Class Reference Template

This template provides a standardized format for documenting TaesLab class methods in markdown format.

[ClassName] Reference Guide

Version: 1.8 (R2024b) 01-Oct-2025

Class Overview

[Brief description of the class purpose and main functionality]

Inheritance

- **Parent Class**: ParentClass
- **Child Classes**: ChildClass1, ChildClass2

Key Features

- Feature 1: [Description]
- Feature 2: [Description]
- Feature 3: [Description]

Class Properties

Property Name	Type	Access	Description
Property1	`type`	Public	[Description]
Property2	`type`	Private	[Description]
Property3	`type`	Protected	[Description]

Constructor

[ClassName]

Create an instance of the [ClassName] class.

Syntax:

```
obj = ClassName(arg1, arg2)
obj = ClassName(arg1, 'Parameter', value)
```

Input Arguments:

- `arg1` — [Description]
  Data type: `type` | *Values*: [valid values]
- `arg2` — [Description]
  Data type: `type` | *Values*: [valid values]
- `'Parameter'` — [Description] *(optional)*
  Data type: `type` | *Default*: `default_value`

Output Arguments:

- `obj` — [ClassName] object
  Data type: `ClassName`

Examples:

```
% Basic usage
obj = ClassName(data);

% With optional parameters
obj = ClassName(data, 'Parameter', value);

% Multiple parameters
obj = ClassName(data, 'Param1', value1, 'Param2', value2);
```

Set Methods

setMethod1

[Description of what this setter method does]

Syntax:

```
obj.setMethod1(value)
obj.Property = value  % Alternative syntax
```

Input Arguments:

- `value` — [Description]
  Data type: `type` | *Values*: [valid values]

Examples:

```
% Direct method call
obj.setMethod1('newValue');

% Property assignment
obj.Property = 'newValue';
```

setMethod2

[Description of what this setter method does]

Syntax:

```
obj.setMethod2(value1, value2)
```

Input Arguments:

- `value1` — [Description]
  Data type: `type` | *Values*: [valid values]
- `value2` — [Description]
  Data type: `type` | *Values*: [valid values]

Examples:

```
obj.setMethod2(param1, param2);
```

Get Methods

getMethod1

[Description of what this getter method returns]

Syntax:

```
result = obj.getMethod1()
result = obj.getMethod1(parameter)
```

Input Arguments:

- `parameter` — [Description] *(optional)*
  Data type: `type` | *Default*: `default_value`

Output Arguments:

- `result` — [Description]
  Data type: `type`

Examples:

```
% Basic usage
result = obj.getMethod1();

% With parameter
result = obj.getMethod1('parameter');
```

Analysis Methods

analysisMethod1

[Description of the analysis performed by this method]

Syntax:

```
results = obj.analysisMethod1()
results = obj.analysisMethod1('Parameter', value)
```

Input Arguments:

- `'Parameter'` — [Description] *(optional)*
  Data type: `type` | *Values*: [valid values] | *Default*: `default`

Output Arguments:

- `results` — [Description]
  Data type: `cResultInfo`

Examples:

```
% Basic analysis
results = obj.analysisMethod1();

% With parameters
results = obj.analysisMethod1('ShowResults', true, 'SaveAs', 'filename.xlsx');
```

analysisMethod2

[Description of the analysis performed by this method]

Syntax:

```
results = obj.analysisMethod2(state)
results = obj.analysisMethod2(state, 'Parameter', value)
```

Input Arguments:

- `state` — [Description]
  Data type: `char` | *Values*: [valid state names]
- `'Parameter'` — [Description] *(optional)*
  Data type: `type` | *Default*: `default`

Output Arguments:

- `results` — [Description]
  Data type: `cResultInfo`

Examples:

```
% Analysis for specific state
results = obj.analysisMethod2('design');

% With optional parameters
results = obj.analysisMethod2('design', 'Method', 'advanced');
```

Validation Methods

isValidMethod1

[Description of what this validation method checks]

Syntax:

```
isValid = obj.isValidMethod1()
isValid = obj.isValidMethod1(parameter)
```

Input Arguments:

- `parameter` — [Description] *(optional)*
  Data type: `type`

Output Arguments:

- `isValid` — [Description]
  Data type: `logical`

Examples:

```
% Check validity
if obj.isValidMethod1()
    % Proceed with operation
end

% Check with parameter
isValid = obj.isValidMethod1(parameter);
```

Display Methods

showMethod1

[Description of what this display method shows]

Syntax:

```
obj.showMethod1()
obj.showMethod1('Format', format)
```

Input Arguments:

- `'Format'` — Display format *(optional)*
  Data type: `char` | *Values*: `'console'` | `'table'` | `'html'` | *Default*: `'console'`

Examples:

```
% Show in console
obj.showMethod1();

% Show in HTML format
obj.showMethod1('Format', 'html');
```

Save/Export Methods

saveMethod1

[Description of what this save method exports]

Syntax:

```
obj.saveMethod1(filename)
obj.saveMethod1(filename, 'Format', format)
```

Input Arguments:

- `filename` — Output filename
  Data type: `char`
- `'Format'` — File format *(optional)*
  Data type: `char` | *Values*: `'xlsx'` | `'csv'` | `'mat'` | *Default*: Auto-detect from extension

Examples:

```
% Save as Excel file
obj.saveMethod1('results.xlsx');

% Save with specific format
obj.saveMethod1('results', 'Format', 'csv');
```

Utility Methods

utilityMethod1

[Description of what this utility method does]

Syntax:

```
result = obj.utilityMethod1(input)
result = obj.utilityMethod1(input, 'Option', value)
```

Input Arguments:

- `input` — [Description]
  Data type: `type`
- `'Option'` — [Description] *(optional)*
  Data type: `type` | *Default*: `default`

Output Arguments:

- `result` — [Description]
  Data type: `type`

Examples:

```
result = obj.utilityMethod1(input);
result = obj.utilityMethod1(input, 'Option', value);
```

Examples

Basic Usage Example

```
% Create class instance
data = ReadDataModel('model.json');
obj = ClassName(data);

% Set properties
obj.setMethod1('value');

% Perform analysis
results = obj.analysisMethod1();

% Display results
obj.showMethod1('Format', 'table');

% Save results
obj.saveMethod1('output.xlsx');
```

Advanced Usage Example

```
% Create with parameters
obj = ClassName(data, 'Parameter1', value1, 'Parameter2', value2);

% Configure analysis
obj.setMethod1('advanced');
obj.setMethod2(param1, param2);

% Run comprehensive analysis
if obj.isValidMethod1()
    results1 = obj.analysisMethod1('ShowResults', false);
    results2 = obj.analysisMethod2('design', 'Method', 'detailed');

    % Compare results
    summary = obj.utilityMethod1(results1);

    % Export everything
    obj.saveMethod1('comprehensive_results.xlsx', 'Format', 'xlsx');
end
```

Notes

- **Performance**: [Any performance considerations]
- **Dependencies**: Requires [dependencies]
- **Compatibility**: MATLAB R2019b or later, Octave 6.0+
- **See Also**: RelatedClass1, RelatedFunction1, RelatedMethod1

Reference Links