Topic: Advanced Inheritance, Multiple Interfaces, and State Management

The Scenario: "Lehman Smart Home Ecosystem"

You are designing a centralized controller for a Smart Home. The system manages different types of devices (Lights, Thermostats, and Security Cameras). Some devices are "Dimmable," some are "Recordable," and all are "Powerable."

Your Task

Create Java files that implements the following multi-layered architecture:

1. The Interfaces

- **Powerable**: Defines methods `void turnOn()` and `void turnOff()`.
- **Adjustable**: Defines a method `void setLevel(int level)` (e.g., for brightness or temperature).

2. The Abstract Class: SmartDevice

Create an abstract class named `SmartDevice` that implements `Powerable`.

- **Fields:**
  - `protected String deviceName`
  - `protected boolean isOn`
- **Constructor:** Initializes `deviceName` and sets `isOn` to `false` by default.
- **Static Member:** `static int activeDevicesCount` to track how many devices are currently "On" across the entire house.
- **Abstract Method:** `abstract void performSelfDiagnostic();`

3. The Concrete Subclass: SmartLight

Create a class `SmartLight` that extends `SmartDevice` and implements `Adjustable`.

- **Additional Field:** `private int brightness` (0-100).
- **Requirement:** * Implement `turnOn()` and `turnOff()`. Ensure you increment/decrement the `activeDevicesCount` only if the state actually changes.
  - Implement `setLevel(int level)`. If the device is off, printing "Cannot adjust: Device is OFF."
  - Implement `performSelfDiagnostic()` to print "Checking LED health..."

4. The Concrete Subclass: SmartThermostat

Create a class `SmartThermostat` that extends `SmartDevice` and implements `Adjustable`.

- **Additional Field:** `private int temperature`.
- **Requirement:** * Implement `setLevel(int level)` specifically for temperature logic (range 60-80 degrees).

    - Override `turnOn()` to print "HVAC System Starting..." before calling `super.turnOn()`.

Challenge Logic (Main Method)

In your `main` method, perform the following:

1. Create an `ArrayList<SmartDevice>` named `homeHub`.
2. Add two `SmartLight` objects (Living Room, Kitchen) and one `SmartThermostat` (Hallway).
3. **The Logic Test:**
    - Turn "On" the Living Room Light and the Hallway Thermostat.
    - Set the Kitchen Light brightness to 75 (this should fail if you haven't turned it on yet!).
    - Print the total `activeDevicesCount` from the `SmartDevice` class.
4. **Polymorphism:** Loop through the `homeHub` and call `performSelfDiagnostic()` on every device.

Checklist for Success

- [ ] Does `SmartLight` implement **both** the abstract class and the `Adjustable` interface?
- [ ] Did you use `static` correctly to track the global count of *active* devices?
- [ ] Does your `SmartThermostat` use `super.turnOn()` to avoid code duplication?
- [ ] Did you handle the "Device is OFF" logic inside `setLevel`?

Submission: Upload your working code to Github and provide the link before the end of the period for in-class credit!