Weather Station V2

Wednesday, November 5, 2025 7:52 PM

Let's step into **Phase 2 — Intermediate (Real-world Safety)**.

Assignment 1 — Weather Station V2 (RAII + weak_ptr)

Goal

Upgrade your Weather Station to be **production-safe**:

- No dangling observers
- No ownership cycles
- Safe self-detach during notify

What to build (requirements)

1) Interfaces (pull model)

- IObserver { void update(); }
- ISubject { Subscription subscribe(std::shared_ptr<IObserver>); }
- Observers will **pull** data from the subject via getters (getTemperature(), etc.).

2) RAII Subscription token

- class Subscription (move-only) that holds a cancel function.
- On destruction (or cancel()), it **auto-unsubscribes** the observer.
- You'll return this token from subscribe(...).

3) Subject storage & safety

- WeatherData:
  - Stores observers as std::unordered_map<size_t, std::weak_ptr<IObserver>>
    (no strong ownership → no cycles).
  - Maintains a monotonic nextId key.
  - subscribe():
    - Adds a weak_ptr slot.
    - Returns a Subscription capturing id and a weak reference to the subject that removes that id.
  - notify():
    - Takes a **snapshot** of live observers by locking weak_ptrs.
    - Prunes expired ones.
    - Calls update() **outside** the lock (if you add a mutex later).

4) Pull getters

- float getTemperature() const;
- float getHumidity() const;
- float getPressure() const;

5) Concrete observers (pull)

- CurrentDisplay pulls and prints current values.
- OnceHighTempAlert(threshold):
  - On first temp >= threshold, prints and **cancels its own Subscription** (store the token inside the observer).

6) Thread-readiness (still single-threaded)

- Add a std::mutex and guard:
  - subscription table updates,
  - state updates,
  - snapshot creation.
- Do **delivery outside the lock**.

7) Acceptance flow (in main)

1. Create std::shared_ptr<WeatherData>.
2. Create observers as std::shared_ptr<...>; subscribe() and keep the returned tokens.
3. Set measurements a few times; verify:
   - Multiple observers print.
   - OnceHighTempAlert fires once then stops.
4. Call token.cancel() for one observer; verify it stops receiving.
5. Let an observer go out of scope (destroyed) without manual cancel; verify **no crash** and it's cleaned up (expired weak_ptr pruned on next notify).