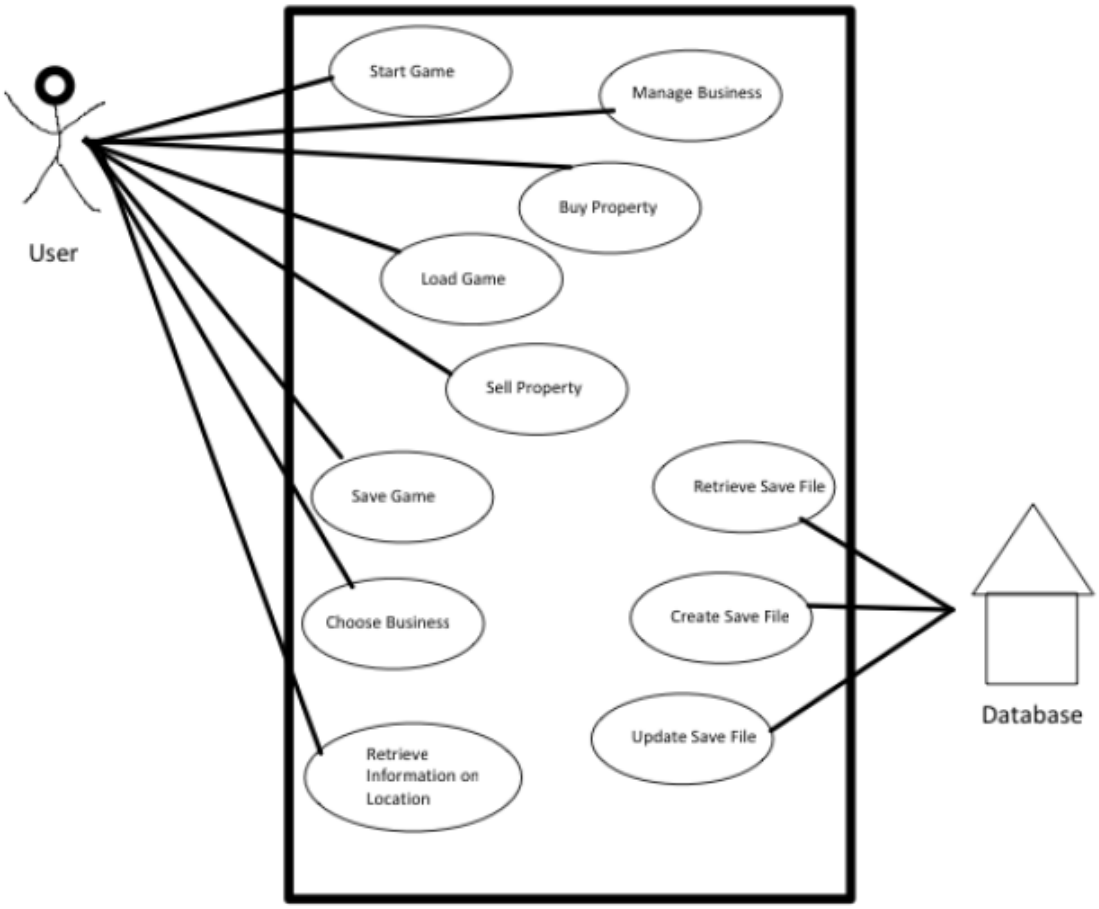


Figure 1 – Business Startup Game Use Case Diagram

2 Second Release

The second release expanded upon the first release. One of the first things that was improved in the second release was the implementation of the map. The map was divided into multiple different classes and called in a simpler way in the main code. The other functions that were added during this release include the "Manage Business" option. All of the "Manage Business" functionality was completed during this release, with the exception of a few bugs. The "Manage Business" option allowed users to "Hire" and "Fire" employees in "Employee Management." It also allowed the user to "Buy Inventory" for the owned stores. It also contained a "Pay Bills" option, which accumulated all the expenses from these various places and allowed the user to pay in one place. The final option added to "Manage Business" was the option to "Sell Business," which allowed the user to sell their business. All these options were added within the backend and they affected the Total Wealth of the business owner and it also affected the number of employees employed. The options also had a front end, which was mainly handled through button clicks by the user. Each of the

options are highlighted in detail in the Group27ManageBusinessSenerio2.pdf. There is a detailed view how all the options work in conjunction with one another in the UML diagram below.

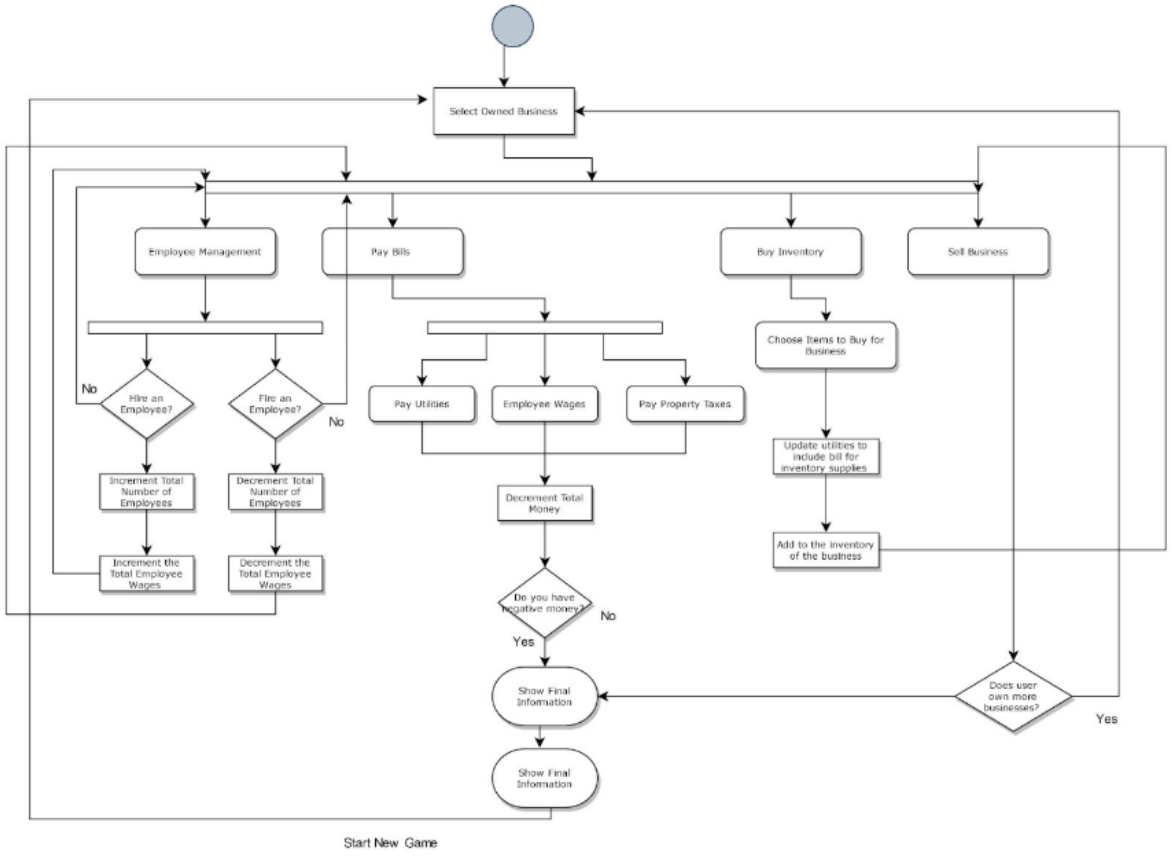


Figure 2 – Manage Business UML Diagram

3 Comparison with Original Project Design Document

The project mostly stuck to the original game; however, some liberties were taken in interpretation. One of the things removed from the original design document was the employee satisfaction. The focus of the game was on money management; therefore, the employee satisfaction option was not added. The user would basically hire, and fire employees as needed based on their profit and the total amount of wealth they had. The only other difference between the original and the implementation was the addition of a simulation. The addition of a simulation was so the user could feel as if days were passing by in the game. This allowed the user to make decisions in the game based on the passage of days. For example, every two weeks the user would pay employees their wages and lose some amount from their total wealth. The simulation was an important addition to the game for simulating a real-world business.

III Testing

1 Items to be Tested

Item ID# 01 – Google Map
Description: Item includes all decisions made by the user that concern the Google map presented to the user at the starting of the game.
Item ID# 02 – Employees Class
Description: Covers all options related to employees in the game.
Item ID# 03 – Inventory Class
Description: Covers all options related to buying inventory in the game.
Item ID# 04 – Pay Bills Class
Description: Covers all options related to paying bills in the game.
Item ID# 05 – Stores Bought
Description: Item covers any decision related to buying a business.
Item ID# 06 – Sell Option
Description: Item covers any decision related to selling a business.

2 Test Specifications

ID# 01 – Gathering Information Test

Description: Test whether the software is able to provide the user the correct information about their owned stores at the end.

Items covered by this test: NA

Requirements addressed by this test: Use Case ID# 06 – Save Game, Use Case ID# 08 – Retrieve Information on Location, Use Case ID# 09 – Retrieve Save File

Environmental needs: No special environment needed to run this test.

Intercase Dependencies: NA

Test Procedures: Testing procedure included selecting a store and then ending the game. The game had 3 store locations and the test was repeated with each store location.

Input Specification: Selection of Store 1, Store 2, or Store 3.

Output Specifications: Ending popup on the game was required to display the selected store, either Store 1, Store 2, or Store 3.

Pass/Fail Criteria: Test case worked as expected for each store selection.

ID# 02 – Fire Employees Test

Description: Test whether the correct number of employees are fired using the firing option for employees.

Items covered by this test: Item ID# 02 – Employees Class

Requirements addressed by this test: NA

Environmental needs: No special environment needed to run this test.

Intercase Dependencies: NA

Test Procedures: Running JUnit test cases and requiring certain outputs from the tests created.

Input/Output Specification:

Starting No. of Total Employees	Input Number of Employees to be Fired	Output Correct No. of Total Employees
5	2	3
10	8	2
2	3	Invalid, Result: 2
6	3	3
1000	999	1

Pass/Fail Criteria: Test is accepted if it passes all the tests including the invalid test case.

ID# 03 – Location Test

Description: Display the correct location on the map using Google API based on the user's selection.

Items covered by this test: Item ID# 01 – Google Maps

Requirements addressed by this test: ID# 09 – Launching App, ID# 06 – Location Statistics

Environmental needs: No special environment needed to run this test.

Intercase Dependencies: NA

Test Procedures: Select a location and then validate whether the correct location is displayed on the map. Repeat with three different locations.

Input/ Output Specification:

Input Selected Location	Output Correct Displayed Location
Palo Alto, USA	Palo Alto, USA
Tokyo, Japan	Tokyo, Japan
Mexico City, Mexico	Mexico City, Mexico

Pass/Fail Criteria: Test case worked for each location as selected.

ID# 04 – Store Location Test

Description: Test whether all stores are placed at a valid location at the selected location.

Items covered by this test: Item ID# 01 – Google Map

Requirements addressed by this test: ID# 09 – Launching App, ID# 06 – Location Statistics, ID# 16 – Download all data for specific region

Environmental needs: No special environment needed to run this test.

Intercase Dependencies: ID# 03 – Location Test

Test Procedures: Test procedure was to select the location of Palo Alto, USA and then test whether all three stores selected were placed using their map markers on valid locations on the Google map.

Input Specification: Selection of Store 1, Store 2, and Store 3.

Output Specifications: Valid location of Store 1, Store 2, and Store 3 on the correct location that was selected.

Pass/Fail Criteria: Test case worked as expected for each store selection.

ID# 05 – Hire Employees Test

Description: Test whether the correct number of employees are hired using the firing option for employees.

Items covered by this test: Item ID# 02 – Employees Class

Requirements addressed by this test: NA

Environmental needs: No special environment needed to run this test.

Intercase Dependencies: NA

Test Procedures: Running JUnit test cases and requiring certain outputs from the tests created.

Input/ Output Specification:

Starting No. of Total Employees	Input Number of Employees to be Hired	Output Correct No. of Total Employees
5	2	8
10	8	18
2	3	5
6	3	9
1000	999	1999

Pass/Fail Criteria: Test is accepted if it passes all the tests including the large test case.

ID# 06 – Pay Employee Wages Calculation Test

Description: Test whether the game will pay the correct number of employees, based on the total number of employees owned.

Items covered by this test: Item ID# 04 – Pay Bills Class

Requirements addressed by this test: NA

Environmental needs: No special environment needed to run this test.

Intercase Dependencies: NA

Test Procedures: Running JUnit test cases and requiring certain outputs from the tests created.

Input/ Output Specification:

No. of Total Employees	Input Employee Wages ($600)	Output Correct Employee Wages Paid
5	600	3000

10	600	6000
2	600	1200
6	600	3600
1000	600	600000

Pass/Fail Criteria: Test is accepted if it passes all the tests including the large test case.

ID# 07 – Pay Employee Wages Total Test

Description: Test whether the game will correctly deduct the amount paid to the employees from the total revenue of the business.

Items covered by this test: Item ID# 04 – Pay Bills Class

Requirements addressed by this test: NA

Environmental needs: No special environment needed to run this test.

Intercase Dependencies: ID#06 – Pay Employee Wages Calculation Test

Test Procedures: Running JUnit test cases and requiring certain outputs from the tests created.

Input/ Output Specification:

No. Amount of Revenue	Input Amount of Employee Wages	Output Correct New Amount of Revenue
10000	3000	7000
5000	6000	Invalid, Result: -1000
6000	1200	4800
700	3600	Invalid, Result: -2900
100000	60000	40000

Pass/Fail Criteria: Test is accepted if it passes all the tests including invalid test cases.

ID# 08 – Inventory Calculation Test

Description: Test whether the inventory calculation is correctly performed by when the option is selected.

Items covered by this test: Item ID# 03 – Inventory Class

Requirements addressed by this test: NA

Environmental needs: No special environment needed to run this test.

Intercase Dependencies: NA

Test Procedures: Running JUnit test cases and requiring certain outputs from the tests created.

Input/ Output Specification:

Input Selection of Inventory Items	Output Correct Total
Mushrooms ($15), Bacon ($30)	45
Mushrooms ($15), Mushrooms ($15), Mushrooms ($15),	45
Peppers ($8), Pickles ($8)	16
Utensils ($10)	10
Empty	0

Pass/Fail Criteria: Test is accepted if it passes all the tests including empty test case.

ID# 09 – Inventory Total Test

Description: Test whether the game will correctly deduct the amount paid for the inventory from the total revenue of the business.

Items covered by this test: Item ID# 04 – Pay Bills Class

Requirements addressed by this test: NA

Environmental needs: No special environment needed to run this test.

Intercase Dependencies: ID#08 – Inventory Calculation Test

Test Procedures: Running JUnit test cases and requiring certain outputs from the tests created.

Input/ Output Specification:

No. Amount of Revenue	Input Total of Inventory	Output Correct New Amount of Revenue
10000	4000	6000
5000	7000	Invalid, Result: -2000
6000	300	5700
700	200	500
100000	5000	95000

Pass/Fail Criteria: Test is accepted if it passes all the tests including invalid test cases.

ID# 10 – Hire Employees When Buying Business Test

Description: Test whether the game will correctly hire employees when a new store is bought by the business.

Items covered by this test: Item ID# 05 – Stores Bought, Item ID# 02 – Employees Class

Requirements addressed by this test: Use Case ID# 03 – Buy Property

Environmental needs: No special environment needed to run this test.

Intercase Dependencies: ID#08 – Inventory Calculation Test

Test Procedures: Running JUnit test cases and requiring certain outputs from the tests created.

Input/ Output Specification:

Initial Number of Employees	Input No. of New Businesses Added	Output Correct Amount of Employees
2	2	8
5	1	7

10	1	12
0	2	4
30	2	34

Pass/Fail Criteria: Test is accepted if it passes all the test cases.

3 Test Results

ID# 01 - Gathering Information Test Results

Date(s) of Execution: Test was only executed once.

Staff conducting tests: Developer Shehab Zalloum

Expected Results: Expected to see results as stated in the output.

Actual Results: Actual results reflected what was listed in the output.

Test Status: Pass, expected results matched actual results.

ID# 02 – Fire Employees Test Results

Date(s) of Execution: Test was only executed once.

Staff conducting tests: Developer Manuel Martinez

Expected Results: Expected to see results as stated in the output.

Actual Results: Actual results reflected what was listed in the output.

Test Status: Pass, expected results matched actual results.

ID# 03 - Location Test Results

Date(s) of Execution: Test was only executed once.

Staff conducting tests: Developers Shehab Zalloum and Manuel Martinez

Expected Results: Expected to see results as stated in the output.

Actual Results: Actual results reflected what was listed in the output.

Test Status: Pass, expected results matched actual results.

ID# 04 - Store Location Test Results

Date(s) of Execution: Test was only executed once.

Staff conducting tests: Developer Mallika Patil

Expected Results: Expected to see results as stated in the output.

Actual Results: Actual results reflected what was listed in the output.

Test Status: Pass, expected results matched actual results.

ID# 05 - Hire Employees Test Results

Date(s) of Execution: Test was only executed once.

Staff conducting tests: Developer Mallika Patil

Expected Results: Expected to see results as stated in the output.

Actual Results: Actual results reflected what was listed in the output.

Test Status: Pass, expected results matched actual results.

ID# 06 – Pay Employee Wages Calculation Test Results

Date(s) of Execution: Test was only executed once.

Staff conducting tests: Developer Manuel Martinez

Expected Results: Expected to see results as stated in the output.

Actual Results: Actual results reflected what was listed in the output.

Test Status: Pass, expected results matched actual results.

ID# 07 – Pay Employee Wages Total Test Results

Date(s) of Execution: Test was only executed once.

Staff conducting tests: Developer Krzysztof Para

Expected Results: Expected to see results as stated in the output.

Actual Results: Actual results reflected what was listed in the output.

Test Status: Pass, expected results matched actual results.

ID# 08 - Inventory Calculation Test Results

Date(s) of Execution: Test was only executed once.

Staff conducting tests: Developers Shehab Zalloum and Manuel Martinez

Expected Results: Expected to see results as stated in the output.

Actual Results: Actual results reflected what was listed in the output.

Test Status: Pass, expected results matched actual results.

ID# 09 - Inventory Total Test Results

Date(s) of Execution: Test was only executed once.

Staff conducting tests: Developer Krzysztof Para

Expected Results: Expected to see results as stated in the output.

Actual Results: Actual results reflected what was listed in the output.

Test Status: Pass, expected results matched actual results.

ID# 10 - Hire Employees When Buying Business Test Results

Date(s) of Execution: Test was only executed once.

Staff conducting tests: Developer Krzysztof Para

Expected Results: Expected to see results as stated in the output.

Actual Results: Actual results reflected what was listed in the output.

Test Status: Pass, expected results matched actual results.

4 Regression Testing

During any future releases, it will be necessary that all the tests created above pass before the new release is accepted. All tests are testing core functions of the project and need to pass, before a new set of features can be accepted into the project.

IV Inspection

1 Items to be Inspected

BusinessGameClient.java

dateSimulator.java

2 Inspection Procedures

Inspections were done electronically. They are discussed through Discord meetings exclusively so everyone is kept up to date. Each of the team member would share a snippet of the code and the team members would comment and make suggestions based on what they wanted to change.

3 Inspection Results

Inspection 1

Who Inspected: Manny

Date & Time: 10/15/20 6 PM

Initially coded by : Kris

Code Inspected:

```
locationList.add("Berlin/Germany");

locationList.add("Madrid/Spain");

 ComboBox<String>locationDropDown = new ComboBox<String>();

locationDropDown.getItems().add(locationList.get(0));
```

Result : Changed the list into a dynamic combo box so it can be changed to whatever country needed.

Inspection 2

Who Inspected: Shehab

Date & Time: 10/15/20 6 PM

Initially coded by : Manny

Code Inspected:

```
WebView webView = new WebView();

WebEngine webEngine = webView.getEngine();

webEngine.load(getClass().getResource("/googlemap.html").toString());
```

Result : Used a webview for the google maps API

Inspection 3

Who Inspected: Malika

Date & Time: 10/15/20 6 PM

Initially coded by : Shehab

Code Inspected:

```
addBuisiness.setDisable(false);

        storeList = getListofStores();

        storeDropdown.getItems().addAll(storeList.get(0),storeList.get(1),
storeList.get(2));
```

Result : Decided on how the buttons for adding the business would work

Inspection 4

Who Inspected: Kris

Date & Time: 10/15/20 6 PM

Initially coded by : Malika

Code Inspected:

```
        if (store1 == true) {

        lastStorePurchased.setText(storeChosen);

       infoScreen.setText("you have already bought store 1, buy something else");

       return;
```

Result : Set up a boundary case so if something is already bought, it can not be bought again

Inspection 5

Who Inspected: Manny

Date & Time: 10/27/20 6 PM

Initially coded by : Shehab

Code Inspected:

```
        sim.nextDay();

        simDate.setText("Date: " + sim.getDate());
```

```
System.out.println("The Counter is: " + storeCount);

int dailyRevenue = (1350 * storeCount);

revenue = revenue + (1350 * storeCount);
```

Result : Figured out what to do for the simulate by days, increasing revenues based on how much stores owned

Inspection 6

Who Inspected: Shehab

Date & Time: 10/27/20 6 PM

Initially coded by : Manny

Code Inspected:

```
int addV = Integer.parseInt(employeeCount.getText()) + buyV;

employeeCount.setText(Integer.toString(addV));

buyT.clear();

infoScreen.setText("you just Hired " + buyV + " employees");
```

Result : We had to converse about how we wanted to update the status for letting the player know what happend

Inspection 7

Who Inspected: Kris

Date & Time: 10/27/20 6 PM

Initially coded by : Malika

Code Inspected:

```
int budgetAfterUtilitiesPaid = Integer.parseInt(budget.getText()) - utilities1;

budget.setText(Integer.toString(budgetAfterUtilitiesPaid));

paidUtilities.setText("Paid: "+ utilities1 + " for Store1 Utilities and " + inventory1 + " for inventory.")
```

Result : Decided to create a temp value before updating

Inspection 8

Who Inspected: Malika

Date & Time: 10/27/20 6 PM

Initially coded by : Kris

Code Inspected:

```
Button Cheddar = new Button("Cheddar");

Cheddar.setGraphic(new ImageView(cheddar));

Cheddar.setStyle("-fx-background-radius: 100em;" + "-fx-max-width: 300px; ");

Cheddar.setOnAction( new EventHandler<ActionEvent>()
```

Result : Decided to make a bunch of items like this to implement the inventory for the simulator

Inspection 9

Who Inspected: Shehab

Date & Time: 11/06/20 6 PM

Initially coded by : Malika

Code Inspected:

```
int tempUtilties = utilities2 + inventoryTotal;

value2 = store2Cost + wages2 + propertyTaxes2 + tempUtilties;

paidTaxesText.setText("The Value of the property2 is "+ value2 + " because you invested " + wages2 +" in wages " + propertyTaxes2 + " in taxes " + inventory2 +" in inventory");
```

Result : Decided on how to update the taxes and what constitutes to be added

Inspection 10

Who Inspected: Malika

Date & Time: 11/06/20 6 PM

Initially coded by : Manny

Code Inspected:

```
else if(Integer.parseInt(employeeCount.getText())==2 || Integer.parseInt(employeeCount.getText()) == 1 || Integer.parseInt(employeeCount.getText()) == 0 ) {

employes = 0;

employeeCount.setText(Integer.toString(employes));
```

Result : Implemented a minimum employee count of 0 and ensured it doesn't go below that

Inspection 11

Who Inspected: Kris

Date & Time: 11/06/20 6 PM

Initially coded by : Shehab

Code Inspected:

```
switch (country) {

case "Seoul/South Korea":

stores.add("Seoul Store1");

stores.add("SeoulStore2");
```

Result : Implemented a dynamic way of adding countries for stores

Inspection 12

Who Inspected: Manny

Date & Time: 11/06/20 6 PM

Initially coded by : Kris

Code Inspected:

```
popUpWindow.setTitle("Full Game Instructions");

Label fullGameInstructions = new Label( "Conclusion\n" + "Total Revenue: $" + revenue + "\n" + "Last Store Bought: " + storeChosen+ "\n" + "Country Chosen: " + country + '\n' + "Thank you for playing!");
```

Result : Created a pop up window to showcase how to use the simulator

V Recommendations and Conclusions

SV: Indicate whether or not the items covered have passed their testing and inspection process or not, and what actions should be taken next. (E.g. further testing & inspection, or implementation.)

VI Project Issues

1 Open Issues

One issue that remains open pertains to the amount of freedom that the player should be given. Since the very beginning of the project it was discussed that giving the player too much freedom would hinder that overall purpose of the game which was to give users the experience of running their own businesses. One of the areas that we thought we could not give the user too much freedom was on location selection. We believed that if the user were able to choose anywhere in the world to place a business then there would be players that would pick less than optimal locations. This idea led us to remove a lot of the freedoms the user had when it came to location and we installed predetermined locations that the user could pick from. However, there are various game mechanics that have similar problems when it comes to how much freedom should the user be given. Therefore this is an issue that is still left open where it is important that a consensus is reached on how much freedom should be given to the player without sacrificing sight of what the game is trying to accomplish through its gameplay.

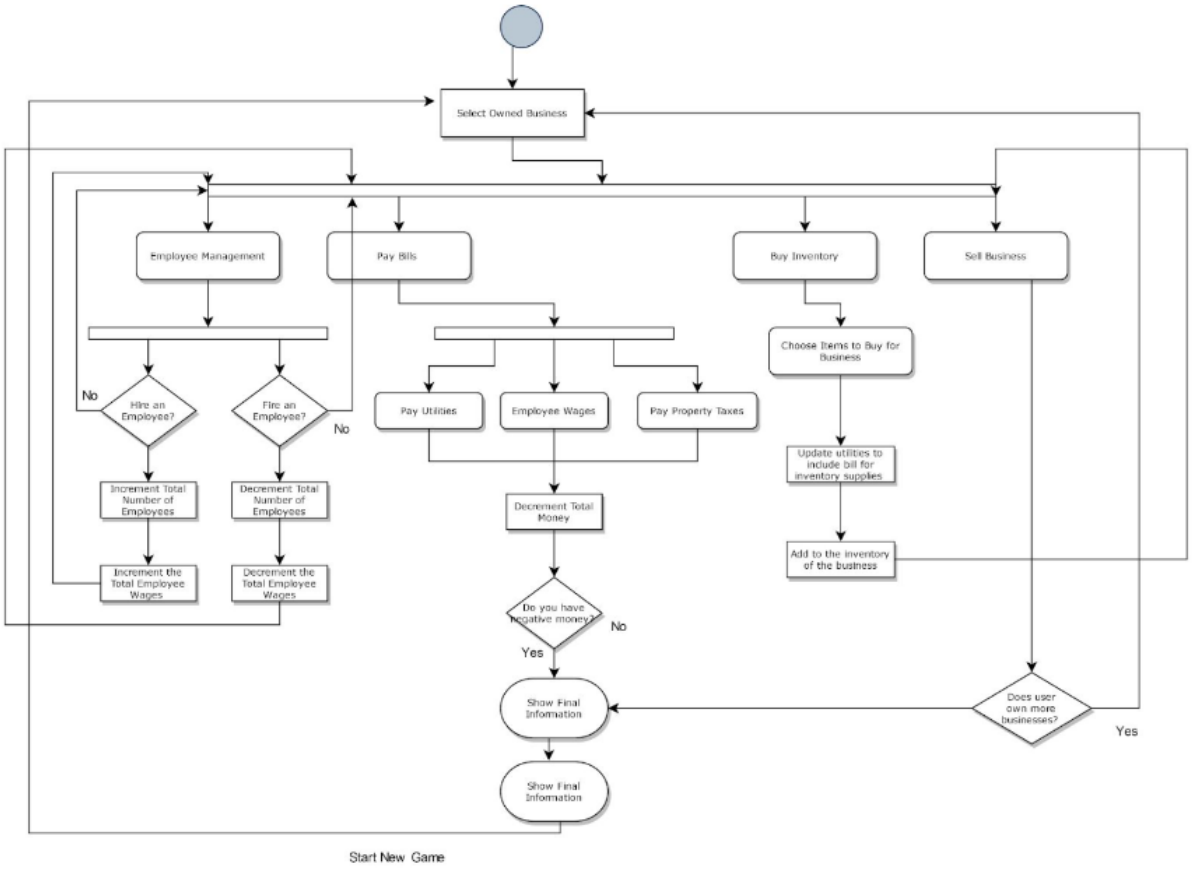
2 Waiting Room

One feature that remains to be implemented is the simulation mechanic. While we were able to implement a basic simulation mechanic it did not simulate more than just the passing of time. While the passage of time is a good indicator of progression, we felt that there should be more than just time simulation. An example would be implementing NPCs that would give the user the ability to interact with other business owners and make the world feel much more alive. These NPCs would interact with the user and offer deals, alliances, or even services. This would make the user think more about how their business fits into the ecosystems of their location and how they can capitalize on the niches that have not been filled by the business of the NPCs.

Another feature that we would like to eventually implement would be the ability for the game to support a multiplayer environment. While NPCs can make the users' world feel much more alive, being able to play with friends would make the game much more enjoyable. If the user is able to invite their friends to play on each other’s worlds, then the user will be able to strategize with friends and have better results when it comes to business decisions.

3 Ideas for Solutions

Figure 3 – Manage Business UML Diagram



4 Project Retrospective

Throughout the development of our project we decided that certain portions of the code would be handled by different members of the group. This meant that certain group members would only deal with the front end of the project while others would focus on the back end. While this worked in the beginning it sometimes came with issues. Some of these issues were not being able to understand what the purpose of certain blocks of code was. In addition, sometimes it was hard to add code without worrying about moving or removing by accident important blocks of the code that would cause problems at execution. Because of these problems we decided that it was important that everyone should understand what the purpose of every block of code is. This would help everyone have more confidence when pushing code through in the repository and would also allow us to be on the same page. This new work method allowed us to be more productive and was a big upgrade from our previous work model.

VII References / Bibliography

[1] Robertson and Robertson, Mastering the Requirements Process.

[2] A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013.

[3] Eric Leon, Alexis Urquiza, Jakub Glebocki and Ahel Guerrero, Business Startup Game Project Report, 2019

[4] M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.

VIII Index