Tan Yeh Han John – Project Portfolio for Farmio

About the Project

This Project began as a Task Manager system named Duke. Together with 4 other members, we were given a choice to optimise the task manager for a more specific target group or to morph the task manager into a different product. After much discussion and research on our potential target audiences, our group found out that the Ministry of Education recently announced that upper-secondary students must attend a coding programme named 'Code for Fun'.

This gave us an inspiration to revamp Duke into a computational thinking game for students who require an elementary introduction into coding which led to the development of Farmio. We aim to teach children basic computing methodology in a fun and interactive way. Through this gameplay, students are exposed to problems that can be solved using programming concepts such as "if-else" statements, "for" and "while" loops.

Figure 1. Welcome Screen of Farmio

Note the following symbols and formatting used in this document:

	This symbol indicates important information
`start`	A grey highlight indicates that this is a command that can be typed into the command line and executed by the game
`Task`	A grey highlight with blue text indicates a component, class or object in the application

Summary of Contributions

This section shows a summary of my coding, documentation and other helpful contributions to the team project.

Core Feature Added: I added the ability for the user to access an In-Game Menu and Show List which consists of the Action List, Condition List, Market List and Task Command List. (Pull Request #218)

- In-Game Menu: The In-Game Menu can be accessed by entering a command `menu`
    - This component is essence for the user to get an overview of all the game commands that can be used at the specific level. The overview will be shown as a frame on the game console.
    - In the event that the user is unable to solve the given objective, the user can go to the In-Game Menu to view a list of other commands that can be used to complete the objective.
    - Considerations were made to allow users to be able to create tasks within the In-Game menu. This means that users can craft and input their commands while referring the different references inside the menu. Users are able to save their game, load a previously saved game and quit the game while viewing the menu.
    -
- Show List: The Show List can be accessed by entering the commands `action`, `condition`, `market` and `task command`.
    - This component is important for users to have a summary of the type of actions, conditions, price of assets and the commands they can use at the current level. The respective lists will be shown as a frame on the game console.
    - In the event that the user is unable to solve the given objective, the user can use the different Show Lists to guide them to complete the level. The commands for all the Show Lists can be seen from the In-Game Menu.
    - The Parser plays an extremely important role in this feature. Within the parser, I have added different conditions to detect the different Show List commands. After the parser detects the type of valid show list commands that the user wants, it will interact with other classes to simulate the respective frames to the user.
    - Considerations were made to allow users to create tasks within the different Show Lists. I believe that it will be more convenient for users to be able to refer to the different actions, commands or task commands when they are crafting their own code to create, edit, insert or deleting a task.

Summary of Contributions

Code Contributed: Please click the link below to see a sample of all the code I have written.

Link: https://nuscs2113-ay1920s1.github.io/dashboard/#search=F14-2&sort=groupTitle&sortWithin=title&since=2019-09-21&timeframe=commit&mergegroup=false&groupSelect=groupByRepos&breakdown=false&tabOpen=true&tabType=authorship&tabAuthor=JohnTYH&tabRepo=AY1920S1-CS2113T-F14-2%2Fmain%5Bmaster%5D

Additional Contributions and Features:

- Enhancement to Places package.
    - Created all conditions and methods to change or access assets within the WheatFarm, ChickenFarm and CowFarm classes.
    - Enhanced the WheatFarm, ChickenFarm, CowFarm and TaskList architecture to route through Farmer. (Pull request #312)
- Implemented 'Add Name' Feature
    - This feature allows users to personalise their own names when playing Farmio.
    - The name add feature will be automatically prompted to the user whenever they start a new game.
    - Implemented name restrictions and added exceptions whenever the user input a non-valid name. (Pull request #208)
- Enhancement to Load Game Feature with regards to Farmer's Name
    - Implemented a bool function 'isValidName()' inside Farmer class to check whether the name that is loaded from the save file is valid. (Pull request #273)
- Testing
    - Implemented JUnit testing for all functions for `WheatFarm`, `ChickenFarm` and CowFarm. (Pull request #312)
    - Coverage for each class is more than 88%.
- Enhancement to Narratives and Feedback
    - Reviewed the game narratives and feedback to make it easier for users especially those of younger age to interpret. (Pull request #282)
- Enhancement to Simulation Frames
    - Implemented `gotoMarket` command simulation (Pull request #222)
    - Implemented `gotoWheatFarm` command simulation (Pull request #222)
    - Implemented `loadgame` command simulation. (Pull request #226)

Contributions to User Guide

The following is an excerpt from our ***Farmio User Guide***. I was in-charge of the Name-Add feature, Menu feature, Task Command List feature, Action List feature, Condition List feature and Market List feature for the User Guide.

Menu Feature:

The menu command allows you to access the in-game menu.

Format: `menu`

This in-game menu will show you a list of commands that can be executed to help you in the game. These commands will be discussed in detail from section 3.4.7 to section 3.4.10 of the User Guide.

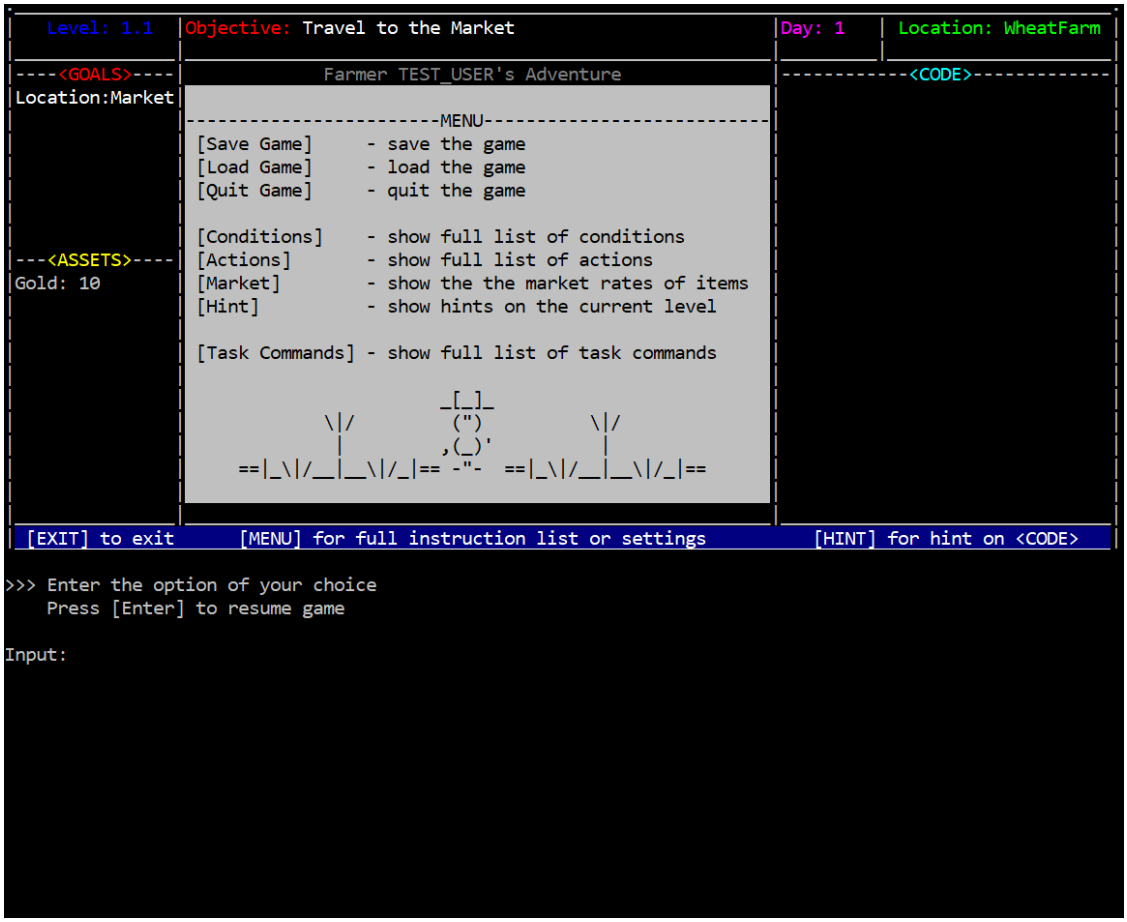

Figure 3.4.6.1. In-game menu

All commands displayed on the menu can be used at any point in the interactive segment. You do not need to go to the menu first.

Contributions to User Guide

Show List feature:

The ActionList is a sub-feature of the Show List feature which displays all the actions that can be used at the current level.

Format: `action`

Displayed action categories:

- Transportation actions. These actions can be used to change the location of the farmer at the current level. You can choose to enter input and create a task at the ActionList screen.
- Market actions that can only be used at the market. These actions mainly consist of buying and selling actions.
- Farm actions that can only be used at the WheatFarm. These actions mainly include planting seeds and harvesting wheat.

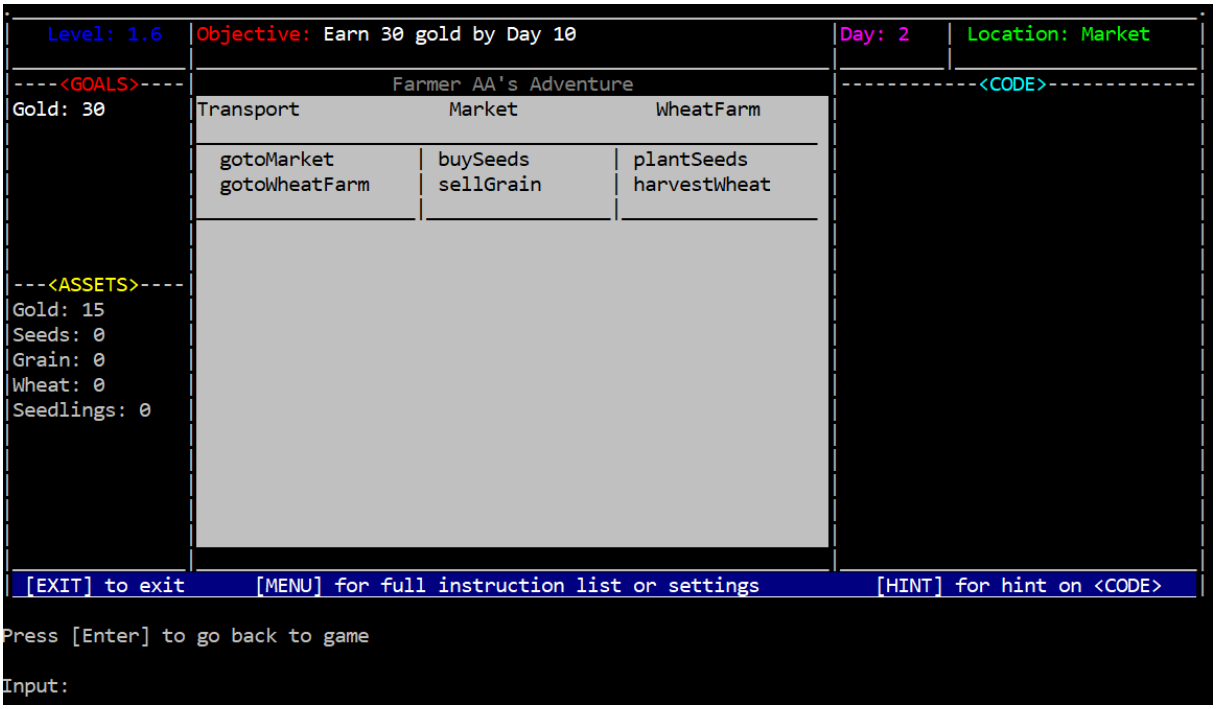

Figure 3.4.8.1. Action List

Additional Contributions to User Guide:

- Add Name Feature can be found in section 3.2.3. of the User Guide
- Other Show List Features can be found in section 3.4.7 – 3.4.10 of the User Guide.

Contributions to Developer Guide

The following section shows my additions to the ***Farmio Developer Guide*** for the Add Name feature, Show List Feature and Running User Created Tasks Feature.

Running User Created Tasks Feature

All Tasks added by the user will be stored in a TaskList. This feature is used when the user is done with his input and starts running the code.

Implementation

This feature goes through the task list after the user has completed creating tasks, and executes the user created tasks one by one.

The running user created tasks feature consists of the following methods:

- Command#CommandDayStart() - Shows and sets the start of a new day.
- Command#CommandTasksRun() - Runs the TaskList and prepare to check if the level's objectives are met.
- Command#CommandCheckObjectives() - Checks if the objectives are met and sets the corresponding stage according to the results.
- Command#CommandDayEnd() - Sets and shows the end of the day and sets the next day.

Given below is an example usage scenario of how the running tasks feature behaves for a tasklist where the user has created only one task that plants seeds.

Step 1: The User will be prompted to input start for the game to run their code. The parser will then initialize Command#CommandDayStart() object. The purpose of Command#CommandDayStart() is to change the stage in Farmio class from DAY_START to DAY_RUNNING.

Step 2: The parser will check which stage is the game at. In this case the current stage is DAY_RUNNING. Upon checking, the parser will call the respective game state command. In this case, the parser will create Command#CommandTasksRun() and Logic will execute it.

Step 3: Command#CommandTasksRun() will call the Farmer#startDay() method in Farmer. The Farmer#startDay() method will execute all the tasks in TaskList which consists of all the tasks added by the user. In this example, the action to be executed in the TaskList will be plantSeedAction.

Step 4: The abstract Action class will receive the task to be executed. If the condition in the task is an ‘if’ condition, IfTask will check whether the condition is satisfied from the Condition#check() method in the ConditionChecker class. In this case, the task type is a ‘do’ task and since ‘plantSeeds’ is a valid action, Action#PlantSeedAction() will be called.

Step 5: Inside the class Action#PlantSeedAction which extends the abstract Action class, the class checks whether the conditions are met. For example, the farmer can only plant seeds if the farmer is in the WheatFarm and has enough seeds. If these conditions are not met, the program will return an error message to the users. If the conditions are met, the simulation will be animated through the simulation feature which will be elaborated on below and we will call the method WheatFarm#plantSeeds() in the class WheatFarm through Farmer#farmer.plantSeeds() to change the value of seeds and seedlings the user has.

Step 6: After executing all the tasks in the TaskList, Command#CommandTasksRun() will change the game state to CHECK_OBJECTIVES. The parser will then call the Command#CommandCheckObjectives() method in the CommandCheckObjective Class to check whether the objectives are met.

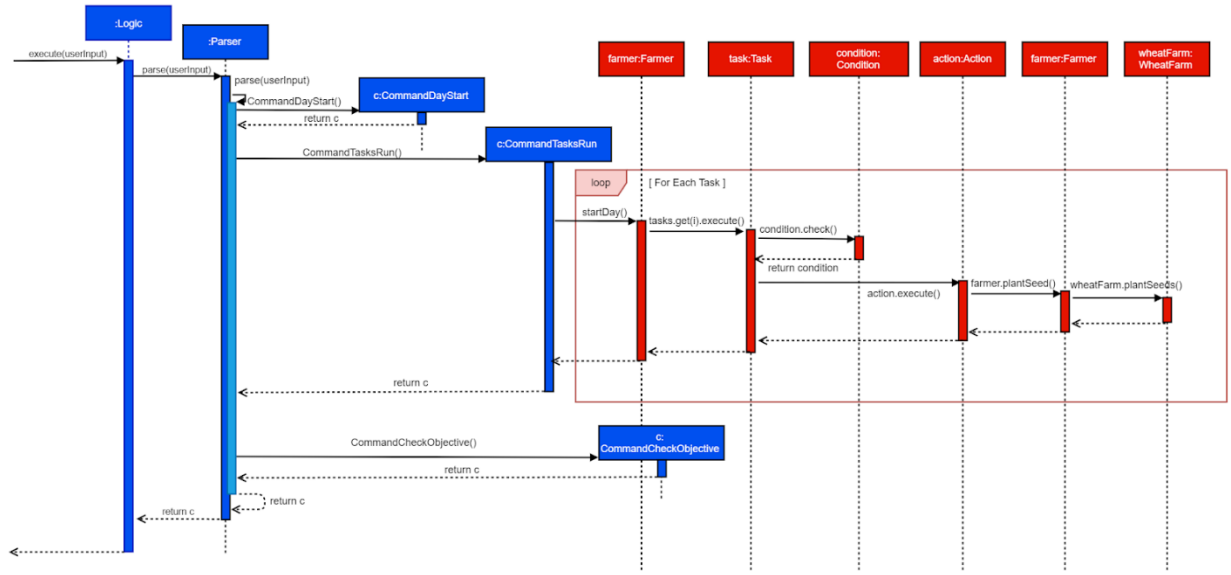

Figure 4.4.1.1. Sequence Diagram for Task Running Mechanism

Contribution to Developer Guide

Design Considerations

Aspect: Execution of tasks in the task list

- Alternative 1 (current choice): Use the execute method in the Task object
    - Pros: Improves abstraction, improves maintainability of the Farmer#startDay() method.
    - Cons: Increases complexity of Task object, as it has to do condition validation instead of simply being a container for a condition and action.
- Alternative 2: TaskList checks the Condition contained in the Task and calls the execute method of the Action.
    - Pros: The Task class will be much simpler and easy to maintain, as it would just be a container
    - Cons: Poor maintainability for TaskList, and poor cohesion as it would then become responsible for the low-level steps of task execution instead of simply calling the execute method of the Task object.

Additional Contributions to Developer Guide:

- Add Name Feature can be found in section 4.2. of Farmio Developer Guide.
- In Game Menu Feature can be found in section 4.9.1 of Farmio Developer Guide.
- Instructions for Manual Testing for Add Name Feature can be found in Appendix F section F2. of Farmio Developer Guide.
- Instructions for Manual Testing for Running User Created Task Feature can be found in Appendix F section F4. of Farmio Developer Guide.
- Instructions for Manual Testing for In-Game Menu Feature can be found in Appendix F section F9. of Farmio Developer Guide.
- Instructions for Manual Testing for Show List Feature can be found in Appendix F section F9. of Farmio Developer Guide.