```
classdef Radiometer < handle & matlab.mixin.CustomDisplay
    %Radiometer Class for Dexter Radiometer connected to ADS1115
    %{
    The constructor sets up the I2C device object. The standard address is
    0x48, but can be set to 0x49,0X4A,or 0x4B. For Radiometer, make sure
    address is 0x48 or 0x49 and for thermistors it is  0x4A or0x4B.

    Default pin and voltage scale are 0 and 0.256. To change simply add in
    constructor.

    The Read method requires the input of 2 thermistor classes to help
    compensate for cold junction and ambient heat. You can also add a
    correction for viewing angle(in degrees),  but if you do not enter one it is
    assumed to be 0 degrees.

    F is in mW/cm^2 and L is in mW/(cm^2*sr)

    n,F1,F2,and F3 are constants that depend on geometry and radiative
    properties of each surface involved. A 4 point calibration will be done
    with a blackbody in Professor Miller's Lab to determine their values.
    %}

    properties
        PINS = [0,1];
        V_SCALE = 0.256;
        Address = hex2dec('0x48');
        AR
        Rad
        theta = 0;
        n = 1;
        F1 = 1;
        F2 = 1;
        F3 = 1;
        Res = 1;
        sr = 1;
    end

    methods
        function obj = Radiometer(arduino,addrs,v_scale,pins)
            %UNTITLED4 Construct an instance of this class
            %   Detailed explanation goes here
            switch nargin
                case 2
                    obj.Address = addrs;
                case 3
                    obj.V_SCALE = v_scale;
                    obj.Address = addrs;
                case 4
                    obj.V_SCALE = vscale;
                    obj.Address = addrs;
                    obj.PINS = pins;
```

```
            end

            obj.AR = arduino;
            obj.Rad = ads1115(obj.AR, obj.Address);

            obj.Rad.VoltageScale = obj.V_SCALE;
        end

        function [L,F] = Read(obj,thermistor_1,thermistor_2,theta)
            %Requires 2 thermistor objects as input,can factor in viewing angle
            switch nargin
                case 4
                    obj.theta = theta;
            end

            v = readVoltage(obj.Rad,obj.PINS(1),obj.PINS(2));

            T_d = thermistor_1.Read();
            T_opt = thermistor_2.Read();


            L = ((v/obj.Res) - (obj.F1 * (T_opt^obj.n)) + (obj.F2 * (T_d^obj.n))) * (obj.↙
F3/cosd(theta));
            F = L * obj.sr;
        end
    end
    methods
        %This simply allows it to take hex or numeric input
        function set.Address(obj, value)
            if isnumeric(value)
                validateattributes(value, {'numeric'}, ...
                    {'scalar', 'nonnegative'}, '', 'Address');
            else
                validateattributes(value, {'char'}, ...
                    {'nonempty'}, '', 'Address');
                value = obj.hex2dec(value);
            end
            if (value < obj.hex2dec('0x48')) || (value > obj.hex2dec('0x51'))
                error('raspi:ads1115:InvalidI2CAddress', ...
                    'Invalid I2C address. I2C address must be one of the following: 0x48,↙
0x49, 0x50, 0x51');
            end
            obj.Address = value;
        end
    end
end
```