CS 2340 — Milestone 5: Project Iteration 4

Random Events, Fertilizing, Pesticides, and DCD

BACKGROUND: For this iteration of the project, your team will implement multiple farm interactions along with random events. Random events can occur whenever the game moves forward a day. While these events can negatively affect the player's crops, you will also provide new ways to improve their farm's output through fertilizing and pesticide application.

PURPOSE: Design class diagrams allow you to map the structure of your application through the specification of its software classes and their relationships.

TASK: For this milestone there is one design deliverable, one testing deliverable, a required feature set, and an extra-credit opportunity. The details of the implementation requirements are outlined below. Your app implementation/functionality will be graded during a demo, which will occur the week after milestones are due.

Design Deliverable: Design Class Diagram

1. Create a design class diagram for your implementation from M4 that includes the following:
    - Classes
    - Attributes with basic types
    - Access modifiers
    - Operations
    - Associations with multiplicities
2. Your DCD should include any classes necessary for planting, watering, and harvesting.
3. You do not need to include UI classes or getters/setters in your DCD.

Implementation Requirements

1. Random events.

- Events should have a random chance of occurring when the game advances a day. This chance should increase based on game difficulty and/or season.
- You should implement the following random events:
  - **Rain**: Increase the water level of all plots by a random amount.
    - If the max water level is exceeded, the plant should die.
  - **Drought**: Decrease water levels of all plots by a random amount.
    - If the water level drops below the minimum, the plant should die.
  - **Locusts**: Kill plots randomly.
    - The number of affected plots should increase based on difficulty.
- The player should be alerted of any event that occurred through a popup of some kind.
  - Rain: Inform the player how much water levels were increased.
  - Drought: Inform the player how much water levels were decreased.
  - Locusts: Inform the player how many crops were killed.

2. Pesticides
   - **Pesticides** should be available for purchase in the market. Pricing should vary based on game difficulty.
   - Applying pesticides to a crop should protect the crop from locusts.
   - Crops that have been sprayed with pesticide should have a lower selling price in the market.
   - Pesticides should only affect the crop that they are applied to. After a crop is harvested and a new crop is planted in its place, pesticides must be re-applied to have an effect.
3. Fertilizers
   - **Fertilizers** should be available for purchase in the market. Pricing should vary based on game difficulty.
   - Fertilizers will speed up a crop's growth cycle and have a chance of increasing its yield.

- Fertilizer effects shouldn't last forever. Develop and display **fertilizer levels** for each plot (either graphically or textually).
- Fertilizing a plot should increase the plot's current fertilizer level.
    - There is no penalty for too little or too much fertilizer.
    - Each day the fertilizer level should decrease until it reaches zero, at which point any growth multiplier and yield bonus will cease.
    - More fertilizer can be applied at any point to increase fertilizer levels (up to a maximum).

Testing Requirements

1. Write **unit tests** to verify the functionality the newly implemented features.
    a. There is no code coverage requirement, but you should make sure that your unit tests cover meaningful functionality.
    b. *You should have at least 5 unit tests.*
2. **Testing Deliverable**: Include with your submission a brief writeup describing your testing process for the milestone. Explain which components were chosen for testing and why. Additionally, explain how your tests verify that the code functions as expected.

Extra Credit Requirements [+10 points]

1. Farm workers

- **Farm workers** should be available to hire for a wage (daily, weekly, etc.) at the market. Wage costs should vary between difficulty levels.
- Farm workers will automate harvesting and selling at the market to remove menial tasks for the player.
- Farm workers' wages should vary based on skill-level. A more efficient farm worker will ask for a higher wage.

- If the player does not have enough money to pay the farm worker(s) wages, then they will leave the farm at the next payment cycle.

The extra credit requirements are worth up to 10 extra points in total.

Checkstyle

During demo your team will be required to run the checkstyle script (located under files>checkstyle>Java Guide.pdf). This script will give your project a score out of 10 and will account for 10 points of your M5 final grade. Be sure to run the checkstyle script prior to submission to avoid unforeseen deductions.

Milestone Tagging

Tags are a way of marking a specific commit and are typically used to mark new versions of software. To do this, use "git tag" to list tags and "git tag –a tag_name –m description".

Submission Requirements

In addition to your diagrams, ensure that you include a link to your GitHub repository in your submission. Also, ensure that you have added your grading TA(s) as collaborators so that they may view your private repository. **Repositories must be located on the Georgia Tech GitHub and must be set to private!** Points may be deducted if these guidelines are not followed!

CRITERIA: You will be graded according to the rubrics on the final pages of this assignment document. **Groups are required to demo in order to receive credit for the features they have implemented.**