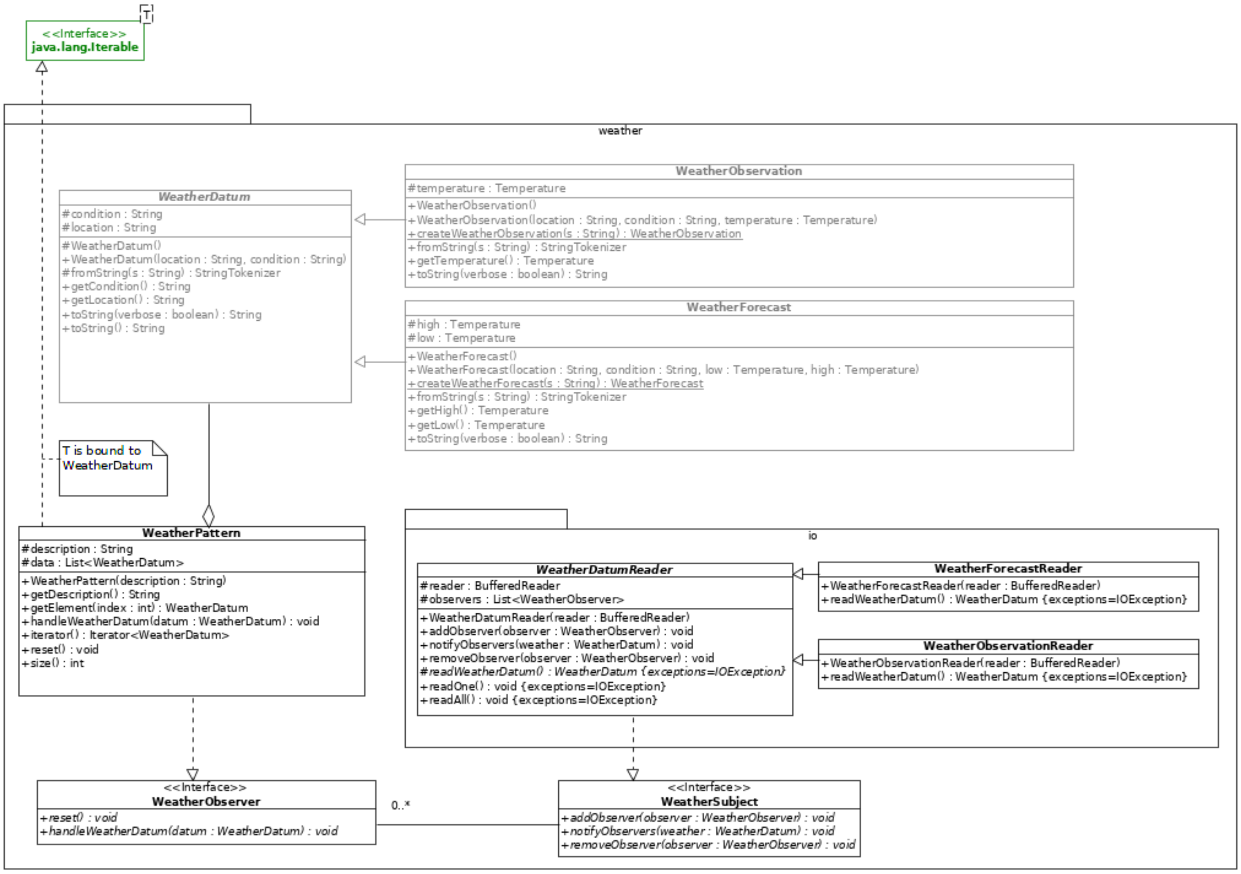
`weather` Package v2.0

Class Diagram

The relationships between the various classes and interfaces of the system are illustrated in the following UML class diagram.


T
<<Interface>>
java.lang.Iterable
weather
WeatherDatum
#condition : String
#location : String
#WeatherDatum()
+WeatherDatum(location : String, condition : String)
#fromString(s : String) : StringTokenizer
+getCondition() : String
+getLocation() : String
+toString(verbose : boolean) : String
+toString() : String
WeatherObservation
#temperature : Temperature
+WeatherObservation()
+WeatherObservation(location : String, condition : String, temperature : Temperature)
+createWeatherObservation(s : String) : WeatherObservation
+fromString(s : String) : StringTokenizer
+getTemperature() : Temperature
+toString(verbose : boolean) : String
WeatherForecast
#high : Temperature
#low : Temperature
+WeatherForecast()
+WeatherForecast(location : String, condition : String, low : Temperature, high : Temperature)
+createWeatherForecast(s : String) : WeatherForecast
+fromString(s : String) : StringTokenizer
+getHigh() : Temperature
+getLow() : Temperature
+toString(verbose : boolean) : String
T is bound to WeatherDatum
WeatherPattern
#description : String
#data : List<WeatherDatum>
+WeatherPattern(description : String)
+getDescription() : String
+getElement(index : int) : WeatherDatum
+handleWeatherDatum(datum : WeatherDatum) : void
+iterator() : Iterator<WeatherDatum>
+reset() : void
+size() : int
io
WeatherDatumReader
#reader : BufferedReader
#observers : List<WeatherObserver>
+WeatherDatumReader(reader : BufferedReader)
+addObserver(observer : WeatherObserver) : void
+notifyObservers(weather : WeatherDatum) : void
+removeObserver(observer : WeatherObserver) : void
#readWeatherDatum() : WeatherDatum {exceptions=IOException}
+readOne() : void {exceptions=IOException}
+readAll() : void {exceptions=IOException}
WeatherForecastReader
+WeatherForecastReader(reader : BufferedReader)
+readWeatherDatum() : WeatherDatum {exceptions=IOException}
WeatherObservationReader
+WeatherObservationReader(reader : BufferedReader)
+readWeatherDatum() : WeatherDatum {exceptions=IOException}
<<Interface>>
WeatherObserver
+reset() : void
+handleWeatherDatum(datum : WeatherDatum) : void
0..*
<<Interface>>
WeatherSubject
+addObserver(observer : WeatherObserver) : void
+notifyObservers(weather : WeatherDatum) : void
+removeObserver(observer : WeatherObserver) : void

Classes that are shown in gray were part of v1.0 of the weather package, the new classes (and sub-packages) are shown in black. Interfaces that are in jade green are part of the Java API.

In addition to the specifications that are contained in this class diagram, the implementation must comply with the following specifications.

The `WeatherSubject` Interface

A `WeatherSubject` is a subject (in the sense of the observer pattern) of `WeatherDatum` objects. In other words, it "produces" `WeatherDatum` objects.

The `WeatherObserver` Interface

A `WeatherObserver` is an observer (in the sense of the observer pattern) of `WeatherDatum` objects. In other words, it "consumes" `WeatherDatum` objects.

The `WeatherPattern` Class

A `WeatherPattern` object is a `WeatherObserver` that contains a collection of `WeatherDatum` objects that are supplied to it by a `WeatherSubject`.

Methods

```
handleWeatherDatum(final WeatherDatum datum)
```

Must add the given `WeatherDatum` to the collection if it is non-`null`, otherwise it must do nothing.

```
reset()
```

Must reset (i.e., remove all elements from) the collection.

The `WeatherDatumReader` Class

A `WeatherDatumReader` object is a `WeatherSubject` that reads one or more `WeatherDatum` objects from a stream, notifying its observers as each one is read.

Methods

```
readOne()
```

Must read a single `WeatherDatum` object (using the abstract `readWeatherDatum()` method) and notify all of the observers (but only if the WeatherDatum object is non-`null`).

```
readAll()
```

Must read a all of the `WeatherDatum` object (using the abstract `readWeatherDatum()` method, until it returns `null`) and notify all of the observers.

The `WeatherObservationReader` Class

A `WeatherObservationReader` object is a `WeatherDatumReader` that reads `String` representations of `WeatherObservation` objects.

Methods

`readWeatherDatum()`

Must read a single `String` representation of a `WeatherObservation` from the inherited attribute named `reader`, construct a `WeatherObservation` object from it, and return it. If the `reader` has reached the end-of-stream, then this method must return `null`.

This method must assume that there is one String representation per "line" in the stream, and that the "line" contains a terse `String` representation of a `WeatherObservation` object.

The `WeatherForecastReader` Class

A `WeatherForecastReader` object is a `WeatherDatumReader` that reads `String` representations of `WeatherForecast` objects.

Methods

`readWeatherDatum()`

Must read a single `String` representation of a `WeatherForecast` from the inherited attribute named `reader`, construct a `WeatherForecast` object from it, and return it. If the `reader` has reached the end-of-stream then this method must return `null`.

This method must assume that there is one String representation per "line" in the stream, and that the "line" contains a terse `String` representation of a `WeatherForecast` object.

Data Files

Two data files have been created for testing purposes, one that contains current observations (`current.obs`) and one that contains forecasts (`30August2021.for`).