# *DASC511 Week 4 Classes Assignment*

This module provides classes that support the manipulation and analysis of measurements. Measurements supported by this module are composed of a number and a unit of measurement. For example, the measurement '3 ft' includes 3 as the number and feet as the unit of measurement. The **Measurement** class provides an object template for measurements. This class encapsulates the logic to convert from-and-to different unit of measurements and adding and subtracting measurements. Instances of the **Measurement** class have an attribute to track their unit-of-measurement and the number of these unit-of-measurements.

This documentation is organized as follows:

A. UML diagram of classes & their interactions
B. Overview information for each class
C. Detailed information on the assigned attributes & methods of each class
D. Examples of calling the methods within the classes

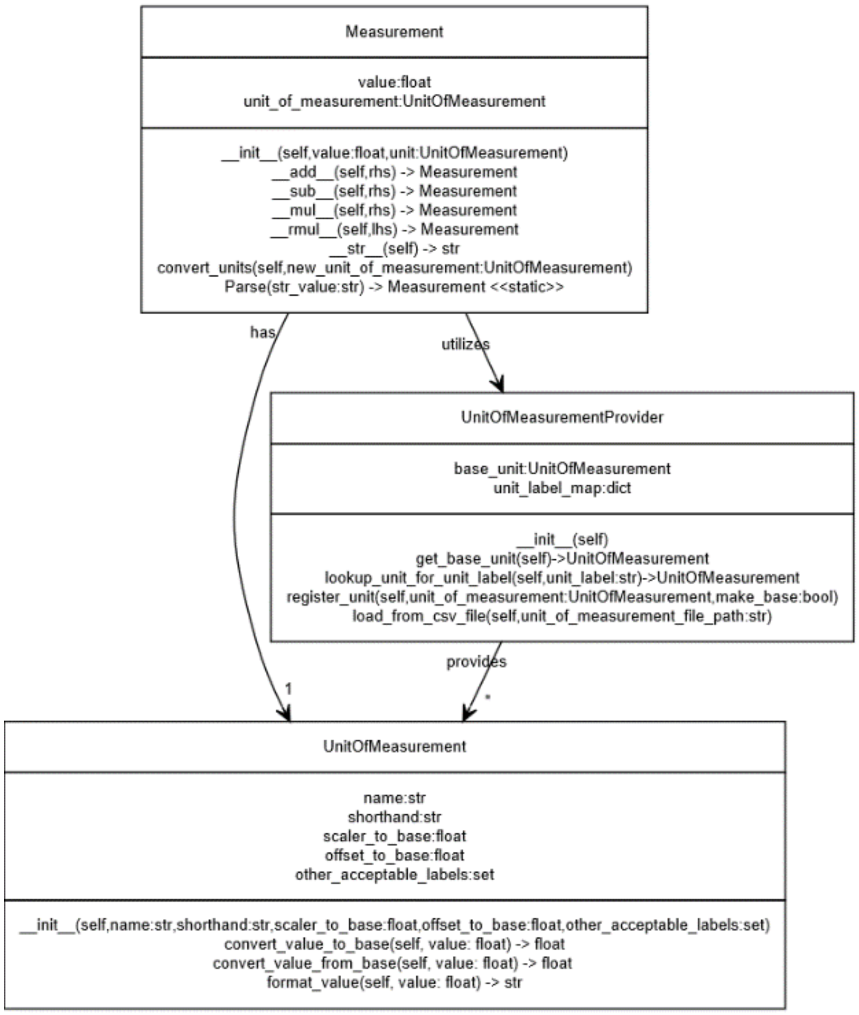
## A. UML diagram of classes & their interactions

Here is a UML Class Diagram of the classes within this module.

The lines between the classes show important associations, discussed above, between the classes.

Note that no inheritance association is depicted.

The diagram below uses "-> type" syntax at the end of the methods to specify the type of object that should be returned.


Measurement
value:float
unit_of_measurement:UnitOfMeasurement
__init__(self,value:float,unit:UnitOfMeasurement)
__add__(self,rhs) -> Measurement
__sub__(self,rhs) -> Measurement
__mul__(self,rhs) -> Measurement
__rmul__(self,lhs) -> Measurement
__str__(self) -> str
convert_units(self,new_unit_of_measurement:UnitOfMeasurement)
Parse(str_value:str) -> Measurement <<static>>
has
utilizes
UnitOfMeasurementProvider
base_unit:UnitOfMeasurement
unit_label_map:dict
__init__(self)
get_base_unit(self)->UnitOfMeasurement
lookup_unit_for_unit_label(self,unit_label:str)->UnitOfMeasurement
register_unit(self,unit_of_measurement:UnitOfMeasurement,make_base:bool)
load_from_csv_file(self,unit_of_measurement_file_path:str)
provides
1
*
UnitOfMeasurement
name:str
shorthand:str
scaler_to_base:float
offset_to_base:float
other_acceptable_labels:set
__init__(self,name:str,shorthand:str,scaler_to_base:float,offset_to_base:float,other_acceptable_labels:set)
convert_value_to_base(self, value: float) -> float
convert_value_from_base(self, value: float) -> float
format_value(self, value: float) -> str

**B. Overview information for each class**

The **UnitOfMeasurement** class provides an object template for unit of measurements such as ‘feet’ and ‘meters’.

- This class and instance attributes for each instance encapsulate the logic and the data to transform values to-and-from to a base unit of measurement.
- We can use this class to represent these unit-of-measurements as objects in the program.
- This class provides object attributes to hold text descriptions (*name, shorthand, other_acceptable_label*s) and attributes used in the logic to convert values to-and-from a base unit of measurement (*scaler_to_base, offset_to_base*).
- The base unit of measurement is tracked by an instance of the **UnitOfMeasurementProvider** class.

The **UnitOfMeasurementProvider** class provides an object template for objects that keep track of **UnitOfMeasurement** objects in a dictionary object and the base unit of measurement that of all these **UnitOfMeasurement** objects know how to convert values to-and-from.

- An instance of **UnitOfMeasurementProvider** can hold multiple **UnitOfMeasurement** instances
- These instances know how to convert measurement values to-and-from the base unit of measurement; this base unit of measurement is tracked with the *base_unit* attribute.
- The *register_unit* method registers **UnitOfMeasurement** objects so that other code could use the *lookup_unit_for_unit_label* method to find these **UnitOfMeasurement** objects with their text representations.
- If *register_unit* method’s *make_base* parameter is True, then the **UnitOfMeasurement** object that is being registered should assign the *base_unit* attribute to the unit-of-measurement passed in as well.

Note that the **Measurement** class refers to an instance of the **UnitOfMeasurementProvider** class created within the module.

- The **Measurement** class uses this instance to lookup the **UnitOfMeasurement** objects that correspond to a text/string description of the unit-of-measurements within its static *Parse* method.
- **Measurement** can access an instance of the **UnitOfMeasurementProvider**.
- In this way, **Measurement** instance methods can convert **Measurement** value attributes to the base unit-of-measurement and to another unit-of-measurement tracked in the **UnitOfMeasurementProvider** instance.

*DASC511 Week 4 Classes Assignment*

**C. Detailed information on the assigned attributes & methods within each class**

1. Class **UnitOfMeasurement**:

    a. **__init__** method is used to initialize UnitOfMeasurement objects

    b. **convert_value_to_base** method converts a number representing from the unit-of-measurement in the instance self to the base unit-of-measurement

    c. **convert_value_from_base** method converts a number from the base unit-of-measurement to the equivalent number in the unit-of-measurement of self

    d. **format_value** method converts a number representing the unit-of-measurement in the instance self to a string text representation of this number. For example, if the *float* object 3.0 was passed into this method and the unit of measurement represents feet, then the returned string should be "3.0 ft".

2. Class **UnitOfMeasurementProvider**:

    **base_unit** attribute represents the base unit of measurement object of type UnitOfMeasurement that all of the UnitOfMeasurements objects provided by this class use as conversion base

    **unit_label_map** attribute represents a mapping object of type dict that uses str keys to point to the appropriate UnitOfMeasurement object as the key's value

    a. **__init__** method is used to initialize **UnitOfMeasurementProvider** objects:

    b. **get_base_unit** method returns *self*'s *base_unit* attribute.

    c. **lookup_unit_for_unit_label** method uses the passed in string object as the key to retrieve the value from the *unit_label_map* key-value dictionary.

    d. **register_unit** method assign the *base_unit* attribute the *unit_of_measurement* parameter if the passed in *make_base* parameter is True. This method also adds the passed in *unit_of_measurement* object to the *unit_label_map* dictionary attribute. The *unit_of_measurement* object represents the value to every appropriate key-value addition to this dictionary. It uses the *name* attribute, the *shorthand* attribute, and every string in the *other_acceptable_labels* attribute as keys. After this method is called for a unit of measurement, the *lookup_unit_for_unit_label* can be used to retrieve the *unit_of_measurement* object by using any one of the string representations of the unit.

e. **load_from_csv_file** method opens the file specified by the file path parameter. It assumes the file is a CSV formatted file with a header line. See the provided text file with the homework for the file format. It parses this file, creating *UnitOfMeasurement* objects from each line's contents, and registering every *UnitOfMeasurement* object by calling the *register_unit* method on *self*. The first unit of measurement in the file is used as the *base_unit*.

3. Within the module a variable named **unit_of_measurement_provider** is created and assigned a new *UnitOfMeasurementProvider* object

4. The method **load_from_csv_file** on the *unit_of_measurement_provider* object is called, passing in the argument "length_measurements.csv" so that the CSV file contents, representing units of measurements and provided with the homework assignment, is loaded into the *unit_of_measurement_provider* object.

5. Class **Measurement**

   a. **__init__** method initializes **Measurement** objects

   b. **__add__** method overrides the addition operator, '+'. Assumes the *rhs* parameter is of type *Measurement*. Uses the unit_of_measurement object's conversion methods on *self* and *rhs* to appropriately convert the *value* attributes of *self* and *rhs*. Then adds these converted *float* values and creates a new *Measurement* object with the appropriate unit of measurement holding this new value.

   c. **__sub__** method overrides the subtraction operator '-'. Assumes the *rhs* parameter is of type *Measurement*. Use the *unit_of_measurement* attribute's conversion methods to appropriately convert the *value* attributes of *self* and *rhs*. Then subtracts the *rhs* converted value from the *self* converted value and creates a new *Measurement* object with the appropriate unit of measurement holding this new value.

   d. **__mul__** method overrides the multiplication operator, '*'. Assumes the *rhs* parameter is of type *int* or *float*. Multiplies the *value* attribute of *self* with *rhs* and creates a new *Measurement* object with the appropriate unit of measurement holding this new value.

   e. **__rmul__** method overrides the multiplication operator when objects of type *Measurement* are on the right-hand-side (rhs) of the '*' operator. Assumes the *lhs* parameter is of type *int* or *float*. Multiplies the *value* attribute of *self* with *lhs* and creates a new *Measurement* object with the appropriate unit of measurement holding this new value.

f. **__str__** method overrides the default string representation conversion of the *Measurement* objects, such as when these objects are printed to the terminal using the *print* command. Returns the string result of the *unit_of_measurement*'s *format_value* method when passed *self*'s *value* attribute.

g. **convert_units** method converts the *value* attribute from the starting value to the equivalent value of the passed in *new_unit_of_measurement* parameter. Assumes *new_unit_of_measurement* parameter is an object of type *UnitOfMeasurement.* After changing the *value* attribute, reassigns *self*'s *unit_of_measurement* attribute to the *new_unit_of_measurement* object.

h. **Parse** static method converts a string representation of a measurement and creates the equivalent *Measurement* object. The equivalent *Measurement* object is returned. This method utilizes the *unit_of_measurement_provider* object in the global scope to lookup the appropriate *UnitOfMeasurement* object.

*DASC511 Week 4 Classes Assignment*

**D. Examples of calling the methods within the classes**

- Assigning a variable **yard** as the returned value of the *Measurement.Parse* static method when passed “3 ft”

- Looking up a unit and converting

    - Using the *unit_of_measurement_provider* object to lookup the *UnitOfMeasurement* object representing meters and assigning a variable **meters_unit** to this *UnitOfMeasurement* object.

    - Then, assigning a variable **yard** as the returned value of the *Measurement.Parse* static method when passed “3 ft”.  Then calls the *convert_units* method passing in the *UnitOfMeasurement* object representing meters, *meters_unit*.