```
classdef InjectionObj < handle
%InjectionObj. Injection Object
% InjectionObj represents a injection point into the well structure.
 It
% includes viscosity blending calculations according Cragoe and ASTM
 D7152.
%
%   Properties
%    - Qd             : flowrate diluent @local conditions  [m3/d]
%    - Qo             : flowrate oil     @local conditions  [m3/d]
%    - Qw             : flowrate water   @local conditions  [m3/d]
%    - Tb             : blending temperature                [C]
%    - drho           : density diluent                     [kg/m3]
%    - orho           : density oil                         [kg/m3]
%    - mrT            : reference matrix temperature        [C]
%    - mrmu           : reference matrix viscosity          [cP or
 cSt]
%    - BlendingMethod : blending method
%    - CragoeA        : Cragoe's method A coefficient
%    - CragoeB        : Cragoe's method B coefficient
%    - bmrmu          :
%    - bmrT           :
%
%   Dependent properties
%    - status         : programming property (to be used in further
 versions)
%    - brho           : density blend                       [kg/m3]
%    - bmu            : viscosity blend                     [cP]
%    - bSG            : specific gravity blend              [-]
%    - WC             : water cut                           [-]
%
%   Methods
%    - BlendingReference
%        calculation of two referencial matrices including viscosity
 at 5
%        different temperature values.
%
 ------------------------------------------------------------------------
% Development
%   By: Ruben Ensalzado
%       2015
%   Rev 00 151216 original release
%   Rev 01 151230 description change: Qd, Qo, Qw
%                 inclusion property: bmrmu, bmrT
%                 inclusion method: BlendingReference

% Developer notes:
% - mrmu : row orientation per reference
% - mrT  : row orientation per reference

    properties
        Qo
```

```
        Qd
        Qw
        Tb
        mrT
        mrmu
        bmrmu
        bmrT
        orho
        drho
        BlendingMethod
        CragoeA
        CragoeB
    end

    properties (Dependent = true)
        status
        bmu
        brho
        bSG
        WC
        wt
    end

    properties (Hidden = true, Constant = true)
        wrho = 1000;        %P = 1 atm, T = 4 C
    end

    methods (Hidden = true)
        % -- initialization method --

        function BL = InjectionObj
            BL.BlendingMethod = 'astm_d7152';
            BL.CragoeA = 5e-2;
            BL.CragoeB = 1e3*log(20);
        end
    end

    methods
        % -- error verification methods --

        function set.BlendingMethod(BL, umethod)
            if strcmpi(umethod, 'cragoe') ||
strcmpi(umethod, 'astm_d7152')
                BL.BlendingMethod = umethod;
            else
                error('InjectionObj:BadArgument', ...
                      'The available methods are: CRAGOE and
ASTM_D7152!')
            end
        end

        function set.mrmu(BL, umrmu)
            if size(umrmu, 2) == 2
                BL.mrmu = umrmu;
```

```
            else
                error('InjectionObj:BadArgument', ...
                      'mrmu must be a matrix nx2, where n is the
number of fluids!')
            end
        end

        function set.mrT(BL, umrT)
            if size(umrT, 2) == 2
                BL.mrT = umrT;
            else
                error('InjectionObj:BadArgument', ...
                      'mrT must be a matrix nx2, including the
reference temperature value!')
            end
        end

        function set.Tb(BL, uTb)
            if size(uTb, 1) == 1 && size(uTb, 2) == 1
                BL.Tb = uTb;
            else
                error('InjectionObj:BadArgument', ...
                      'Tb must be a single temperature value!')
            end
        end
    end

        methods
            % -- dependent properties --

        function status = get.status(BL)
            %Status. Independent property verification

            req = zeros(4, 1);

            req(1) = isempty(BL.wt);
            req(2) = isempty(BL.mrT);
            req(3) = isempty(BL.Tb);
            req(4) = isempty(BL.mrmu);

            if all(~req)
                status = true;
            else
                status = false;
            end
        end

        function bmu = get.bmu(BL)
            %DynamicViscosity. (blend)
            %Units
            % bmu : cP

            switch lower(BL.BlendingMethod)
                case 'cragoe'
```

```
                bmu = BlendingCragoe(BL);
            case 'astm_d7152'
                bmu = BlendingASTM_D7152(BL);
        end
    end

    function WC = get.WC(BL)
        %WaterCut.
        %Units
        % Qw : m3/d
        % Qo : m3/d
        % Qd : m3/d

        WC = (BL.Qw)/(BL.Qo + BL.Qd + BL.Qw);
    end

    function bSG = get.bSG(BL)
        %SpecificGravity. (blend) Definition
        %Units
        % brho : kg/m3
        % wtho : kg/m3

        bSG = BL.brho/BL.wrho;
    end

    function brho = get.brho(BL)
        %Density. (blend) Mass balance
        %Units
        % drho : kg/m3
        % orho : kg/m3
        % Qo   : m3/d
        % Qd   : m3/d
        % brho : kg/m3

        brho = (BL.orho*BL.Qo + BL.drho*BL.Qd)/(BL.Qo + BL.Qd);
    end

    function wt = get.wt(BL)
        %WeightFraction.
        %Units
        % drho : kg/m3
        % orho : kg/m3
        % Qo   : m3/d
        % Qd   : m3/d

        wt = zeros(2, 1);
        wt(1) = BL.orho*BL.Qo/(BL.orho*BL.Qo + BL.drho*BL.Qd);
        wt(2) = 1 - wt(1);
    end

end

methods
    function BlendingReference(BL)
```

```
            %BlendingReference. Reference for blend viscosity
            %Units
            % Tb    : C
            % bmrT  : C
            % bmrmu : cP

            T0 = BL.Tb;
            BL.bmrmu = zeros(5, 1);
            BL.bmrT = linspace(T0 - 10, T0 + 10, 5)';

            for i = 1:5
                BL.Tb = BL.bmrT(i);
                BL.bmrmu(i) = BL.bmu;
            end

            BL.Tb = T0;
        end

    end


    methods (Access = protected, Hidden = true)
        % -- viscosity calculation methods --

        function bmu = BlendingCragoe(BL)
            %BlendingCragoe. Blending according Cragoe's method (1933)
            %Units
            % vscm : cP
            % bnu  : cP
            % T    : C

            A = BL.CragoeA;
            B = BL.CragoeB;

            vscm = BL.mrmu;
            lij = B./log(vscm/A);
            lbm = lij(:, 1) + (BL.Tb - BL.mrT(:, 1)).*(lij(:, 2) -
lij(:, 1))./...
                                (BL.mrT(:, 2) - BL.mrT(:, 1));
            lmx = sum(BL.wt.*lbm);
            bmu = A*exp(B/lmx);
        end

        function bnu = BlendingASTM_D7152(BL)
            %BlendingASTM_D7152. Blending according ASTM D7152 (2011)
            %Units
            % vscm : cSt
            % bmu  : cSt
            % T    : C

            vscm = BL.mrmu;
            zij = vscm + 0.7 + exp(-1.47 - 1.84*vscm - 0.51*vscm.^2);
            wij = log10(log10(zij));
            Tij = log10(BL.mrT + 273.15);
```

```
            Ttb = log10(BL.Tb + 273.15);
            mnv = (Tij(:, 2) - Tij(:, 1))./(wij(:, 2) - wij(:, 1));

            wb = (Ttb + sum(BL.wt.*(mnv.*wij(:, 1)-Tij(:, 1))))/
(sum(BL.wt.*mnv));
            zb = 10^(10^wb) - 0.7;

            bnu = zb - exp(-0.7487 - 3.295*zb + 0.6119*zb^2 -
 0.3191*zb^3);
        end
    end
end
```

Published with MATLAB® R2015a