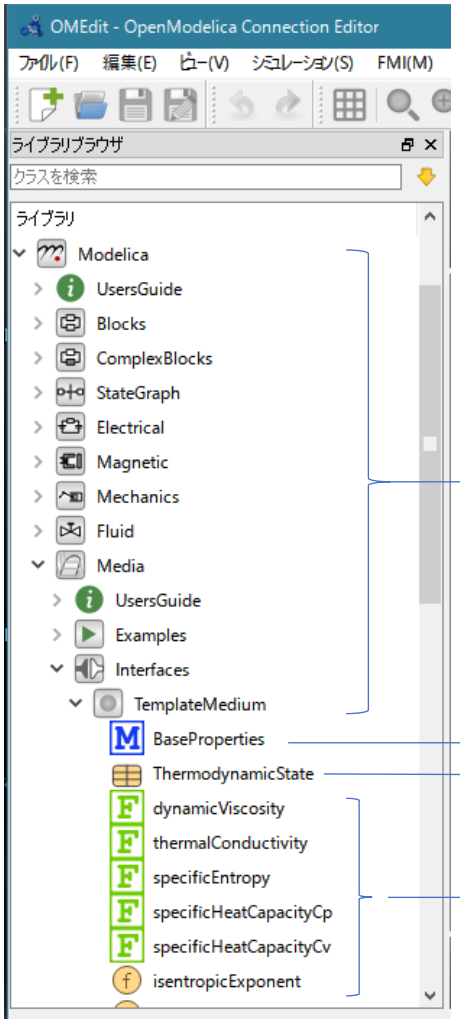
OpenModelica講習中級 Modelica.Fluidライブラリ解説

1. Modelicaのクラスの概要

2017年12月7日　田中 周(有限会社アマネ流研)

1. Modelicaのクラスの概要



OpenModelica https://www.openmodelica.org/
OMEdit のライブラリブラウザ

ライブラリはクラス
によって構成されている。

package

model
record

function

クラス

Modelica のクラス
① package
② type
③ class
④ record
⑤ model
⑥ function
⑦ connector
⑧ block

Examples

まず、ライブラリを構成するクラスの特徴と使い方について概観する。

ClassExample1 簡単なシミュレーションモデルを作ってみる
ClassExample2 record を使って変数をグループ化する
ClassExample3 block を使ってモデル化する
ClassExample4 connector 付き model を使ってモデル化する
ClassExample5 クラスを継承してみる
ClassExample6 ローカルクラスを作ってみる
ClassExample7 交換可能なローカルクラスを作ってみる
ClassExample8 Modelica.Media を使って空気の物性値を調べる
ClassExample9 Modelica.Media を使って室温の変化を調べる

ClassExample1
簡単なシミュレーションモデルを作ってみる

クラスの宣言方法は２種類ある

A. 既存クラスを代入するような宣言方法

class クラス名 = 元になるクラス名(...)

type, connector,...

```
type A = B(... );
```

B. 構成要素を列記する宣言方法

class クラス名と end クラス名の間に構成要素を列記する。

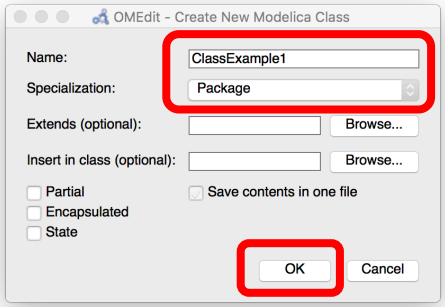
class, package, model,
block, connector,...

```
package C
  ...
end C;
```

```
class D
  ...
end D;
```

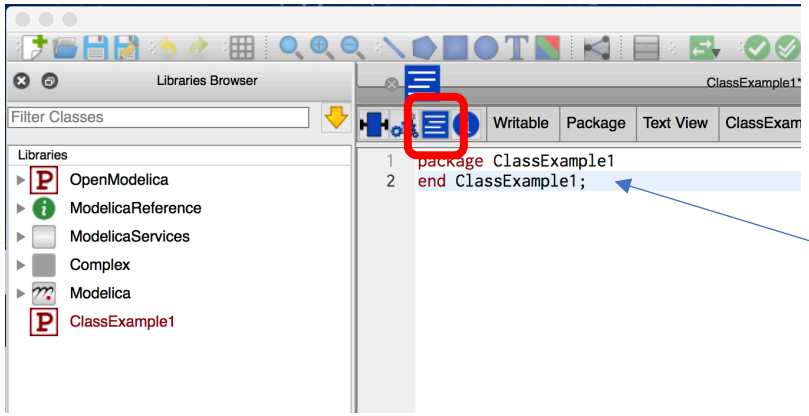
ClassExample1

1. File > New Modelica Class（ファイル＞Modelicaクラス新規作成）



名前: ClassExample1
クラス・タイプ: Package

2. Text View に切り替える。



3. コードを入力する。

ClassExample1

```
package ClassExample1 // package    (1)
  constant Acceleration g = -9.8;

  // type
  type Acceleration = Real(quantity = "Acceleration", unit = "m/s2");   (2)
  type Mass = Real(quantity = "Mass", unit = "kg");                     (3)
  type Force = Real(quantity = "Force", unit = "N");                    (4)
  type Velocity = Real(quantity = "Velocity", unit = "m/s");            (5)
  type Position = Real(quantity = "Length", unit = "m");                (6)

  // class
  class MassA   (7)
    parameter Mass m = 1.0;
    parameter Force f = m*g;
    parameter Velocity v0 = 5.0;
    parameter Position x0 = 0.0;
    Position x(start = x0);
    Velocity v(start = v0);
  equation
    v = der(x);
    f = m*der(v);
  end MassA;

end ClassExample1;
```

自由落下する質点のモデル

（Modelica.SIunitsにあるものを説明のために再宣言しています。）

パラメータ

m：質量，f：外力，

x_0, v_0：初期条件

変数　x：変位，v：速度

方程式

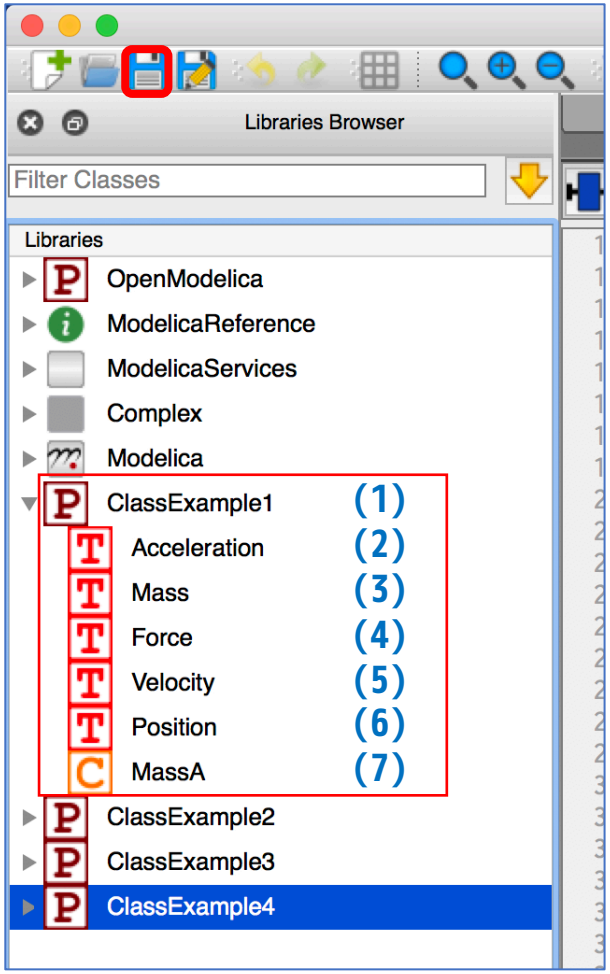
$$v = \frac{dx}{dt}, f = m\frac{dv}{dt}$$ ：質点の運動方程式

構成要素

(1)～(7)のクラスを作成した。

ClassExample1

4. 保存する。

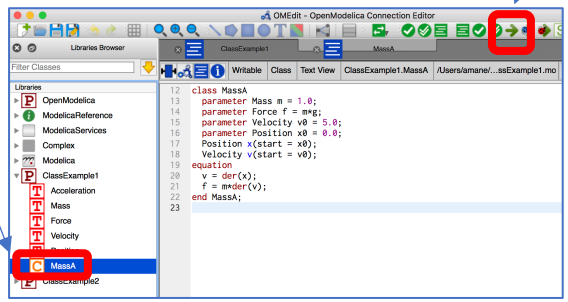


ライブラリブラウザ

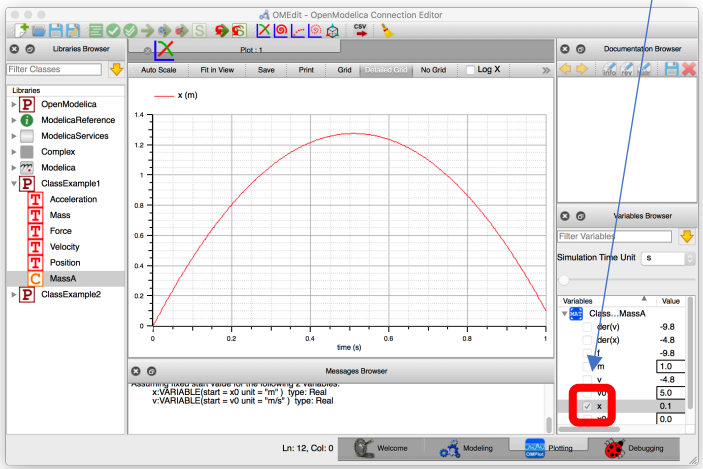
5. 実行する。

i. MassAを
ダブルクリック

ii. Simulate(シミュレート)
をクリックする。

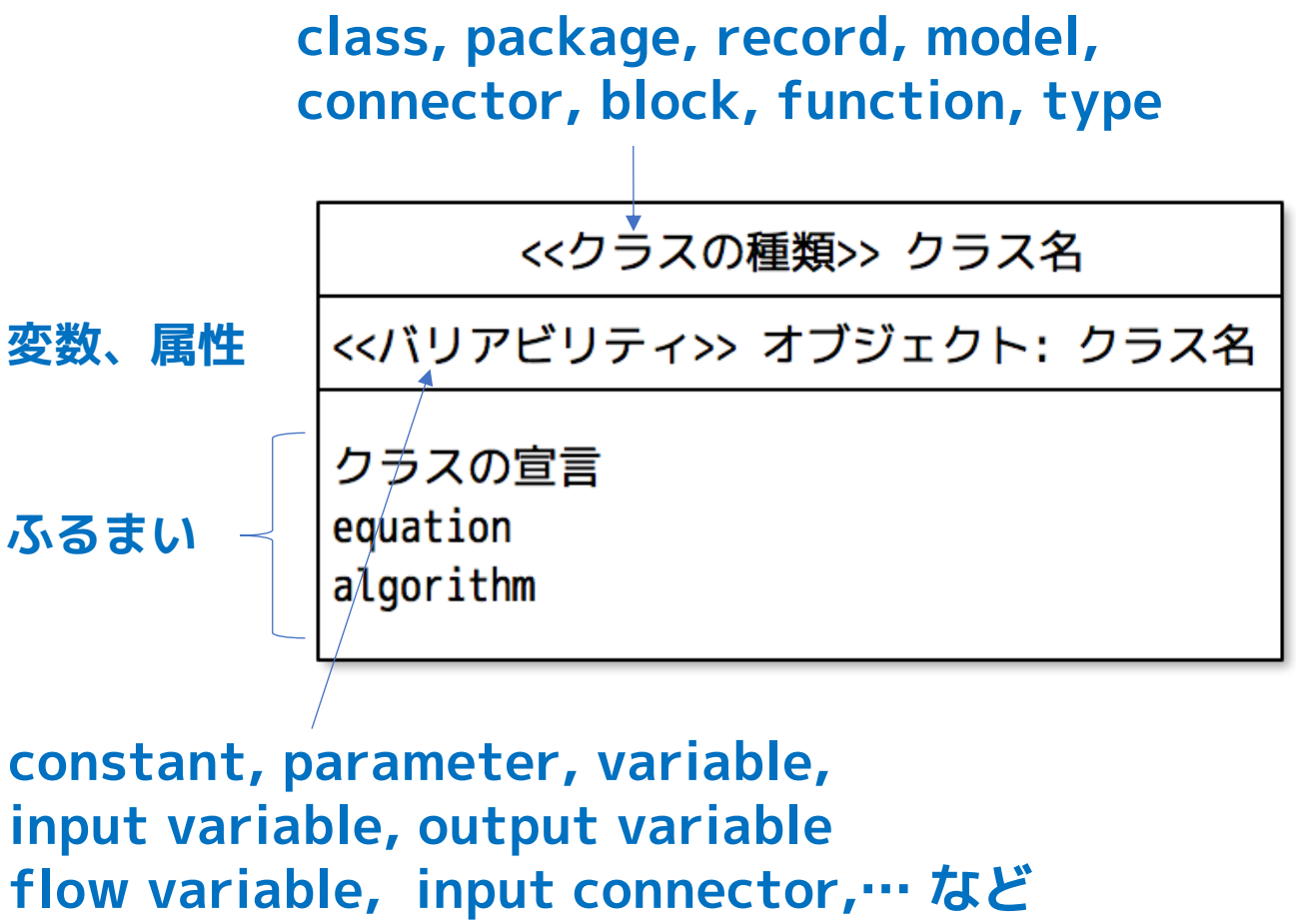


iii. xをチェックする。



ClassExample1

Modelica のクラスの特徴を整理するため UML 的に表現し、作成したクラスの特徴を概観する。



ClassExample1

① package (1)

- 内部にクラスの宣言と定数の定義のみが記述できる。

<<package>> ClassExample1
<<constant>> g: Acceleration
type Acceleration type Mass type Force type Velocity type Position class MassA

定数

クラスの宣言

② type (2)〜(6)

- predefined type とその配列、type を継承したクラスからのみ宣言できる。

```
type A = B(... );
```

predefined type

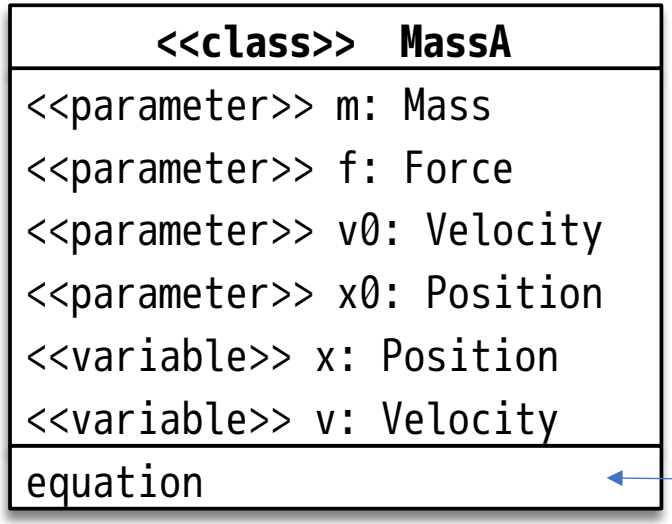
predefined type (built-in class)

- **Real**
- **Integer**
- **Boolean**
- **String**
- **enumeration(…)**

ClassExample1

③ class（汎用クラス）(7)

- 制限や強化機能のない汎用的なクラスを表す。model と同じものである。
- ほとんどの場合、class を使用せずに model や block など、特定の用途に特化されたクラスを使用することが推奨されている。



パラメータ、変数
などを表すオブジェクト

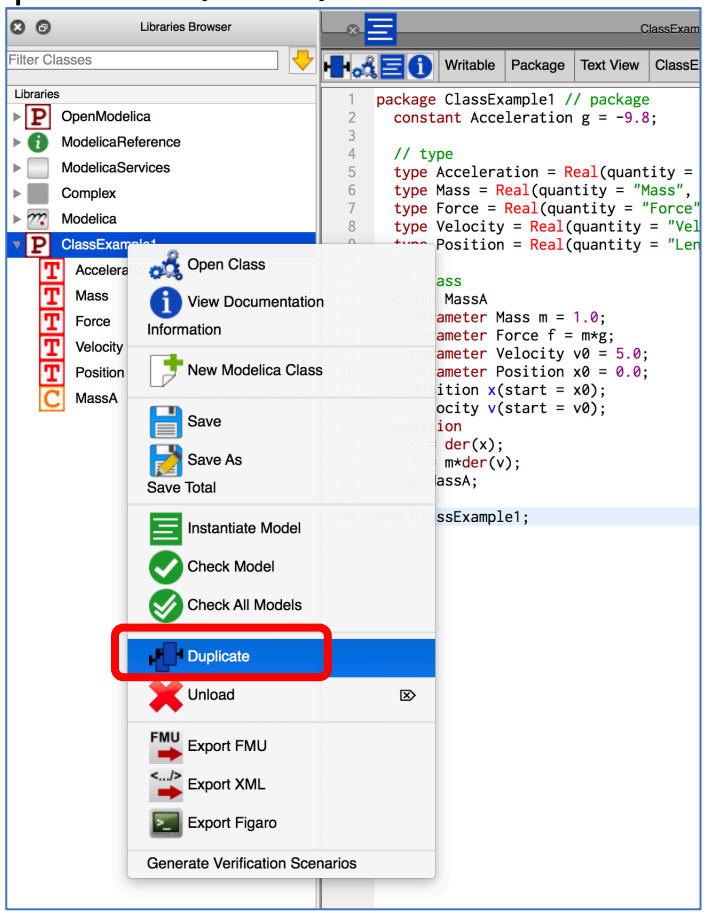
方程式またはアルゴリズム、
クラスの宣言

シミュレーションモデルを記述することも可能

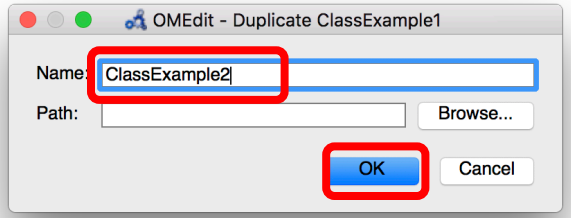
ClassExamle2

record を使って変数をグループ化する

1. ライブラリブラウザのClassExample1 を右クリックして Duplicate(複製)を選択する。



2. クラス名を入力する。



3. 次のスライドのように編集する。

ClassExample2

```
package ClassExample2 // package

  constant Acceleration g = -9.8;

  // type
  type Acceleration = Real(quantity = "Acceleration", unit = "m/s2");
  type Mass = Real(quantity = "Mass", unit = "kg");
  type Force = Real(quantity = "Force", unit = "N");
  type Velocity = Real(quantity = "Velocity", unit = "m/s");
  type Position = Real(quantity = "Length", unit = "m");

  // record
  record State
    Position x;
    Velocity v;
  end State;

  // model
  model MassB
    parameter Mass m = 1.0;
    parameter Force f = m * g;
    parameter State s0(x = 0, v = 5);
    State s(x(start = s0.x), v(start = s0.v));
  equation
    s.v = der(s.x);
    f = m * der(s.v);
  end MassB;

end ClassExample2;
```

質点の運動状態を表す
record

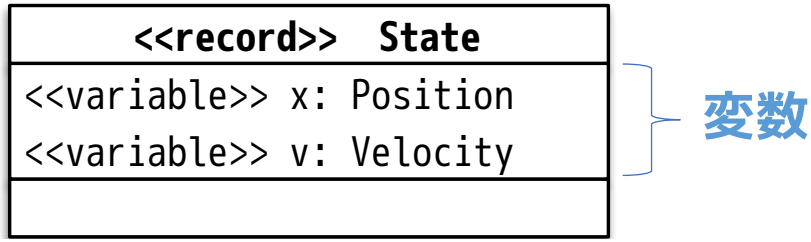
class の代わりに model にする。

Stateクラスのオブジェクト
state を状態変数にして
MassA を書き直す。

ClassExample2

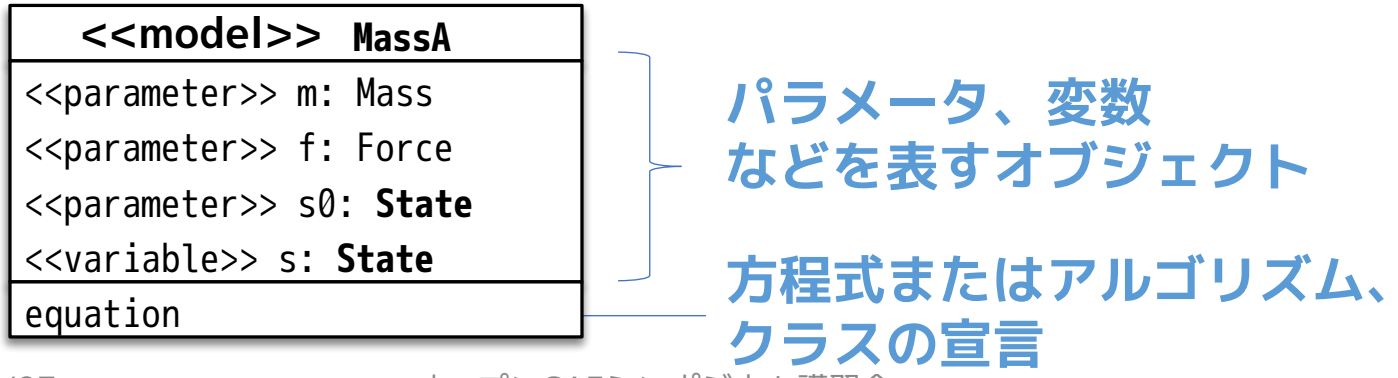
④ record

- 変数をグループ化するのに用いる。
- 変数の定義で、接頭語などを付けることができない。
- ふるまいに相当するものは記述しない。



⑤ model

- 主に パラメータ、変数、方程式などを実装してシミュレーションモデルを作成するのに使用する。
- equation または algorithm を記述する。
- 実質、class と同じもの。

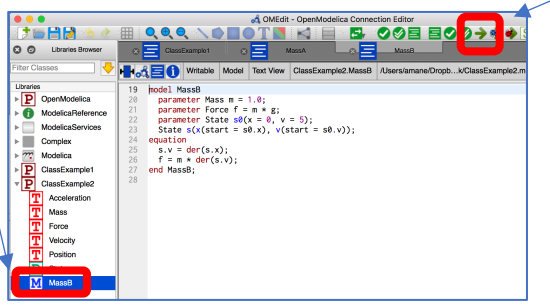


ClassExample2

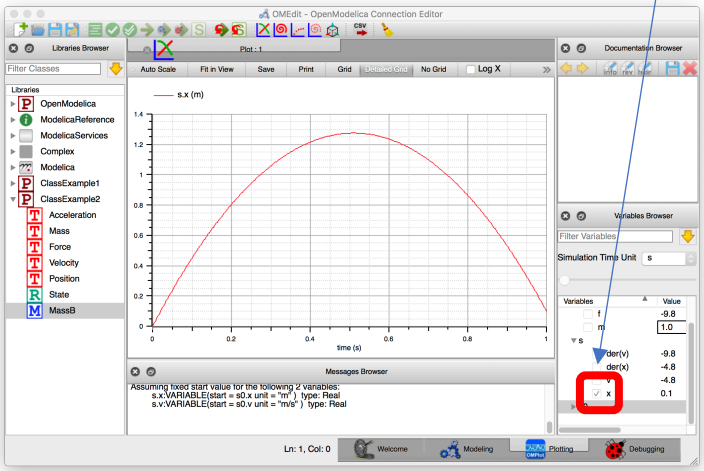
4. シミュレーションを実行する

i. MassBをダブルクリック

ii. Simulate(シミュレート)をクリックする。



iii. sを展開してxをチェックする。



ClassExamle3

block を使ってモデル化する

```
package ClassExample3  // package
  constant Acceleration g = -9.8;

  // type
  type Acceleration = Real(quantity = "Acceleration", unit = "m/s2");
  type Mass = Real(quantity = "Mass", unit = "kg");
  type Force = Real(quantity = "Force", unit = "N");
  type Velocity = Real(quantity = "Velocity", unit = "m/s");
  type Position = Real(quantity = "Length", unit = "m");
  type SpringConstant = Real(quantity = "Spring Constant", unit = "N/m");

  // record
  record State
    Position x;
    Velocity v;
  end State;

  // function
  function hookesLaw
    input SpringConstant k;
    input Position x;
    output Force f;
  algorithm
    f := -k*x;
  end hookesLaw;
```

ClassExample2 を
複製して編集する。

フックの法則を表す function（関数）

ClassExamle3

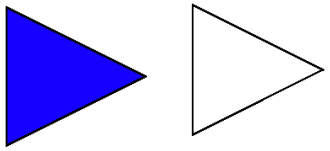
```
 // connector （Modelica.Block.Interfaces にあるものを再宣言してます。）
connector RealInput = input Real "'input Real' as connector" annotation( ...);
connector RealOutput = output Real "'output Real' as connector" annotation( ...);

 // block
 block MassBlock
    parameter Mass m = 1.0;
    parameter State s0(x = 0, v = 0);
    State s(x(start = s0.x), v(start = s0.v));
    RealInput f annotation( ...);
    RealOutput x annotation( ...);
 equation
    s.v = der(s.x);
    f = m * der(s.v);
    x = s.x;
    annotation( ...);
 end MassBlock;

 block SpringBlock
    parameter SpringConstant k = 1.0;
    RealInput x annotation( ...);
    RealOutput f annotation( ...);
 equation
    f = hookesLaw(k,x);
    annotation( ...);
 end SpringBlock;
```

運動方程式

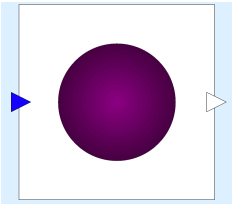
前のスライドの関数を使用！

RealInput: 実数の入力コネクタ

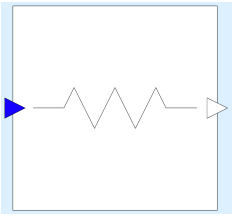
RealOutput:実数の出力コネクタ



外力で動く質点の block



変位に対して弾性力が生じるバネの block



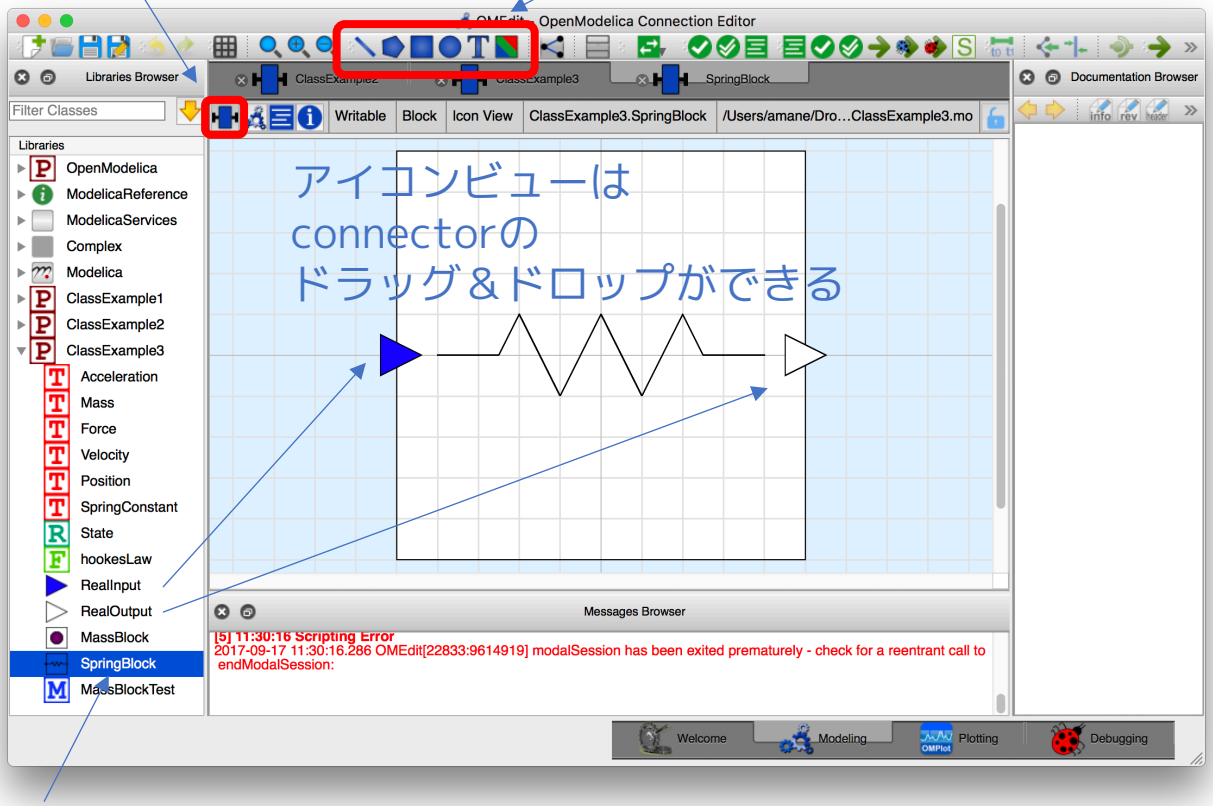
ClassExamle3

クラスのアイコンの編集方法

3. 描画する。

- 線
- 多角形
- 長方形
- 楕円
- テキスト
- イメージ

2. Icon View (アイコンビュー)に切り替える。



1. 編集したいクラスをダブルクリックしてアクティブにする。

ClassExamle3

⑥ function

- 関数の定義に使用する。
- input/output を付けた変数を宣言する。
- algorithm文のみが定義できる。
- 通常、function 名は小文字である。

<<function>> hookslaw
<<input variable>> k: SpringConstant <<input variable>> x: Position <<output variable>> f: Force
algorithm

入力変数
出力変数

アルゴリズム

⑦ connector（単純なもの）

block や model を接続するコネクタ。
type と似ているが、接頭語をつけることができる。

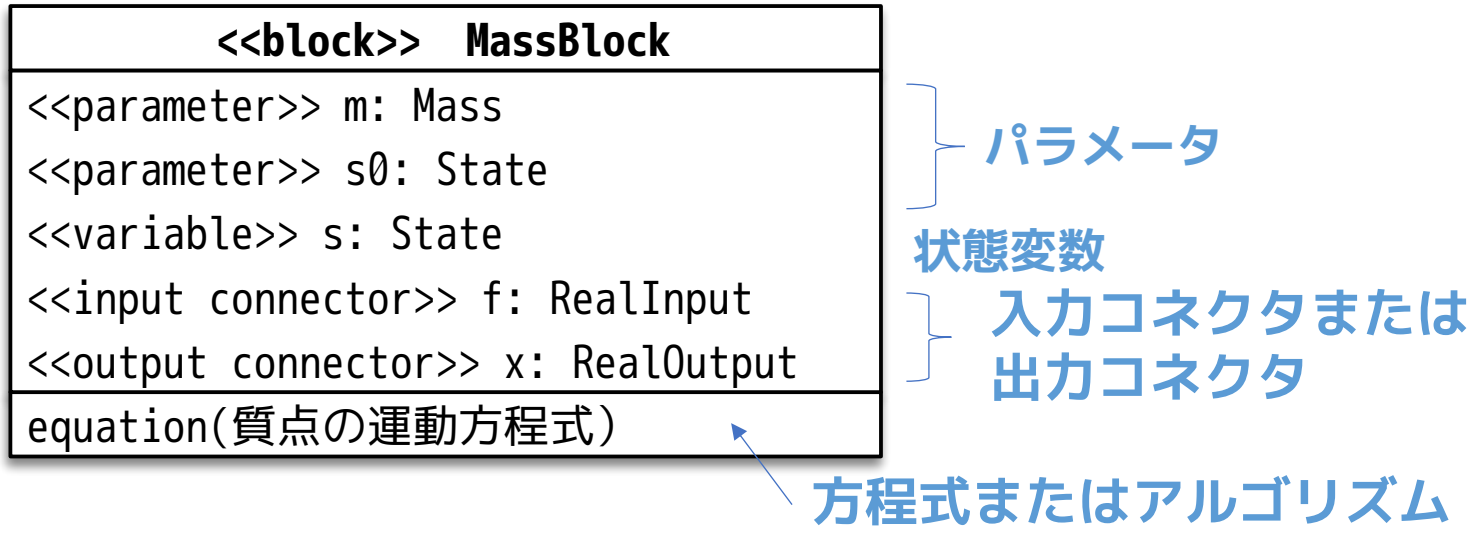
- input　クラス外で値を計算する
- output　クラス内で値を計算する

```
connector A = input B;
```

ClassExamle3

⑧ block

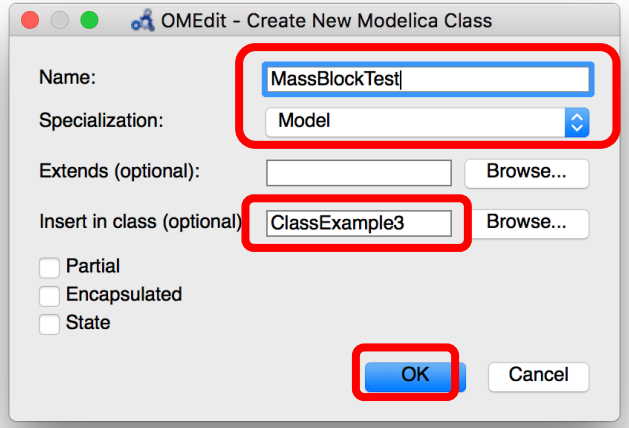
- modelとほぼ同じであるが、input または output の接頭語がついた connector を必ず使用する。
- ブロック線図のブロックを作成するのに使用する。



ClassExamle3

block のテスト用モデルを作成する

1. File > New Modelica Class（ファイル＞Modelicaクラス新規作成）



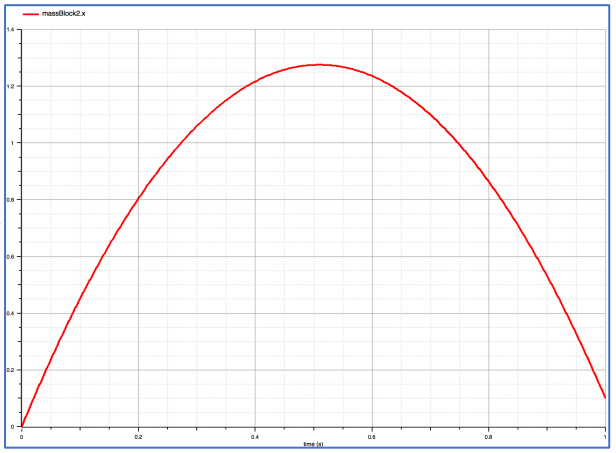
名前: MassBlockTest
クラス・タイプ: Model
挿入するクラス: ClassExample3

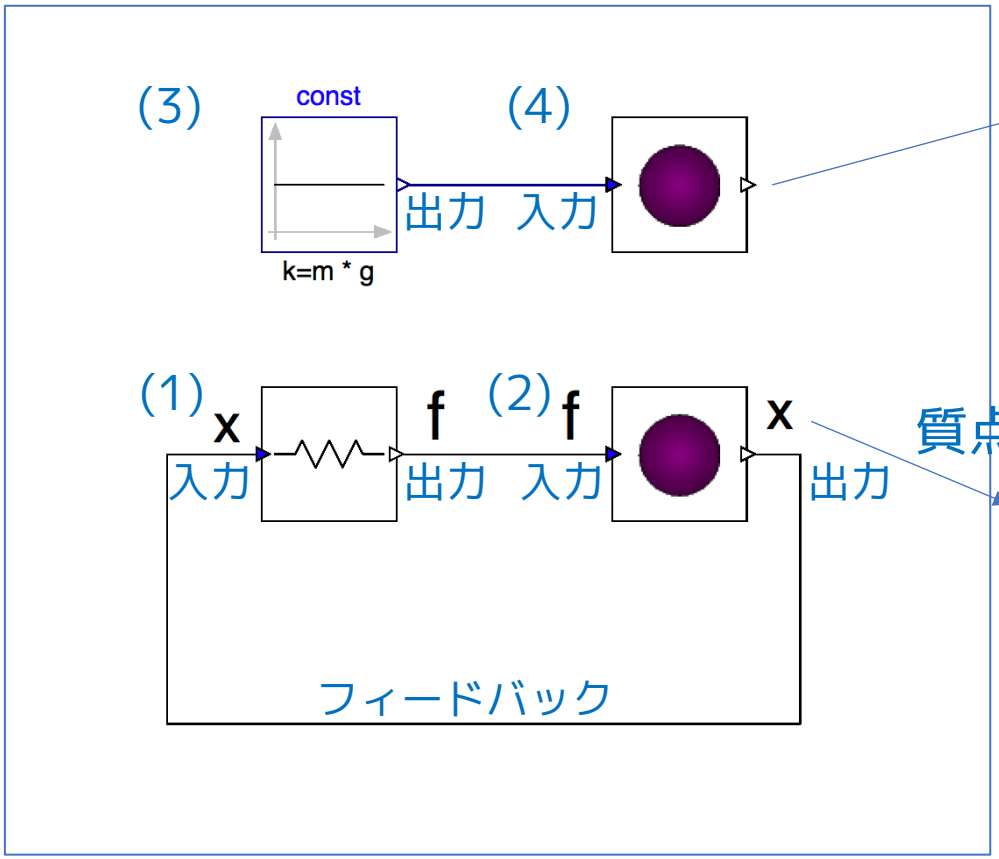
2. 次のスライドのように block を配置して接続する。
パラメータの設定は、次の次のスライドのソースコードを参照してください！

ClassExamle3

MassBlockTest

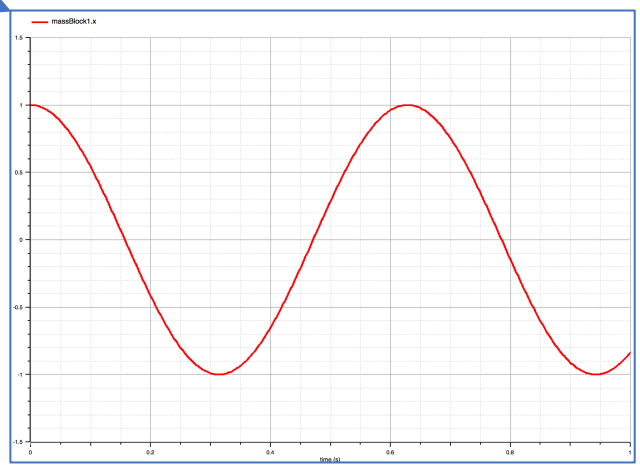
質点に外力として重力を与える。





モデル（ブロック線図）

質点に外力として弾性力を与える。



シミュレーション結果

ClassExamle3

MassBlockTest

```
// model
model MassBlockTest
    parameter Mass m = 1.0;
    SpringBlock springBlock1(k = 100) annotation( ...);                    (1)
    MassBlock massBlock1(m = m, s0(x = 1.0, v = 0.0)) annotation( ...);    (2)
    Modelica.Blocks.Sources.Constant const(k = m * g) annotation( ...);    (3)
    MassBlock massBlock2(m = m, s0(x = 0, v = 5.0)) annotation( ...);      (4)
equation
    connect(const.y, massBlock2.f) annotation( ...);
    connect(massBlock1.x, springBlock1.x) annotation( ...);
    connect(springBlock1.f, massBlock1.f) annotation( ...);
    annotation( ...);
end MassBlockTest;


end ClassExample3;
```

クラスを接続するときは、オペレータ connect() を
使って接続関係を方程式として定義する。

ClassExamle4

connector 付き model を使ってモデル化する

```
package ClassExample4  // package
  constant Acceleration g = -9.8;

  // type
  type Acceleration = Real(quantity = "Acceleration", unit = "m/s2");
  type Mass = Real(quantity = "Mass", unit = "kg");
  type Force = Real(quantity = "Force", unit = "N");
  type Velocity = Real(quantity = "Velocity", unit = "m/s");
  type Position = Real(quantity = "Length", unit = "m");
  type SpringConstant = Real(quantity = "Spring Constant", unit = "N/m");

  // record
  record State
    Position x;
    Velocity v;
  end State;

  // connector
  connector Flange
    Position s "Absolute position of flange";
    flow Force f "Cut force directed into flange";
    annotation( ...);
  end Flange;
```

ClassExample2 を
複製して編集する。

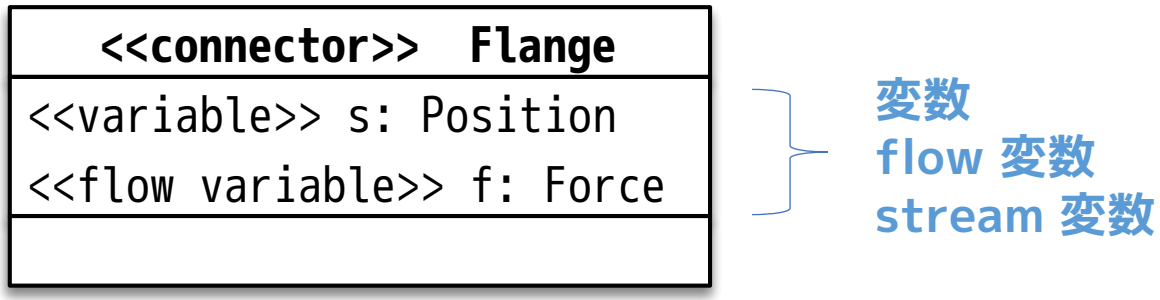
Flange: 1次元並進モデルを接続する
connector
(Modelica.Mechanics.Translational.Interfaces.Flange_a
と同じものを再宣言してます。)

接続されたコネクタ間で
- 位置 s は同じ値になる。
- 力 f は和がゼロになる。

ClassExamle4

⑦ connector

- record と似ているが、クラスを接続するためのコネクタを表す。
- equation や algorithm は定義できない。
- 変数の定義で、接頭語 flow, stream などが使用できる。
- 外部に入出力を表す接頭語 input, output をつけることができる。



変数：接続されたコネクタ間で値が同じ
flow 変数：接続されたコネクタ間で和がゼロ
stream 変数：「3_ModelicaFluidライブラリ.pdf」 参照

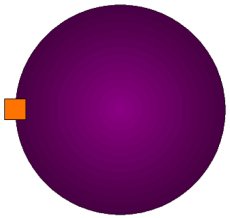
ClassExamle4

コネクタ変数 Flange を使用したモデルを作成する。

```
// model
 model MassModel
   parameter Mass m = 1.0;
   parameter State s0(x = 0, v = 0);
   State s(x(start = s0.x), v(start = s0.v));
   Flange flange annotation( ...);
 equation
   s.v = der(s.x);
   flange.f = m * der(s.v);
   flange.s = s.x;
   annotation( ...);
 end MassModel;

 model SpringModel
   parameter SpringConstant k = 1.0;
   Flange flange annotation( ...);
 equation
   flange.f = k * flange.s;
   annotation( …);
 end SpringModel;
```

運動方程式を実装

質点のモデル



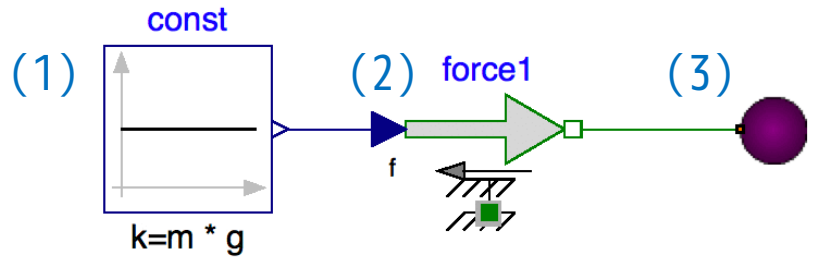
弾性力の計算式を実装

バネのモデル

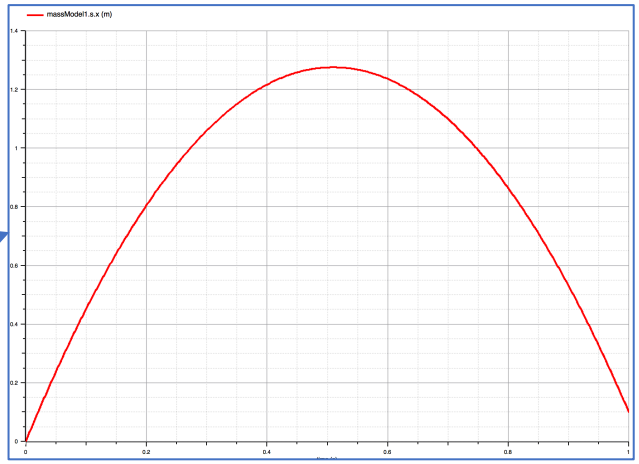
ClassExamle4

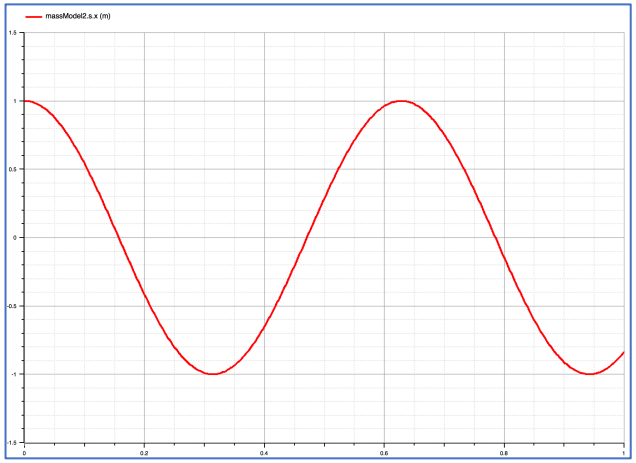
MassModelやSpringModel のテスト用モデル MassModelTest

質点に Forceコンポーネント
を接続して重力を与える



質点にバネを接続する





シミュレーション結果

ClassExamle4

MassModelやSpringModel のテスト用モデル MassModelTest

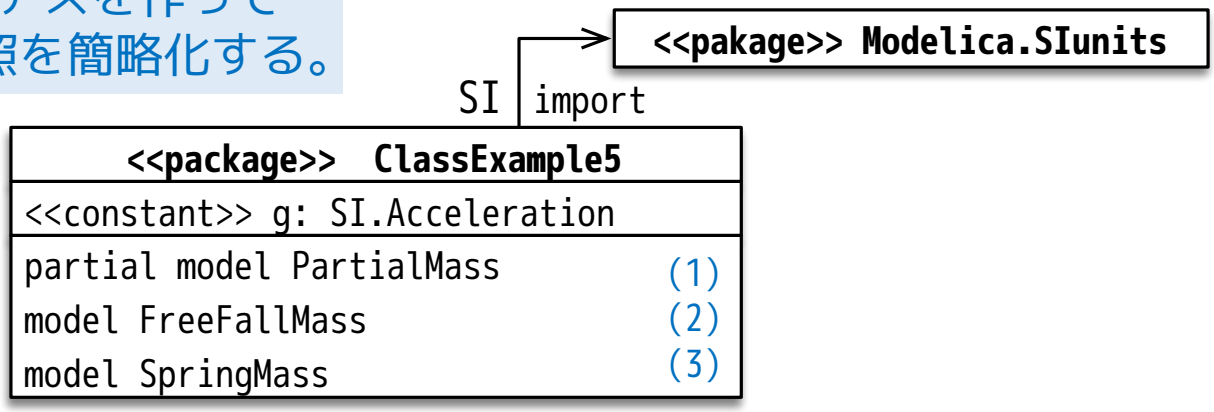
```
model MassModelTest
   parameter Mass m = 1.0;
   Modelica.Blocks.Sources.Constant const(k = m * g)  annotation( ...);          (1)
   Modelica.Mechanics.Translational.Sources.Force force1 annotation( ...);       (2)
   MassModel massModel1(m = m, s0(x = 0.0, v = 5.0)) annotation( ...);           (3)
   SpringModel springModel1(k = 100)  annotation( ...);                          (4)
   MassModel massModel2(m = m, s0(x = 1.0, v = 0.0))  annotation( ...);          (5)
equation
   connect(springModel1.flange, massModel2.flange) annotation( ...);
   connect(const.y, force1.f) annotation( ...);
   connect(force1.flange, massModel1.flange) annotation( ...);
end MassModelTest;

end ClassExample4;
```

ClassExamle5
クラスを継承してみる

block や connector 付き model は、connector を接続することによってクラスどうしを関連づける。
クラスどうしを関連づける方法は他にもある。

- エイリアスによる参照(import文)
- 継承・拡張(extends, partial class)
- ローカルクラス
- 交換可能なローカルクラス(replaceable, redeclare)

import文 エイリアスを作って
他のクラスの参照を簡略化する。



ClassExamle5

(1)

<<partial model>> PartialMass
<<parameter>> m: SI.Mass <<parameter>> v0: SI.Velocity <<parameter>> x0: SI.Position <<variable>> f: SI.Force <<variable>> x: SI.Position <<variable>> v: SI.Velocity
equation(質点の運動方程式)

(2)

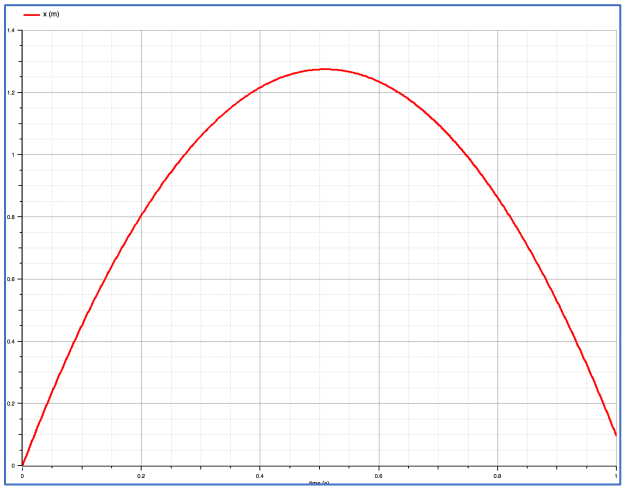
<<model>> FreeFallMass
equation(重力の計算式)

(3)

<<model>> SpringMass
<<parameter>> k:SI.TranslationalSpringConstant
equation(バネ弾性力の計算式)

partial class（部分クラス）
変数、equation, algorithm などが足りないクラス。

extends（継承、拡張）
既存のクラスを継承して新たなクラスを作る。



ClassExamle5

```
package ClassExample5
  import SI=Modelica.SIunits;
  constant SI.Acceleration g = -1 * Modelica.Constants.g_n;

  // partial model
  partial model PartialMass (1)
    parameter SI.Mass m = 1.0;
    parameter SI.Velocity v0 = 5.0;
    parameter SI.Position x0 = 0.0;
    SI.Force f;
    SI.Position x(start = x0);
    SI.Velocity v(start = v0);
  equation
    v = der(x);
    f = m * der(v);
  end PartialMass;

  // model
  model FreeFallMass extends PartialMass; (2)
  equation
   f = m * g;
  end FreeFallMass;

  model SpringMass extends PartialMass(x0 = 1.0, v0 = 0.0); (3)
    parameter SI.TranslationalSpringConstant k = 100;
  equation
   f = -k * x;
  end SpringMass;

end ClassExample5;
```

import文 エイリアスを作ってクラスの参照を簡略化する。

変数の数に対して方程式が足りない!!

変数3個

方程式2個
(質点の運動方程式)

継承先の model で方程式を
加えることによって
model が完成する!!

力として重力を計算する方程式

力としてバネの弾性力を計算する方程式

ClassExamle6
ローカルクラスを作ってみる

運動方程式を実装する

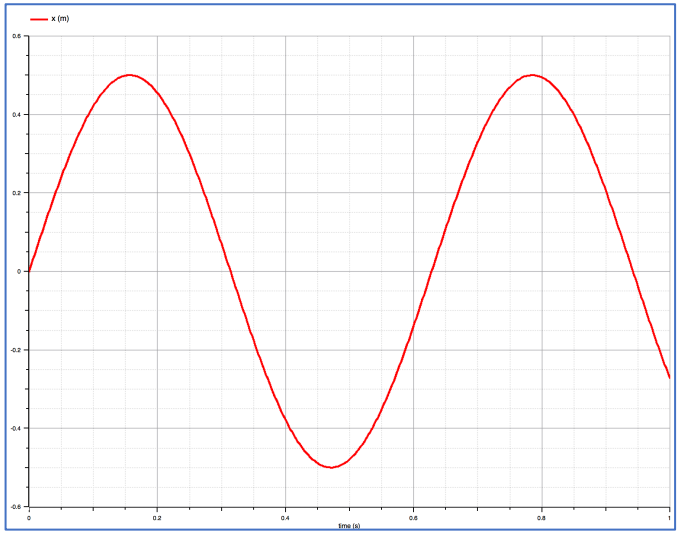
<<model>> MassC
<<parameter>> m: SI.Mass <<parameter>> v0: SI.Velocity <<parameter>> x0: SI.Position <<variable>> x: SI.Position <<variable>> v: SI.Velocity <<variable>> af: AppliedForce
model AppliedForce equation(質点の運動方程式)

ローカルクラス
クラスの中にクラスを作る

ローカルクラスを作ることによってクラス毎の役割を分けることができる。

作用する力の計算式を実装する

<<model>> AppliedForce
<<parameter>> k:SI.TranslationalSpringConstant <<input variable>> x: SI.Position <<output variable>> f: SI.Force
equation(バネ弾性力の計算式)



MassCの計算結果

ClassExamle6

```
package ClassExample6 (1)
  import SI = Modelica.SIunits;
  constant SI.Acceleration g = -1 * Modelica.Constants.g_n;

  // model
  model MassC  (2)
    parameter SI.Mass m = 1.0;
    parameter SI.Velocity v0 = 0.0;
    parameter SI.Position x0 = 1.0;
    SI.Position x(start = x0);
    SI.Velocity v(start = v0);
    SI.Force f;
    AppliedForce af(k=100);

    // local model
    model AppliedForce  (3)
      parameter SI.TranslationalSpringConstant k = 100;
      input SI.Position x;
      output SI.Force f;
    algorithm
      f := -k * x;
    end AppliedForce;

  equation
    v = der(x);
    f = m * der(v);
    x = af.x;
    f = af.f;
  end MassC;

end ClassExample6;
```

質点の運動を表すモデル

ローカルクラス(local model)
のオブジェクト

local model の宣言

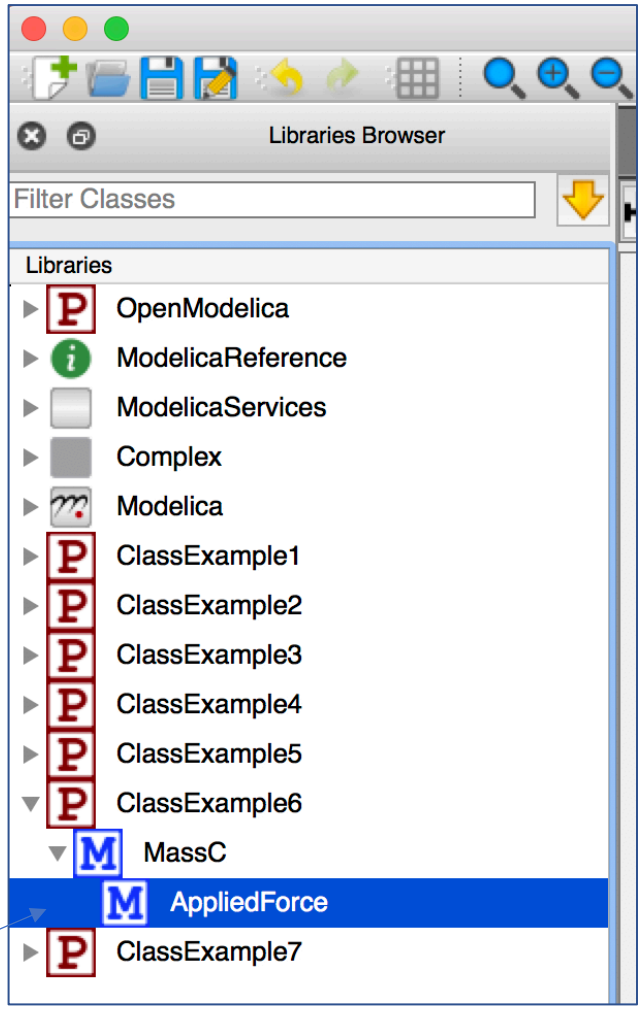
フックの法則により
弾性力を計算する。

質点に作用する力を
計算する
local model

質点の運動方程式

local modelとの接続関係を表す式

ClassExamle6



(1)
(2)
(3)

ローカルクラス

ローカルクラスはライブラリブラウザで確認できる。

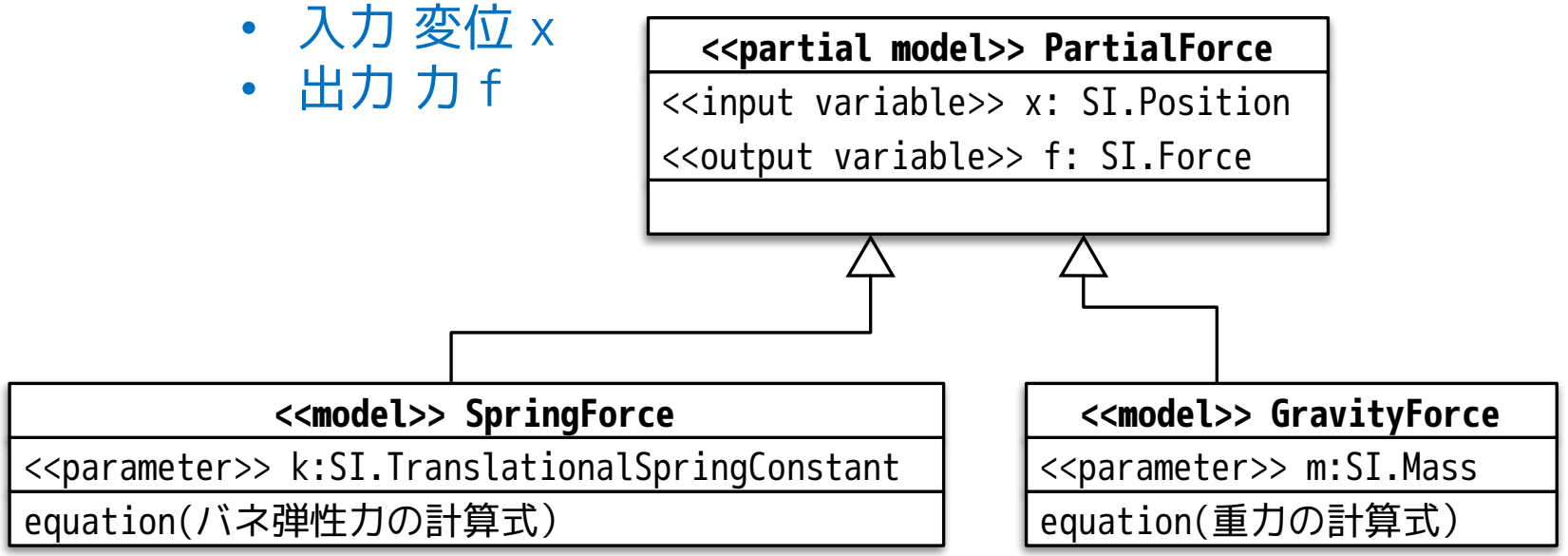
ClassExamle7

交換可能なローカルクラスを作ってみる

まず、質点に作用する力のモデルを数種作る。

(1) 質点に作用する力のベースモデル(部分モデル)

- 入力 変位 x
- 出力 力 f



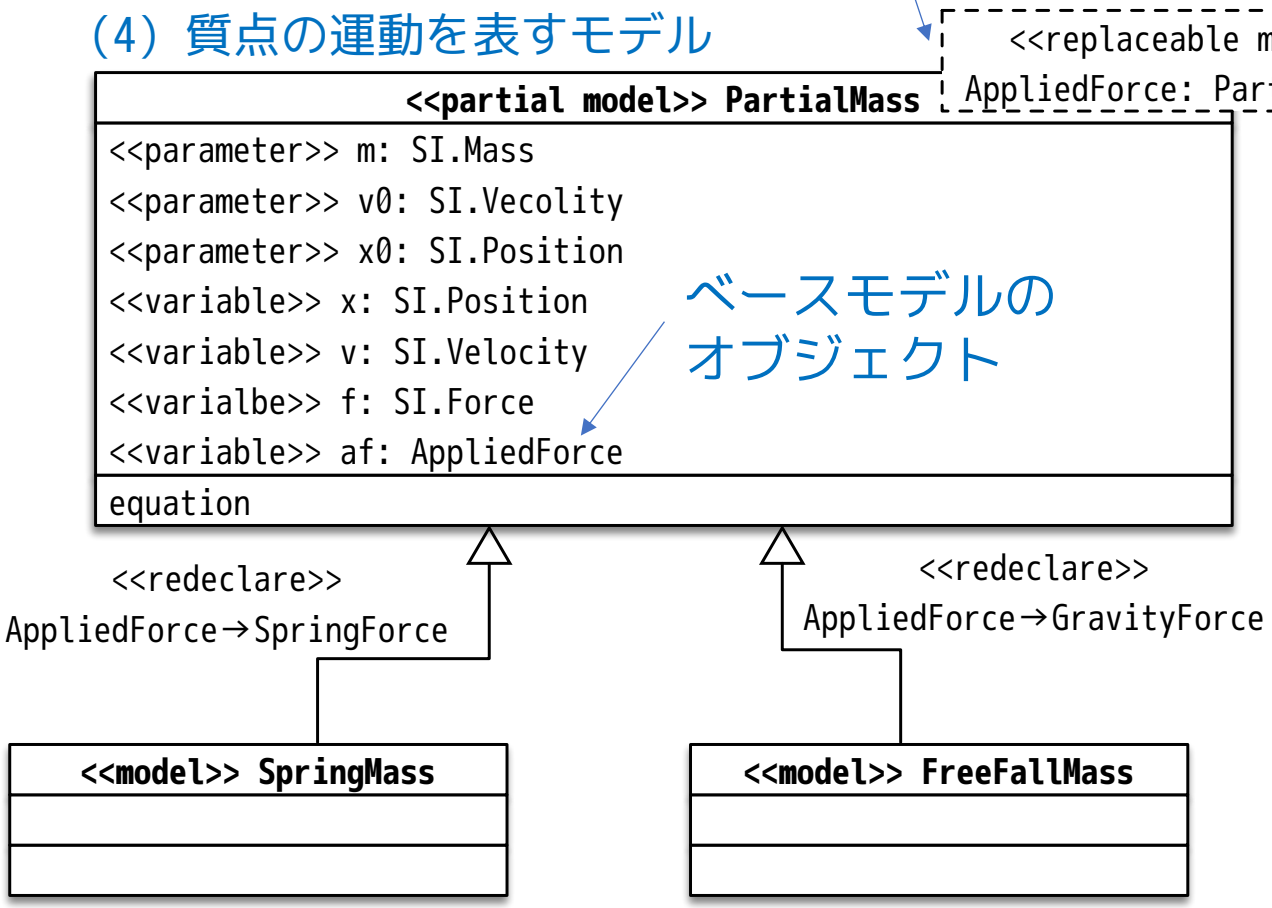
(2) バネ弾性力のモデル

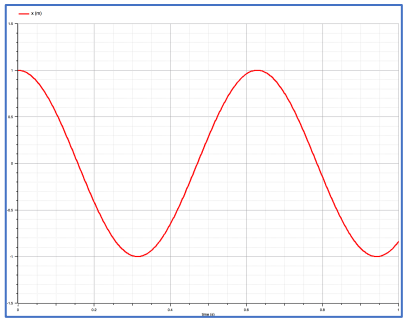
(3) 重力のモデル

ClassExamle7

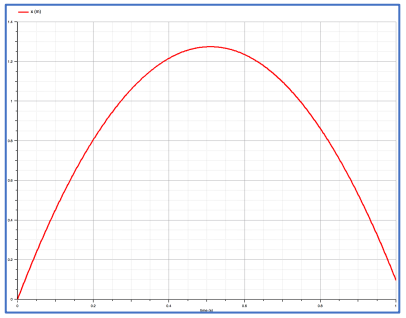
交換可能ローカルクラスを
もつクラスを作る。

交換可能ローカルクラスはテンプレート
のようにクラスの右上に書くことにする。





(5)の結果



(6)の結果

ClassExamle7

```
package ClassExample7
  import SI = Modelica.SIunits;
  constant SI.Acceleration g = -1 * Modelica.Constants.g_n;

  // force models
  partial model PartialForce
    input SI.Position x;
    output SI.Force f;
  end PartialForce;

  model SpringForce extends PartialForce;
    parameter SI.TranslationalSpringConstant k = 1;
  algorithm
    f := -k * x;
  end SpringForce;

  model GravityForce extends PartialForce;
    parameter SI.Mass m = 1.0;
  algorithm
    f := m * g;
  end GravityForce;
```

(1) 質点に作用する力のベースモデル(部分モデル)

(2) バネ弾性力のモデル

(3) 重力のモデル

ClassExamle7

```
// mass models
 model PartialMass
   parameter SI.Mass m = 1.0;
   parameter SI.Velocity v0 = 5.0;
   parameter SI.Position x0 = 0.0;
   SI.Position x(start = x0);
   SI.Velocity v(start = v0);
   SI.Force f;
   AppliedForce af;
   replaceable model AppliedForce = PartialForce;
 equation
   v = der(x);
   f = m * der(v);
   x = af.x;
   f = af.f;
 end PartialMass;

 model SpringMass
 extends PartialMass(m=1.0,  x0=1.0, v0=0.0, redeclare model AppliedForce = SpringForce(k=100));
 end SpringMass;

 model FreeFallMass
 extends PartialMass(m = 1.0, v0=5.0, x0=0.0, redeclare model AppliedForce = GravityForce(m=1.0));
 end FreeFallMass;

end ClassExample7;
```

（4）質点の運動を表すモデル

replaceable class クラス名
交換可能なローカルクラスを宣言する。

redeclare class クラス名
ローカルクラスを再宣言する。

（5）バネにつながった質点

バネ弾性力モデルをバインドする。

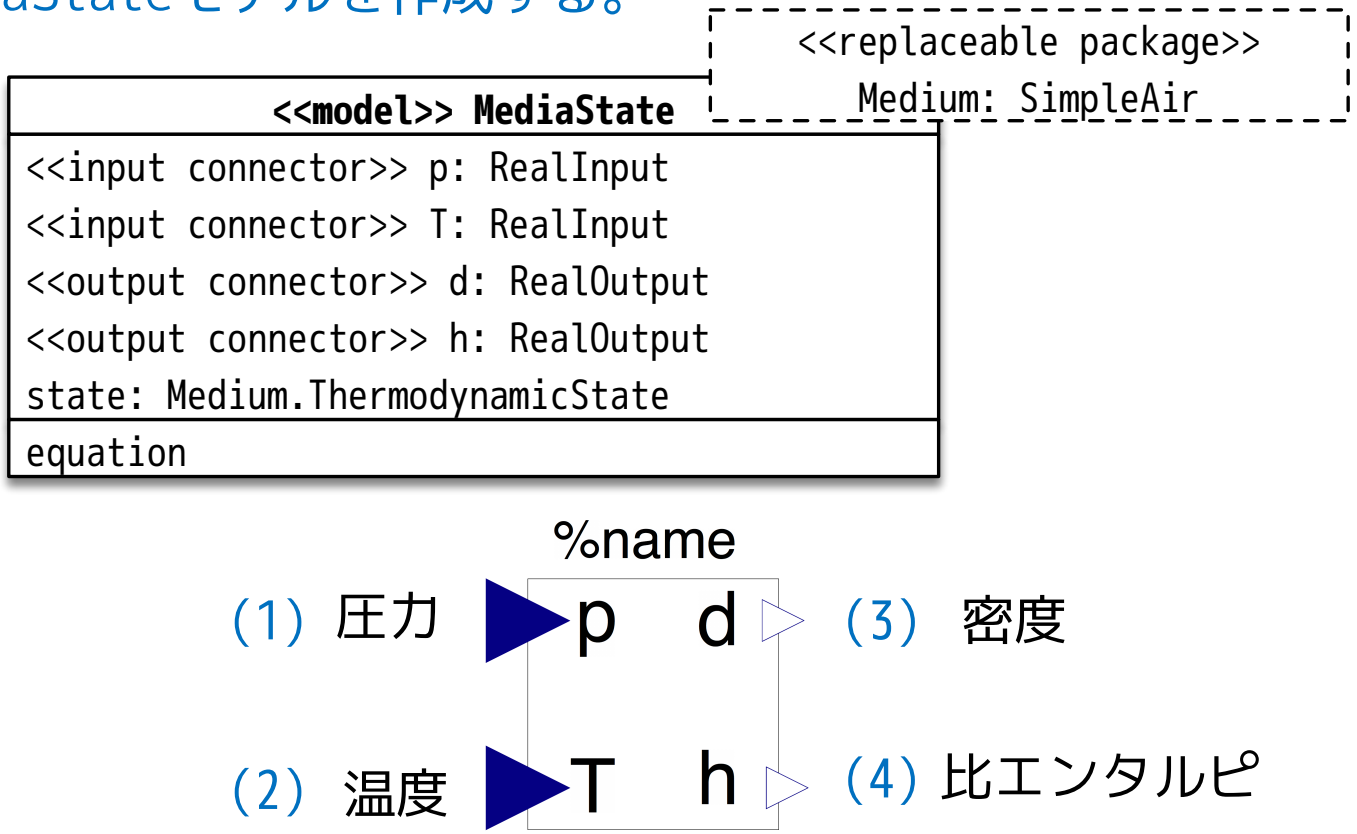
（6）自由落下する質点

重力モデルをバインドする。

ClassExamle8

Modelica.Mediaを使って空気の物性値を調べる

温度と圧力を入力すると、空気の密度と比エンタルピを出力する
MediaStateモデルを作成する。



ClassExamle8

交換可能なローカルパッケージとして、Medium を宣言し SimpleAir(空気の物性モデル）をバインドする。

```
package ClassExample8
  import Modelica.Media;

  model MediaState
    replaceable package Medium = Media.Air.SimpleAir;
    Medium.ThermodynamicState state;
    Modelica.Blocks.Interfaces.RealInput p annotation( ...); (1)
    Modelica.Blocks.Interfaces.RealInput T annotation( ...); (2)
    Modelica.Blocks.Interfaces.RealOutput d annotation( ...); (3)
    Modelica.Blocks.Interfaces.RealOutput h annotation( ...); (4)
  equation
    state = Medium.setState_pT(p, T);
    d = Medium.density(state);
    h = Medium.specificEnthalpy(state);
    annotation( ...);
  end MediaState;
```

熱力学的状態を表す record 変数

圧力、温度、密度、比エンタルピの入出力コネクタ

圧力pと温度Tから熱力学的状態表す変数 state を決定する。

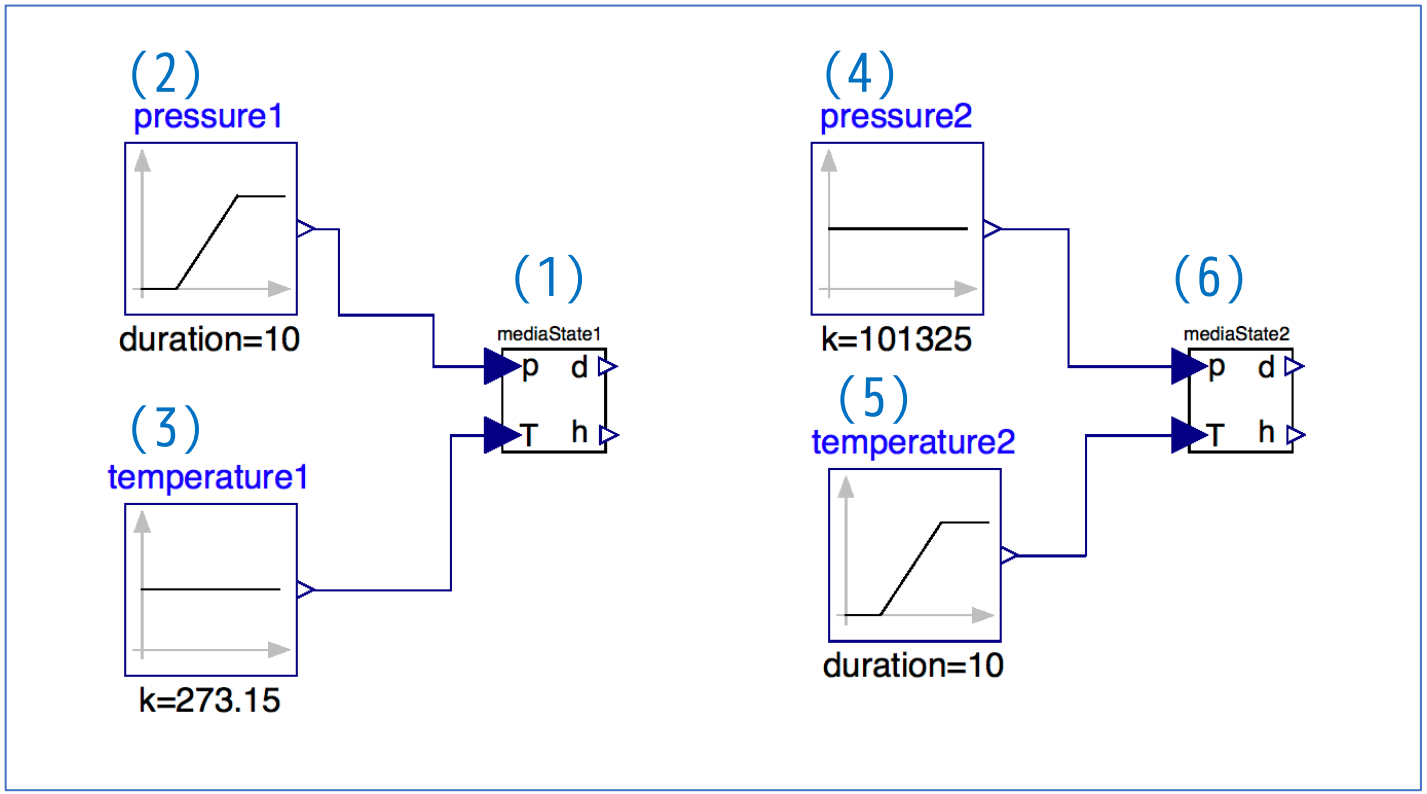
変数 state から、密度 d, 比エンタルピ h を求める。

ClassExamle8

テストモデル MediaStateTest

温度を固定して
圧力を変化させる

圧力を固定して
温度を変化させる



ClassExamle8

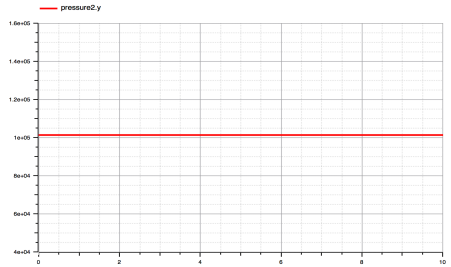
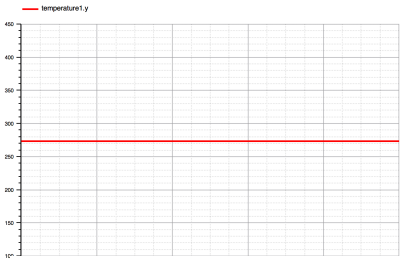
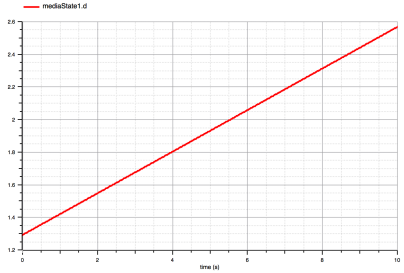
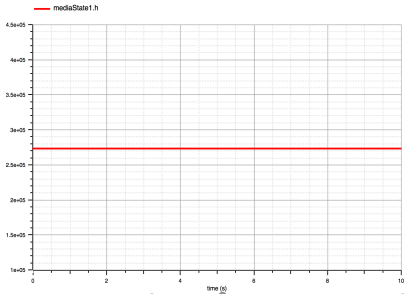
物性モデルを DryAirNasa に変更してみる。

```
model MediaStateTest
  replaceable package Medium = Media.Air.DryAirNasa;
  MediaState mediaState1(redeclare package Medium = Medium) annotation( ...);              (1)
  Modelica.Blocks.Sources.Ramp pressure1(duration = 10, height = 100000, offset = 101325)  (2)
    annotation( ...);
  Modelica.Blocks.Sources.Constant temperature1(k = 273.15)  annotation( ...);             (3)
  Modelica.Blocks.Sources.Constant pressure2(k = 101325) annotation( ...);                 (4)
  Modelica.Blocks.Sources.Ramp temperature2(duration = 10, height = 100, offset = 273.15)  (5)
   annotation( ...);
  MediaState mediaState2(redeclare package Medium = Medium) annotation( ...);              (6)
equation
  connect(temperature2.y, mediaState2.T) annotation( ...);
  connect(pressure2.y, mediaState2.p) annotation( ...);
  connect(pressure1.y, mediaState1.p) annotation( ...);
  connect(temperature1.y, mediaState1.T) annotation( ...);
end MediaStateTest;
```

こんな感じでいろんな種類（純物質）の流体の物性を調べられる。

ClassExamle8

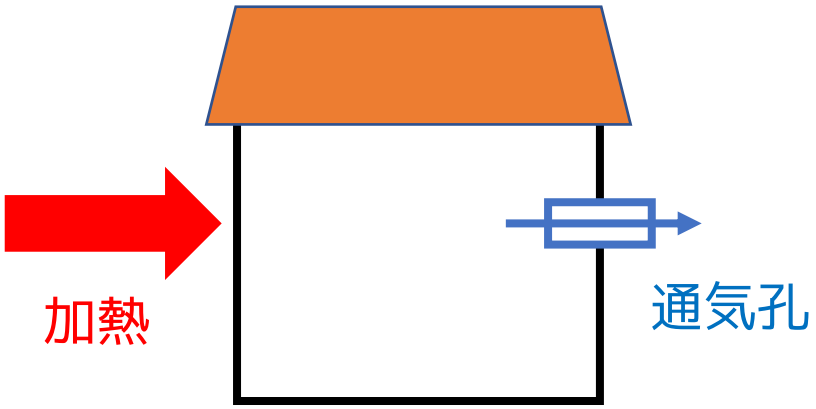
テストモデル
の計算結果

	mediaState1	mediaState2
圧力 p		
温度 T		
密度 d		
比エンタルピ h		

ClassExamle9

Modelica.Mediaを使って室温の変化を調べる

- 圧力を一定に保った部屋を加熱する。
- 部屋には通気孔があり、膨張した空気が流出する。



ClassExamle9

まず、単体モデルを作ってみる。

```
package ClassExample9
  import Modelica.Media;
  import SI = Modelica.SIunits;

  model RoomA
    package Medium = Media.Air.DryAirNasa;
    parameter SI.Temperature T_amb = 293.15;
    parameter SI.Pressure p_amb = 101325;
    parameter SI.Volume V = 22.0;
    parameter SI.HeatFlowRate Q_flow = 100;

    Medium.BaseProperties medium;
    SI.MassFlowRate m_flow(start = 0.0);
    SI.Mass M;
    SI.Energy U;
  equation
    M = medium.d * V;
    U = medium.u * M;
    der(M) = m_flow;
    der(U) = Q_flow + medium.h * m_flow;
    medium.p = p_amb;
  initial equation
    medium.T = T_amb;
  end RoomA;
```

DryAirNasa(空気の物性モデル)をローカルパッケージ Medium として宣言する。

雰囲気の状態

部屋の容積（４畳半ぐらい）

熱流量 100 W

初期温度

方程式

$M = \rho V$

$U = Mu$

室内全体の空気の質量と内部エネルギー

$$\frac{dM}{dt} = m\_flow$$

$$\frac{dU}{dt} = Q\_flow + h \cdot m\_flow$$

質量保存則とエネルギー保存則

$p = p_{amb}$ 圧力は雰囲気圧力で固定

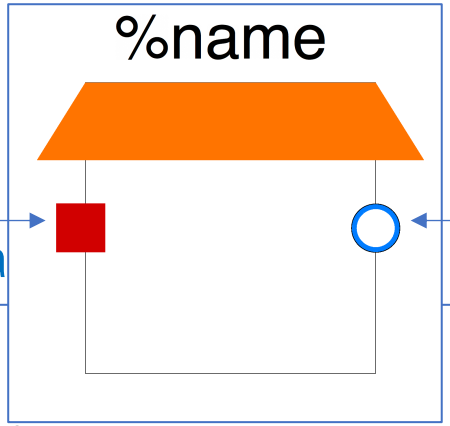
ClassExamle9

コネクターを使って、コンポーネント化する。

```
model RoomB
  replaceable package Medium = Media.Air.DryAirNasa;
  parameter SI.Volume V = 22.0;
  Modelica.Fluid.Interfaces.FluidPort_b port_b(redeclare package Medium = Medium) (1)
    annotation( ...);
  Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a port_a annotation( ...);   (2)
  Medium.BaseProperties medium;
  SI.Mass M;
  SI.Energy U;
equation
  M = medium.d * V;
  U = medium.u * M;
  der(M) = port_b.m_flow;
  der(U) = port_a.Q_flow + actualStream(port_b.h_outflow) * port_b.m_flow;
  port_b.p = medium.p;
  port_b.h_outflow = medium.h;
  port_a.T = medium.T;
initial equation
  medium.T = 293.15;
  annotation( ...);
end RoomB;
```

port_b を流れる流体を設定する。

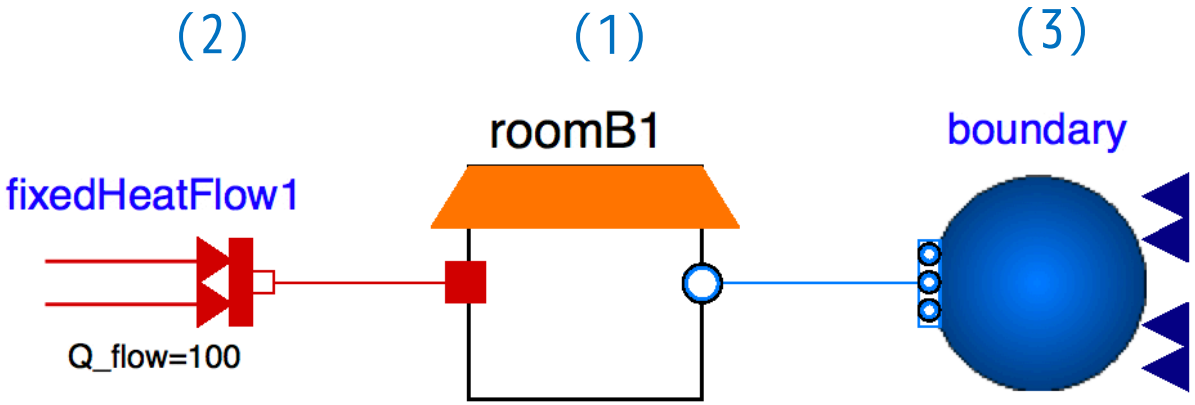
(2)
port_a:
HeatPort_a



(1)
port_b:
FluidPort_b

ClassExamle9

テストモデルを作成する。



RoomBTest

ClassExamle9

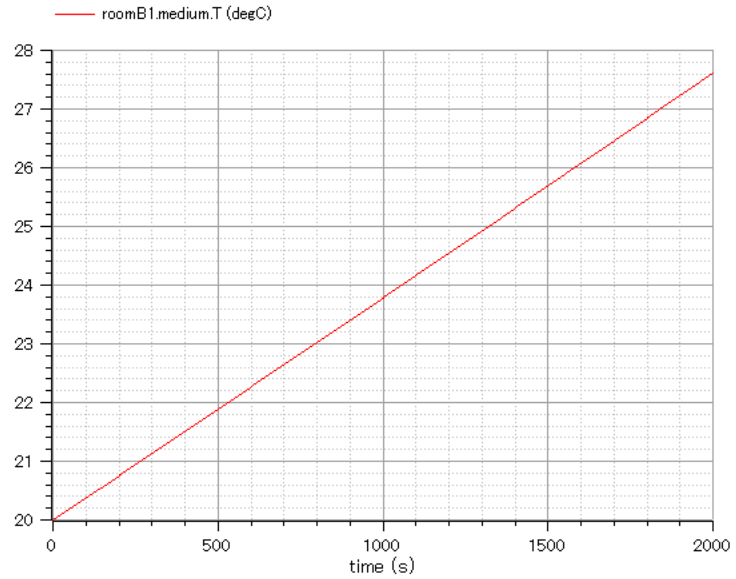
RoomTestのソースコード

コンポーネントを流れる流体を設定する。

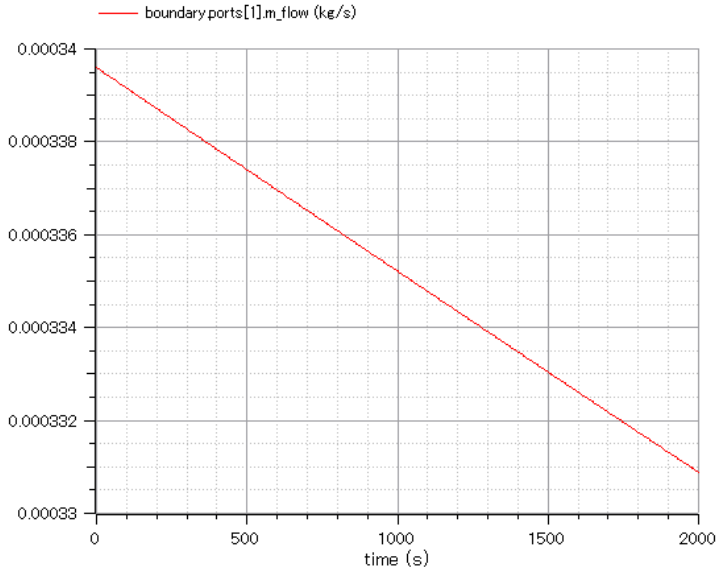
```
model RoomBTest
  replaceable package Medium = Media.Air.DryAirNasa;
  ClassExample9.RoomB roomB1(redeclare package Medium = Medium)                    (1)
    annotation( ...);
  Modelica.Thermal.HeatTransfer.Sources.FixedHeatFlow fixedHeatFlow1(Q_flow = 100)  (2)
    annotation( ...);
  Modelica.Fluid.Sources.Boundary_pT boundary(redeclare package Medium = Medium,    (3)
    T = 293.15, nPorts = 1, p = 101325)  annotation( ...);
equation
  connect(roomB1.port_b, boundary.ports[1]) annotation( ...);
  connect(fixedHeatFlow1.port, roomB1.port_a) annotation( ...);
end RoomBTest;

  annotation( ...);
end ClassExample9;
```

ClassExamle9

テストモデルのシミュレーション結果



室温の変化



通気口の流量

まとめ

- Modelicaの８種類のクラスについて特徴を示した。
- クラスの基本的な使い方について述べた。

Modelica のクラス

① package
② type
③ class
④ record
⑤ model
⑥ function
⑦ connector
⑧ block

クラスの基本的な使い方

- コネクタによる接続
- エイリアスによる参照(import文)
- 継承・拡張(extends, partial class)
- ローカルクラス
- 交換可能なローカルクラス (replaceable, redeclare)

Modelica Standard Library License

https://github.com/modelica/ModelicaStandardLibrary/blob/master/LICENSE