İHSAN DOĞRAMACI BİLKENT UNIVERSITY

SPRING 2018

CS319 – OBJECT ORIENTED SOFTWARE ENGINEERING

ITERATION 2 – ANALYSIS REPORT

FARMIO – GROUP 2F

GROUP MEMBER:

1. Nursena Kal
2. Eray Şahin
3. Fuad Ahmad
4. Demir Topaktaş

Content

1. INTRODUCTION

Farmio is a 2D farming simulator video game we planned to develop. The main aim of this game is to manage a farmland by planting different types of seeds, taking care of plants, gathering their grown crops to either sell or eat. If the player chooses to sell the grown crops, this will facilitate in generating income, helping the player to invest in different types of seeds to attain more money. Alternatively, the player may also choose to eat the grown crops so that the health of the farmer can be kept at its maximum. The player has to balance these two actions properly since either running out of money or worsening health means the end of the game. In other words, the game will be over when the player loses all of his/her money - and there is no investment in plants to provide income- or when the farmer represented in the game loses his/her health.

Additionally, we will implement Farmio in Java programming language by Object-Oriented Programming principles and it will be a desktop application. This report contains the game overview, basic game objects, and the basic structure of the game. Besides, we also added the functional, non-functional requirements as well as the use-case, class, activity and state diagrams.

2. OVERVIEW

2.1 ENTITIES FROM PREVIOUS REPORT

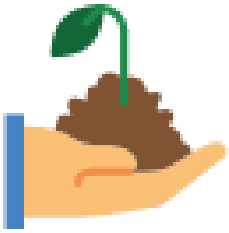
2.1.1. Gameplay

The game is played using the mouse. Some actions are to be handled through right clicking to choose one of the several possible actions.

2.1.2. Map

The map initially consists of a farm house whose location is fixed (defined by the game). Surrounding the farm house are the “slots” of soil on which the player may plant and grow seeds depending on the type of the soil (see the next section 2.1.3. Soil for details).

2.1.3. Soil



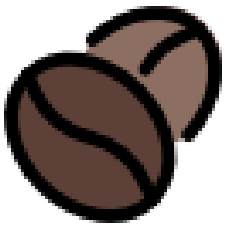
The soil slots can be considered in two different categories as “grass” and “pit”. Grasses are blocks of soil which are not suitable for planting. On the other hand, pits are the ones on which the player can plant and grow seeds.

2.1.4. Store

The player is able to buy seeds of different kinds from the store. The store, which is to be made available directly on the game screen, displays the available seeds and their respective prices as small icons.

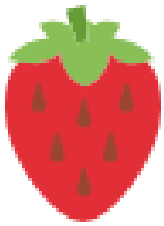
Note that, at the beginning of a new game, the player is provided with some initial money to buy some seeds to begin planting. Except this initial money, the actual income is supposed to be generated through growing crops and selling them.

2.1.5. Seeds



The game represents strawberry, corn sunflower, corn,tomato, potato seeds available for purchase at the store. These seeds differ by their growing times and prices. Having planted seeds, as long as they are watered once using the “watering can” tool, they grow continually and form grown crops. On the other hand, when they are not watered for a specific time limit, they will spoil, making the farm slot in which they were sown available for planting again.

2.1.6. Grown Crops



As already mentioned above, seeds produce grown crops (also called “food” throughout the report) when treated properly. These crops need to be harvested as soon as possible, otherwise they will spoil (just like the seeds that are not watered). Again, rotten food makes the farm slot available for planting different seeds (as if it has never been cultivated before).

2.1.7. Harvesting

When the seeds are grown fully and have produced grown crops, which is indicated by slots’ attaining new icons, the player is expected to harvest them before they spoil. To harvest the grown crops, the player simply clicks on these new icons.

2.1.8. Inventory

The purchased seeds and collected crops appear directly on the inventory, which also is continuously accessible on the game screen for ease of use. The player can switch back and forth between both the purchased seeds to plant and also between gathered crops to eat or sell.

2.1.9. Selling

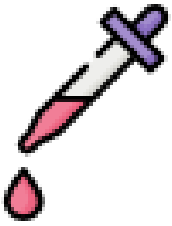
To sell the gathered crops and generate income, the player should go to store. Note that each crop will yield a different amount of money, depending on the kind of the seed from which it is grown.

2.1.10. Farmer's Health

As one of the main objectives of the game, in addition to maintaining money, the player needs to keep the health of the farmer highest so as to keep the game continuing. This involves the player's letting the farmer eat some of the gathered crops through using the inventory. Note that there will not be a farmer represented by a person moving around the farmland. Instead, it will be conceptually referred by its health and money (that is, to be displayed within a box on the screen).

2.2 ADDED ENTITIES

2.2.1 GMC



When farmer wants to grow each plant fast, he/she can buy GMC from store. It will reduce the time required for growing of related plant but also it will lower the price that will be gained from any crop.

2.2.2 Rain

This is one additional power-up but this will not be able to be bought from store. When Rain occurs, the status of any seed's water status will be fulled.

2.2.3. Fertalizer

This will be available on store. Farmer will be able to buy this and use it on seeds. Using this will lower the growing time but also it will not lower the price of grown seed.

3. FUNCTIONAL REQUIREMENTS

3.1. Play Game

3.1.1. Start a New Game

The player is allowed to launch a new game anytime without needing to override any previous progress. In other words, the player may maintain multiple “save files” with each having a possibly different progress.

3.1.2. Load Game

Since the player is granted access to multiple save files as mentioned above, this will allow the player to choose and load one of the previously saved games.

3.2. Credits

This screen displays some information about us as the developers of the game.

3.3. Help

The player can access this screen to learn more about the game controls as well as the objective of the game.

3.4. Exit Game

The player may exit the game through this option.

3.5. ADDITIONAL CHANGES IN FUNCTIONAL REQUIREMENTS

3.5.1 Buy Seeds and Trees

The player should be able to buy seeds and trees from store. Main function of this game is the ability of a player to play real world farming game.

3.5.2 Sell Grown Crops and Harvested Items

The player should be able to sell the items that he/she has planted and took care of.

3.5.3 Remove Spoiled Trees and Crops

When player does not take care of the items that he/she has planted then they will spoil. After they are spoiled, player can remove these spoiled items.

3.5.4 Water Planted Seeds and Planted Trees

Player can water the items that he/she has planted to take care of those items.

3.5.5 Use GMC or Fertalizer

Player will be able to buy GMC and Fertalizer from Store and use them on the seeds and trees to help them grow faster with some consequences.

3.5.6 Plant Trees and Seeds

Player can plant trees and seeds on his /her farm.

3.5.7 Harvest Grown Items

Player can harvest all of the grown items and sell them to gain coins.

4. NON-FUNCTIONAL REQUIREMENTS

4.1. USABILITY

The game should represent an intuitively understandable user interface. More specifically, the positions of the store icon, inventory and the farmer icon should be easy to identify and convenient to interact with.

4.2. PERFORMANCE

4.2.1. RESPONSE TIME

The game should respond to the input provided by the user as quickly as possible. Primarily, the images need to be updated instantly to indicate the changes the player has made, also providing a smoother gameplay.

4.3. SUPPORTABILITY

Since the player can start a new game anytime in addition to the previously saved ones (also mentioned in 1.1.1 Start a New Game), the generation of this new game should not cause any changes to the previously saved games. That is, the game system should handle all save files separately, independent of each other.

4.4 ADDITIONAL CHANGES IN NON-FUNCTIONAL REQUIREMENTS

Roboutness

We aim to minimize errors and handle problematic cases using better choices of function definitions (having learned about architectural styles and design patterns).

5. SYSTEM MODELS

In this section, there will be detailed information about planned use cases and related sequence diagrams. Their explanations will be bellow and detailed dynamic models are available.

5.1. USE CASE DIAGRAM

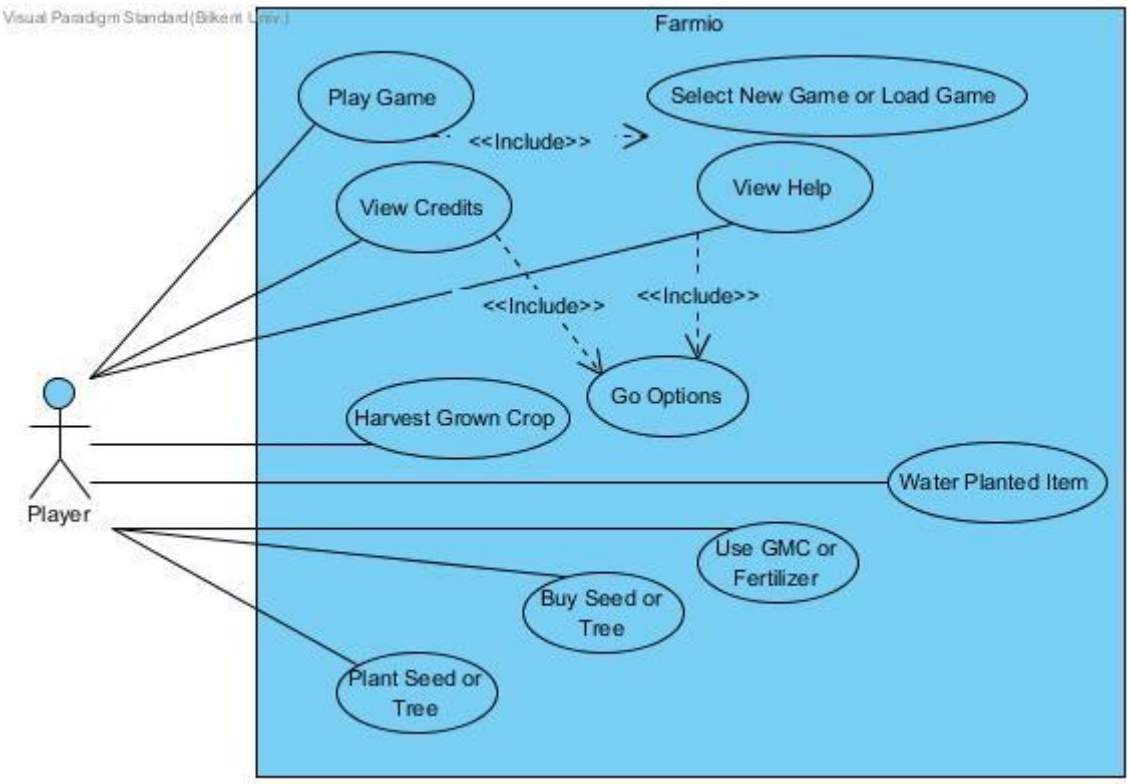


Figure 5.1 Use Case Model Diagram

Use Case #1

Use case name: Play Game

Participating actors: Player

Entry condition: Player has to open main Menu and has to select “Play” button

Exit condition:

1. Player has to lose game by finishing his money.
   OR
2. Player has to exit the game by clicking the exit button.

Main Flow of Events:

1. Player clicks the "Play Game" Button
2. System comes up with selection page that asks if player wants to load game or start a new game
3. Player plays the game and returns to main menu at the time player wants.
4. Player returns to the main menu of the game.

Alternative Flow of Event:

1. Player loses all of his/her money and system comes up with pop-up message that “Game Over” and game returns to Main Menu
2. Player can exit the game any time.

Use Case #2

Use case name: Select New Or Load Game

Participating actors: Player

Entry condition: Player chose “Play” button from main menu.

Exit condition: Player returns to main menu.

Main Flow of Events:

1. Player clicks "Play Game" button from main menu.
2. Player chooses to play new game or load game.
3. Player returns to the main menu of the game after playing game.

Alternative Flow of Event:

- Player returns to main menu without playing the game.

Use Case #3

Use case name: View Help

Participating actors: Player

Entry condition: Player firstly opens the game and goes to Options page.

Exit condition: Player returns to main menu.

Main Flow of Events:

1. “Help” button is chosen by Player.
2. Player gets informed about the instructions of the game.
3. Player returns to the Option menu.

Use Case #4

Use case name: View Credits

Participating actors: Player

Entry condition: Player firstly opens the game and goes to Options page.

Exit condition: Player returns to main menu.

Main Flow of Events:

1. "Credits" button is chosen by Player.
2. Player gets informed about the creators of the game.
3. Player returns to the Option menu.

Use Case #5

Use case name: Go Options

Participating actors: Player

Entry condition: Player firstly opens the game and goes to Options page.

Exit condition: Player returns to main menu.

Main Flow of Events:

1. "Options" button is chosen by Player.
2. Player chooses between options, help and credits from this menu.
3. Player returns to the Option menu.

Use Case #6

Use case name: Use GMC / Use Fertilizer

Participating actors: Player

Entry condition: Player should have at least one seed already planted and have enough money to afford GMC or Fertilizer

Exit condition:

1. Player has to lose game by finishing his money, OR
2. Player has to cause the farmer starve (game over), OR
3. Player has to exit the game by clicking the exit button OR
4. The plant (with GMC/Fertilizer) must get spoiled, OR
5. The plant (with GMC /Fertilizer) grows and gets harvested.

Main Flow of Events:

1. Player clicks the "Play Game" button.
2. Player chooses to load an existing game.
3. Player launches store.
4. Player purchases “GMC” or “Fertilizer”.
5. Player selects GMC/ Fertilizer from the inventory and clicks on the planted seed.

Use Case #7

Use case name: Water Planted Item

Participating actors: Player

Entry condition: Player should have an available planted seed or tree to be watered.

Exit condition:

1. Player has to lose game by finishing his money, OR
2. Player has to cause the farmer starve (game over), OR
3. Player has to exit the game by clicking the exit button OR
4. The plant that is watered must get spoiled, OR
5. The plant or tree that is watered grows and gets harvested.

Main Flow of Events:

1. Player clicks the "Play Game" button.
2. Player chooses to load an existing game.
3. Player clicks on planted item and selects “Water Can”.
4. Player waters the planted item.

Use Case #8

Use case name: Buy Seed or Tree

Participating actors: Player

Entry condition: Player should have enough coin and available slot to plant any tree or a seed .

Exit condition:

1. Player has to lose game by finishing his money, OR
2. Player has to cause the farmer starve (game over), OR
3. Player has to exit the game by clicking the exit button OR
4. The planted item must get spoiled, OR
5. The planted item grows and gets harvested.

Main Flow of Events:

1. Player clicks the "Play Game" button.
2. Player chooses to load an existing game.
3. Player launches store.
4. Player purchases any type of “Seed” or “Tree”.

Use Case #9

Use case name: Plant Seed or Tree

Participating actors: Player

Entry condition: Player should have bought seed or tree.

Exit condition:

1. Player has to lose game by finishing his money, OR
2. Player has to cause the farmer starve (game over), OR
3. Player has to exit the game by clicking the exit button OR

Main Flow of Events:

1. Player clicks the "Play Game" button.
2. Player chooses to load an existing game.
3. Player clicks on the available slot and plants the bought seed or tree.

Use Case #10

Use case name: Harvest Grown Crop

Participating actors: Player

Entry condition: Player should have take care of the planted tree or seed and enough time should be passed.

Exit condition:

6. Player has to lose game by finishing his money, OR
7. Player has to cause the farmer starve (game over), OR
8. Player has to exit the game by clicking the exit button OR

Main Flow of Events:

5. Player clicks the "Play Game" button.
6. Player chooses to load an existing game.
7. Player clicks on a grown item(can be Grown Plant or Grown Tree).
8. Player uses tools to harvest grown item.

5.2. DYNAMIC MODELS

5.2.1. SEQUENCE DIAGRAMS

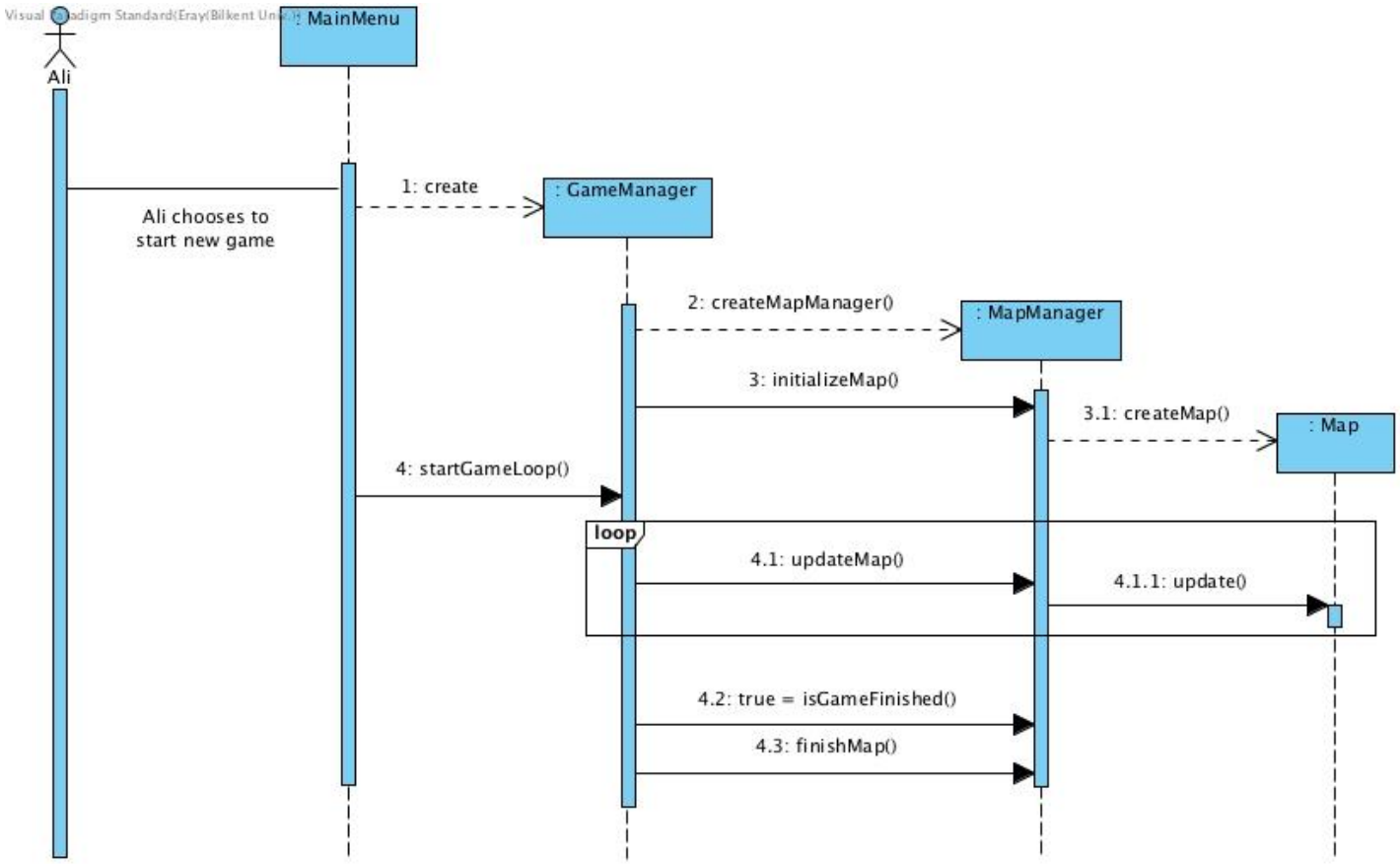


Figure 5.2.1.1 Loading Game Sequence Diagram

Loading the game

Although not being a complete scenario on its own, the above diagram demonstrates the common process as part of most of the following scenarios (i.e. to starting a new game). Note that the case for loading an existing game is similar, with the exception that the map is initialized using the data saved in the respective text file (previously saved game).

Ali creates a new game by clicking on the respective button in the menu. The map is initialized and displayed on the screen.

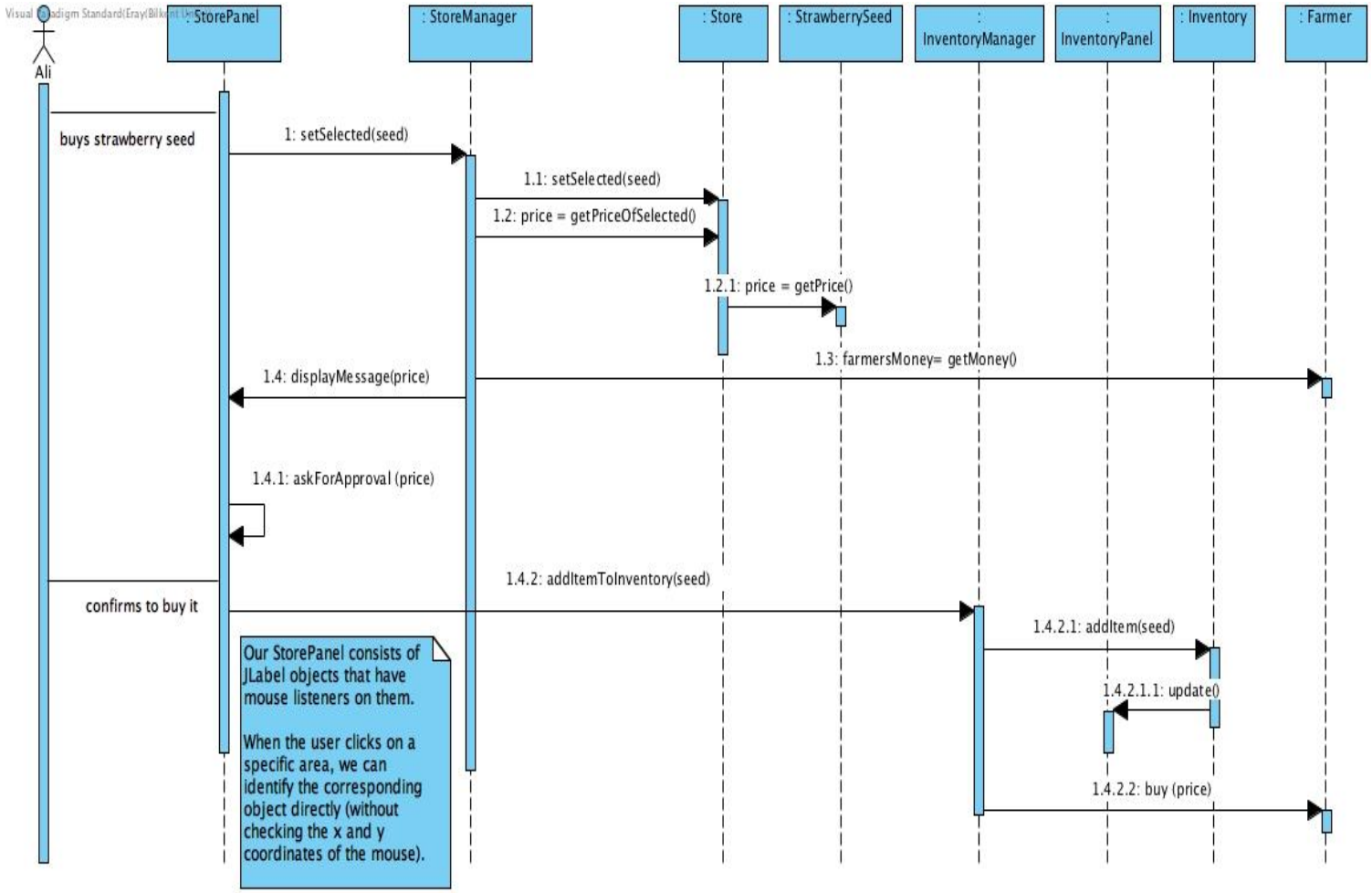


Figure 5.2.1.2 Buying A Seed Sequence Diagram

Buying a seed

The above diagram illustrates how Ali buys a strawberry seed from the store. Firstly, Ali chooses the strawberry seed on the store screen and clicks on “buy”. Then, the chosen item is identified by the system. The price of the strawberry seed is compared with the current money of the farmer. Since Ali has already collected sufficient money to afford that seed, a message appears to ask Ali whether he really wants to buy that seed. Then, Ali

confirms to buy it. Finally, the item is added to the inventory and the money of the farmer is decremented by the price of the strawberry seed.

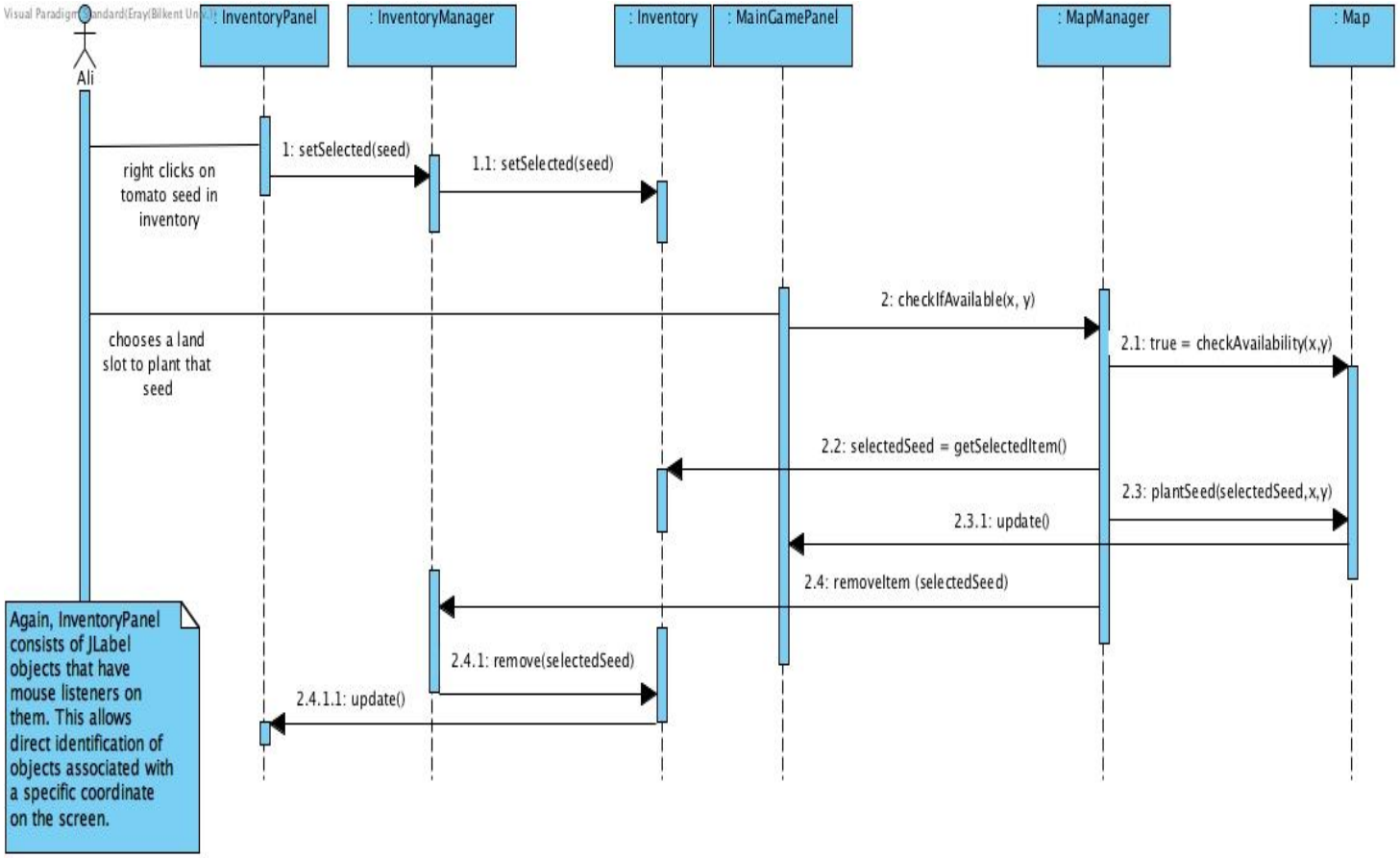


Figure 5.2.1.3 Planting A Seed Sequence Diagram

Planting a seed

In the scenario conveyed by the above diagram, Ali wants to plant the tomato seed he has in the inventory. Firstly, he selects the tomato seed from the inventory. Then, he clicks on a farm slot where this seed is to be planted. The system checks if the chosen slot is available,

having ensured that nothing was planted there, and adds the seed to that specific slot. The inventory reflects the decrease in the number of tomato seeds.

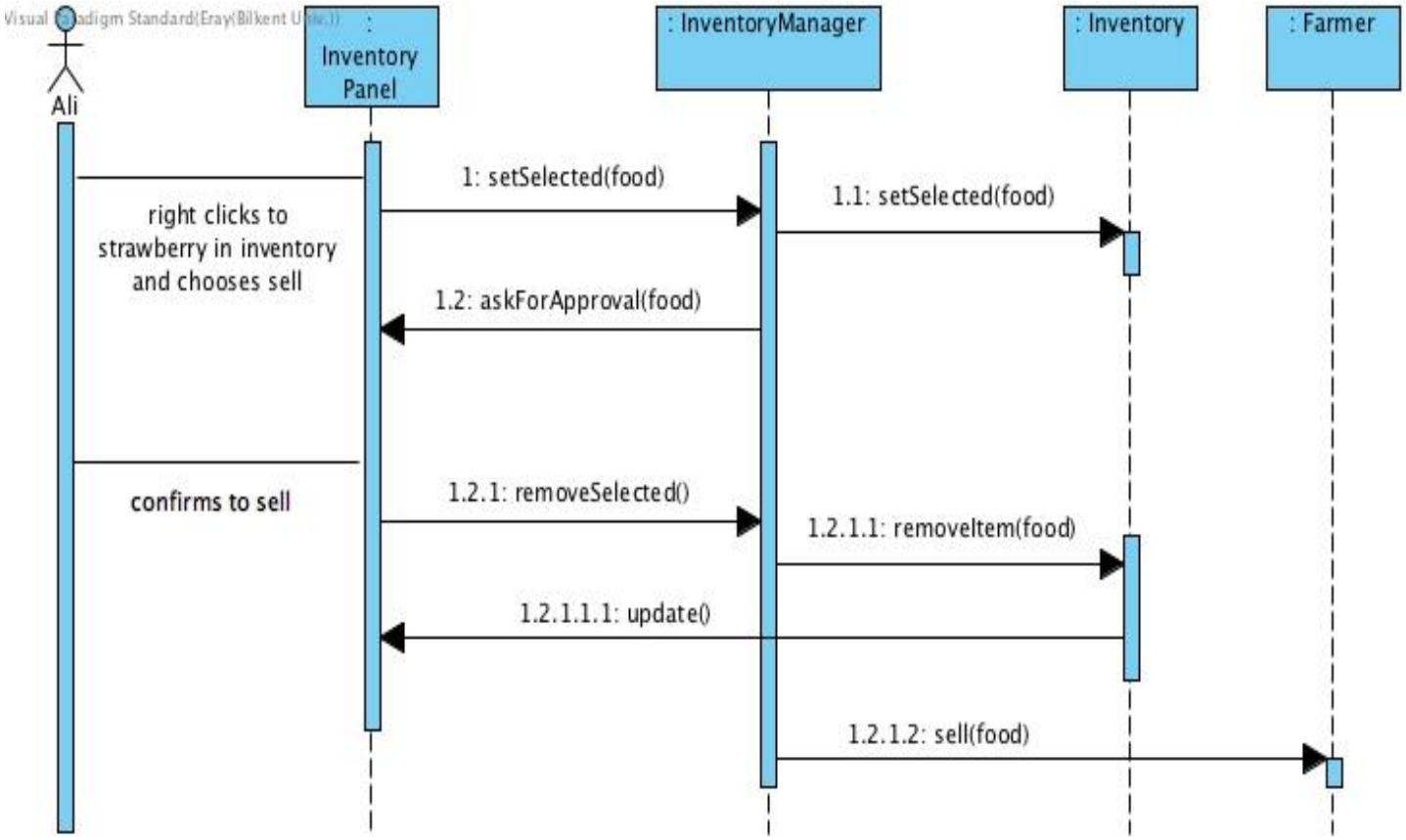


Figure 5.2.1.4 Selling A Food Sequence Diagram

Selling a Food

Here, Ali wants to sell the strawberry (not the seeds but the grown strawberries) which he collected before. To do that, he selects the strawberry from the inventory, right-clicks on it and chooses “Sell”. Then, a message asking Ali whether he is sure about his decision is displayed. Ali confirms his attempt and the strawberry is sold. The inventory reflects the decrease in strawberry and farmer receives the money.

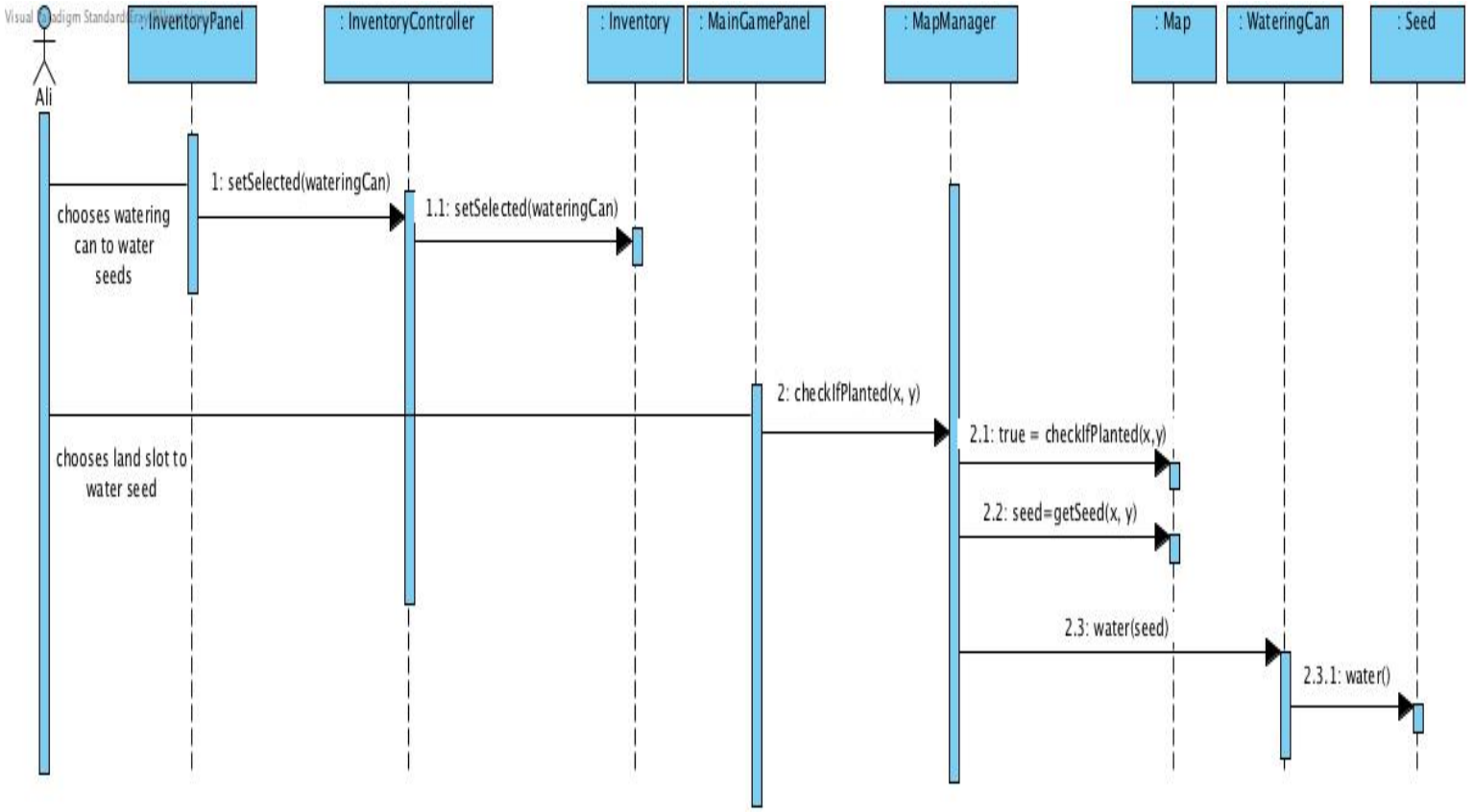


Figure 5.2.1.5 Watering Seeds Sequence Diagram

Watering Seeds

Ali, in this scenario, equips the watering can to water a specific seed. He chooses the slot where this seed is planted. The system, after checking that there is a seed in the chosen slot, waters the seed that Ali has chosen.

5.2.2. STATE DIAGRAM

5.2.2.1 Development Of A Tree Object

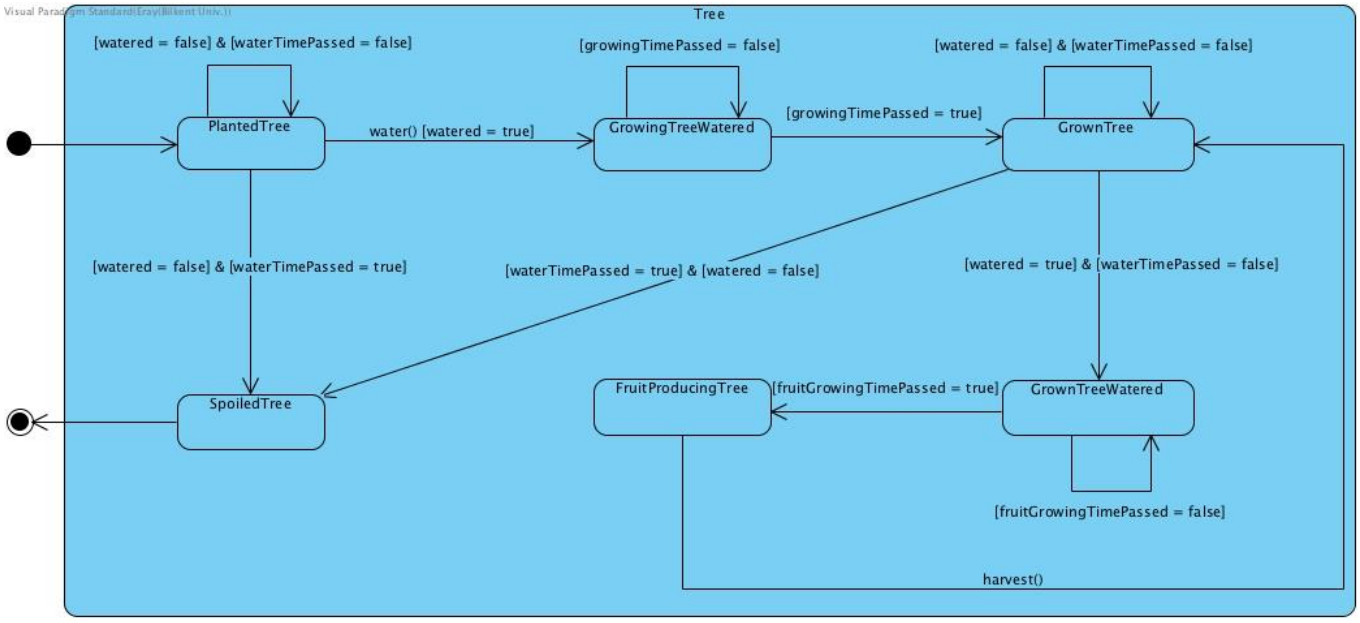


Figure 5.2.2.1 Development Of A Tree State Diagram

In this state diagram, an example of the Tree Object is shown . Initially, after having been planted on an available slot, trees need to be watered within a specified time(determined by *waterTimPassed)* . If these sown trees are not watered, they will be spoiled. On the other hand, *watered* Trees form growing trees. They stay in this state until theform their GrownTree. Just like PlantedTrees, GrownTrees are spoiled when they are not watered. In all cases of spoilage – whether it is a tree, food or seed that spoils- or gathering of grown products, the land slot becomes available for planting new tree.

5.2.3 Development Of A Seed State

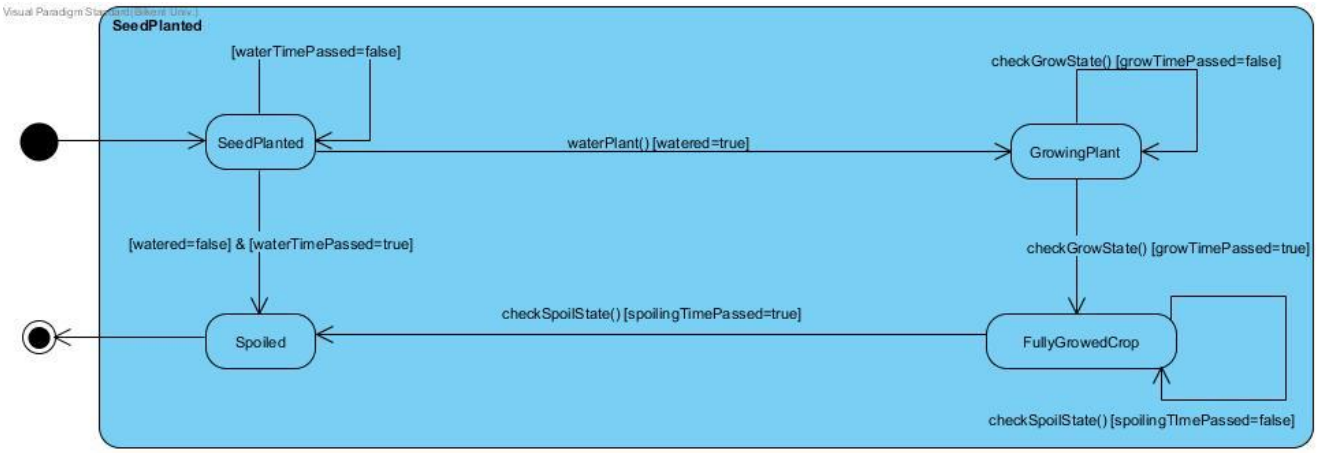


Figure 5.2.2.1 Development Of A Seed State Diagram

The above diagram illustrates the states through which seeds proceed. Initially, after having been planted on an available slot, seeds need to be watered within a specified time (determined by waterTimePassed). If these sown seeds are not watered within that time, they are spoiled. On the other hand, watered seeds form growing plants. They stay in this state (keep growing) until they form their FullyGrownCrop. Just like seeds, FullyGrownCrops are spoiled when they are not harvested within a time limit (to be clear, spoilingTimePassed determines grown crops' spoiling time whereas waterTimePassed defines that of the seeds). In all cases of spoilage -whether it is a seed or food that spoils- or gathering of the grown products, the land slot becomes available for planting new seeds. In other words, spoilage and harvesting cause land slots to be "reset", making them available for planting from scratch as also mentioned in the overview.

5.2.3. ACTIVITY DIAGRAM

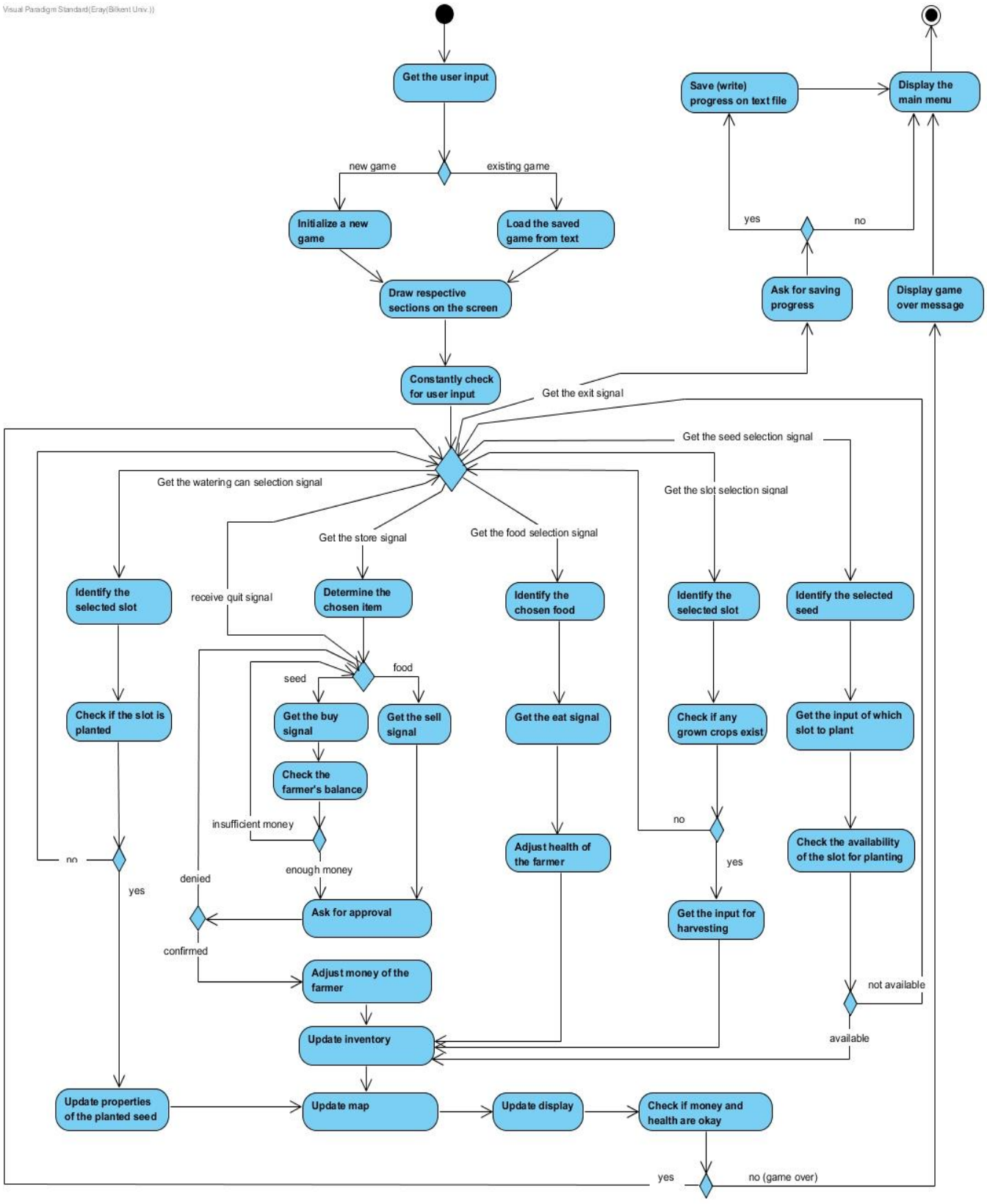


Figure 5.2.3.1 Activity diagram

The above diagram illustrates the activities performed by the system for the “Play Game” use case. Firstly, the system receives the player’s input to understand whether it is a new game to be initialized or an existing one to be loaded. Then, in either case, the respective panels are drawn on the screen.

The system, at this point, checks the input provided by the player. If the player clicks on an already-owned seed to plant, for instance, the system identifies the selected seed. Then, it expects the player to choose a slot to plant that seed. Having identified that a slot is selected, the system ensures that it does not bear any seeds which might be previously sown. Finally, the seed is planted and the inventory is updated.

Having planted some seeds, the player may want to water them. In this case, the system finds that the player has chosen the watering can. Then, the system checks the slot selected by the player to see if it has seeds that can be watered. If there are seeds in that slot, the system updates the properties of that seed as well as the display.

Since seeds form “food” after they grow, the player is able to harvest them. For this case, the player clicks on slots bearing grown seeds. The system checks if the clicked slot has grown crops in it and, if so, allows the player to harvest them. The system updates the inventory, map and display.

The collected food can be eaten to maintain the health of the farmer. When the player chooses a food that is already collected, the system, firstly, identifies that food. Then, having received the eat signal, it adjusts the health of the farmer by using the health contribution of that food. The inventory is updated.

Finally, the player is able to purchase seeds and sell collected crops (food) using the store. For “purchasing seeds” case, the player, firstly, chooses a seed. The system identifies

that seed and compares its price with the farmer's balance. If the farmer has enough money, a message appears for the player to confirm that purchase attempt. If the player confirms, the seed is purchased. For the selling case, the system identifies the chosen food and displays a similar message for the player to confirm that decision. In each case, for the confirmed ones, the money of the farmer and the inventory are updated.

Note that there is a chain of activities performed at the end of each of the above activities. More specifically, the map (where objects are located) and the display are always updated as part of all in-game actions. Furthermore, the health and money of the farmer are checked to see whether the game is over.

6. OBJECT AND CLASS MODEL

In this section, object model of Farmio is described. First, general 2 part version of UML Class diagram is given. After these two parts, detailed versions are given. Changes made after first iteration is also added to this part bellow.

Tree: Trees have three kinds: apple, cherry or raspberry. They have different growing times and a status of development. Trees produce fruit when they are watered, just like seeds' forming grown crops.

Power-up: This concept of power-ups is a new feature and has two versions as GMC (genetically modified crop) and fertilizer. They boost growing status of the plants.

Rain: Rain randomly waters every plant in the farm. In contrast with power-ups, rain can't be bought from store, it randomly happens.

GameObject: As the game requires items to be visualized on the screen, most of the objects are expected to have a name, x-y coordinates and an icon. Therefore, this generalization of objects in the frame of "game object" is crucial to our game.

Item: This represents a general concept of items that can be equipped. More specifically, watering can, seed and food are also items to make the interactions of store and inventory easier and operate on the same “unit”.

Seed: Seeds have different kinds: potato, tomato, corn, sunflower and strawberry seeds. They have attributes representing their water condition, price at the store, growth time, state of development, assigned power-ups (if any) and spoilage condition.

SeedManager: This is to check the growing state of seeds and regulate them with a timer and even warn the player with pop-up messages.

Food: As referred multiple times throughout this report, seeds produce grown crops and these grown crops are considered as “food”. Therefore, a food can be of any of the following types: strawberry, corn, sunflower, potato, pomato, cherry, raspberry and apple. Also, with our last addition of trees to the game, not all food are derived from seeds but also from trees. In other words, there are five types of seeds -potato, tomato, corn, sunflower and strawberry- and three types of trees -cherry, raspberry and apple trees- that add up to eight, compensating the number of different kinds of food. That is, foods are produced both from seeds and trees. Note that each of these foods has different contributions to the health of the farmer when eaten, in addition to their selling prices’ being different.

Farmer: The farmer, which is to be represented as a static image on the screen, has attributes of money and health. When the farmer buys / sells items, the money will be increased / decreased. On the other hand, in case of food consumption, the health will be increased.

FarmHouse: There will be a farmhouse with a fixed location at the beginning of a new game (of course, it will remain there when the same game is continued). Therefore, this class has location-related attributes and operations.

Soil: This class just appears as a conceptual entity and is a generalization of “pits” and “grasses”. The main focus will be on the pit (which is, again, a type of soil) that can bear seeds. Therefore, pit has an intermediate role in the interactions of the player with seeds.

WateringCan: WateringCan class simply represents a watering can that can be equipped, has a concrete water level which decreases as it gets used on soil slots bearing seeds.

Inventory: The inventory is a medium for the player to interact with owned items. The purchased seeds and gathered food, both of which are considered “items”, are to be maintained in the inventory.

InventoryManager: This is the manager of inventory as its name implies. This class modifies other objects after interactions in inventory class.

Store: As the name suggests, this is the store to purchase seeds and power-ups. In addition to buying seeds and power-ups, the player may also sell foods here to earn money.

StoreManager: Store manager is also like inventory manager which modifies other objects after every user interaction with store.

Map: Map maintains the game objects as well as the available slots and planted ones. It plays a crucial role in the game and constantly receives updates depending on the player’s actions. For instance, when a seed gets planted, map has to know about this seed, its location, type etc.

MapManager: This manager will keep most of the manager objects. Therefore, it is the core manager class which will modify map and accomplish map-specific tasks.

GameManager: This class will have the essential logic of the game, which include functions for starting and finishing game.

MainMenu: This class is responsible for menu-related actions. As available options are drawn on the screen and user's events are reported to here, this class may choose to start the game (new or an existing one), launch help screen, view credits or display settings.

FileManager: Reading the saved properties of a game from a text file as well as recording new data to the same (or even new) text file(s) are the responsibilities of this class.

GamePanel: Having obtained the properties of an existing game (or defined those of a new one), this class draws the inventory, store and farmer boxes on the screen.

HelpPanel: This is a simple class to indicate the help screen. The information of game controls is stored within this class.

CreditsPanel: Similar to the help section, a description about us is given through this class.

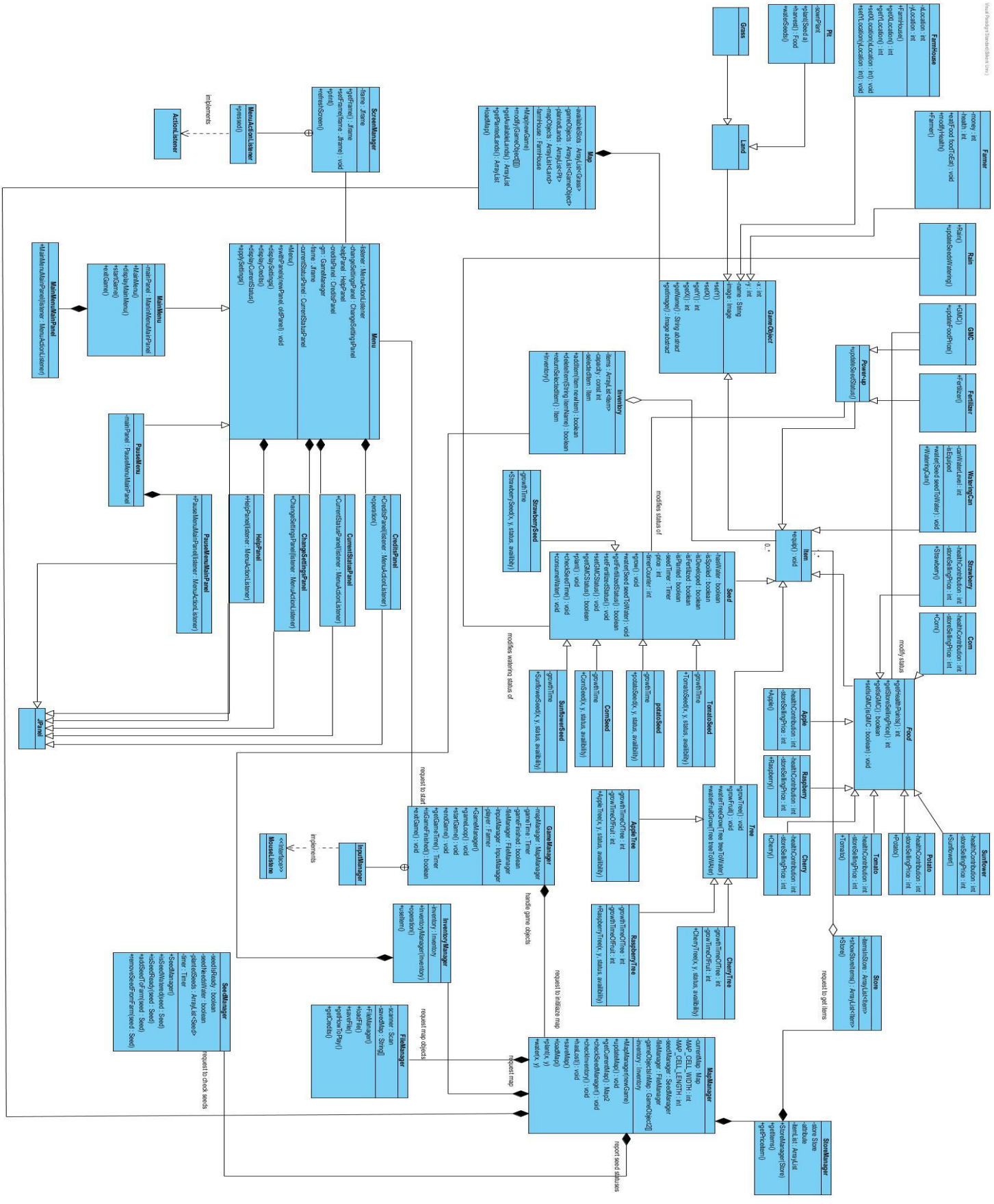


Figure 6.1 Class Diagram General View Part 1

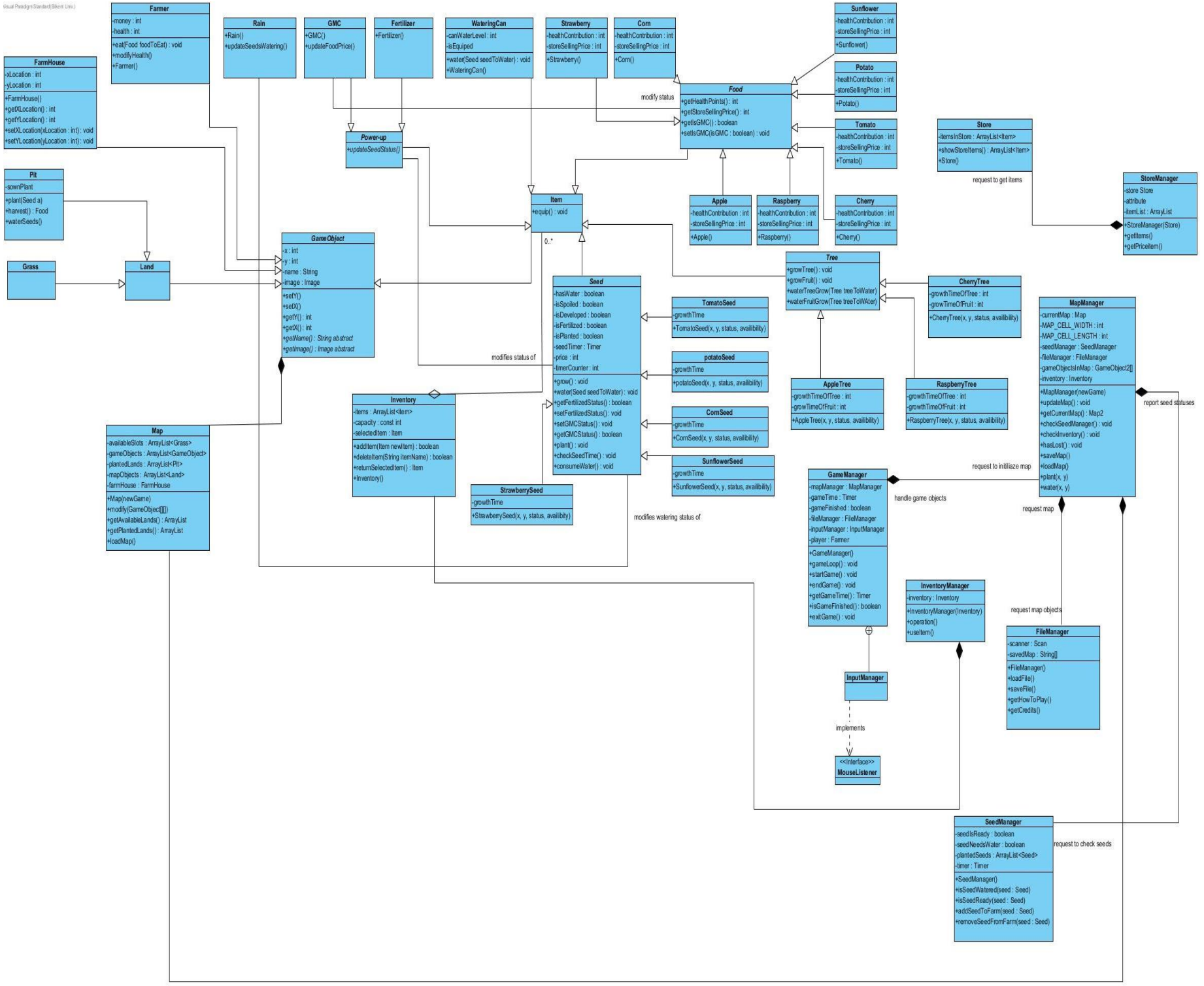


Figure 6.2 Class Diagram General View Part 2

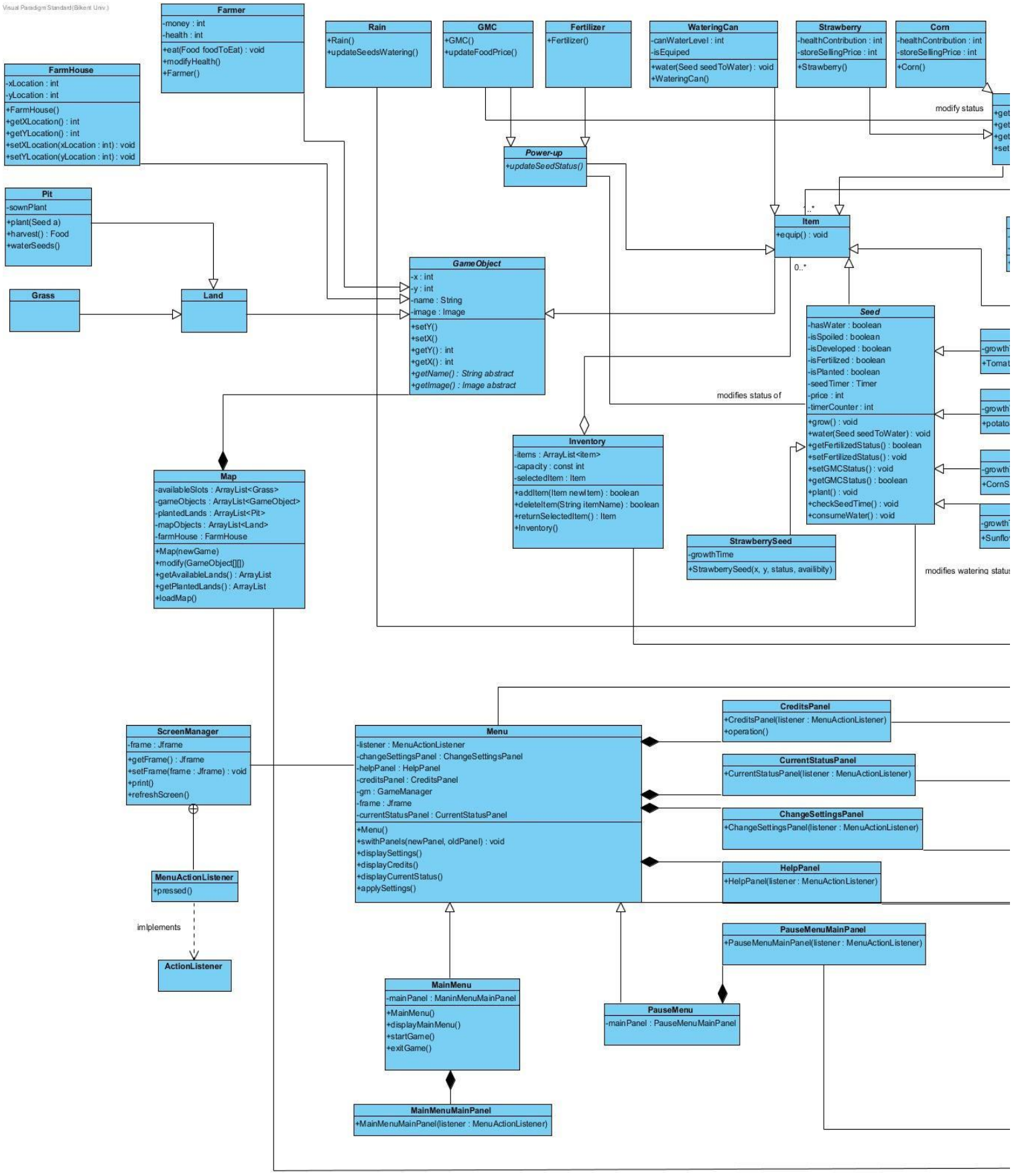


Figure 6.3 Class Diagram Detailed View Part 1

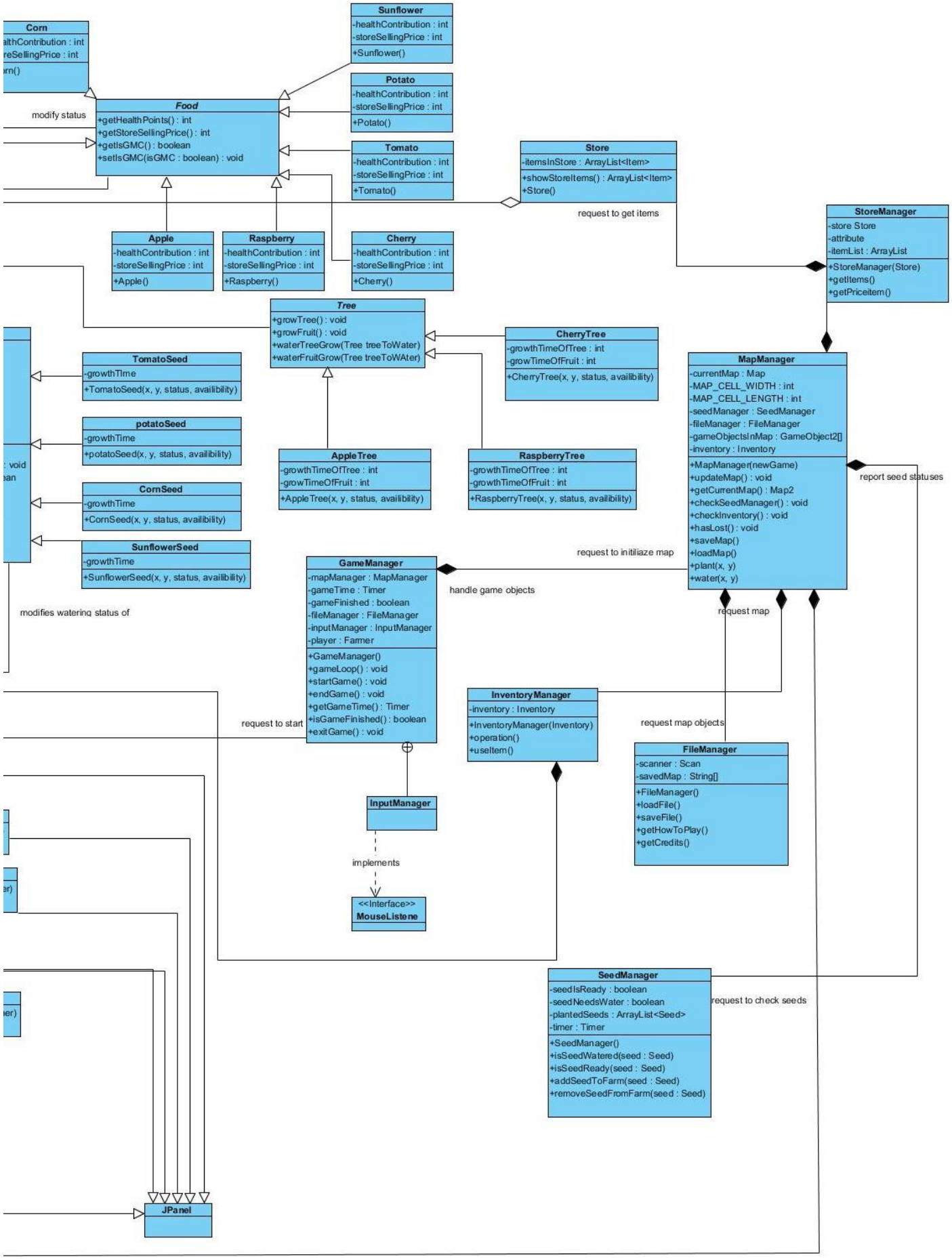


Figure 6.4 Class Diagram Detailed View Part 2

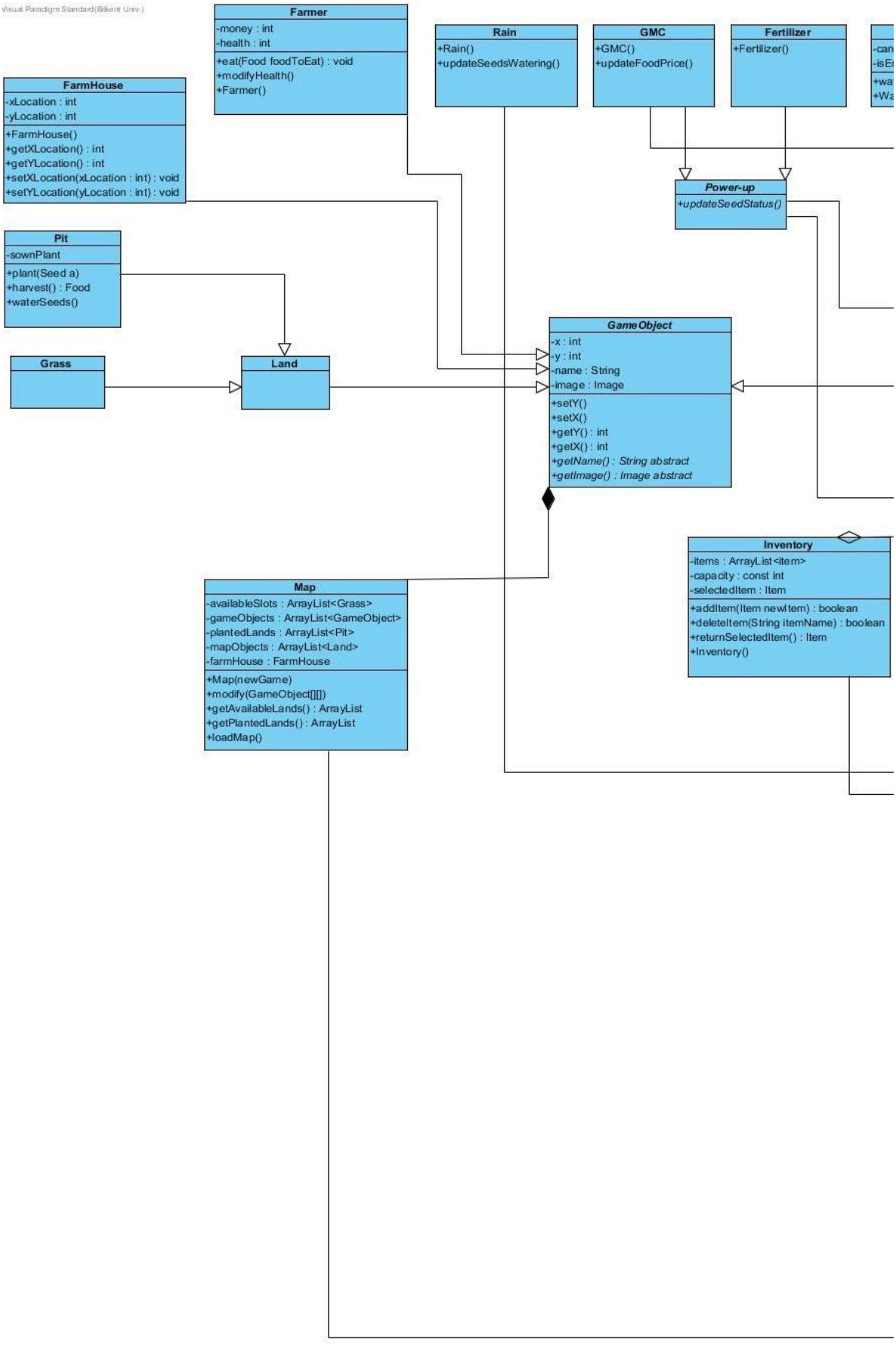


Figure 6.4 Class Diagram Detailed View Part 3

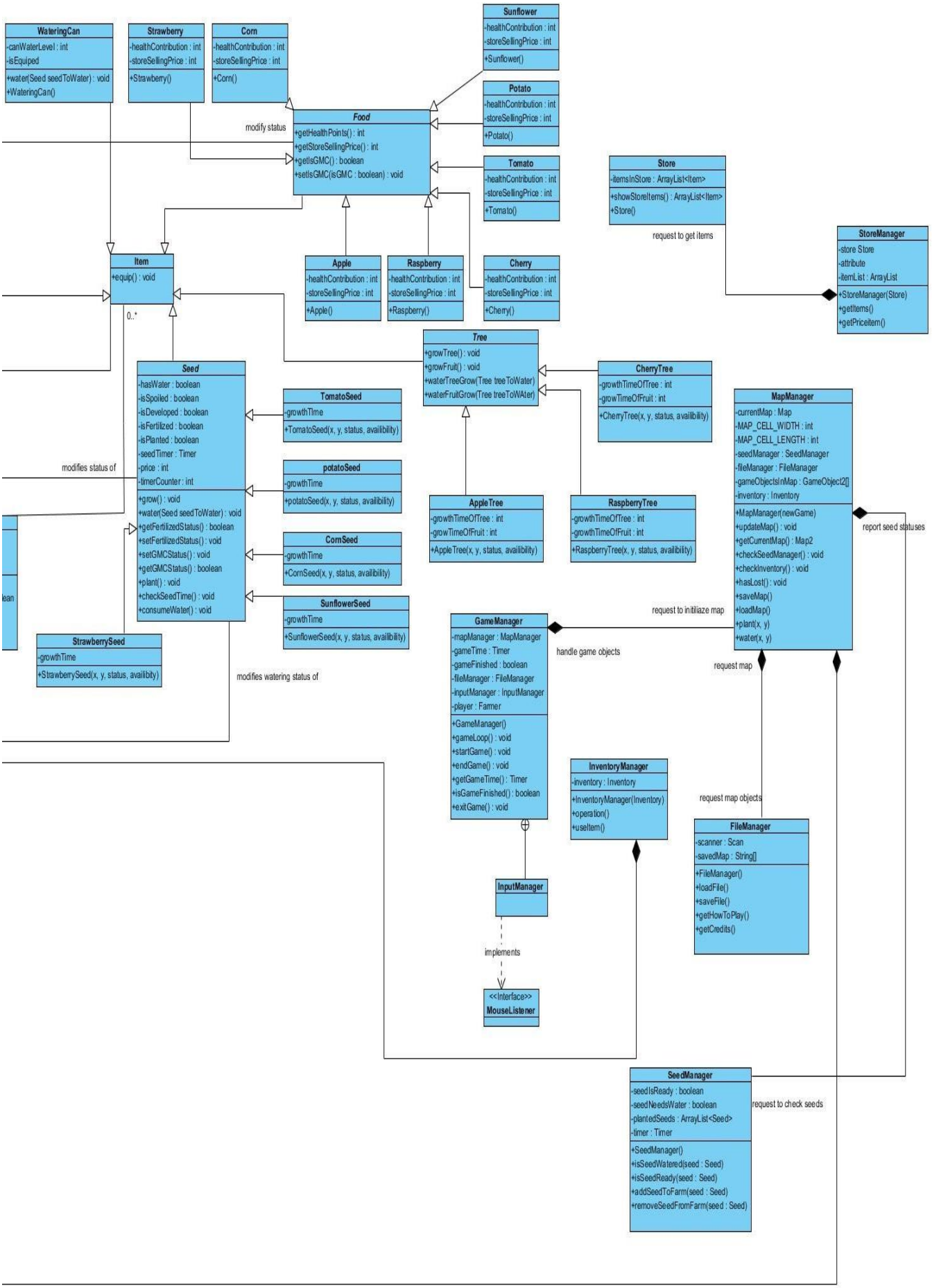


Figure 6.5 Class Diagram Detailed View Part 4

7. UI – NAVIGATIONAL PATHS AND SCREEN MOCK -UPS

7.1 Menu for Pause and Main Menu

This is the representation of menu for both load game, pause menü and start page.

Figure 7.1.1 Menu Page Mockup

7.2 Game Play

7.2.1 Inventory

Figure 7.2.Inventory Mockup

7.2.2 Planting Seeds And Trees

Figure 7.2.2.1 Planting Seed and Tree Mockup

7.2.3 Growing Process Of Seeds

Figure 7.2.3.1 Growing Process of Seeds Mockup

7.2.4 Grown Example Of Seeds

Figure 7.2.4.1 Grown Seed Mockup

7.2.5 Store

Figure 7.2.5.1 Store Section Mockup

7.2.6 Example Map


Figure 7.2.6.1 Map Mockup

7.3 Credits

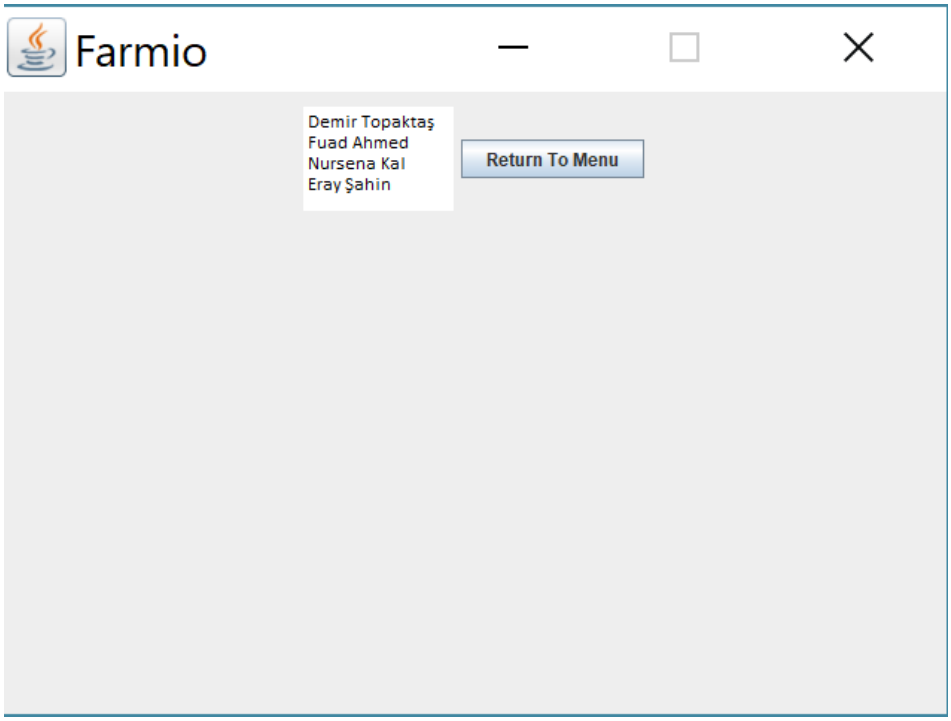


Figure 7.3.1 Credits Page Mockup

7.4 Help

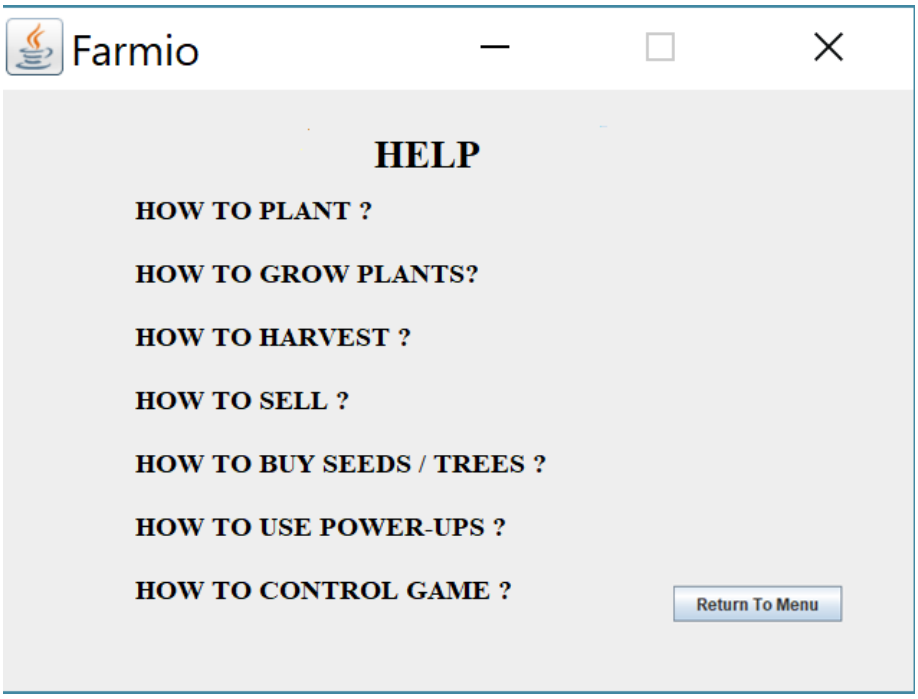


Figure 7.4.1 Help Page Mockup

8 . Referances

Original Farmville Game ： https://www.zynga.com/games/farmville