Okay, hier ist eine überarbeitete Aufgabenstellung, die mehr auf Eigenständigkeit und Analyse der Schüler setzt, mit subtileren Hinweisen:

Aufgabenstellung: Wetterdaten-Visualisierung – Entkopplung durch Beobachtung (Gruppenarbeit - 2 Personen)

Ziel:

In dieser Aufgabe sollt ihr die bestehende Wetterdaten-Visualisierung verbessern, indem ihr die Art und Weise der Datenübertragung zwischen dem `WeatherDataSimulator` und dem `WeatherVisualizer` überdenkt. Das Ziel ist, das System flexibler zu gestalten, sodass Änderungen an einem Teil des Programms weniger Auswirkungen auf den anderen Teil haben.

Szenario:

Ihr habt ein JavaFX-Programm, das Wetterdaten simuliert und anzeigt. Der `WeatherDataSimulator` erzeugt Wetterdaten und sorgt aktuell dafür, dass diese im `WeatherVisualizer` angezeigt werden. Wenn ihr euch den Code anseht, werdet ihr feststellen, dass der Simulator und der Visualizer sehr eng miteinander verbunden sind. Das kann problematisch sein, wenn man das Programm erweitern oder verändern möchte.

Eure Aufgabe:

Eure Aufgabe ist es, die **Kommunikation** zwischen `WeatherDataSimulator` und `WeatherVisualizer` zu **verbessern**. Findet heraus, wie ihr die Abhängigkeit zwischen diesen beiden Programmteilen **reduzieren** könnt, sodass sie unabhängiger voneinander agieren.

Detaillierte Schritte zur Problemlösung:

1. **Analyse des bestehenden Codes:**
    - **Untersucht die Java-Dateien:**
        - `CurrentWeatherData.java`, `WeatherData.java`, `WeatherCondition.java`: Versteht, welche Informationen Wetterdaten enthalten.
        - `WeatherDataSimulator.java`: Wie werden Wetterdaten erzeugt? **Wie werden die Daten aktuell an den Visualizer weitergegeben?** Sucht nach Stellen im Code, die direkt mit dem `WeatherVisualizer` interagieren.
        - `WeatherVisualizer.java`: Wie werden die Wetterdaten angezeigt? **Wer übergibt dem Visualizer die Daten aktuell?**
        - `WeatherVisualizationApp.java`: Wie werden Simulator und Visualizer im Hauptprogramm verknüpft?
        - `WeatherDataObserver.java`: **Achtung! Schaut euch dieses Interface genauer an. Was könnte seine Rolle sein?** Der Name könnte euch einen Hinweis geben.
2. **Identifiziert die Abhängigkeit:**
    - **Wo besteht die direkte Verbindung?** In welchen Klassen gibt es Felder oder Methodenaufrufe, die eine direkte Beziehung zwischen Simulator und Visualizer zeigen?
    - **Was passiert, wenn ihr den Visualizer austauschen wolltet?** Wäre das mit der aktuellen Implementierung einfach? Was müsste geändert werden?

3. **Überlegt euch eine flexiblere Lösung:**
    - **Wie könnte der Simulator Wetterdaten "bekannt machen", ohne direkt zu wissen, wer sie anzeigt?** Stellt euch vor, der Simulator "veröffentlicht" die Daten, und andere Programmteile können sich "interessieren" und diese Daten erhalten.
    - **Hinweis:** Denkt an das Interface `WeatherDataObserver.java`. **Wer könnte ein "Beobachter" sein? Wer könnte "beobachtet" werden?**
    - **Wie könnten "Beobachter" sich beim "Beobachteten" anmelden, um über neue Daten informiert zu werden?**
4. **Plant die Änderungen:**
    - **Welche Klassen müsst ihr verändern?** Welche Klassen werden neue Verantwortlichkeiten bekommen?
    - **Welche Methoden müsst ihr hinzufügen oder ändern?** Skizziert grob, wie die Kommunikation zukünftig ablaufen soll.
5. **Implementiert eure Lösung:**
    - Beginnt mit der Klasse, die Wetterdaten erzeugt. **Wie könnt ihr diese so anpassen, dass sie "Beobachter" verwalten und informieren kann?**
    - Passt dann die Klasse an, die die Daten anzeigen soll. **Wie kann diese sich als "Beobachter" anmelden und auf neue Daten reagieren?**
    - Vergesst nicht, die `WeatherVisualizationApp.java` anzupassen, um eure neue Kommunikationsmethode zu nutzen.
6. **Testet und überprüft:**
    - Startet die Anwendung. Werden die Wetterdaten immer noch korrekt angezeigt?
    - **Überlegt euch, wie ihr testen könnt, ob eure Lösung wirklich flexibler ist.** Was wäre, wenn ihr einen zweiten Visualizer hinzufügen wolltet, der die Daten anders darstellt? (Bonusaufgabe: Implementiert einen zweiten, einfachen "Text-basierten" Observer, der die Daten in der Konsole ausgibt.)

Hinweise und Tipps:

- **Startet mit der Analyse:** Ein gutes Verständnis des bestehenden Codes ist der Schlüssel.
- **Denkt abstrakt:** Konzentriert euch auf die *Idee* der Entkopplung, bevor ihr euch in Details des Codes verliert.
- **Das `WeatherDataObserver`-Interface ist wichtig!** Es gibt euch einen Rahmen vor.
- **Schrittweise vorgehen:** Verändert nicht alles auf einmal. Testet eure Änderungen regelmäßig.
- **Gruppenarbeit:** Besprecht eure Ideen und Lösungsansätze regelmäßig in der Gruppe.

Erwartetes Ergebnis:

- Ein funktionierendes Wettervisualisierungsprogramm, bei dem `WeatherDataSimulator` und `WeatherVisualizer` **weniger stark voneinander abhängen**.
- Ein Programm, das **erweiterbarer** ist (z.B. Hinzufügen neuer Anzeigemöglichkeiten).
- Ein Verständnis dafür, wie man durch **indirekte Kommunikation** (wie durch "Beobachtung") die Flexibilität von Programmen erhöhen kann.

Abgabe:

- Pusht die Änderungen in eurem Git-Repostory immer am Ende der Stunde