TME 3

Documentation file

Odinakachukwu Nzekwe

Athabasca University

**Computer Science 308:**

Java for Programmers

COMP 308

The Greenhouse Controls System

This Greenhouse Controls system is a Java-based application designed to simulate and manage the operations of a greenhouse. It features an event-driven framework for controlling various aspects of the greenhouse, such as temperature, lighting, and water. This system is implemented using object-oriented principles, including abstraction, inheritance, and polymorphism, ensuring modularity and maintainability.

**Overview of the Components**

**Controller Class**

The Controller class serves as the backbone of the application. It provides a mechanism to manage events through an event list and includes methods to add, execute, and remove events. The run method iterates over the list of events and executes the action method of each event when it is ready. Additionally, the shutdown method facilitates system shutdown operations, ensuring a clean and organized closure

.

**Event Class**

The Event class is an abstract representation of an event. It encapsulates a delay time and calculates the readiness of an event based on the system's current time. Subclasses of Event define specific behaviors through the action method, which is invoked when the event is executed. This design enables easy extension by adding new event types.

**GreenhouseControls Class**

The GreenhouseControls class extends Controller and introduces domain-specific functionalities for greenhouse operations. It includes variables to represent the state of the fans, windows, and power supply. Additionally, it overrides the shutdown method to log errors and serialize the current state to a file, ensuring recoverability in case of a failure.

This class also contains nested event classes that simulate specific operations, such as turning lights on or off, adjusting the thermostat, and managing water supply. Each nested class defines an action method to perform the corresponding task. For example, ThermostatNight sets the thermostat to night mode, and LightOn turns the lights on.

**Fixable Interface**

The Fixable interface represents an abstraction for resolving issues. It defines methods for fixing and logging issues, which are implemented by specific classes, such as PowerOnFixable and WindowFixable. These classes encapsulate the logic for repairing power outages and window malfunctions, respectively.

**ControllerException Class**

The ControllerException class extends Exception to represent custom errors in the system. It allows meaningful error messages to be generated and aids in debugging and maintaining the application.

**Restore Class**

The Restore class is responsible for restoring the system state from a serialized file. It reads the saved state using an ObjectInputStream and reinstates the GreenhouseControls object. If the restored system has an issue, the appropriate Fixable implementation is invoked to resolve the problem. This ensures system resilience and continuity.

**Key Functionalities**

**Event Management**: The system schedules and executes various events, enabling precise control over greenhouse operations.

**Error Handling**: By defining a custom exception and fixable strategies, the system ensures robust error management and quick recovery from failures.

**State Persistence**: The shutdown method logs errors and saves the system's state, facilitating a seamless restart.

**Modularity**: The use of abstract classes and interfaces promotes extensibility and ease of maintenance.

**Execution Workflow**

When the program starts, it can either restore a previously saved state or initialize a new GreenhouseControls instance. The main method adds a sequence of events to the controller and executes them in the specified order. The application continues running until all events are processed or the Terminate event is triggered.

**Practical Applications**

This system is a simulation that can be extended to control actual greenhouse hardware. By integrating with sensors and actuators, it could automate real-world greenhouse operations, such as maintaining optimal temperature, humidity, and light levels. Its modular design makes it adaptable for other domains requiring event-driven automation.

**Conclusion**

The Greenhouse Controls system exemplifies a well-architected, object-oriented application. It demonstrates how abstract design principles can be applied to create a flexible and maintainable solution for complex automation tasks. Its ability to handle errors, persist state, and facilitate recovery highlights its robustness and practicality in real-world scenarios.