Gardening System API

Your automated gardening system must expose several public methods that will be invoked by a script to simulate various environmental conditions affecting your garden. It is crucial that your implementation ensures the survival of the plants under these conditions.

class GertenSimulationAPI

void initializeGarden()

This method initializes the garden with a predefined set of plants. You may define a configuration file with the types and numbers of the plants in your garden based on your defined plant types. This method is intended to be called first in any simulation run.
Note: While the GUI allows users to add new plants and specify their needs like water requirements and susceptibility to parasites, this method is specifically for seeding data programmatically from a config file for testing purposes or further 3rd party automation.

Map<String, Object> getPlants()

Returns a map containing:

- "plants": a List<String> of plant names,
- "waterRequirement": a List<Integer> detailing how much water each plant needs,
- "parasites": a List<List<String>> listing parasites each plant is vulnerable to.

This method retrieves all plant definitions set in the initializeGarden method.
Example:

```
{
   plants = [Rose, Tomato],
   waterRequirement = [10, 15],
   parasites = [[pest1, pest2], [pest1]]
}
```

void rain(int)

Simulates rainfall in the garden. The automated system should adjust water levels for the plants based on the rainfall amount.
Input range: Range will be determined based on the output received from the getPlants.waterRequirement values

void temperature(int)

If your system includes a temperature response module, use this method to receive and handle daily temperature data.
Input range: 40 to 120 F

void parasite(str)

Triggers a parasite infestation based on each plant's vulnerabilities as defined in the getPlants() method.

void getState()

Logs details about the garden's current state, including which plants are alive, which have died, and any other relevant data.

Operational Overview

Upon calling the initializeGarden(), the simulation's clock starts i.e. beginning of your first day. Each hour post-initialization represents a new day in the simulation. A script will randomly invoke rain(), temperature(), or parasite() every hour, simulating the passage of days. After 24 simulated hours (i.e., 24 days), getState() will be invoked to assess the performance of your automated gardening system.

Additionally, don't forget the requirement to maintain a log.txt file that records all events.

Here's a pseudo-code of how your application will be tested in a simulation

```java
public class GardenSimulator {
    public static void main(String[] args) {

        GardenSimulationAPI gardenAPI = new GardenSimulationAPI();

        // beginning of the simulation
        gardenAPI.initializeGarden(); // this marks the beginning of the clock
        HashMap<String, Object> initialPlantDetails = gardenAPI.getPlants();

        gardenAPI.rain(25);
        sleepOneHour(); // end of first day

        // beginning of the second day
        gardenAPI.temperature(60);
        gardenAPI.parasites("insects");
        sleepOneHour(); // end of second day

        //.... after 24 days

        gardenAPI.getStatus();
}}
```