# Step 1: Build JaamSim Model

All events which triggers a control policy has to be modelled with the blocks under the HCCMLibrary, because only these blocks can send signals of the events to the controllers. All events which doesn’t trigger or aren’t triggered by a control policy can mostly be modelled by the normal blocks of the JaamSim library.

**HCCMActiveEntity:**

A renamed SimEntity without extra functionality.

This entity can be used for modelling an Active Entity

**HCCMArriveEvent:**

A modified EnitityGenerator with the ability to send an event signal to the specified controller(s) when an entity arrives in the system.

This entity can be used to generate Active Entities in the system.

**HCCMLeaveEvent:**

A modified EnititySink with the ability to send an event signal to the specified controller(s) when an entity leaves the system.

This entity can be used to remove Active Entities from the system.

**HCCMEntityDelay:**

A modified EntityDelay with the ability to send a start signal to the specified controller(s) when an entity starts the activity and an end signal to the specified controller(s) when an entity ends the activity.

This entity can be used for modelling a scheduled activity which only involves one active entity.

**HCCMControlActivity:**

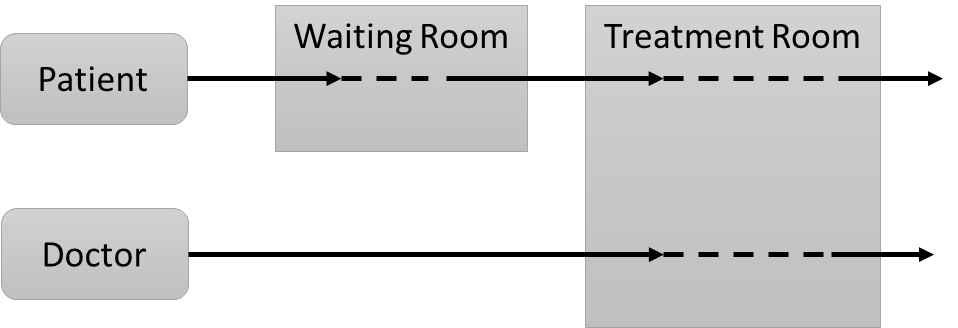
A modified Queue with the ability to send a start signal to the specified controller(s) when an entity starts the activity and an end signal to the specified controller(s) when an entity ends the activity.

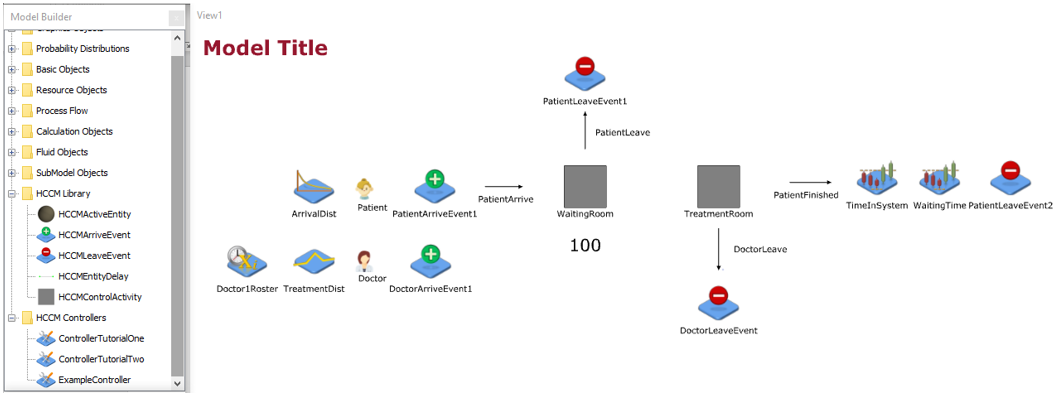
This entity can be used to model the following activities:

* Activity which ends by request
* Scheduled Activity with an active entity as server and an active entity as customer

These activities can happen in a Passive Entity.

**Example Tutorial One:**



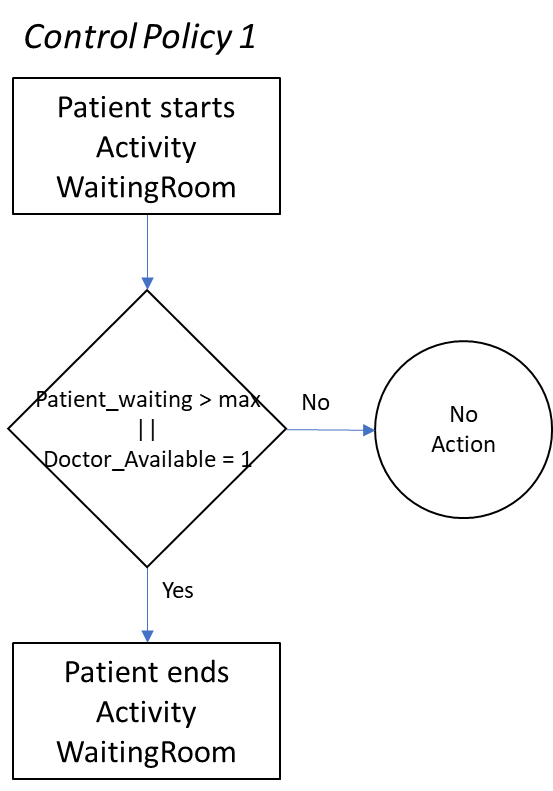
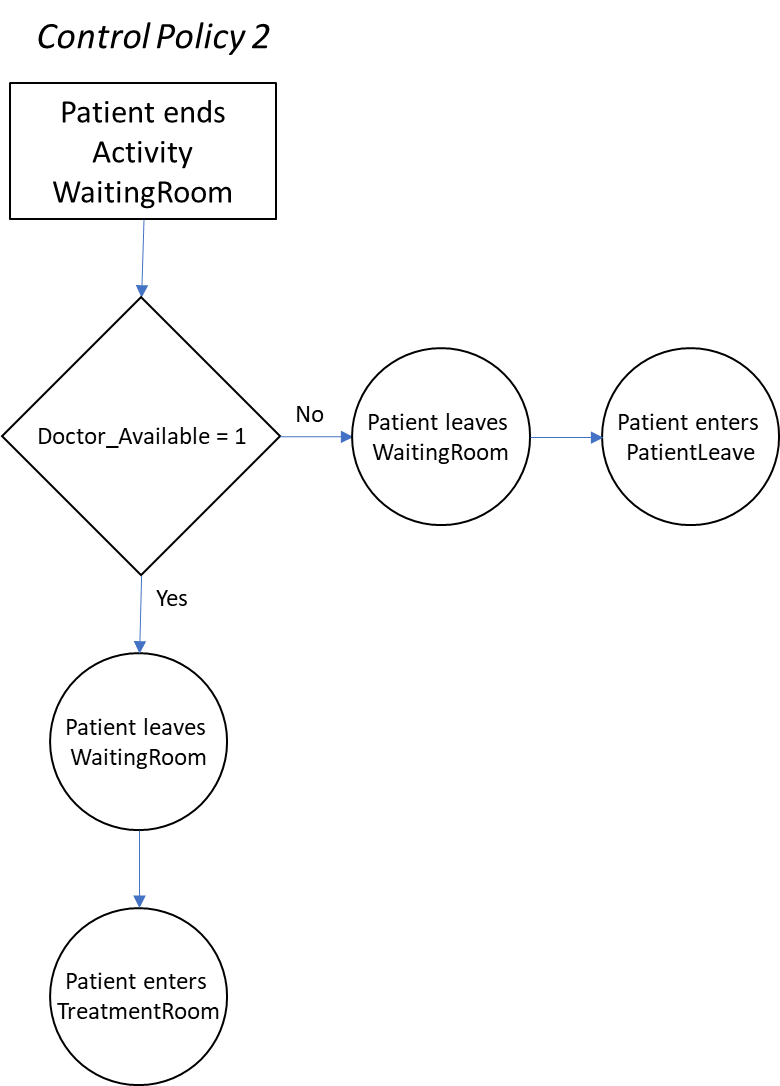
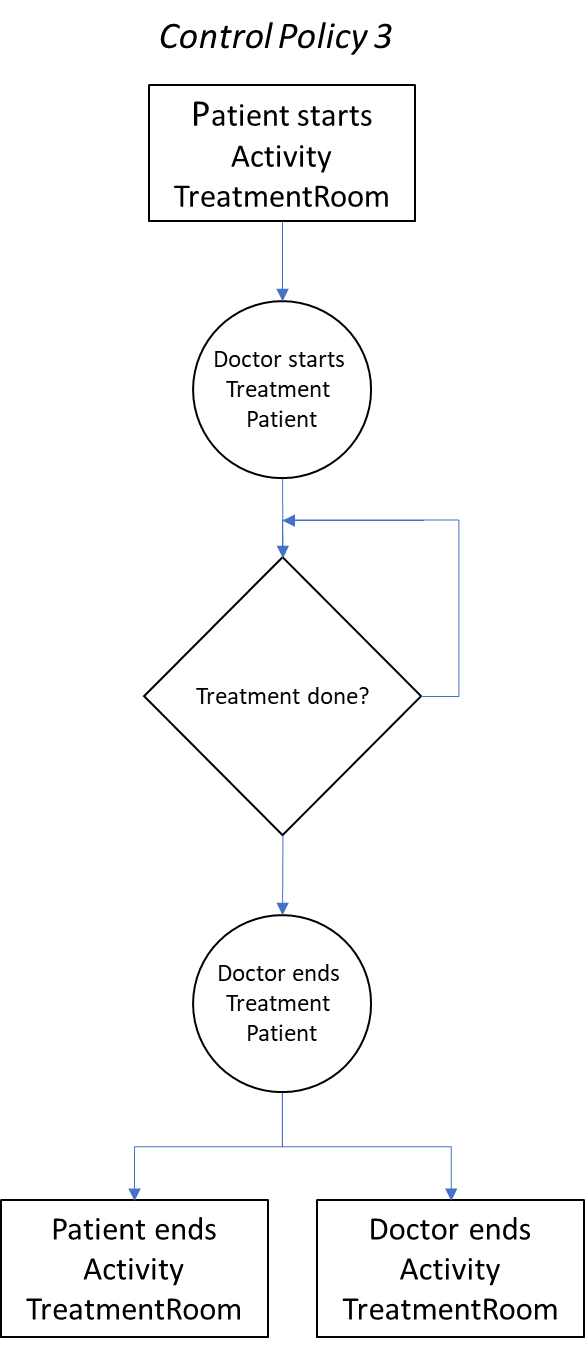


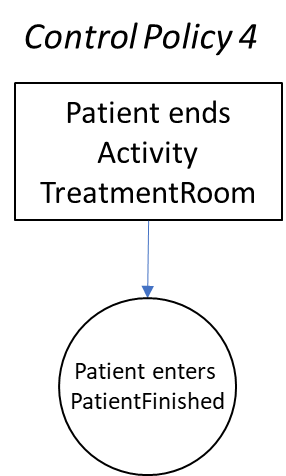
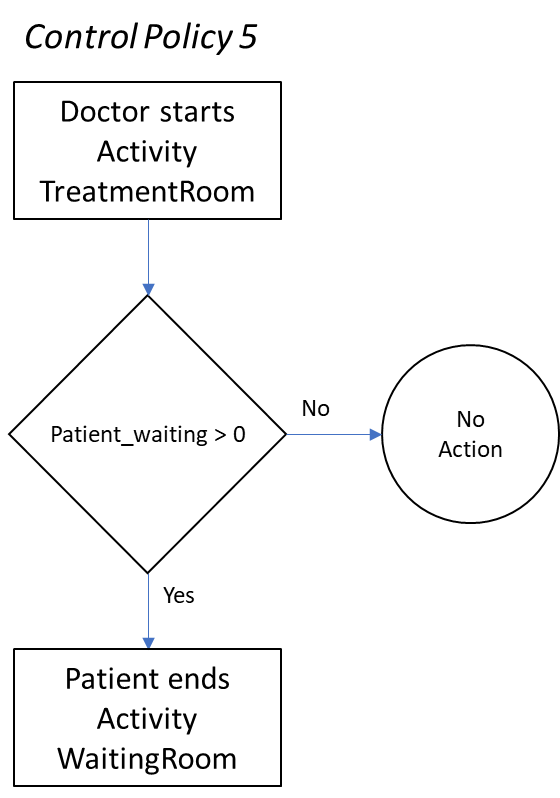
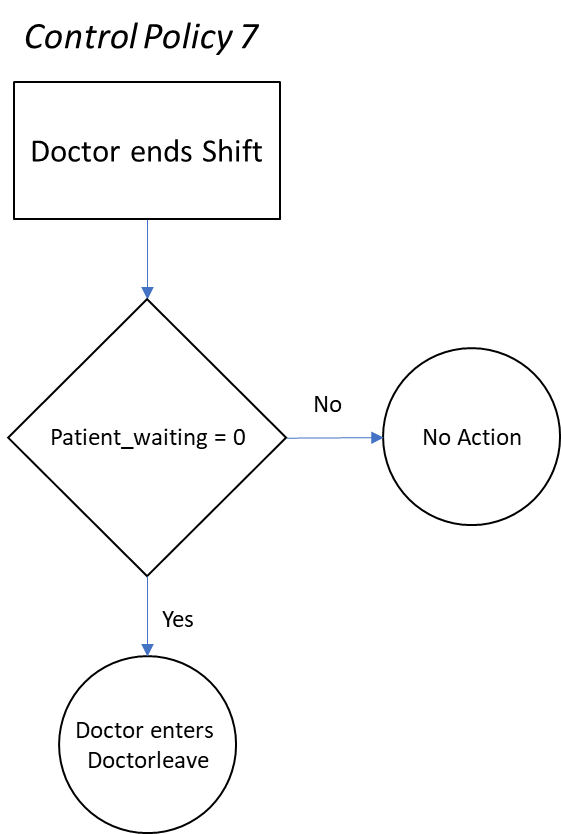
# Step 2: Convert Activities and Control Policies to HCCM Library structure

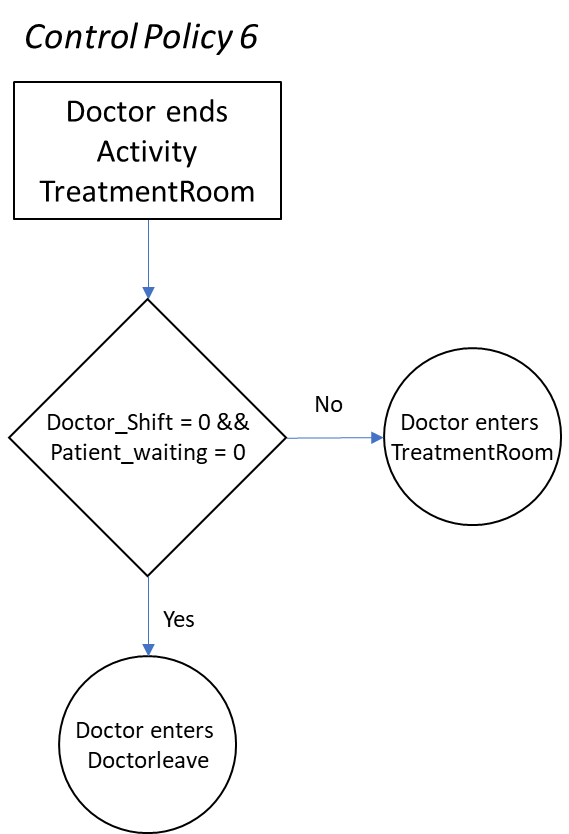
For the HCCM library in JaamSim it is needed to adjust the activities and the structure of the Control Policies of the Conceptual Model, because the JaamSim model only works if the correct signals are send to the controller and if the correct actions are triggered when a signal is received by the controller. For example, if the entity ControlActivity is used in the model it is needed to define the next component to which the active entity has to be send. And it has to be defined when the End signal has to be send.

**Example Tutorial One:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Signal** | **Triggered by:** | **Actions** | **Triggers:** |
| 1. | Patient StartActivity WaitingRoom | Patient enters WaitingRoom | Patient\_State = “Wait” | Control Policy 1 |
| 2. | Patient EndActivity WaitingRoom | Control Policy 1  Control Policy 5 |  | Control Policy 2 |
| 3. | Patient StartActivity TreatmentRoom | Patient enters TreatmentRoom | Doctor\_Available = 0  Doctor\_State = “Working”  Patient\_State = “Treat” | Control Policy 3 |
| 4. | Patient EndActivity TreatmentRoom | Control Policy 3 |  | Control Policy 4 |
| 5. | Doctor StartActivity TreatmentRoom | Doctor enters TreatmentRoom | Doctor\_Available = 1  Doctor\_State = “Idle” | Control Policy 5 |
| 6. | Doctor EndActivity TreatmentRoom | Control Policy 3 | Doctor\_Available = 0  Doctor\_State = “” | Control Policy 6 |
| 7. | Doctor ends Shift | TimeSeries |  | Control Policy 7 |

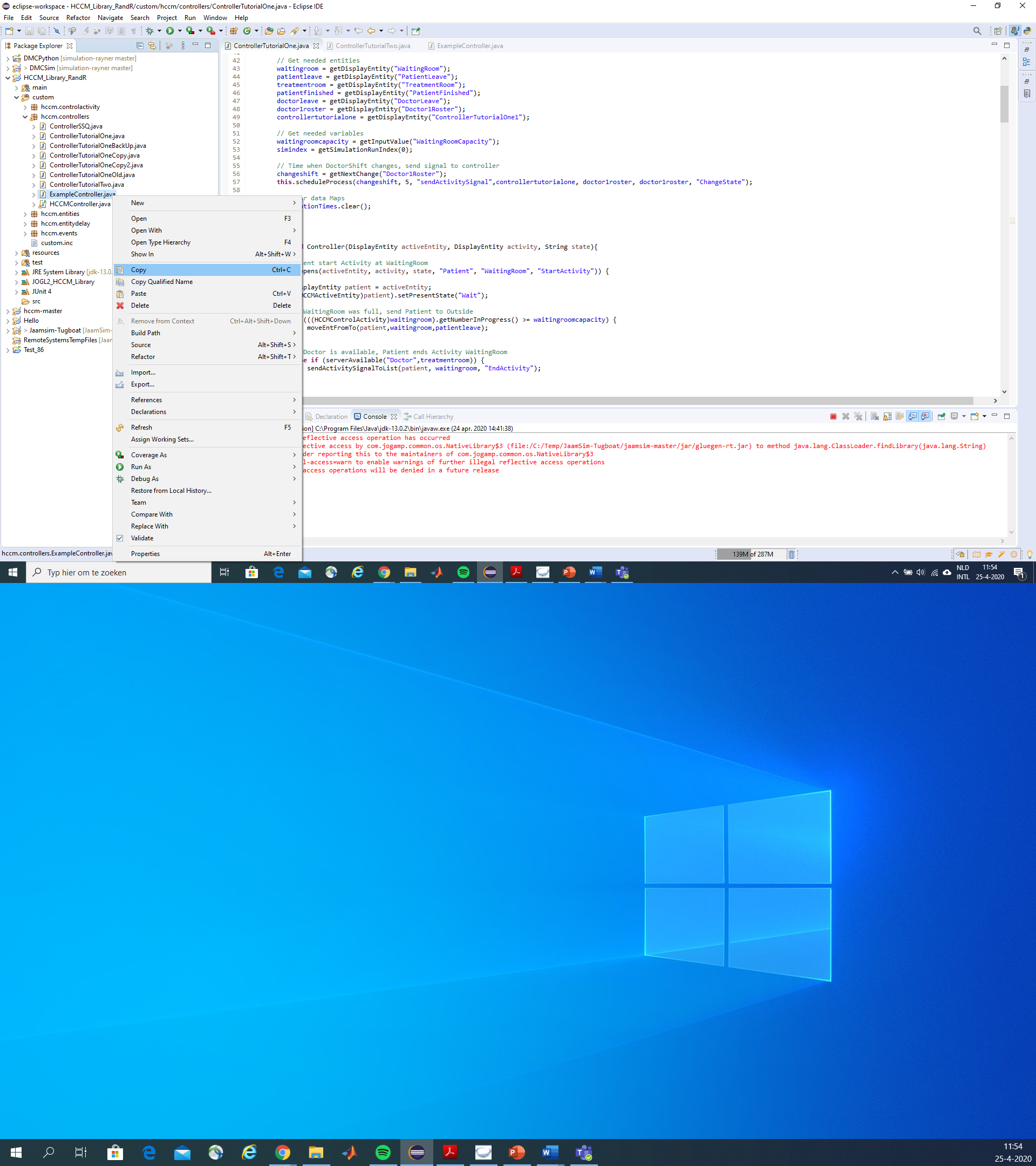






# Step 3: Create controller(s)

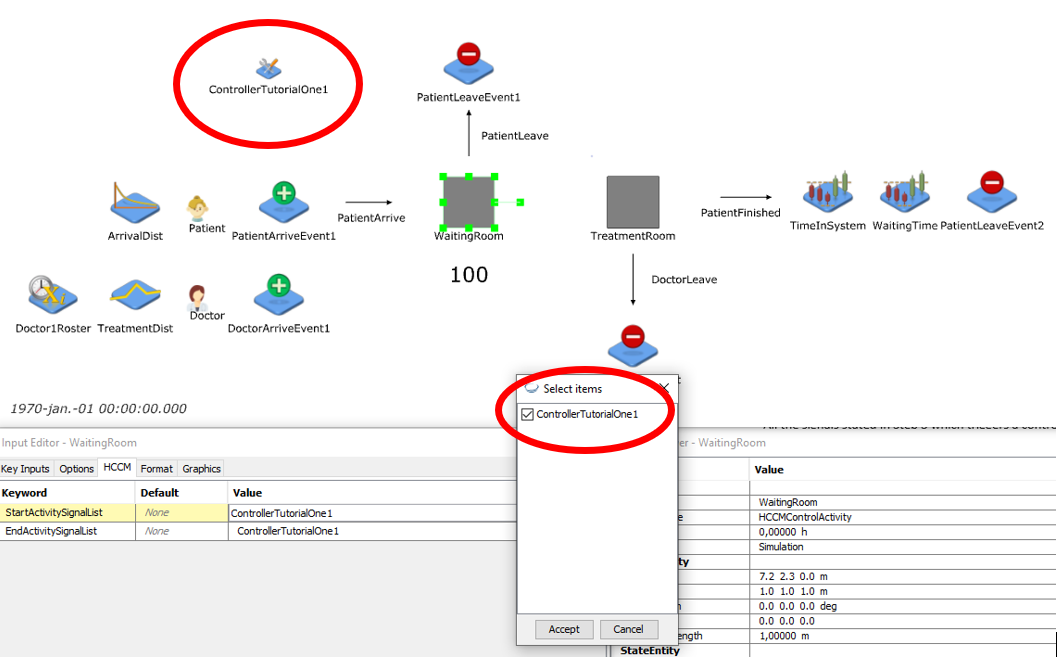
## Step 3.1 Create java file:

* Make a copy of ExampleController.java and give it a specific name (figure …)
* Add this controller to the library by copying the code of the ExampleController in the custom.inc (figure ..)
* 



## Step 3.2 Add controller to JaamSim model

To be able to send signals to the controller, the controller has to be added to the JaamSim model. All the signals (stated in Step 2) which triggers a control policy has to be send to the correct controller. This has to be done in the JaamSim model within the correct entities.

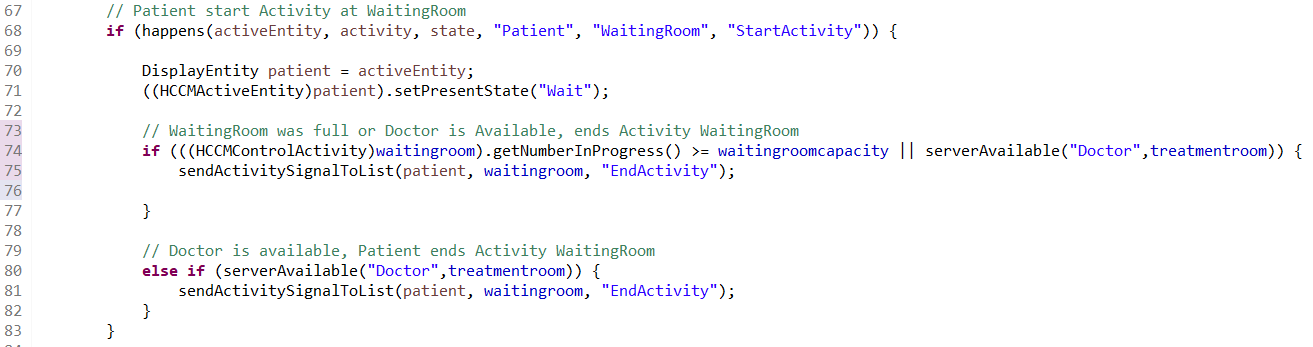


# Step 4: Program the Controller

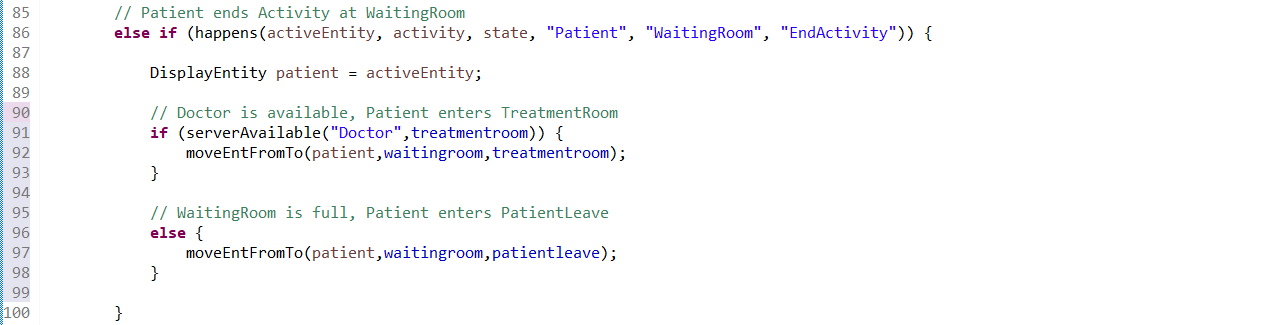
The Control policies stated in step 2 has to be programmed in the java script of the controller. Most of the needed functions are defined in the ExampleController.java but it is possible to add extra code and functions to the java script.

**Example Tutorial One:**

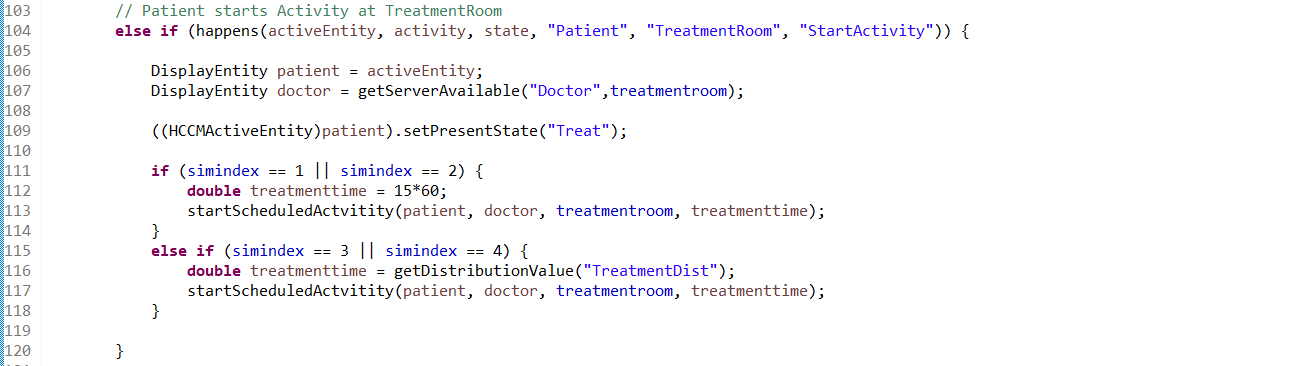
**Control Policy 1:**



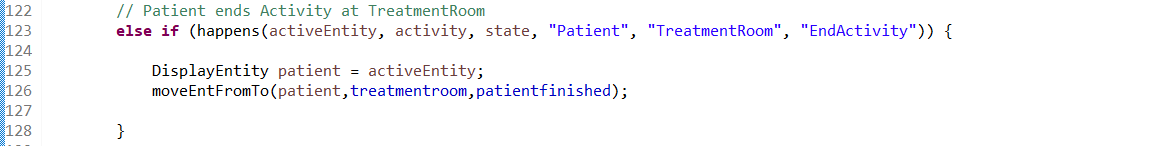
**Control Policy 2:**



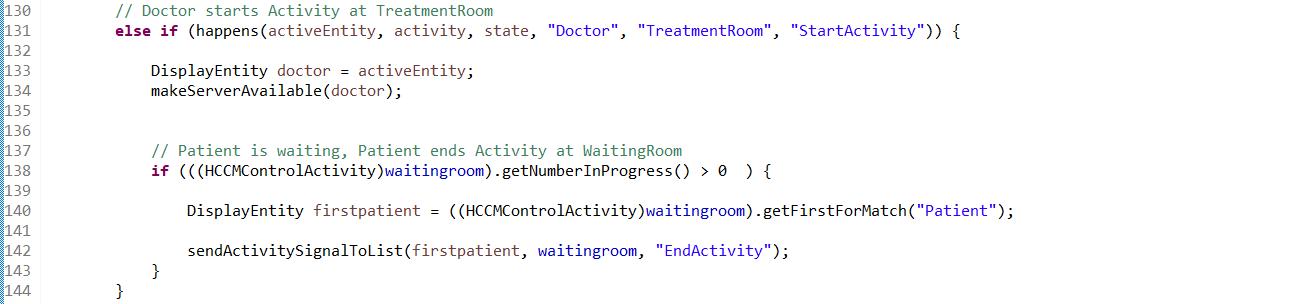
**Control Policy 3:**



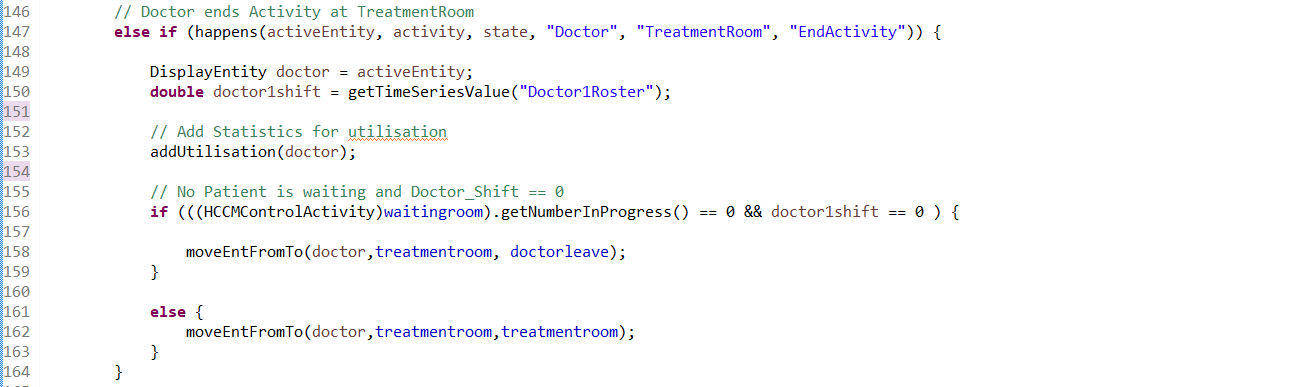
**Control Policy 4:**



**Control Policy 5:**



**Control Policy 6:**



**Control Policy 7:**

