CMPSCI 2261- Object Oriented Programming with Java.

Project 2: Smart Farming -100 Points

You can choose to work in groups of 2 or 3 or individually (Please complete on time if working on the project individually). You can select your group members from the people's list on canvas, email and ask them for forming groups.

If you are working in groups, you must write your partner's name on top of the file when submitting on canvas. If you are working individually, you must write that you have worked individually on the project. Please also submit your. java files and outputs screenshots with pdf of full code. Read the full document thoroughly. Plan a diagram for ease of coding to understand the flow of the project.

Grading Policy

Some classes/interfaces/menus are given to you in this document. If you need any more classes or interfaces, using your own creativity, you can create required classes, interfaces, menus, methods, objects, array lists, functionalities and interlink them with details -75 points.

Required Outputs – 25 points.

Task Description

In this project, you are required to implement a simulation of a **Smart Farming system**. you should assume that some input values (user inputs) are coming from Sensors, since we are not incorporating sensor hardware's in this project. Before starting this project, research about the domain of

Smart Farming and try to understand about the required inputs and outputs. Configure them as needed.

Here, you are supposed to model various types of sensors and irrigation strategies.

1. Define an interface named **Sensor** with methods to read sensor data such as **readMoistureLevel**() and **readWeatherCondition**().

2. Implement classes **SoilMoistureSensor** and **WeatherSensor** that inherit from the **Sensor** interface. These classes should simulate sensors to read soil moisture levels and weather conditions, respectively.

3. Create an **abstract** class named **IrrigationStrategy** with methods such as **determineIrrigationAmount**() and **scheduleIrrigation**().

4. Implement concrete subclasses of **IrrigationStrategy**, such as **BasicIrrigationStrategy** and **AdvancedIrrigationStrategy**, which utilize different algorithms to determine when and how much to irrigate based on sensor data.

BasicIrrigationStrategy:

1. **determineIrrigationAmount(double moistureLevel, String weatherCondition):**

- This method calculates the amount of water needed for irrigation based on the current soil moisture level and weather condition.
- It may use simple heuristics or predefined thresholds to determine the irrigation amount.

2. **scheduleIrrigation():**

- This method schedules irrigation sessions based on predetermined intervals or fixed times of the day.
- It does not take into account dynamic factors such as weather forecasts or soil moisture trends.

AdvancedIrrigationStrategy

1. determineIrrigationAmount(double moistureLevel, String weatherCondition, double cropWaterRequirement):

- This method calculates the optimal amount of water needed for irrigation considering the current soil moisture level, weather condition, and specific water requirements of the crop.

2. adjustIrrigationSchedule():

- This method dynamically adjusts the irrigation schedule based on real-time sensor data, weather forecasts, and crop water demand.

3. considerSoilTypeAndTopography():

- This method takes into account the soil type and topography of the field to adjust irrigation strategies accordingly.

5. Develop a **SmartIrrigationSystem** class that coordinates the interaction between sensors and irrigation strategies. This class should have methods to collect data from **sensors (user inputs)** and invoke appropriate irrigation strategies.

6. Use the existing **SmartIrrigationSystem** class to include methods for crop management, such as **monitorCropHealth() and applyFertilizer().**

7. Implement classes for any 2 types of crops, each inheriting from a common **Crop** superclass. Each crop class should include properties such as growth stage, nutrient requirements, and susceptibility to diseases.

8. Create interfaces for the decision-making module, such as **DecisionMaker**, with methods like **makeIrrigationDecision**(), **makeFertilizationDecision**(), and **makePestControlDecision**().

9. Implement concrete classes that implement the **DecisionMaker** interface.

10. Include additional functionality, such as detecting pest presence or measuring nutrient levels in the soil.

11. Incorporate a menu interface component that allows farmers to interact with the system, view sensor data, and adjust settings for crop management.

12. Develop classes to represent different types of livestock, such as cattle, poultry, and sheep, each inheriting from a common **Livestock** superclass. Include properties such as health status, diet requirements, and production metrics (ex. milk yield, egg production).

13. Implement classes for managing livestock health and productivity, with properties such as **LivestockHealthMonitor** and **LivestockProductionManager**, with methods to monitor health indicators, administer medication, and optimize feeding schedules.

14. Enhance the menu interface to provide farmers with insights into both crop and livestock health, enabling them to make informed decisions for overall farm productivity.

15. Implement messages to notify farmers of potential health issues or production anomalies in crops and livestock.

16. Utilize inheritance to create subclasses for specific types of **crops** (ex., Wheat, Corn, Tomatoes) and **livestock** (ex. Cows, Chickens, Sheep), inheriting common behaviours and properties from parent classes.

17. **Water and Energy Usage Optimization:**

1. Define an interface named **ResourceOptimization** with methods such as **optimizeWaterUsage**() and **optimizeEnergyUsage**().

2. Implement classes for irrigation systems and equipment control that implement the **ResourceOptimization** interface. These classes will override the interface methods to optimize water and energy usage based on sensor data and environmental conditions.

18. **Waste Management:**

1. Create a class named **WasteManagement** with methods.

for composting units and recycling facilities, that implement the **manageWaste**() and **recycleMaterials**().

2. These classes will provide functionalities to handle waste disposal and recycling efficiently.

19. **Carbon Footprint Reduction:**

1. Define an interface named **CarbonFootprint** with methods such as **trackEmissions**() and **reduceEmissions**().

2. Implement classes for carbon footprint tracking and reduction strategies that implement the **CarbonFootprint** interface. These classes will track emissions from various farm activities and provide methods to identify and implement measures for reducing the farm's carbon footprint.

Sample inputs, outputs and calculations.

(Your inputs and outputs need not be exactly same. This is just a sample. The requirements are given to you. You should use your own creativity, study about the domain of Smart farming and you can come up with your own inputs and outputs.)

Sample menu and submenus with outputs (You can customize the menu)

1. View Sensor Data:
2. Configure Settings:
3. Perform Actions:
4. Generate Reports:
5. Manage Livestock:
6. Manage Crops
7. System Information:
8. Exit

1. View Sensor Data:

- Sample Input: Select option to view soil moisture levels.
- Sample Output:

Current Soil Moisture Level: 40%

2. Configure Settings:

- Sample Input: Set irrigation scheduling preferences (ex, frequency, duration).
- Sample Output:

Irrigation Scheduling Preferences Updated: - Frequency: Twice a day - Duration: 15 minutes per session.

3. Perform Actions:

- Sample Input: Initiate sensor readings.
- Sample Output:

Sensor Readings: - Soil Moisture Level: 35% - Weather Condition: Sunny

4. Generate Reports:

- Sample Input: Generate report on resource usage (water, energy).
- Sample Output:

Water Usage: 100 gallons per day Energy Usage: 5 kWh per day

5. Manage Livestock:

- Sample Input: View information about individual livestock (ex. age, breed).
- Sample Output:

Livestock Information: - Name: Daisy - Age: 2 years - Breed: Holstein

6. Manage Crops:

- Sample Input: View information about planted crops (ex. type, growth stage).
- Sample Output:

Crop Information: - Type: Tomato - Growth Stage: Flowering

7. System Information:

- Sample Input: Display system status and health.
- Sample Output:

System Status: - Overall Health: Good - Software Version: v1.2.0

8. Exit:

- Sample Input: Select option to exit the smart farming program.
- Sample Output: Program exits gracefully.

------**********----------

Scenarios:

- Soil Moisture Level: 20%
- Weather Condition: Sunny

Decision-making process:

1. Read Sensor Data:

- Obtain the current soil moisture level from the sensor: 20%
- Check the current weather condition: Sunny

2. Analyze Soil Moisture Level:

- Evaluate the soil moisture level:
- If the soil moisture level is below a predefined threshold (ex.30%), it indicates that the soil is dry and irrigation may be required.
- In this scenario, the soil moisture level is below the threshold (20%), suggesting that the soil is dry.

3. Consider Weather Conditions:

- Take into account the current weather condition:
- If the weather is sunny, there is a likelihood of evaporation and increased water loss from the soil.

- Considering the sunny weather, the soil is likely to dry out further.

4. Make Decision:

- Based on the low soil moisture level and sunny weather condition, it is determined that irrigation is required to replenish the soil moisture and prevent crop dehydration.

- You can consider some threshold values.

5. Determine Irrigation Amount:

- Calculate the irrigation amount based on the current soil moisture level and crop water requirements.

- For example, irrigate for 20 minutes to ensure adequate moisture replenishment.

6. Schedule Irrigation:

- Schedule the irrigation session for the optimal time, considering factors such as time of day and weather forecasts.

Output:

Decision:

- Based on current conditions, irrigation is recommended.

- Irrigation amount: 20 minutes.

- Irrigation scheduled for 8:00 AM.

The determination of the irrigation duration (20 minutes in this case) would typically involve a more sophisticated calculation based on factors such as the current soil moisture level, crop type, weather conditions, and irrigation system efficiency.

Example of how the irrigation duration could be calculated:

1. Calculate Water Deficit:

- Determine the difference between the current soil moisture level and the desired moisture level (threshold for irrigation). For example:

- Desired moisture level: 50%

- Current soil moisture level: 20%

- Water deficit = Desired moisture level - Current soil moisture level = 50% - 20% = 30%

2. Calculate Irrigation Rate:

- Determine the rate at which water is applied by the irrigation system (e.g., gallons per minute). This value depends on the irrigation system specifications and can be predefined.
- For example, if the irrigation rate is 1 gallon per minute:
- Irrigation rate = 1 gallon/minute

3. Calculate Irrigation Duration:

- Divide the water deficit by the irrigation rate to determine the irrigation duration needed to replenish the water deficit. For example:
- Irrigation duration = Water deficit / Irrigation rate = 30% / (1 gallon/minute) = 30 minutes

To calculate water usage in gallons per day, we typically need to consider the irrigation duration and the irrigation rate (water flow rate) of the system.

Assuming we have the following information:

- Irrigation duration: 20 minutes per session
- Number of irrigation sessions per day: 3 sessions
- Irrigation rate: 1 gallon per minute

We can calculate the total water usage per day as follows:

1. Calculate Total Water Usage per Session:

- Water used per session = Irrigation duration * Irrigation rate = 20 minutes * 1 gallon/minute = 20 gallons

2. Calculate Total Water Usage per Day:

- Total water usage per day = Water used per session * Number of irrigation sessions per day = 20 gallons/session * 3 sessions/day = 60 gallons/day

So, the calculation for water usage in the Resource Usage Report would be:

- Water Usage: 60 gallons per day

To calculate energy usage in kilowatt-hours (kWh) per day, we typically need to consider the power consumption of the equipment used for irrigation and the duration of its operation.

Assuming we have the following information:

- Power consumption of irrigation equipment: 1 kilowatt (kW)
- Duration of irrigation equipment operation: 3 hours per day

We can calculate the total energy usage per day as follows:

1. Calculate Total Energy Usage per Day:

- Total energy usage per day = Power consumption * Duration of operation = 1 kW * 3 hours = 3 kWh

So, the calculation for energy usage in the Resource Usage Report would be:

- Energy Usage: 3 kWh per day

sample inputs and outputs for Monitor and track livestock health indicators.

Sample Inputs for Monitoring and Tracking Livestock Health:

1. View Livestock Health Indicators:

- Sample Input: Select option to view health indicators for individual livestock.
- Sample Output:

Livestock Health Indicators for Daisy: - Weight: 500 kg - Temperature: 38.5°C - Heart Rate: 80 bpm - Respiratory Rate: 20 bpm - Activity Level: Normal

2. Track Livestock Health Trends:

- Sample Input: Select option to track health trends over time.
- Sample Output:

Livestock Health Trends: - Weight Trend: Stable - Temperature Trend: Increasing - Heart Rate Trend: Normal - Respiratory Rate Trend: Decreasing

Sample Inputs for Performing Actions Related to Feeding and Medication:

1. Feed Livestock:

- Sample Input: Select option to feed livestock.
- Sample Output:

  Feed Livestock:

  - Feeding Daisy with 5 kg of hay and 2 kg of grains.

2. Administer Medication to Livestock:

- Sample Input: Select option to administer medication to livestock.

  Sample Output: Administer Medication:

  - Administering vaccination to Daisy.

  - Dosage: 10 ml

  - Injection Site: Intramuscular

Sample Outputs for Performing Actions Related to Feeding and Medication:

1. Feed Livestock:

- This output is generated when a user selects the option to feed livestock (ex. Daisy). The system executes the feeding action, providing details of the feed composition (ex. hay, grains) and the quantity fed to Daisy.

2. Administer Medication to Livestock:

- This output is generated when a user selects the option to administer medication to livestock (ex. Daisy). The system executes the medication administration action, providing details of the medication (ex. vaccination), dosage, and administration method (ex. injection site).