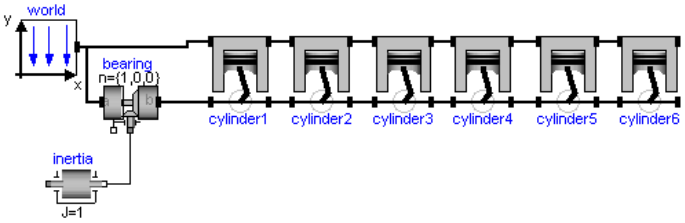
Advanced Modelica Tutorial Exercises

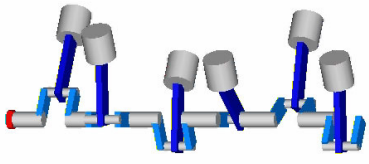
Hilding Elmqvist, Dynasim
Martin Otter, DLR

Refine MultiBody/Engine

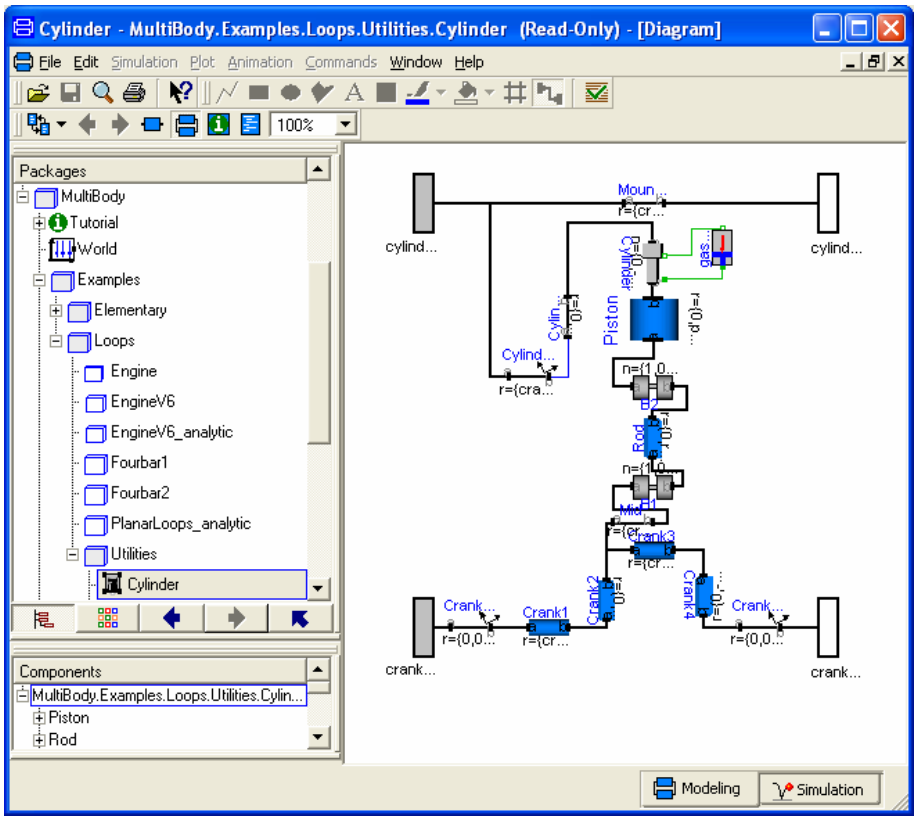
Make Engine example model a reusable component

1. Manage parameters
2. Allow changing number of cylinders
3. Test rig with replaceable engine



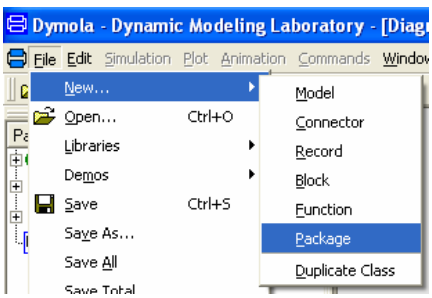


Use Cylinder from Library MultiBody

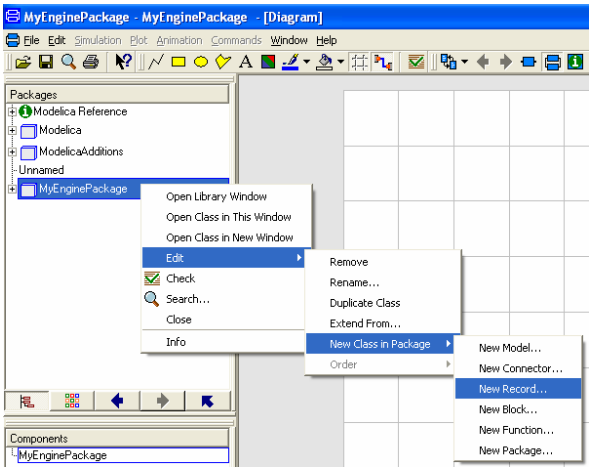


Package structuring

First create package

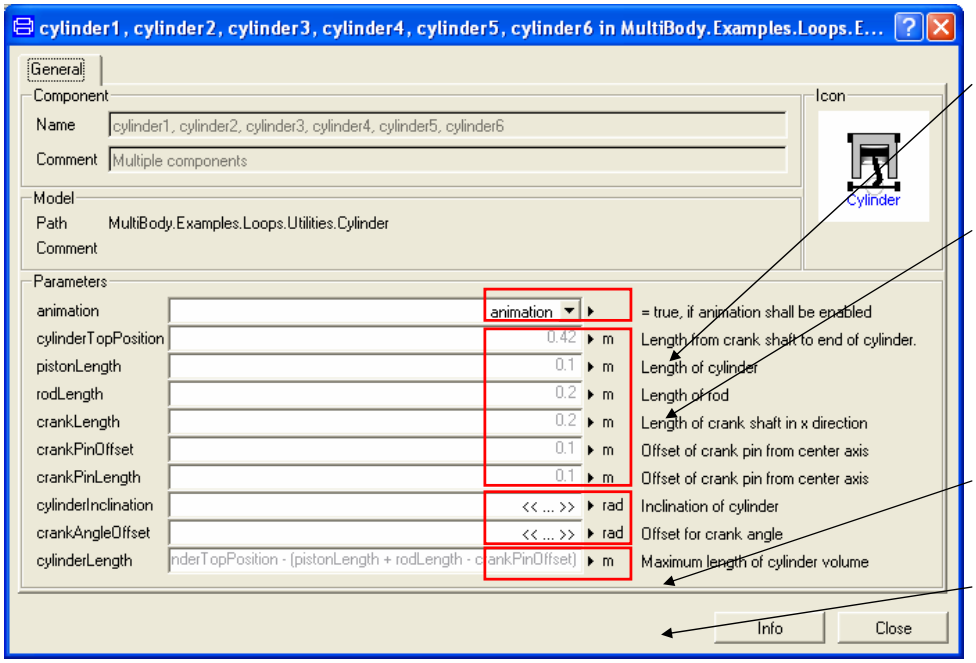


Then create models, etc within the package



Task 1. Handling of parameters

- Now – multiple select

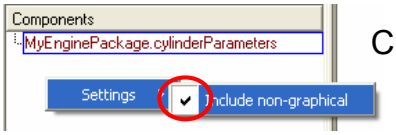


Parameter record

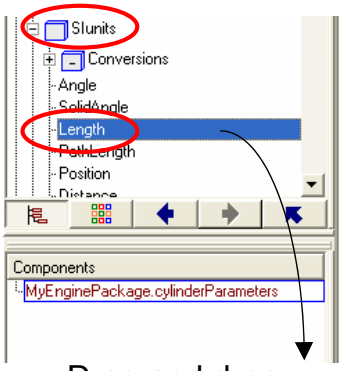
- Introduce parameter record for common parameters
  - Record gives possible reuse of data in different engine configurations
  - Propagate record elements to individual parameters
  - Introduce tabs and groups

To build record in the GUI

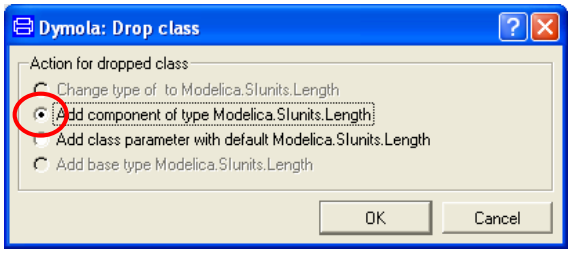
Create **record** cylinderParameters and insert parameters.

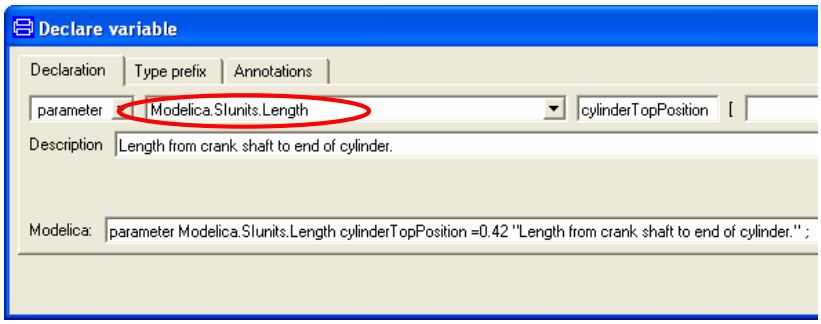


Change of setting needed

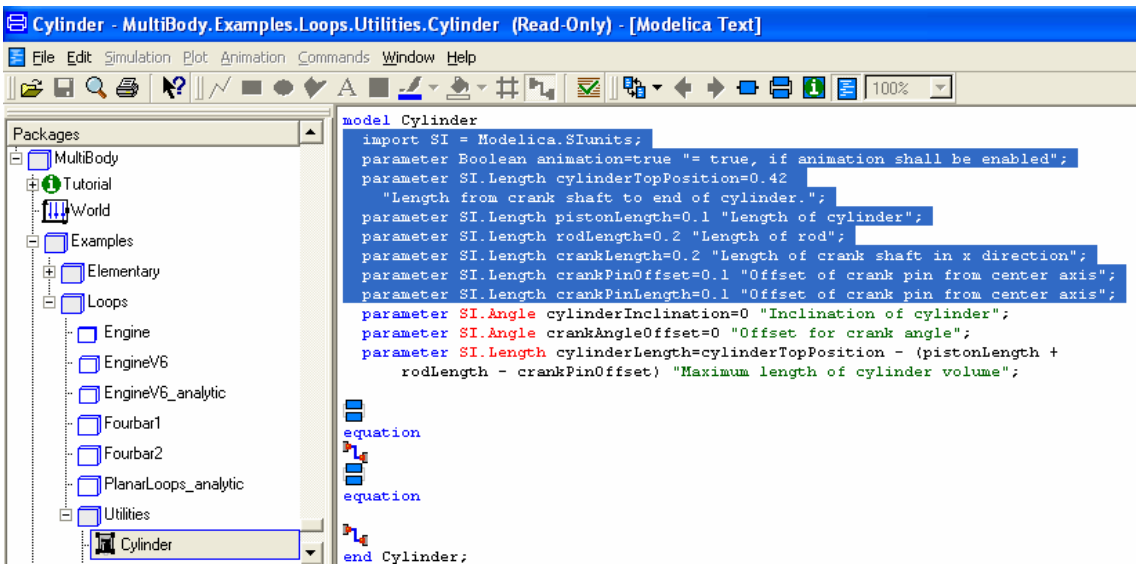


Drag and drop



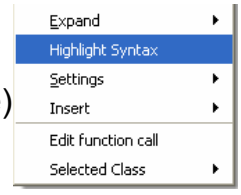


Copy and paste alternative



Context menu

(after paste)

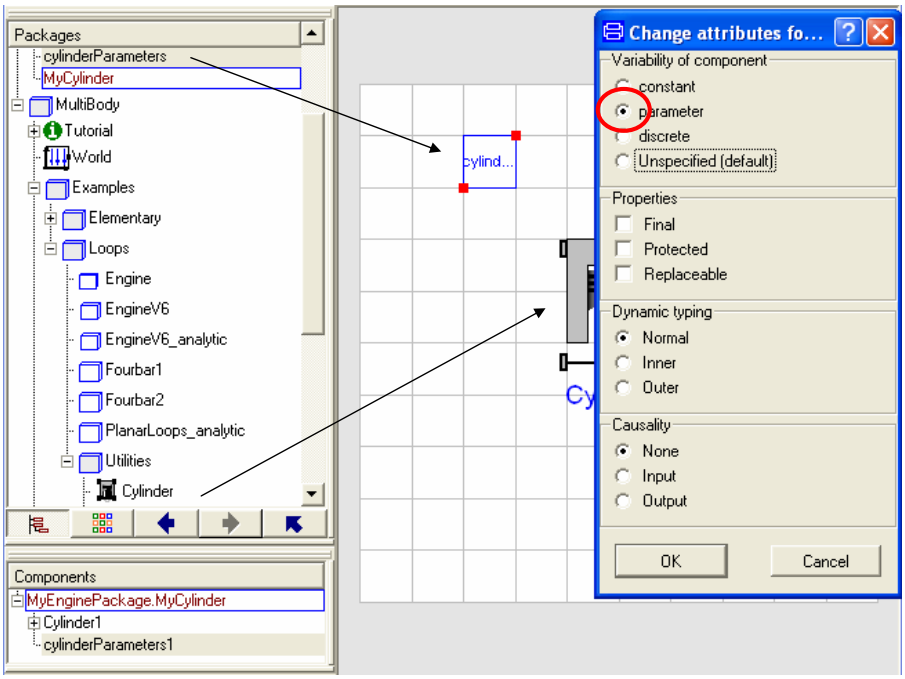


Delete this line

```
record cylinderParameters
  import SI = Modelica.SIunits;
  parameter Boolean animation=true "= true, if animation shall be enabled";
  parameter SI.Length cylinderTopPosition=0.42
    "Length from crank shaft to end of cylinder.";
  parameter SI.Length pistonLength=0.1 "Length of cylinder";
  parameter SI.Length rodLength=0.2 "Length of rod";
  parameter SI.Length crankLength=0.2 "Length of crank shaft in x direction";
  parameter SI.Length crankPinOffset=0.1 "Offset of crank pin from center axis";
  parameter SI.Length crankPinLength=0.1 "Offset of crank pin from center axis";
end cylinderParameters;
```

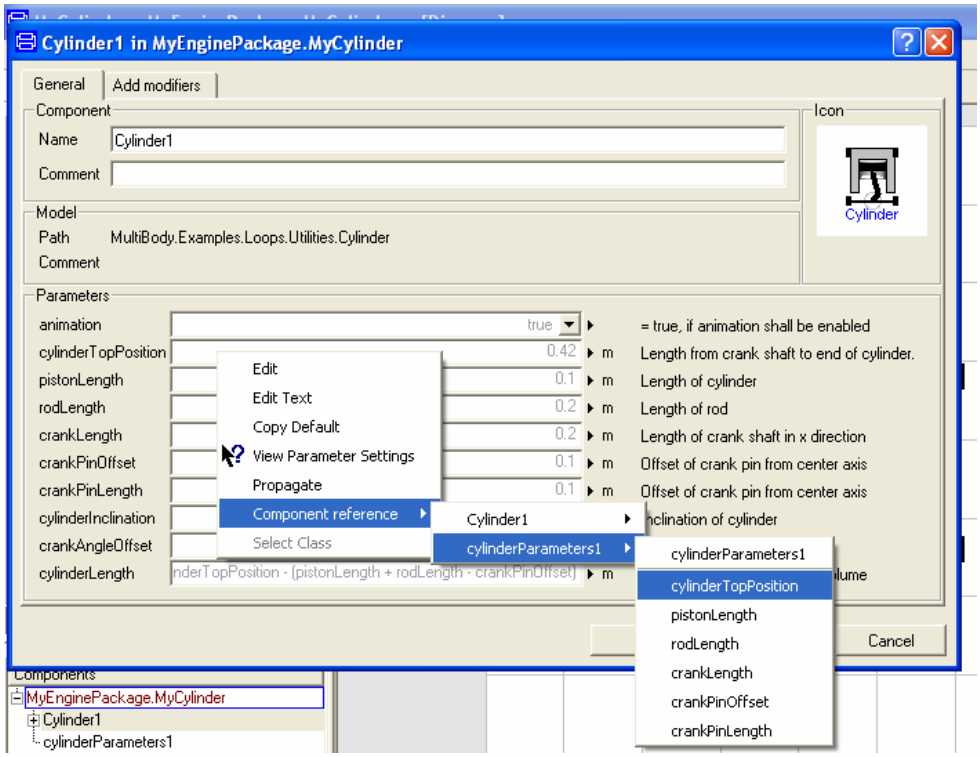
Create MyCylinder

Create **model** MyCylinder and insert a cylinder from the MultiBody library and a parameter record.



Propagate parameters

Propagate each parameter in the record to the corresponding parameter of the MultiBody cylinder. Point to the input field and click the right mouse button and create ”Component reference”.

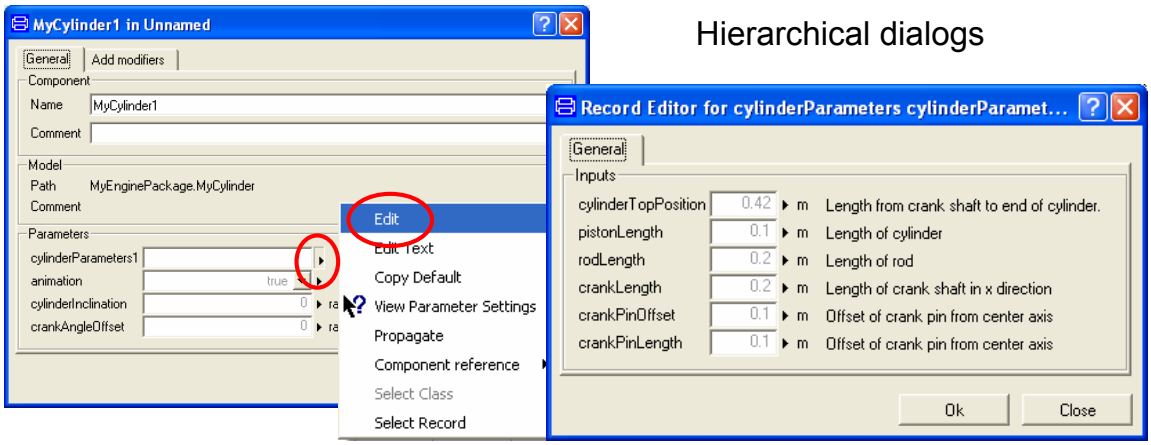


Add remaining parameters

Add remaining 3 parameters.

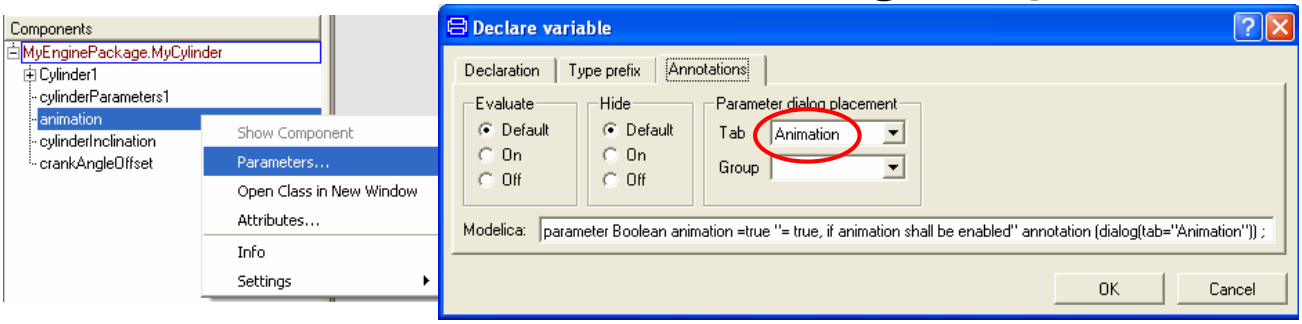
Test MyCylinder by creating a component in Unnamed and double clicking it.

```
model MyCylinder
  MultiBody.Examples.Loops.Utilities.Cylinder Cylinder1(
    cylinderTopPosition=cylinderParameters1.cylinderTopPosition,
    pistonLength=cylinderParameters1.pistonLength,
    rodLength=cylinderParameters1.rodLength,
    crankLength=cylinderParameters1.crankLength,
    crankPinOffset=cylinderParameters1.crankPinOffset,
    crankPinLength=cylinderParameters1.crankPinLength,
    cylinderInclination=cylinderInclination,
    crankAngleOffset=crankAngleOffset,
    animation=animation) annotation (extent=[-20, -20; 20, 40]);
  annotation (Diagram);
  parameter cylinderParameters cylinderParameters1
    annotation (extent=[-60, 60; -40, 80]);
  import SI = Modelica.SIunits;
  parameter Boolean animation=true "= true, if animation shall be enabled";
  parameter SI.Angle cylinderInclination=0 "Inclination of cylinder";
  parameter SI.Angle crankAngleOffset=0 "Offset for crank angle";
end MyCylinder;
```

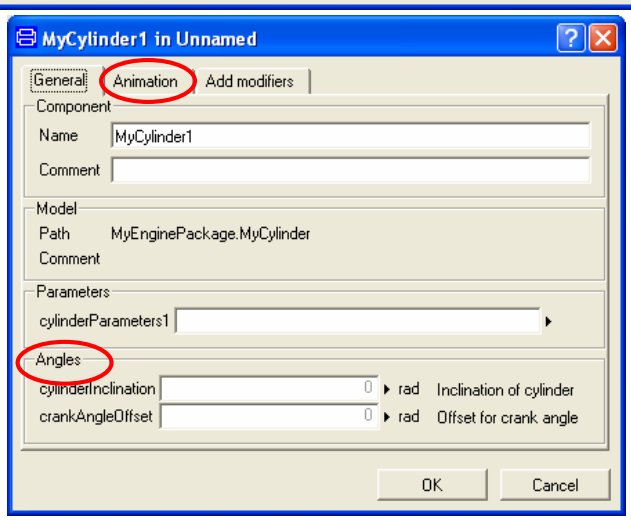


Hierarchical dialogs

Introduce tabs and groups



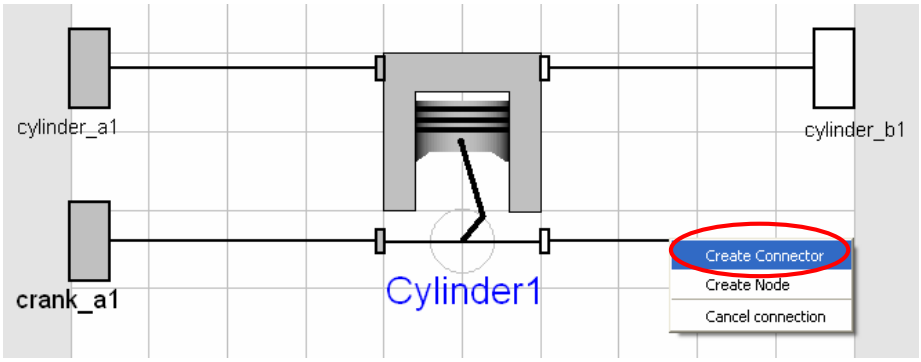
Resulting dialog with tabs and groups



Improve dialog by introducing tabs and groups.

Propagate connectors

Propagate connectors by drawing connections. At end point, click right mouse button and select ”Create connector”

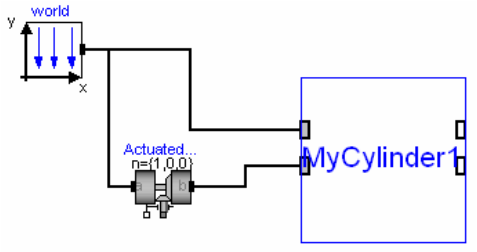


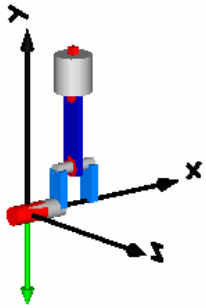
Resulting Modelica code

```
model MyCylinder
  MultiBody.Examples.Loops.Utilities.Cylinder Cylinder1(
    cylinderTopPosition=cylinderParameters1.cylinderTopPosition,
    pistonLength=cylinderParameters1.pistonLength,
    rodLength=cylinderParameters1.rodLength,
    crankLength=cylinderParameters1.crankLength,
    crankPinOffset=cylinderParameters1.crankPinOffset,
    crankPinLength=cylinderParameters1.crankPinLength)
    annotation (extent=[-20, -20; 20, 40]);
  annotation (Diagram);
  parameter cylinderParameters cylinderParameters1
    annotation (extent=[-60, 60; -40, 80]);
  import SI = Modelica.SIunits;
  parameter Boolean animation=true "= true, if animation shall be enabled"
    annotation (dialog(tab="Animation"));
  parameter SI.Angle cylinderInclination=0 "Inclination of cylinder"
    annotation (dialog(group="Angles"));
  parameter SI.Angle crankAngleOffset=0 "Offset for crank angle"
    annotation (dialog(group="Angles"));
  MultiBody.Interfaces.Frame_a cylinder_a1
    annotation (extent=[-110, 26; -90, 46]);
  MultiBody.Interfaces.Frame_a crank_a1 annotation (extent=[-110, -18; -90, 2]);
  MultiBody.Interfaces.Frame_b cylinder_b1 annotation (extent=[90, 26; 110, 46]);
  MultiBody.Interfaces.Frame_b crank_b1 annotation (extent=[90, -18; 110, 2]);
equation
  connect(Cylinder1.cylinder_a, cylinder_a1)
    annotation (points=[-22, 36; -100, 36], style(color=0, thickness=2));
  connect(Cylinder1.crank_a, crank_a1)
    annotation (points=[-22, -8; -100, -8], style(color=0, thickness=2));
  connect(Cylinder1.cylinder_b, cylinder_b1)
    annotation (points=[22, 36; 100, 36], style(color=0, thickness=2));
  connect(Cylinder1.crank_b, crank_b1)
    annotation (points=[22, -8; 100, -8], style(color=0, thickness=2));
end MyCylinder;
```

Make MyEngine

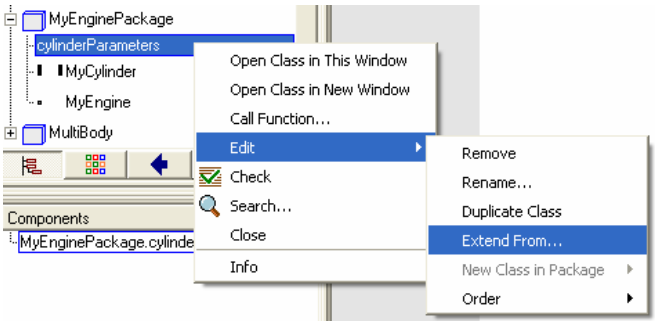
Introduce World object and Revolute joint

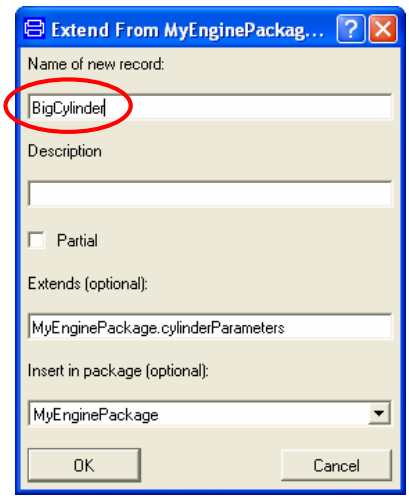




- Axis of rotation for Revolute joint is x-axis
- Set initial angular velocity to 1 deg/s
- Simulate for 0.1 second

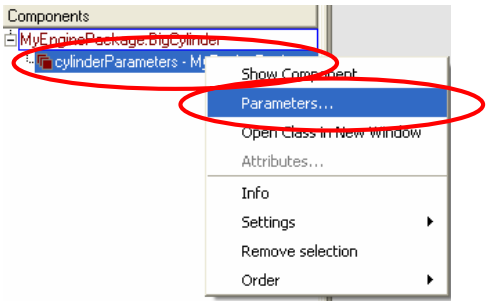
Declare parameter set

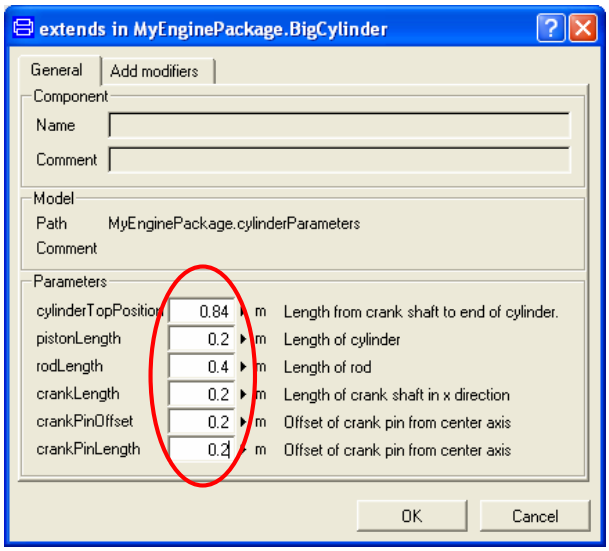




Declare a parameter set by extending the parameter record.

Make parameter set

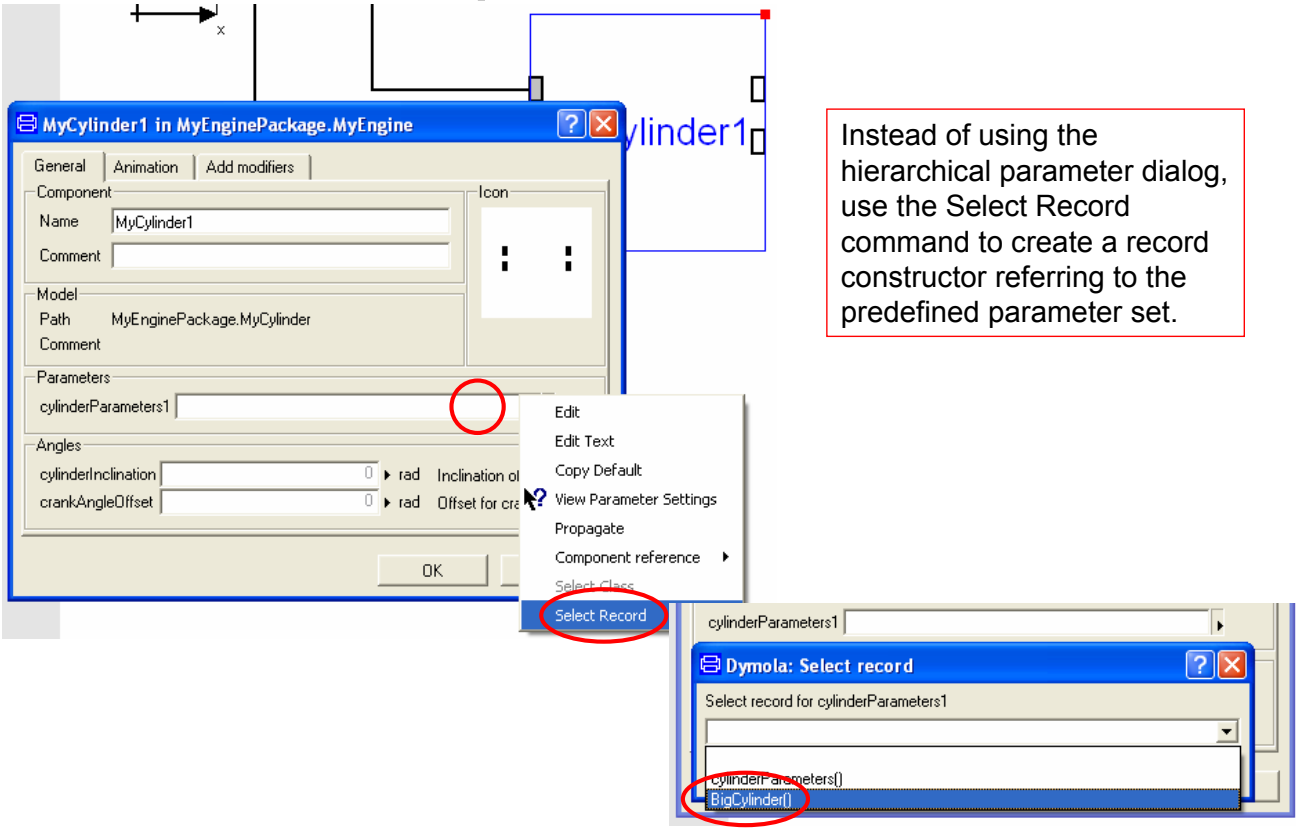





```
record BigCylinder
  extends cylinderParameters(
    cylinderTopPosition=0.84,
    pistonLength=0.2,
    rodLength=0.4,
    crankLength=0.2,
    crankPinOffset=0.2,
    crankPinLength=0.2);
end BigCylinder;
```

Give values to the parameter set by using the Parameter command of the extended class.

Use parameter set



Instead of using the hierarchical parameter dialog, use the Select Record command to create a record constructor referring to the predefined parameter set.

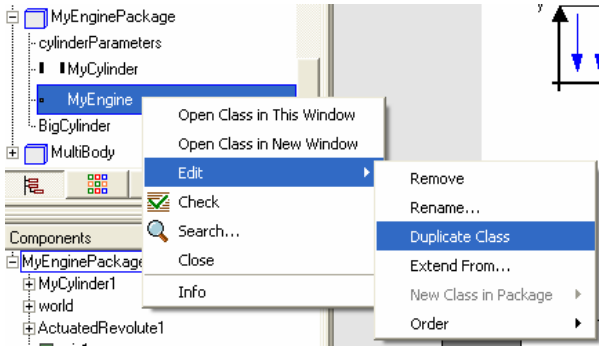
Resulting Modelica Code

```
model MyEngine
  extends EngineBase;
  MyCylinder MyCylinder1(cylinderParametersl=MyEnginePackage.BigCylinder())
    annotation (extent=[20, 0; 80, 60]);
  inner MultiBody.World world annotation (extent=[-80, 60; -60, 80]);
  annotation (Diagram);
  MultiBody.Joints.ActuatedRevolute ActuatedRevolute1(
    n={1,0,0},
    w_start=1,
    initType=MultiBody.Types.Init.PositionVelocity)
    annotation (extent=[-40, 30; -20, 10]);
  Modelica.Mechanics.Rotational.Interfaces.Flange_a axis1
    annotation (extent=[-110, -10; -90, 10]);
equation
  connect(world.frame_b, MyCylinder1.cylinder_a1) annotation (points=[-59, 70;
        -20, 70; -20, 40.8; 20, 40.8], style(color=0, thickness=2));
  connect(ActuatedRevolute1.frame_b, MyCylinder1.crank_a1) annotation (points=[
        -19, 20; 0, 20; 0, 27.6; 20, 27.6], style(color=0, thickness=2));
  connect(ActuatedRevolute1.frame_a, world.frame_b) annotation (points=[-41, 20;
         -50, 20; -50, 70; -59, 70], style(color=0, thickness=2));
  connect(ActuatedRevolute1.axis, axis1)
    annotation (points=[-30, 10; -30, 0; -100, 0], style(color=0));
end MyEngine;
```


Task 2. Array of cylinders

- Make array of cylinders
- Propagate parameter record to each cylinder
- Introduce arrays of individual angles
- Use for loop to connect them

Duplicate MyEngine. Call it MyEngineN. It will contain n cylinders.



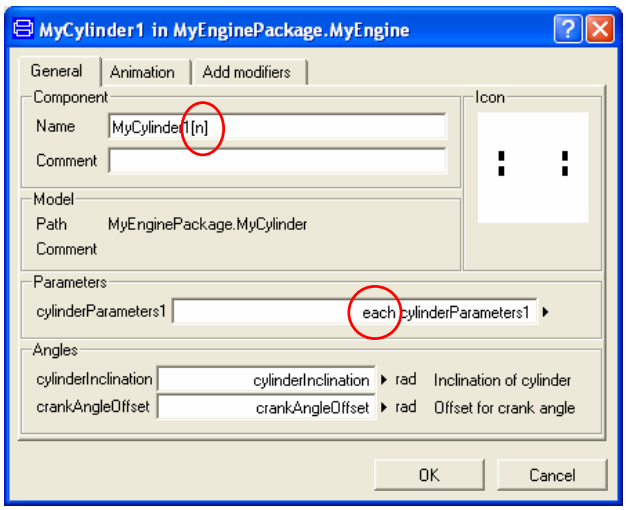
MyCylinder parameter dialog

MyEngineN parameters:

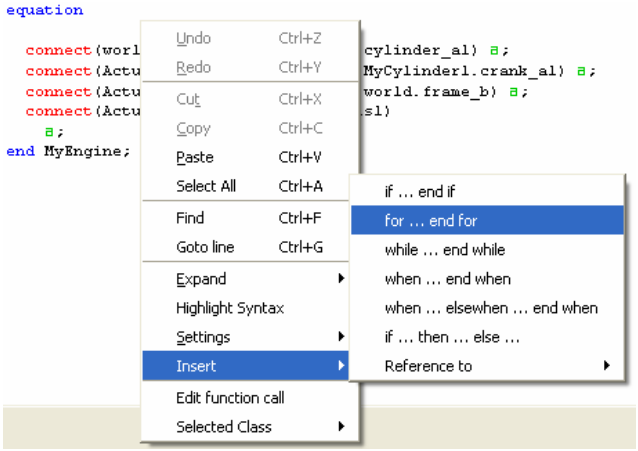
```
parameter cylinderParameters cylinderParameters1
  a;
parameter Integer n=1;
import SI = Modelica.SIunits;
parameter SI.Angle cylinderInclination[n]=zeros(n) "Inclination of cylinder"
  a;
parameter SI.Angle crankAngleOffset[n]=zeros(n) "Offset for crank angle"
  a;
```

For MyEngineN introduce a parameter record, n and the angle vectors.

Introduce an array of cylinders [n] and propagate the parameters. Use the each keyword to the cylinder parameters since all the cylinders have the same set.



Cylinder array connections



Use Modelica Text mode to generate a **for** statement and make the connections.

```
equation
  for i in 1:n - 1 loop
    connect(MyCylinder1[i].cylinder_b1, MyCylinder1[i + 1].cylinder_a1);
    connect(MyCylinder1[i].crank_b1, MyCylinder1[i + 1].crank_a1);
  end for;
  connect(world.frame_b, MyCylinder1[1].cylinder_a1) a;
  connect(ActuatedRevolute1.frame_b, MyCylinder1[1].crank_a1) a;
  connect(ActuatedRevolute1.frame_a, world.frame_b) a;
  connect(ActuatedRevolute1.axis, axis1)
    a;
end MyEngine;
```

Define V6 engine

Use the GUI to define a V6 engine.

```
model MyEngineV6
  extends MyEngineN(
    n=6,
    cylinderInclination=Modelica.SIunits.Conversions.from_deg({-30,30,-30,30,-30,
        30}),
    crankAngleOffset=Modelica.SIunits.Conversions.from_deg({-30,90,-150,-90,300,
        150}));
end MyEngineV6;
```


Task 3. Make engine test rig

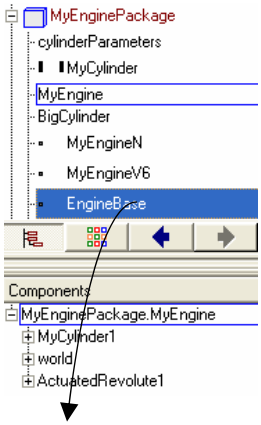
- For structurally different engines (with the same interface)
- Easy to replace engines
- Common engine property – one rotational flange

Introduce engine base

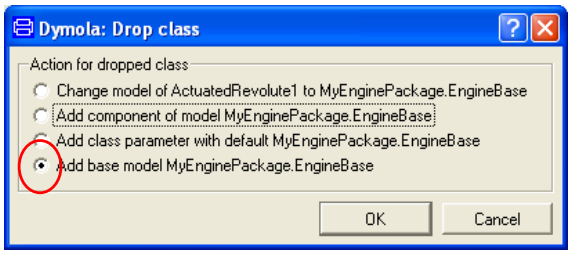
Make a partial model EngineBase with just a rotational flange.


```
partial model EngineBase
  Modelica.Mechanics.Rotational.Interfaces.Flange_a axis1
    annotation (extent=[-110, -10; -90, 10]);
end EngineBase;
```

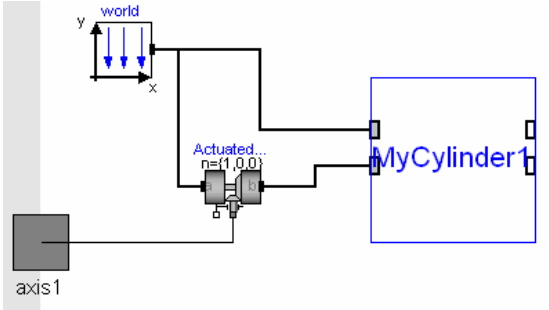

Complement MyEngine and MyEngineN



Drag base engine model to MyEngine



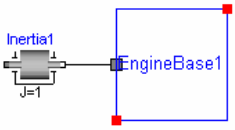
Add the base engine with the rotational flange to MyEngine and MyEngineN and connect the flange to the Revolute joint.

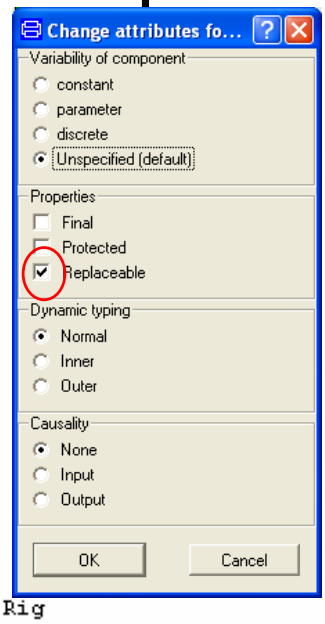


Modelica Text

```
model MyEngine
  extends EngineBase;
  MyCylinder MyCylinder1(cylinderParameters1=MyEnginePackage.BigCylinder())
    annotation (extent=[20, 0; 80, 60]);
  inner MultiBody.World world annotation (extent=[-80, 60; -60, 80]);
  annotation (Diagram);
  MultiBody.Joints.ActuatedRevolute ActuatedRevolute1(
    n={1,0,0},
    w_start=1,
    initType=MultiBody.Types.Init.PositionVelocity)
    annotation (extent=[-40, 30; -20, 10]);
equation
  connect(world.frame_b, MyCylinder1.cylinder_a1) annotation (points=[-59, 70;
        -20, 70; -20, 40.8; 20, 40.8], style(color=0, thickness=2));
  connect(ActuatedRevolute1.frame_b, MyCylinder1.crank_a1) annotation (points=[
        -19, 20; 0, 20; 0, 27.6; 20, 27.6], style(color=0, thickness=2));
  connect(ActuatedRevolute1.frame_a, world.frame_b) annotation (points=[-41, 20;
         -50, 20; -50, 70; -59, 70], style(color=0, thickness=2));
  connect(ActuatedRevolute1.axis, axis1)
    annotation (points=[-30, 10; -30, 0; -100, 0], style(color=0));
end MyEngine;
```


Use replaceable class



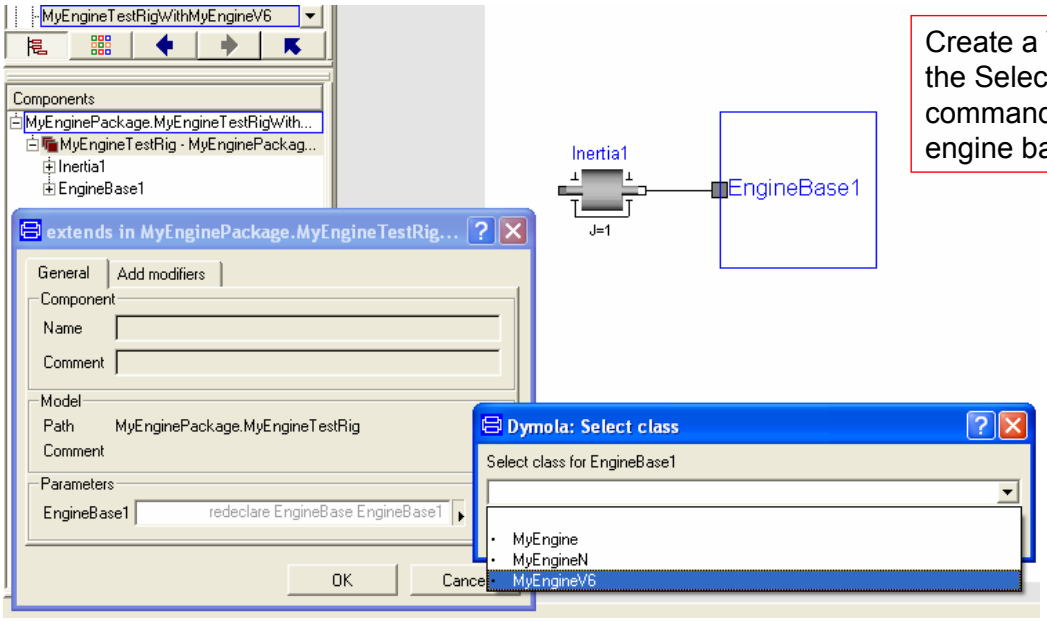


Make test rig model. Make a replaceable component from EngineBase. Add a load inertia.

Can be changed

```
model MyEngineTestRig
  Modelica.Mechanics.Rotational.Inertia Inertial
    annotation (extent=[-80, -10; -60, 10]);
  replaceable EngineBase EngineBase1 annotation (extent=[-40, -20; 0, 20]);
equation
  connect(Inertial.flange_b, EngineBase1.axis1)
    annotation (points=[-60, 0; -40, 0], style(color=0));
end MyEngineTestRig;
```

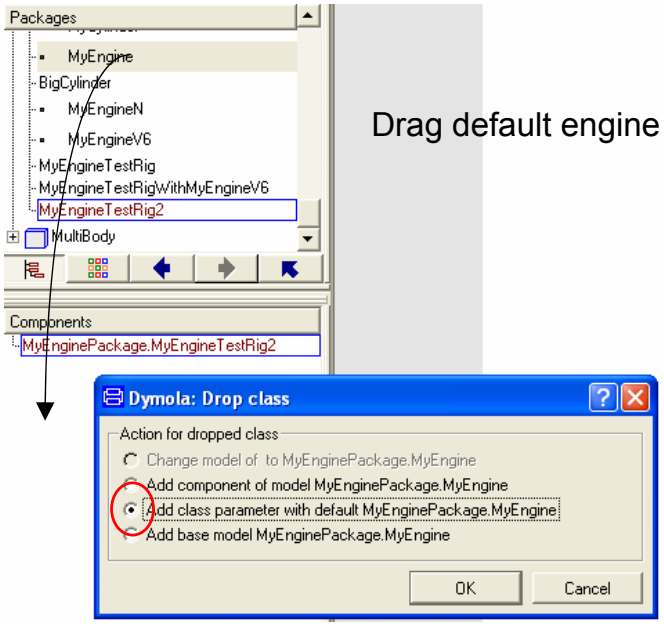
Mount a V6 in the rig



Create a V6 test rig. Use the Select Class command to replace the engine base by a V6.

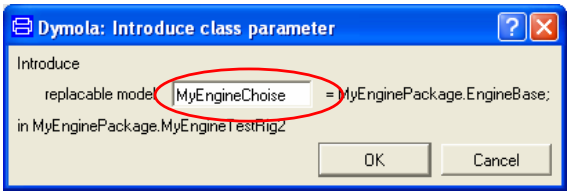
```
model MyEngineTestRigWithMyEngineV6
  extends MyEngineTestRig(redeclare MyEngineV6 MyEngine1);
end MyEngineTestRigWithMyEngineV6;
```


Task 3b. Use of class parameter

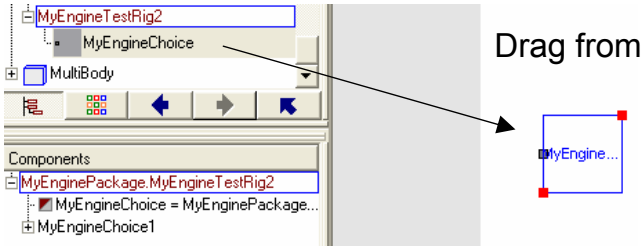


Drag default engine class

Create a class parameter MyEngineChoise based on EngineBase.



Class parameter component

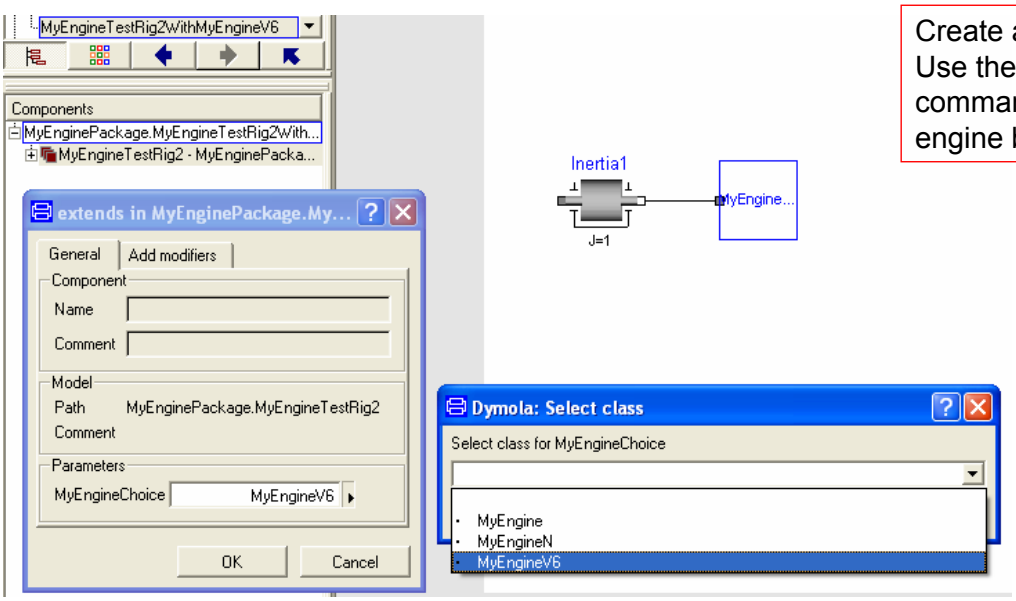


Create an engine component based on the engine base.

Can be changed

```
model MyEngineTestRig2
  replaceable model MyEngineChoice = EngineBase;
  MyEngineChoice MyEngineChoice1 annotation (extent=[-40, -10; -20, 10]);
  Modelica.Mechanics.Rotational.Inertia Inertial
    annotation (extent=[-80, -10; -60, 10]);
equation
  connect(Inertial.flange_b, MyEngineChoice1.axis1)
    annotation (points=[-60, 0; -49.9, 0; -49.9, 0; -40, 0], style(color=0));
end MyEngineTestRig2;
```


Mount a V6 in the rig 2



Create a new V6 test rig. Use the Select Class command to replace the engine base by a V6.

```
model MyEngineTestRig2WithMyEngineV6
  extends MyEngineTestRig2(redeclare model MyEngineChoice = MyEngineV6);
end MyEngineTestRig2WithMyEngineV6;
```