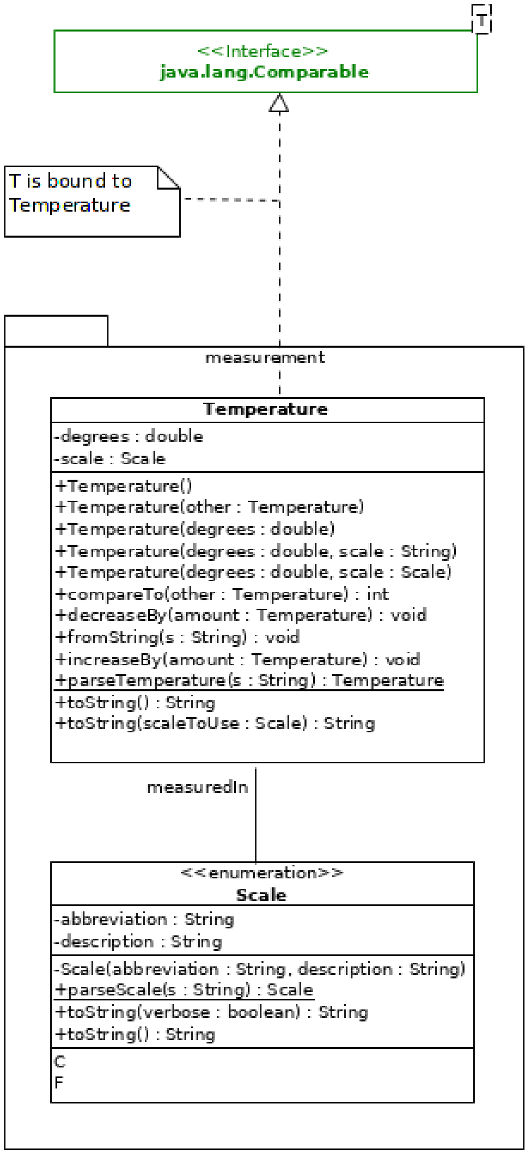
`measurement` Package v1.0

Class Diagram

The relationships between the various classes and interfaces of the system are illustrated in the following abstract UML class diagram.
Classes/interfaces that are shown in jade green are part of the **J**ava API.


T
<<Interface>>
java.lang.Comparable
T is bound to Temperature
measurement
Temperature
-degrees : double
-scale : Scale
+Temperature()
+Temperature(other : Temperature)
+Temperature(degrees : double)
+Temperature(degrees : double, scale : String)
+Temperature(degrees : double, scale : Scale)
+compareTo(other : Temperature) : int
+decreaseBy(amount : Temperature) : void
+fromString(s : String) : void
+increaseBy(amount : Temperature) : void
+parseTemperature(s : String) : Temperature
+toString() : String
+toString(scaleToUse : Scale) : String
measuredIn
<<enumeration>>
Scale
-abbreviation : String
-description : String
-Scale(abbreviation : String, description : String)
+parseScale(s : String) : Scale
+toString(verbose : boolean) : String
+toString() : String
C
F

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

The Scale Enumeration

`Scale` is an enumeration of the different temperature scales used by WeatherBits. WeatherBits currently supports two different temperature scales, Celsius and Fahrenheit (the default is Fahrenheit). WeatherBits does not currently support the Kelvin scale, though support for the Kelvin scale may need to be added in the future, and the implementation must take this into account.

Instances

This enum must define the following instances.

```
C     ("C", "Celsius"),
F     ("F", "Fahrenheit")
```

Instance Variables

This enum must, at a minimum contain the following *private* instance variables.

`abbreviation` A `String` containing the standard one-letter abbreviation for the `Scale`.

`description` A `String` containing the standard one-word description/name of the `Scale`.

Methods

```
parseScale(final String s)
```

Must parse a **terse or verbose** `String` representation (**ignoring case**) of a `Scale` and return the appropriate instance. If the `String` is neither a terse nor verbose representation (ignoring case), then it must return `null`.

```
toString(final boolean verbose)
```

Must return the `description` if `verbose` is `true` and must return the `abbreviation` otherwise.

```
toString()
```

Must return a terse `String` representation of the owning instance.

The Temperature Class

An encapsulation of a temperature that includes degrees and units/scale (e.g., Celsius/centigrade or Fahrenheit).

Instance Variables

This class must, at a minimum, contain the following *private* instance variables. It may contain other instance variables as well.

scale	A Scale.
degrees	A double containing the number of degrees in the units contained in scale.

Methods

```
Temperature()
```

Must construct a Temperature of 0.0 degrees F.

```
Temperature(final Temperature other)
```

Must construct a copy of the given Temperature object.

```
decreaseBy(final Temperature amount)
```

Must decrease the owning Temperature by the given Temperature, accounting for Scale differences. This method must not change the scale attribute of either Temperature.

```
fromString(final String s)
```

Must parse a **terse** String representation of a Temperature and change the attributes of the owning Temperature appropriately. If there is a problem with the String representation then it must leave the attributes of the owning Temperature unchanged.

```
increaseBy(final Temperature amount)
```

Must increase the owning Temperature by the given Temperature, accounting for Scale differences. This method must not change the scale attribute of either Temperature.

```
parseTemperature(final String s)
```

Must create a Temperature object from the given **terse** String representation. This method must return the default Temperature if the String is not a valid terse representation.

`toString()`

Must return a **terse** `String` representation of the owning `Temperature`. The numeric part of the `String` representation must begin with a sign indicator, and must be in a field of width 6 with one place to the right of the decimal. The numeric part must be followed immediately by the terse representation of the `scale` attribute. For example:

```
0123456
+108.1F
 +57.9F
  +0.0F
 -19.1C
```

`toString(final Scale scaleToUse)`

Must return a **terse** `String` representation of the owning `Temperature` using the given `Scale` (which may or may not be different from the `Scale` of the owning `Temperature`).