Setting Up JaamSim and HCCM

In this lab you will walk through the set up of how to run JaamSim from source in an Eclipse project and how to include the HCCM library in JaamSim.

1 Prerequisites

These instructions were prepared using:

1. Git – 2.28.0;
2. GitHub Desktop – 2.9.3 (x64);
3. Java – Amazon Corretto JRE 8;
4. Eclipse for Java Developers – 2021-06.

They should work with more recent versions of the software too. All of this software is standard on Engineering lab machines and Eclipse can also be accessed via FlexIT. Corretto OpenJDK is available on University of Auckland's Software Centre.

If you wish to use FlexIT, go to `flexit.auckland.ac.nz` and login, otherwise go to step 2. Then search for Eclipse – see figure 1, click on the Eclipse link and install the VMware Horizon Client – see figure 2 – if needed. You can install it using Software Centre on University of Auckland machines. Once Horizon Client is installed then you can go back to `flexit.auckland.ac.nz` and run Eclipse. You should enable Eclipse (via Horizon Client) to access your local storage so you can access all the folders and files we will use here.

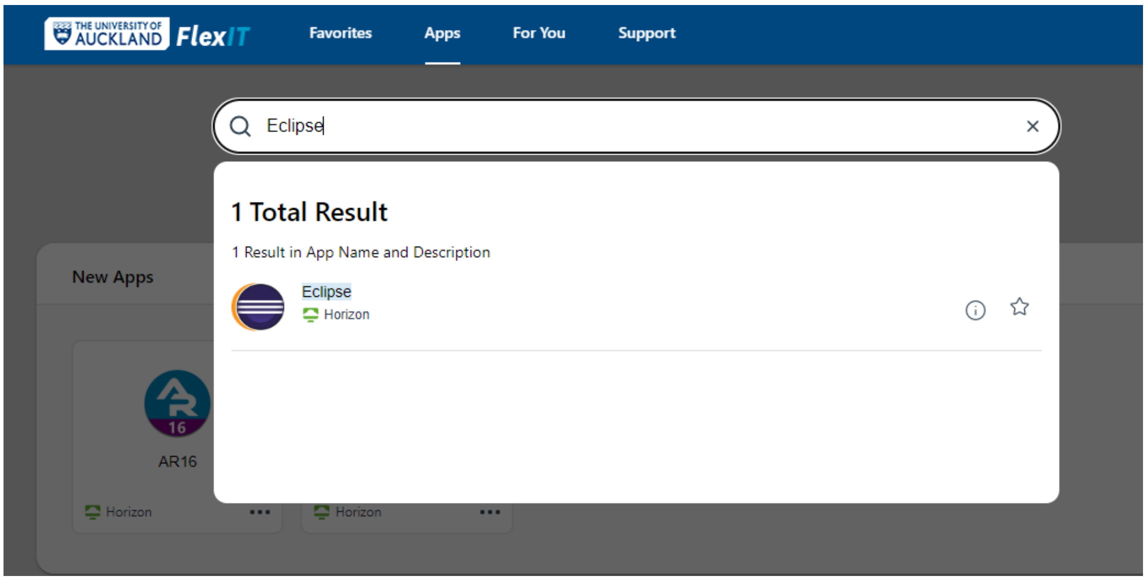


Figure 1: Screenshot of finding Eclipse in FlexIT

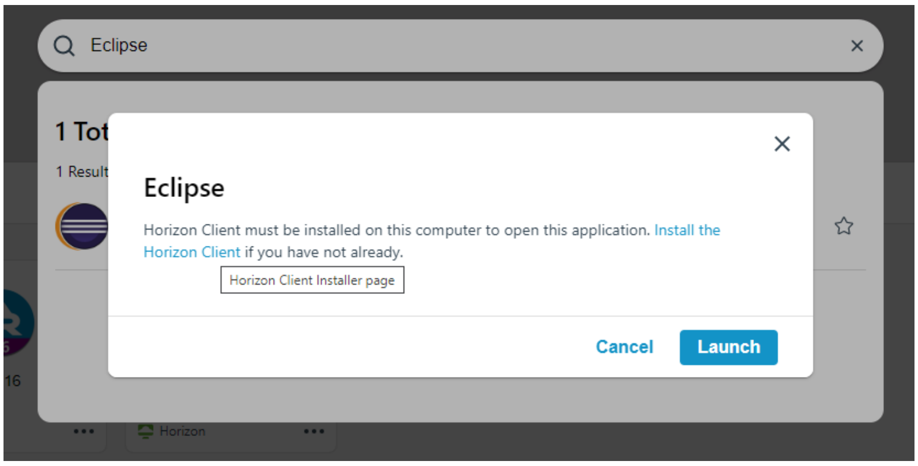

Eclipse
1 Tot
1 Result
Eclipse
Horizon Client must be installed on this computer to open this application. Install the Horizon Client if you have not already.
Horizon Client Installer page
Cancel
Launch
Horizon

Figure 2: Screenshot of starting the VMware Horizon Client install

2 Create the project folder structure

Create a new folder under EngFiles on your H drive called ENGSCI355. Then create two folders within this one, called "sim" and "labs". The "sim" folder will contain the java code for the simulation software Jaamsim, including custom code that you write, and is the focus of these instructions. The "labs" folder will contain subfolders for each lab with the simulation files for each. Create a folder for HCCM logic functions within the "sim" folder. We will use "sim_custom" in these instructions. Figure 3 shows a screenshot of the folder structure.

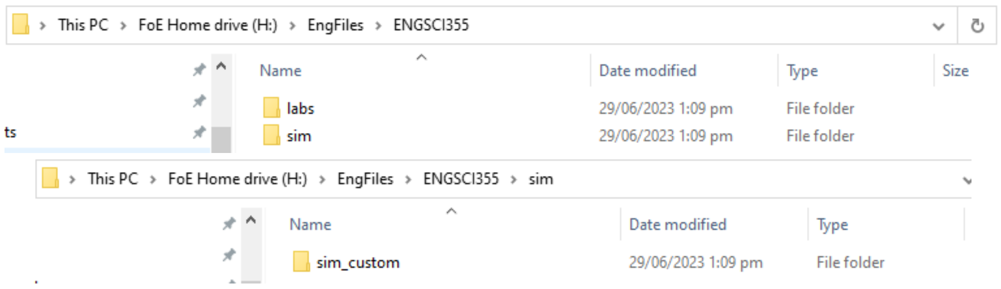

This PC > FoE Home drive (H:) > EngFiles > ENGSCI355
Name
Date modified
Type
Size
labs 29/06/2023 1:09 pm File folder
sim 29/06/2023 1:09 pm File folder
ts
This PC > FoE Home drive (H:) > EngFiles > ENGSCI355 > sim
Name
Date modified
Type
sim_custom 29/06/2023 1:09 pm File folder

Figure 3: Screenshot of folder structure

3 Clone HCCM into the project folder

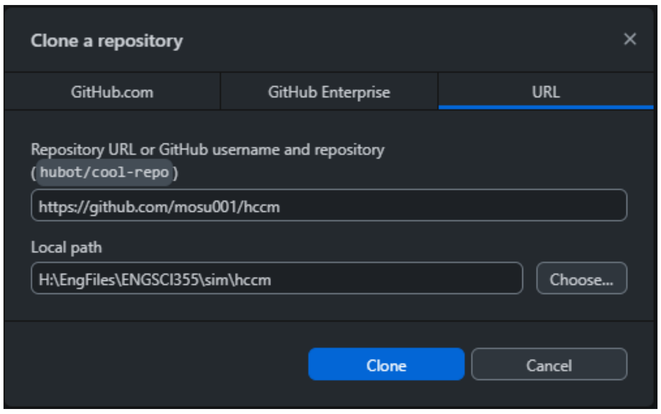
Open GitHub Desktop and clone

```
https://github.com/mosu001/hccm
```

into an hccm folder within the "sim" folder. Figures 4 and 5 are screenshots of the clone process and HCCM folder structure (within the main project folder) respectively.

Note, if you use git from the command line, e.g., Git Bash, you need to add the recurse submodules option

```
git clone --recurse-submodules https://github.com/mosu001/hccm
```


Clone a repository

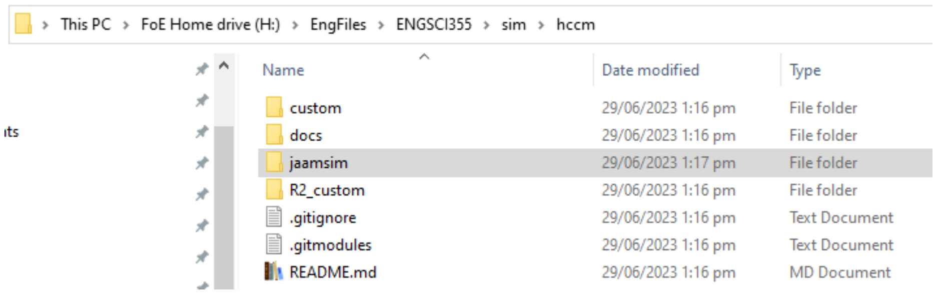
Repository URL or GitHub username and repository (`hubot/cool-repo`)

https://github.com/mosu001/hccm

Local path

H:\EngFiles\ENGSCI355\sim\hccm

Figure 4: Screenshot of cloning the HCCM repository into the project folder


This PC › FoE Home drive (H:) › EngFiles › ENGSCI355 › sim › hccm

Name	Date modified	Type
custom	29/06/2023 1:16 pm	File folder
docs	29/06/2023 1:16 pm	File folder
jaamsim	29/06/2023 1:17 pm	File folder
R2_custom	29/06/2023 1:16 pm	File folder
.gitignore	29/06/2023 1:16 pm	Text Document
.gitmodules	29/06/2023 1:16 pm	Text Document
README.md	29/06/2023 1:16 pm	MD Document

Figure 5: Screenshot of HCCM folder structure

4 Create files to load HCCM and customised components

Create a file autoload.cfg in sim_custom with the content shown in figure 6 (note that this is an extension of autoload.cfg from hccm\custom). This includes all the HCCM components and enables components within sim_custom to be included.

```
Include units.inc
Include sim.inc
Include units-imperial.inc
Include units-knots.inc
Include displayModels.inc
Include graphics.inc
Include probabilityDistributions.inc
Include basicObjects.inc
Include resourceObjects.inc
Include examples.inc
Include processFlow.inc
Include calculationObjects.inc
Include fluidObjects.inc
Include submodels.inc
Include hccm.inc
Include sim_custom.inc
```

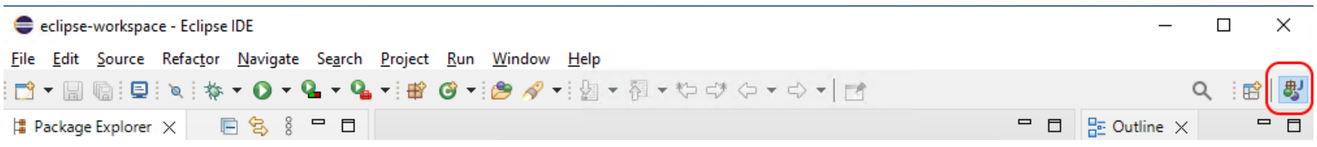
Figure 6: Customised autoload.cfg file

Add a blank sim_custom.inc text file to sim_custom and leave it blank for now.

5 Create an Eclipse project in the main project folder

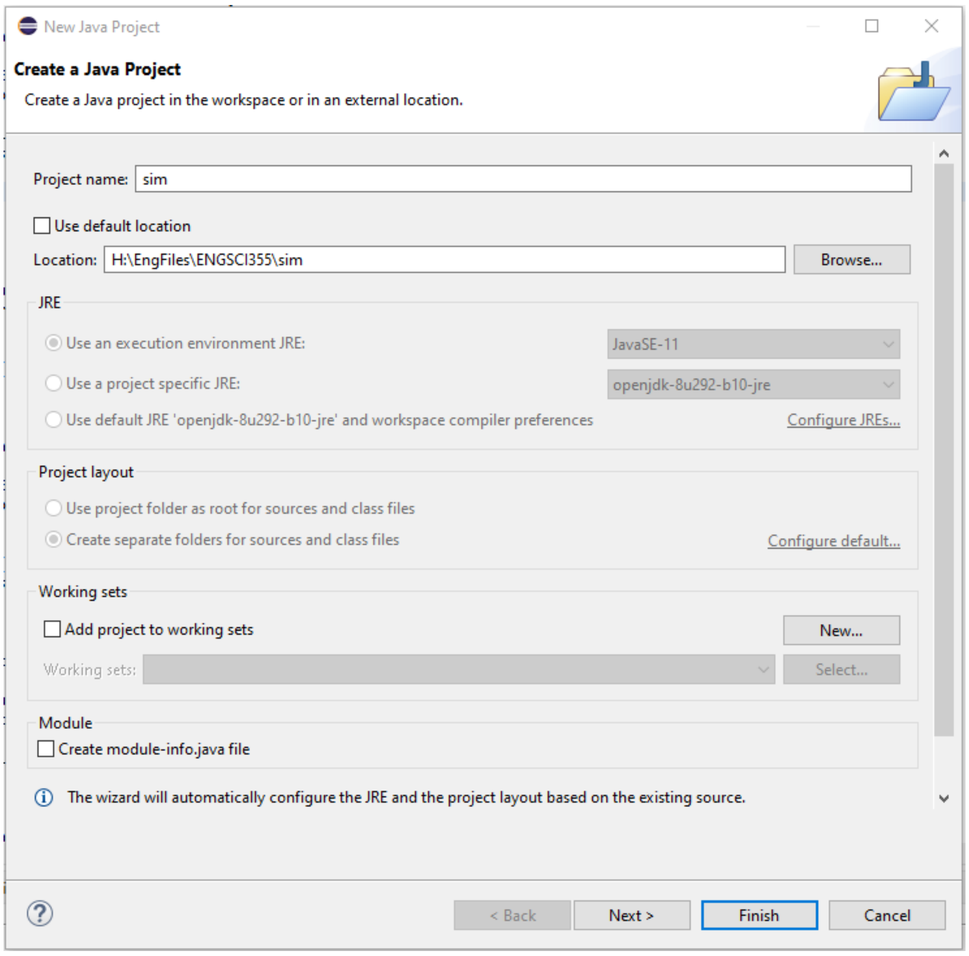
Open Eclipse, make sure you are in the Java perspective (highlighted in red) – see figure 7, then select File > New > Java Project to create a new Java project. Turn off Use default location and browse to the "sim" folder in Location. Turn off Create module-info.java file, you may have to click Next before you see the option to turn this off. Figure 8 shows a screenshot of the project create dialog.

Click on Finish.



eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Outline

Figure 7: Screenshot showing Java perspective



New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: sim

☐ Use default location

Location: H:\EngFiles\ENGSCI355\sim

JRE

- ● Use an execution environment JRE: JavaSE-11
- ○ Use a project specific JRE: openjdk-8u292-b10-jre
- ○ Use default JRE 'openjdk-8u292-b10-jre' and workspace compiler preferences

Project layout

- ○ Use project folder as root for sources and class files
- ● Create separate folders for sources and class files

Working sets

☐ Add project to working sets

Working sets:

Module

☐ Create module-info.java file

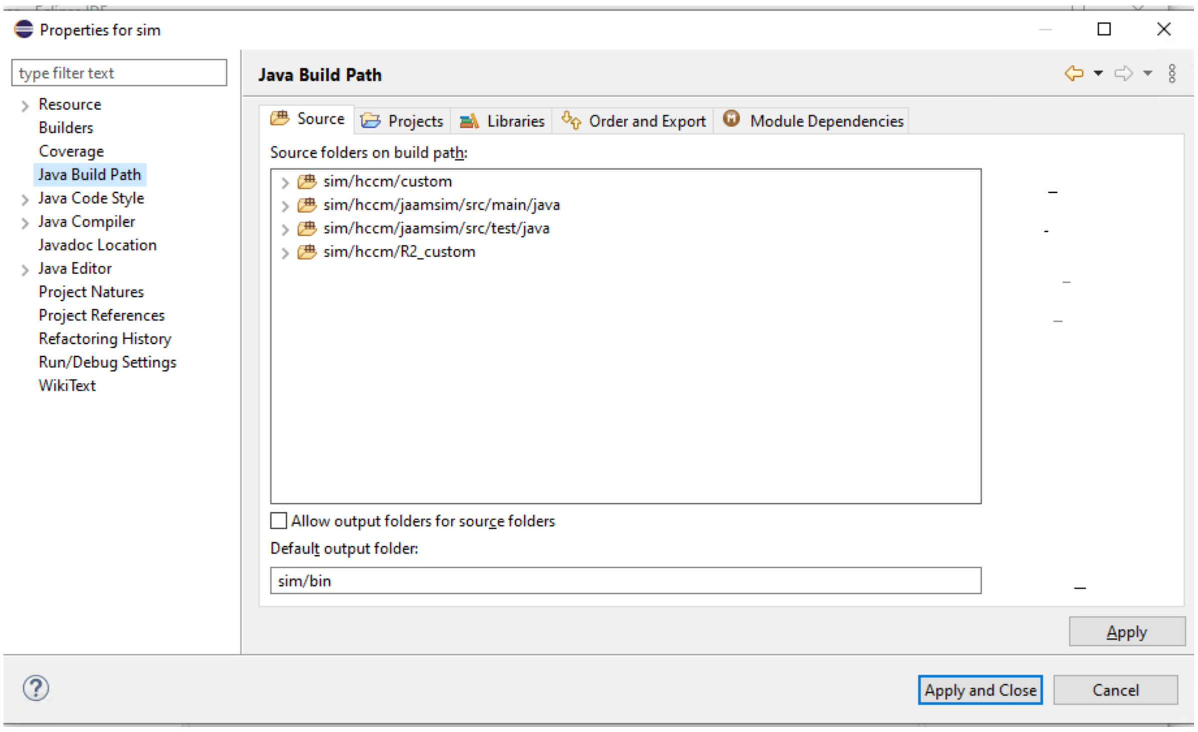
The wizard will automatically configure the JRE and the project layout based on the existing source.

Figure 8: Screenshot of Java project creation

6 Configure Source Folders

Right click on your project – it looks like a folder called "sim" in the Package Explorer pane – and select Properties. Click on Java Build Path in the list of properties on the left.

Go to the Source tab. Figure 9 shows a screen shot of what you should see.



Source folders on build path:

- sim/hccm/custom
- sim/hccm/jaamsim/src/main/java
- sim/hccm/jaamsim/src/test/java
- sim/hccm/R2_custom

☐ Allow output folders for source folders

Default output folder: sim/bin

Figure 9: Screenshot of initial Source tab in project properties

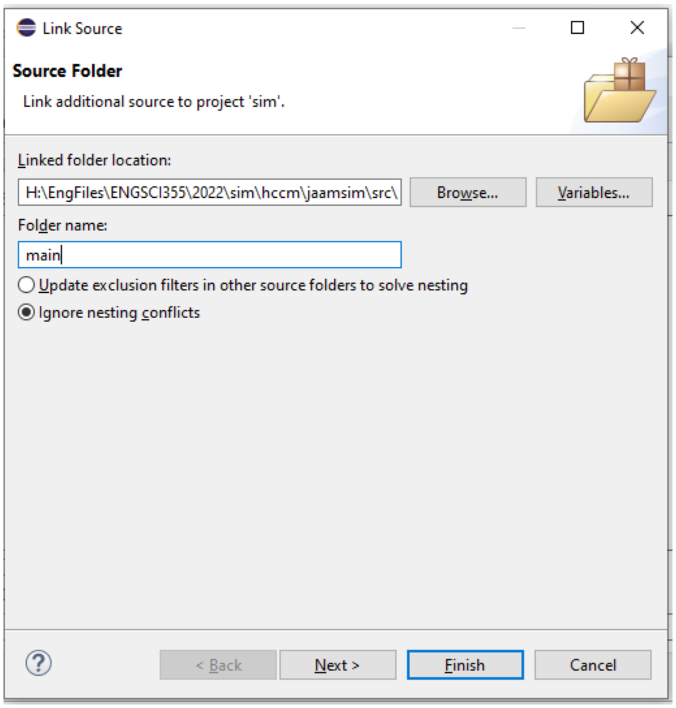
Select sim\hccm\jaamsim\src\main\java and Remove. Also remove sim\hccm\jaamsim\src\test\java and sim\hccm\R2_custom.

Click on Link Source and browse to sim\hccm\jaamsim\src\main\java and set the Folder name to be main. Click on Finish. Figure 10 shows the dialog box for this step.

Similarly use Link Source to set hccm\jaamsim\src\test\java to have the Folder name of test.

Use Link Source one last time to set hccm\jaamsim\src\main\resources to have the Folder name of resources.

Finally, click on Add Folder to add sim\sim_custom. Figure 11 shows the dialog box when all the folders have been configured properly.


Link Source

Source Folder

Link additional source to project 'sim'.

Linked folder location:

H:\EngFiles\ENGSCI355\2022\sim\hccm\jaamsim\src\

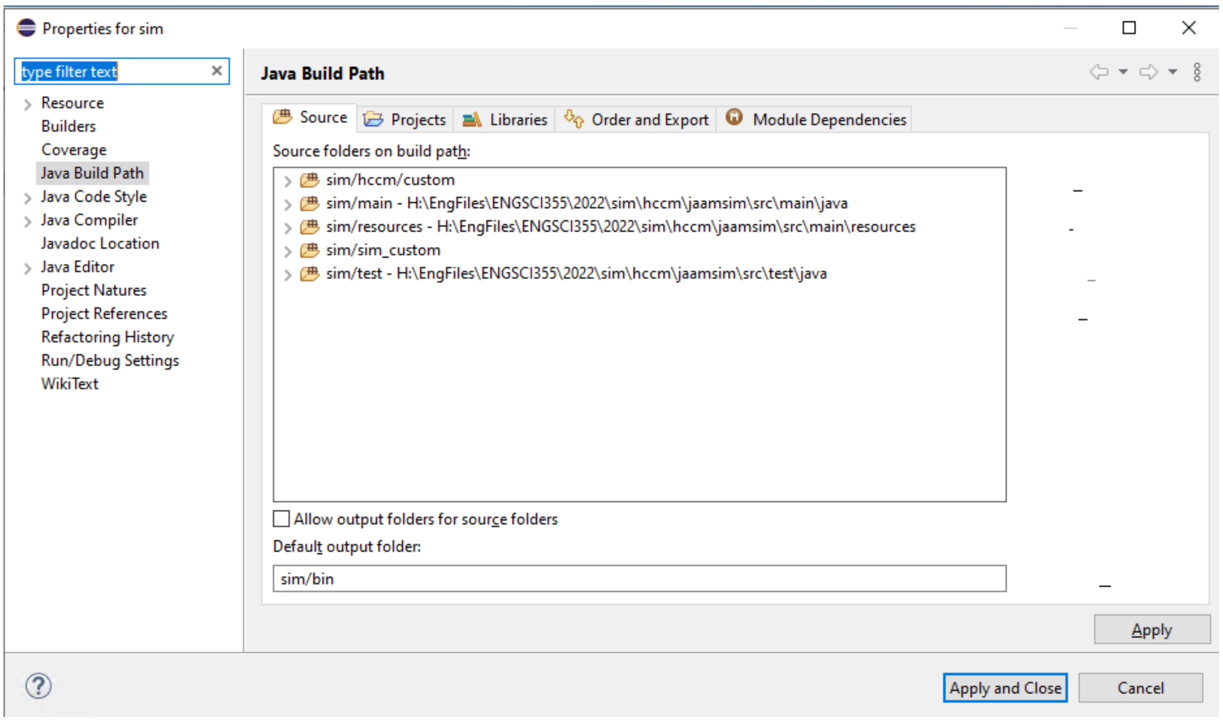
Folder name:

main

( ) Update exclusion filters in other source folders to solve nesting

(●) Ignore nesting conflicts

Figure 10: Screenshot of Link Source dialog box for defining main


Properties for sim

Java Build Path

Source folders on build path:

- sim/hccm/custom
- sim/main - H:\EngFiles\ENGSCI355\2022\sim\hccm\jaamsim\src\main\java
- sim/resources - H:\EngFiles\ENGSCI355\2022\sim\hccm\jaamsim\src\main\resources
- sim/sim_custom
- sim/test - H:\EngFiles\ENGSCI355\2022\sim\hccm\jaamsim\src\test\java

[ ] Allow output folders for source folders

Default output folder:

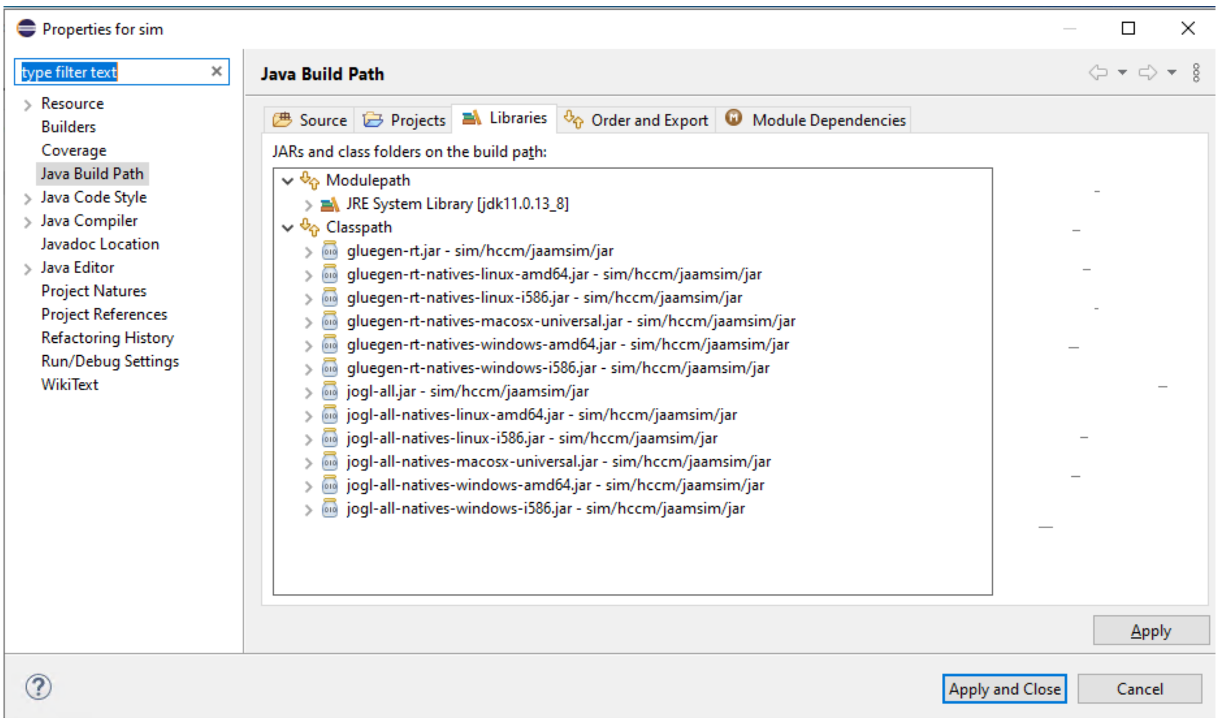
sim/bin

Figure 11: Screenshot of project properties after folders are configured

7 Configure Libraries

Next, click on the Libraries tab (in the project properties dialog) – see figure 12.

You might see Modulepath and Classpath or just a list of libraries. I will give instructions


Properties for sim

type filter text

Resource
Builders
Coverage
Java Build Path
Java Code Style
Java Compiler
Javadoc Location
Java Editor
Project Natures
Project References
Refactoring History
Run/Debug Settings
WikiText

Java Build Path

Source | Projects | Libraries | Order and Export | Module Dependencies

JARs and class folders on the build path:

```
Modulepath
  JRE System Library [jdk11.0.13_8]
Classpath
  gluegen-rt.jar - sim/hccm/jaamsim/jar
  gluegen-rt-natives-linux-amd64.jar - sim/hccm/jaamsim/jar
  gluegen-rt-natives-linux-i586.jar - sim/hccm/jaamsim/jar
  gluegen-rt-natives-macosx-universal.jar - sim/hccm/jaamsim/jar
  gluegen-rt-natives-windows-amd64.jar - sim/hccm/jaamsim/jar
  gluegen-rt-natives-windows-i586.jar - sim/hccm/jaamsim/jar
  jogl-all.jar - sim/hccm/jaamsim/jar
  jogl-all-natives-linux-amd64.jar - sim/hccm/jaamsim/jar
  jogl-all-natives-linux-i586.jar - sim/hccm/jaamsim/jar
  jogl-all-natives-macosx-universal.jar - sim/hccm/jaamsim/jar
  jogl-all-natives-windows-amd64.jar - sim/hccm/jaamsim/jar
  jogl-all-natives-windows-i586.jar - sim/hccm/jaamsim/jar
```

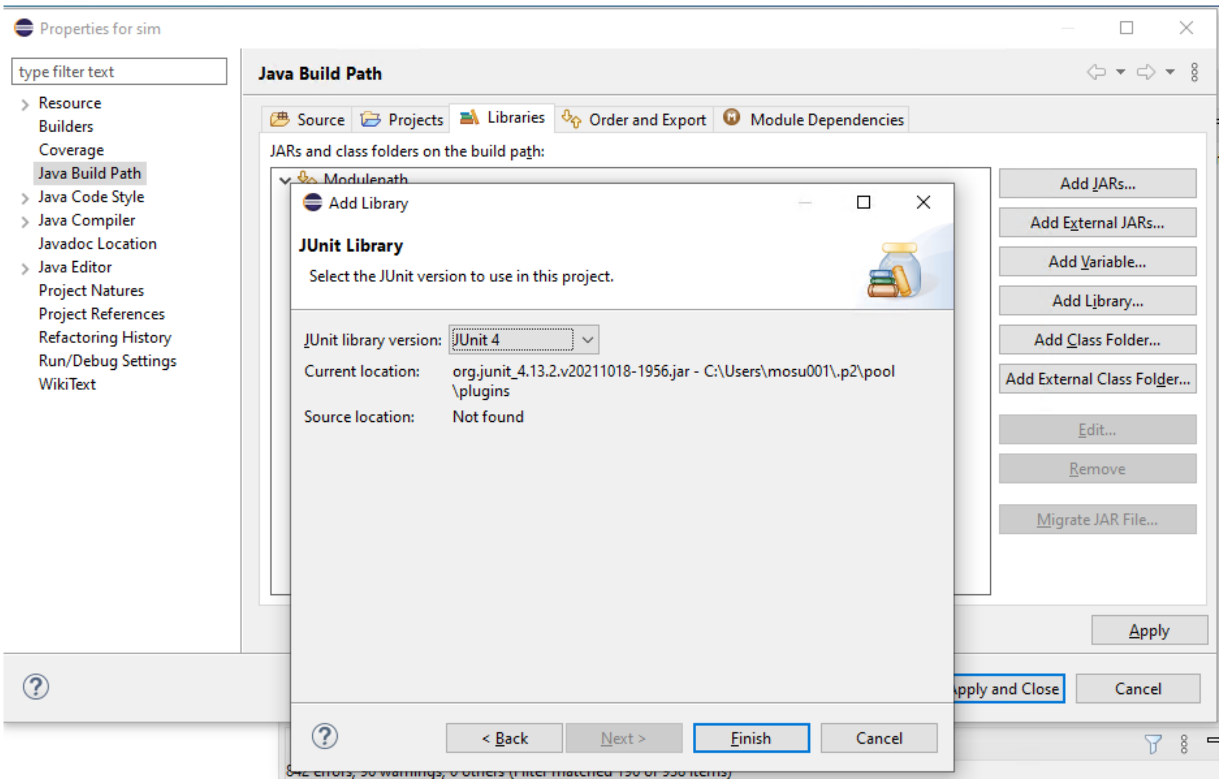
Figure 12: Screenshot of the intial libraries in project properties

for the Modulepath/Classpath configuration but you can follow these instructions for the list of libraries too.

Select all the .jar files on the Classpath and Remove them.

Then select the Classpath and select Add Library. . .

Choose Junit, then Junit 4 and Finish. See figure 13 for a screenshot of adding the JUnit library.


Properties for sim

Java Build Path

JARs and class folders on the build path:

Add Library

JUnit Library

Select the JUnit version to use in this project.

JUnit library version: JUnit 4

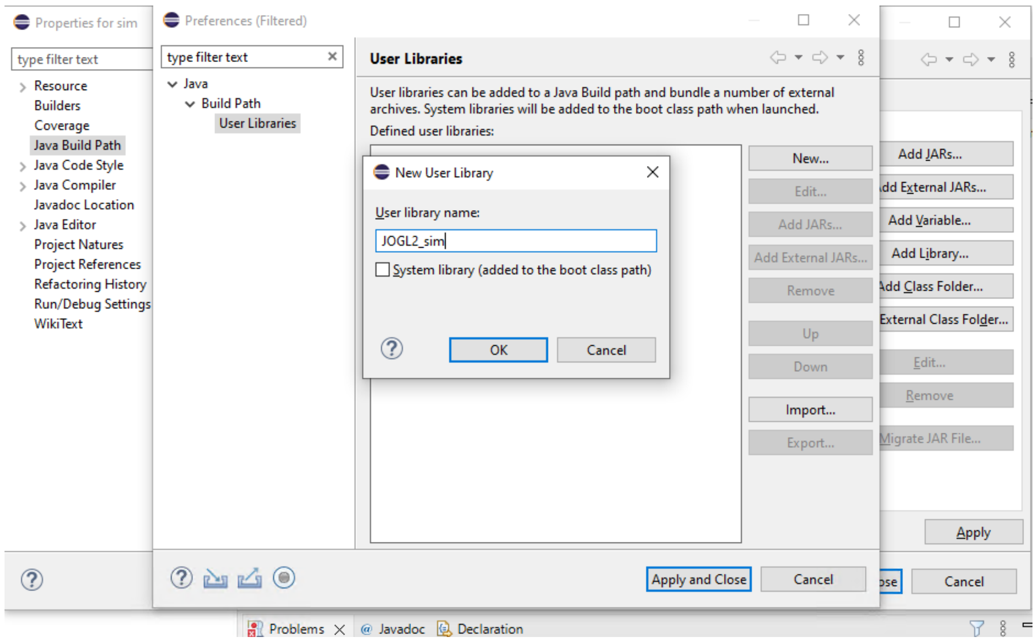
Current location: org.junit_4.13.2.v20211018-1956.jar - C:\Users\mosu001\.p2\pool\plugins

Source location: Not found

Figure 13: Screenshot of JUnit selection

Select Classpath again and then click on Add Library. . .

Select User Library, then Next, then User Libraries, then New. . . and name the library JOGL2 – see figure 14. If JOGL2 already exists then use the name JOGL2_sim (this can be due to an older library or multiple HCCM or custom JaamSim projects).



User libraries can be added to a Java Build path and bundle a number of external archives. System libraries will be added to the boot class path when launched.

Defined user libraries:

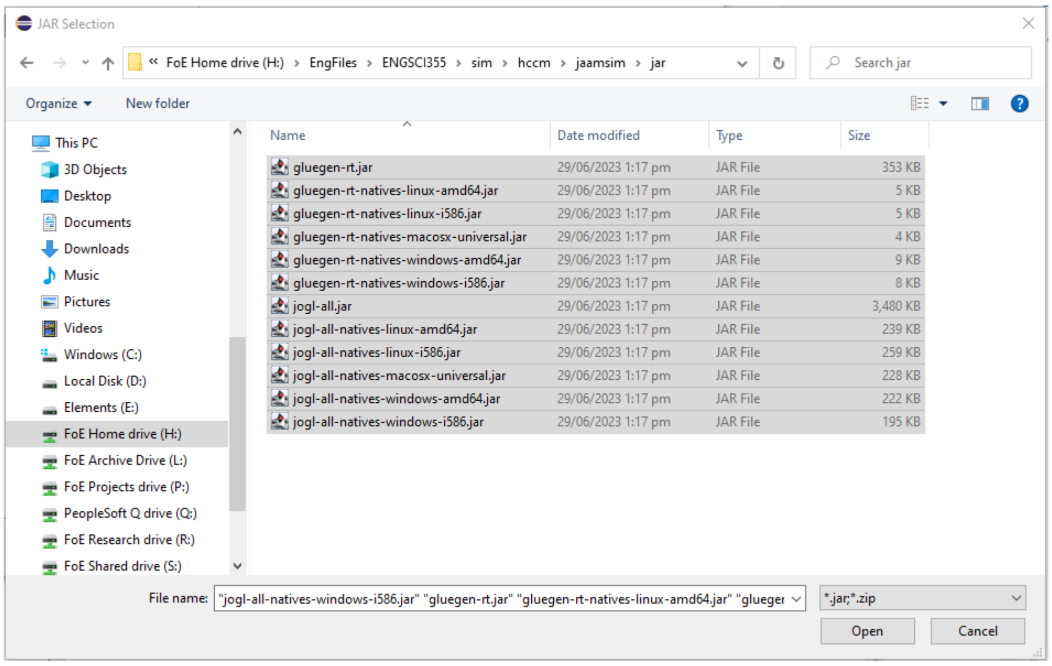
New User Library

User library name: JOGL2_sim

☐ System library (added to the boot class path)

Figure 14: Screenshot of adding a new user library in project properties

Next the library (JOGL2_sim) and click on Add External JARs. . . Navigate to sim\hccm\jaamsim\jar and select all of the .jar files – see figure 15.



Name	Date modified	Type	Size
gluegen-rt.jar	29/06/2023 1:17 pm	JAR File	353 KB
gluegen-rt-natives-linux-amd64.jar	29/06/2023 1:17 pm	JAR File	5 KB
gluegen-rt-natives-linux-i586.jar	29/06/2023 1:17 pm	JAR File	5 KB
gluegen-rt-natives-macosx-universal.jar	29/06/2023 1:17 pm	JAR File	4 KB
gluegen-rt-natives-windows-amd64.jar	29/06/2023 1:17 pm	JAR File	9 KB
gluegen-rt-natives-windows-i586.jar	29/06/2023 1:17 pm	JAR File	8 KB
jogl-all.jar	29/06/2023 1:17 pm	JAR File	3,480 KB
jogl-all-natives-linux-amd64.jar	29/06/2023 1:17 pm	JAR File	239 KB
jogl-all-natives-linux-i586.jar	29/06/2023 1:17 pm	JAR File	259 KB
jogl-all-natives-macosx-universal.jar	29/06/2023 1:17 pm	JAR File	228 KB
jogl-all-natives-windows-amd64.jar	29/06/2023 1:17 pm	JAR File	222 KB
jogl-all-natives-windows-i586.jar	29/06/2023 1:17 pm	JAR File	195 KB

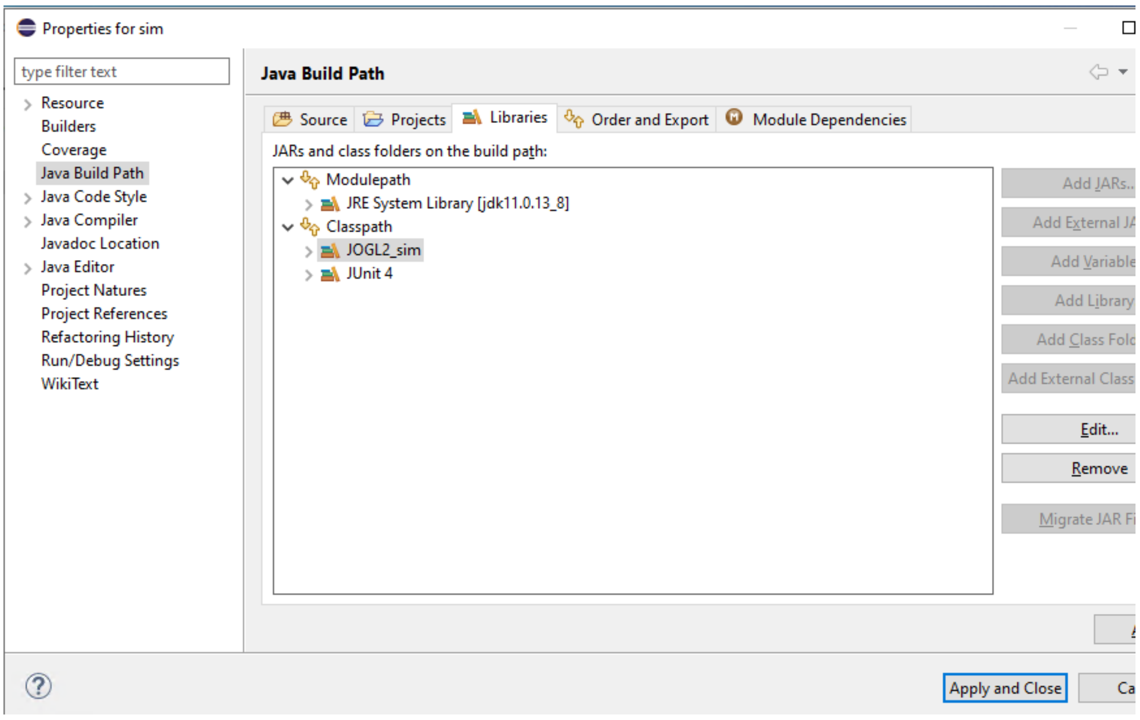
File name: "jogl-all-natives-windows-i586.jar" "gluegen-rt.jar" "gluegen-rt-natives-linux-amd64.jar" "glueger

.jar;.zip

Figure 15: Screenshot of adding JARs to the user library in project properties

Select Open, Apply and Close, then Finish. You will be back at the Libraries tab – see figure 16.

Click Apply and Close.



Properties for sim

type filter text

- Resource
- Builders
- Coverage
- Java Build Path
- Java Code Style
- Java Compiler
- Javadoc Location
- Java Editor
- Project Natures
- Project References
- Refactoring History
- Run/Debug Settings
- WikiText

Java Build Path

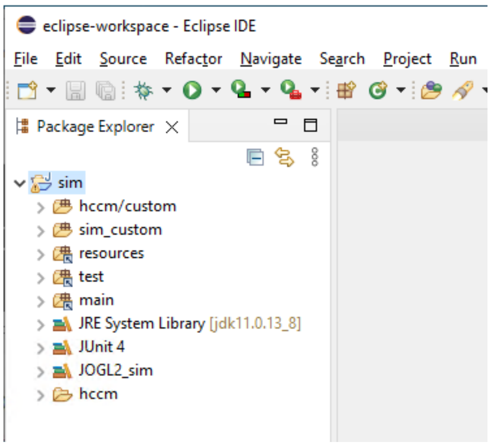
Source | Projects | Libraries | Order and Export | Module Dependencies

JARs and class folders on the build path:

- Modulepath
  - JRE System Library [jdk11.0.13_8]
- Classpath
  - JOGL2_sim
  - JUnit 4

Figure 16: Screenshot of final libraries in project properties

Your Eclipse project should look like the screenshot in figure 17.



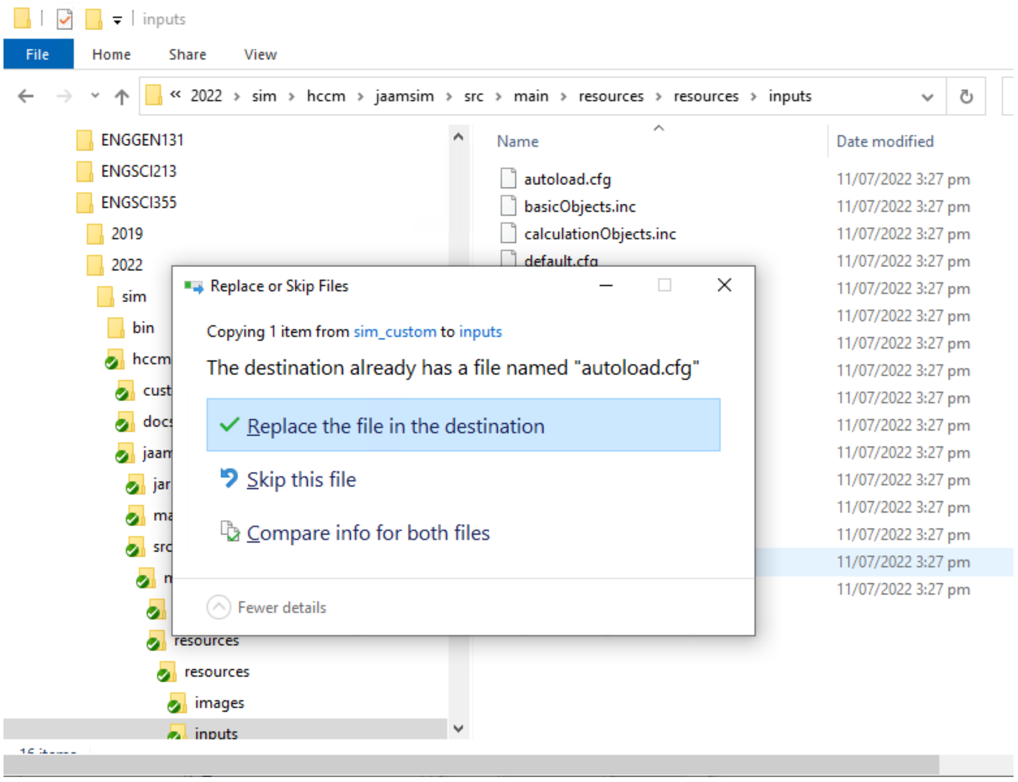
eclipse-workspace - Eclipse IDE

- sim
  - hccm/custom
  - sim_custom
  - resources
  - test
  - main
  - JRE System Library [jdk11.0.13_8]
  - JUnit 4
  - JOGL2_sim
  - hccm

Figure 17: Screenshot of Eclipse project after setting properties

8 Integrate with JaamSim

To integrate HCCM and any custom logic with JaamSim you simply need to copy your autoload.cfg file (from sim_custom) to sim\hccm\jaamsim\src\main\resources\resources\inputs and replace the autoload.cfg file that is currently there – see figure 18. You also need to copy the sim_custom.inc file you created, and the file hccm.inc in hccm\custom to the same location.


Replace or Skip Files

Copying 1 item from sim_custom to inputs

The destination already has a file named "autoload.cfg"

Figure 18: Screenshot of replacing the default autoload.cfg file with the customised one

Once you have copied the sim_custom.inc and hccm.inc files into the inputs folder you will need to refresh this folder in Eclipse. Open the resources folder on the left then open resources.inputs you should see both the files listed. If you don't try right clicking on the resources.inputs folder and choosing refresh.

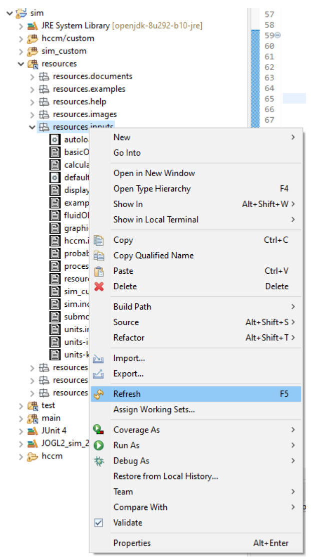
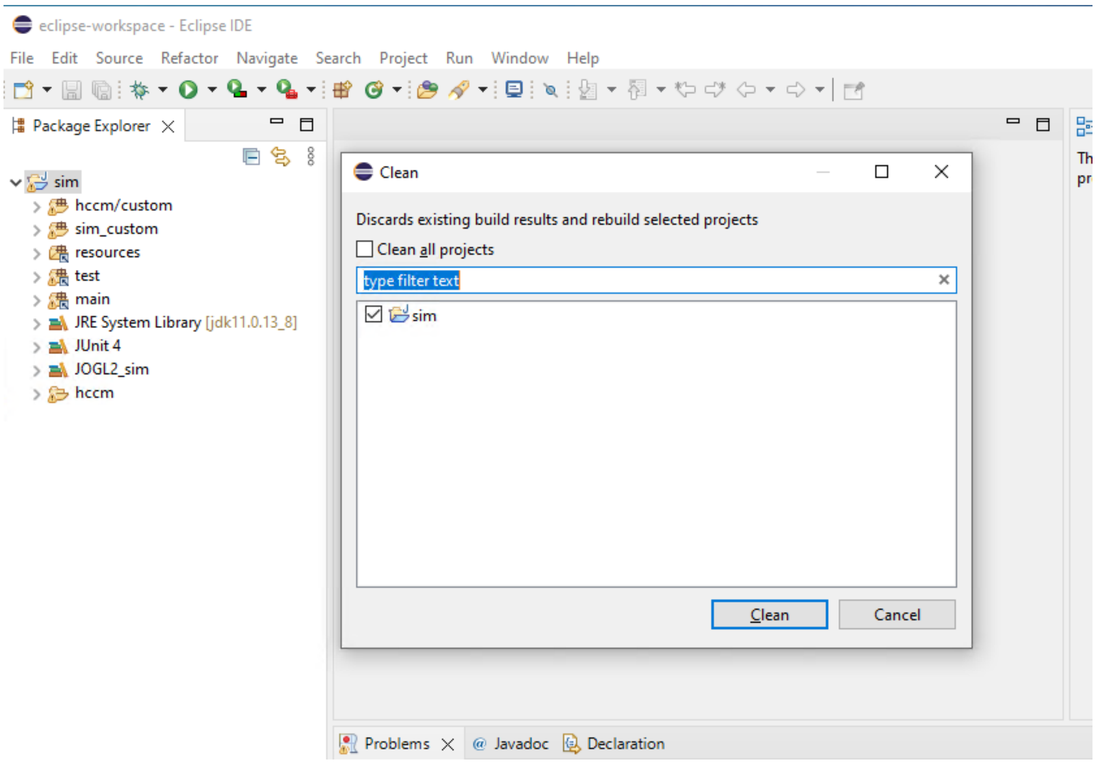


Figure 19: Screenshot of refreshing inputs folder

9 Run Custom JaamSim

To get the customisation to work you may need to click on Project > Clean... and clean your project and wait for the Java code to rebuild – see figure 20.



Discards existing build results and rebuild selected projects

Figure 20: Screenshot of cleaning the main project

Now, within the sim project, expand main, then com.jaamsim.ui and right click on GUIFrame.java. Choose Run As and then choose Run Configurations....

In the Run Configurations dialog, slect Java Application from the list on the left and then click on the New launch configuration button and you should see GUIFrame selected as shown in figure 22.

Go to the Arguments tab and add the following text to VM arguments,

```
--add-exports java.base/java.lang=ALL-UNNAMED
--add-exports java.desktop/sun.awt=ALL-UNNAMED
--add-exports java.desktop/sun.java2d=ALL-UNNAMED
```

Figure 23 shows a screenshot.

Click on Run and JaamSim should launch with the HCCM library included – see figure 24.

Once you exit JaamSim (with HCCM) you can run it again by going to the run button in Eclipse, clicking on the expand arrow and choosing GUIFrame to run – see figure 25.

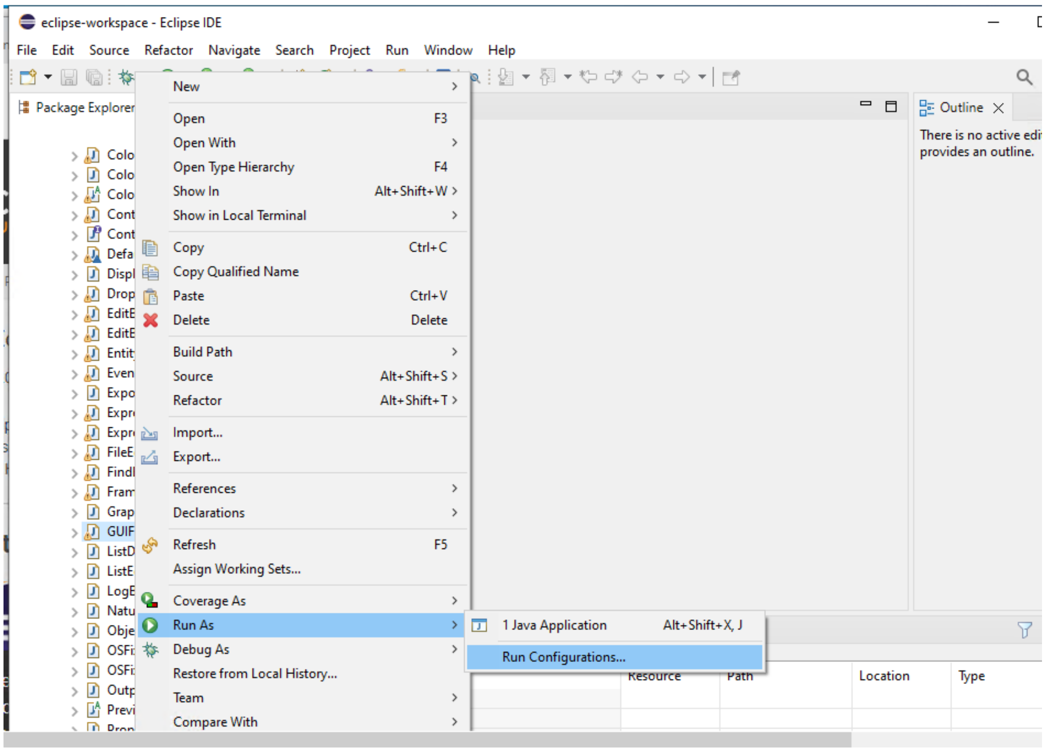
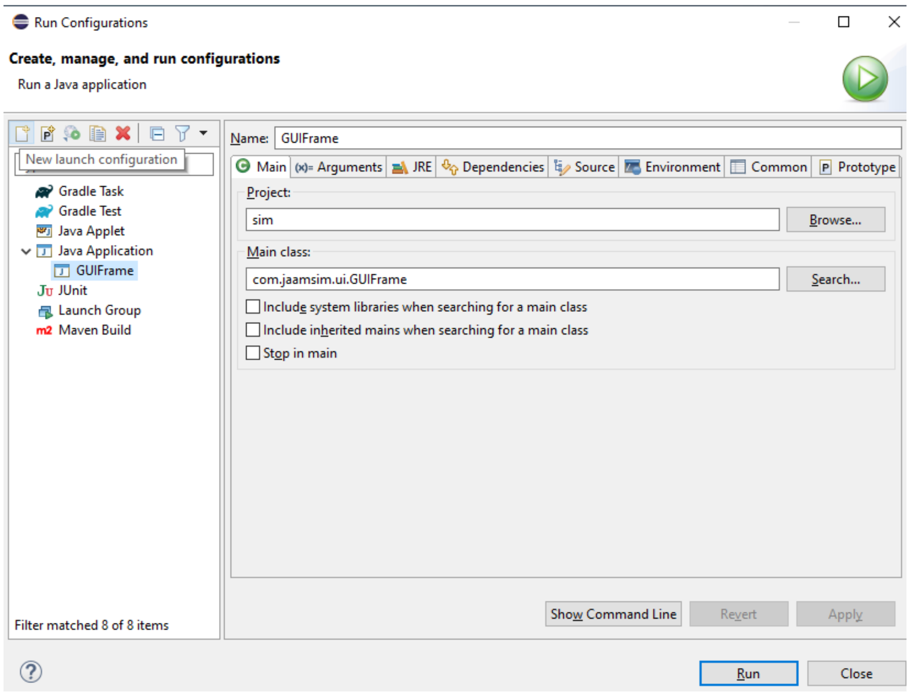


Figure 21: Screenshot of opening the run configuration for JaamSim


Run Configurations

Create, manage, and run configurations

Run a Java application

Name: GUIFrame

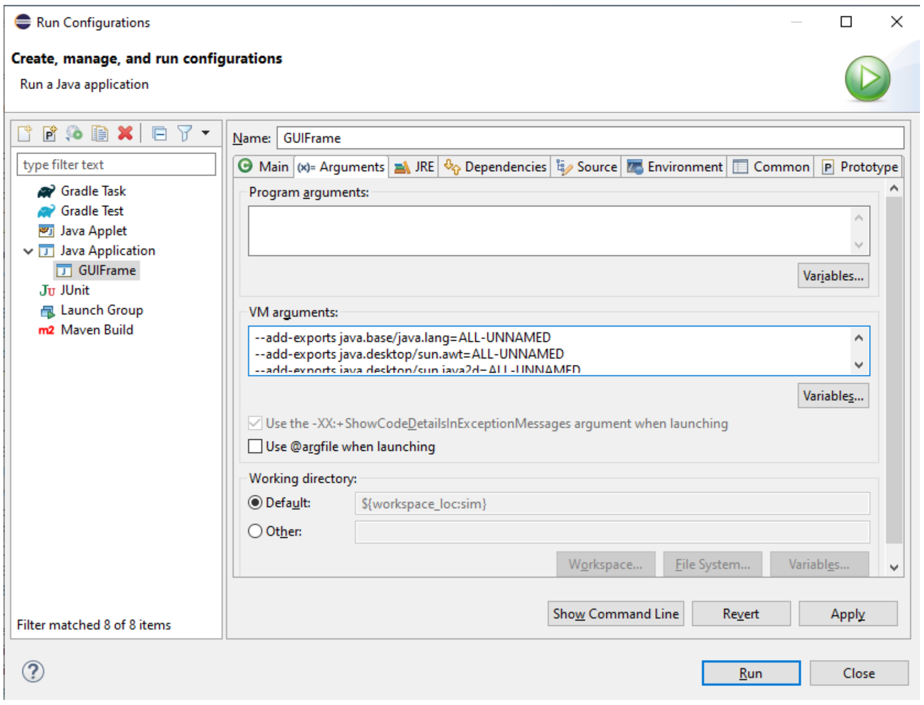
Project: sim

Main class: com.jaamsim.ui.GUIFrame

- [ ] Include system libraries when searching for a main class
- [ ] Include inherited mains when searching for a main class
- [ ] Stop in main

Filter matched 8 of 8 items

Figure 22: Screenshot of creating a new run configuration for JaamSim



Name: GUIFrame

Program arguments:

VM arguments:

```
--add-exports java.base/java.lang=ALL-UNNAMED
--add-exports java.desktop/sun.awt=ALL-UNNAMED
--add-exports java.desktop/sun.java2d=ALL-UNNAMED
```

[✓] Use the -XX:+ShowCodeDetailsInExceptionMessages argument when launching

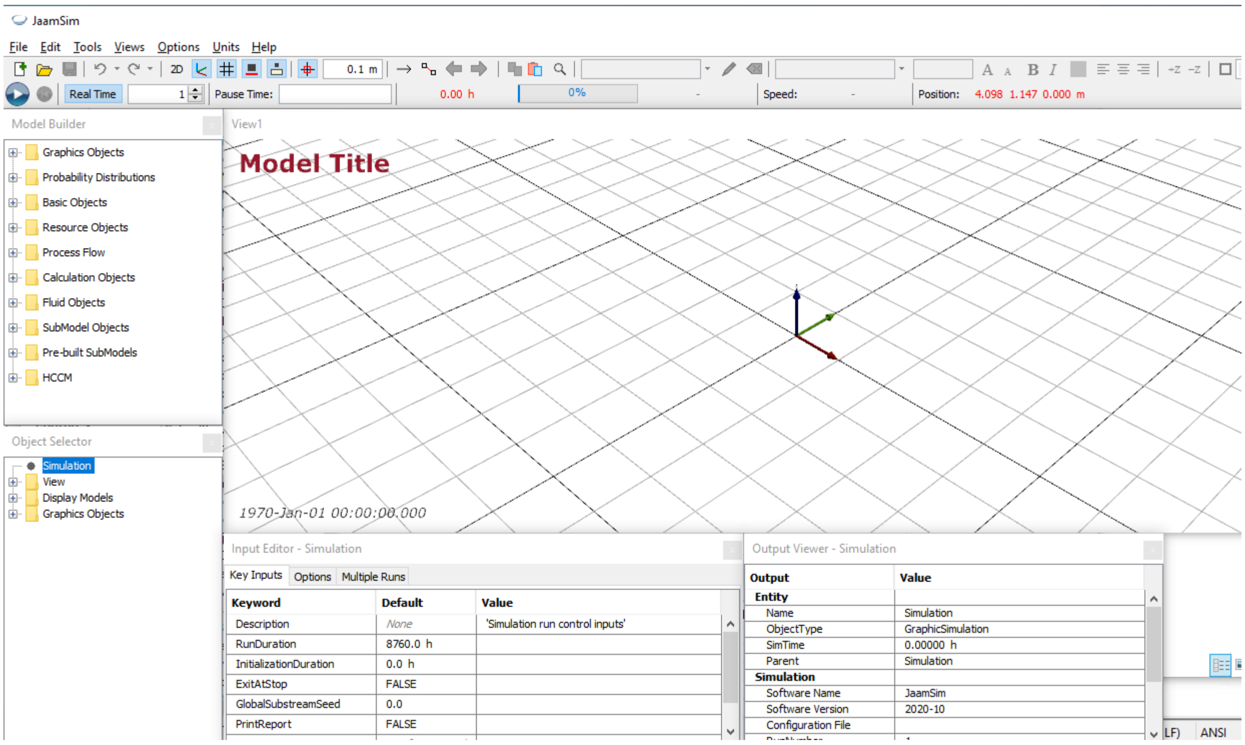
[ ] Use @argfile when launching

Working directory:

[✓ Default:] ${workspace_loc:sim}

Other:

Figure 23: Screenshot of VM arguments for the new run configuration



Input Editor - Simulation

Keyword	Default	Value
Description	None	'Simulation run control inputs'
RunDuration	8760.0 h	
InitializationDuration	0.0 h	
ExitAtStop	FALSE	
GlobalSubstreamSeed	0.0	
PrintReport	FALSE	

Output Viewer - Simulation

Output	Value
Entity	
Name	Simulation
ObjectType	GraphicSimulation
SimTime	0.00000 h
Parent	Simulation
Simulation	
Software Name	JaamSim
Software Version	2020-10
Configuration File	

Figure 24: Screenshot of JaamSim running from source with HCCM included

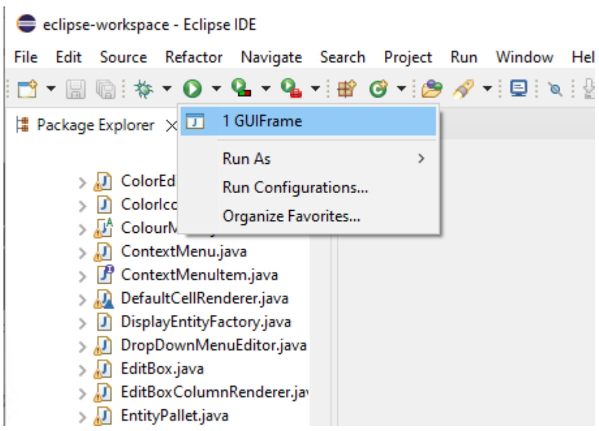
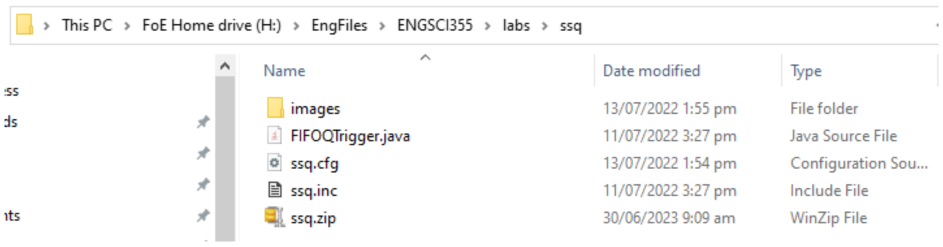


Figure 25: Screenshot of running customised JaamSim using the run button

10 Running an HCCM model

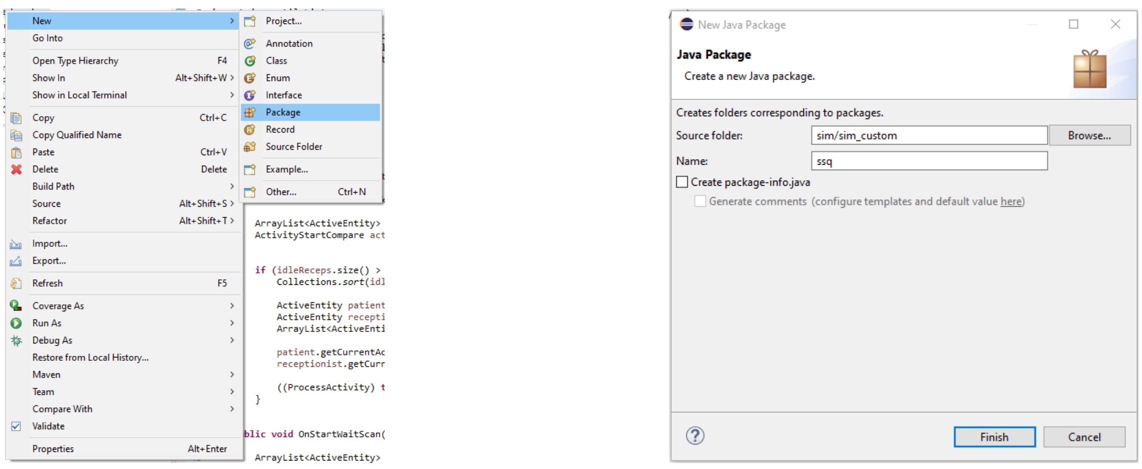
Download the single server queue model's folder from Canvas (ssq.zip). Create a new folder within the "labs" folder called "ssq" and extract ssq.zip into that folder. Figure 26 shows what "ssq" should look like afterwards.


This PC > FoE Home drive (H:) > EngFiles > ENGSCI355 > labs > ssq

Name	Date modified	Type
images	13/07/2022 1:55 pm	File folder
FIFOQTrigger.java	11/07/2022 3:27 pm	Java Source File
ssq.cfg	13/07/2022 1:54 pm	Configuration Sou...
ssq.inc	11/07/2022 3:27 pm	Include File
ssq.zip	30/06/2023 9:09 am	WinZip File

Figure 26: Screenshot of "ssq" after extracting ssq.zip

Now in Eclipse right-click on the sim_custom folder and select New → Package. Enter 'ssq' for the name of the package and click Finish.



```
ArrayList<ActiveEntity>
ActivityStartCompare act

if (idleReceps.size() >
    Collections.sort(idl

    ActiveEntity patient
    ActiveEntity recepti
    ArrayList<ActiveEnti

    patient.getCurrentAc
    receptionist.getCurr

    ((ProcessActivity) t
}

blic void OnStartWaitScan(
    ArrayList<ActiveEntity>
```

Creates folders corresponding to packages.
Source folder: sim/sim_custom
Name: ssq
☐ Create package-info.java
☐ Generate comments (configure templates and default value here)

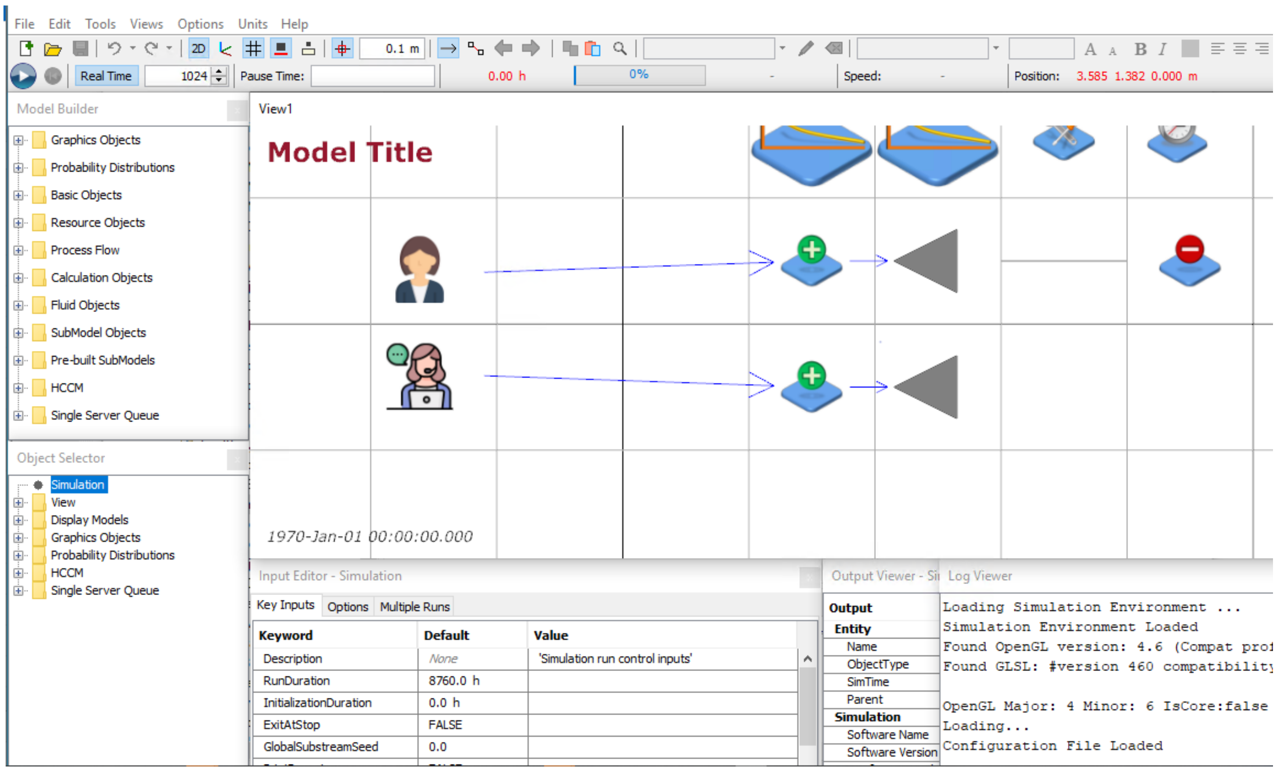
Figure 27: Screenshot of Package Creation

Now go back to the ssq folder you extracted the zip file to and copy the FIFOQTrigger.java file to the newly created package folder under sim → sim_custom → ssq.

Finally we need to make this new object available in Jaamsim. To do this we need to edit the sim_custom.inc file that we put in sim → hccm → jaamsim → src → main → resources → resources → inputs. Open the sim_custom.inc file and also open the ssq.inc file in the ssq folder. Copy the contents of ssq.inc into sim_custom.inc.

Run JaamSim with HCCM from Eclipse again (you might want to Clean your project again just in case). You should now see a Single Server Queue palette in the Model Builder window. It has the FIFO trigger for the Single Server Queue model (you will learn more about triggers in later labs).

Next, open ssq.cfg from the ssq folder. Figure 28 shows what you should see. Run the model and see how the customers and servers join together for service in the queue. Figure 29 shows the model paused with a customer and server in process and a customer in the queue.



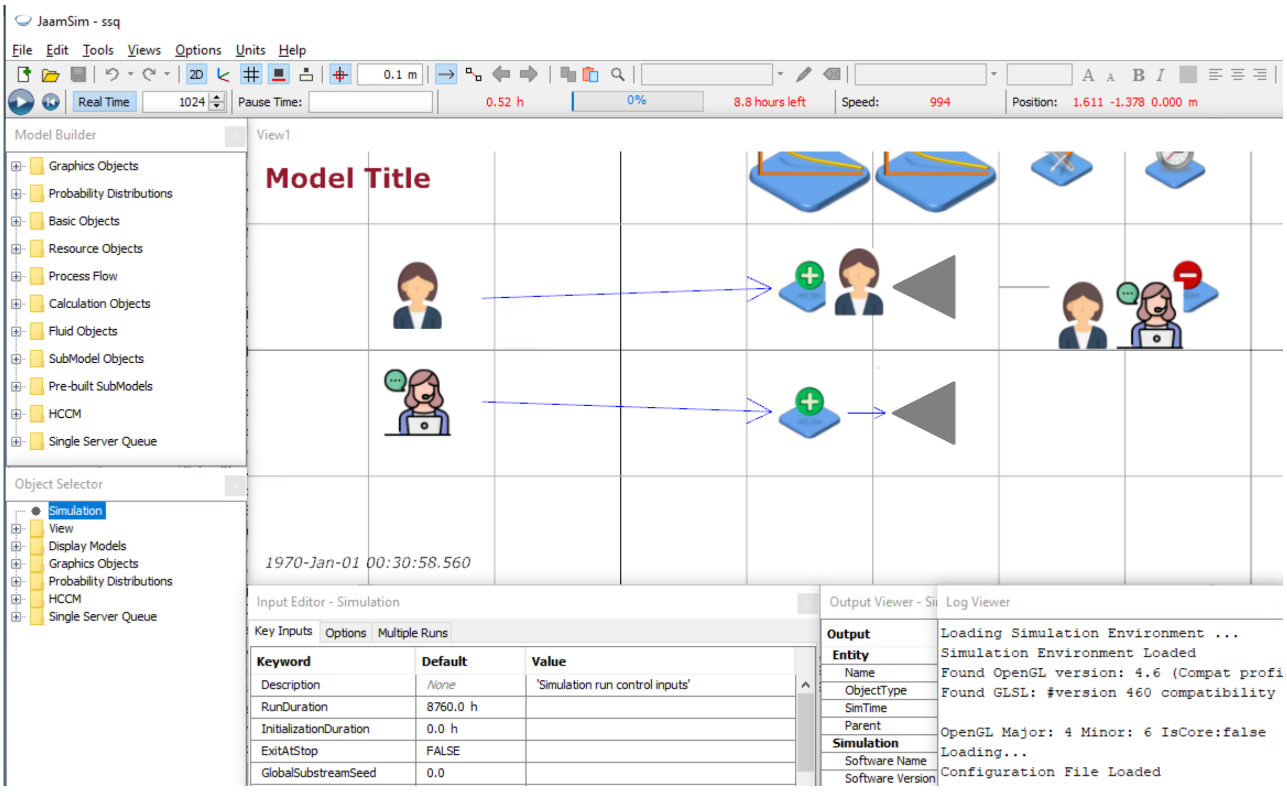
Keyword	Default	Value
Description	None	'Simulation run control inputs'
RunDuration	8760.0 h	
InitializationDuration	0.0 h	
ExitAtStop	FALSE	
GlobalSubstreamSeed	0.0	

Output
Entity
Name
ObjectType
SimTime
Parent
Simulation
Software Name
Software Version

```
Loading Simulation Environment ...
Simulation Environment Loaded
Found OpenGL version: 4.6 (Compat prof
Found GLSL: #version 460 compatibility

OpenGL Major: 4 Minor: 6 IsCore:false
Loading...
Configuration File Loaded
```

Figure 28: Screenshot of ssq.cfg when it has been opened



Keyword	Default	Value
Description	None	'Simulation run control inputs'
RunDuration	8760.0 h	
InitializationDuration	0.0 h	
ExitAtStop	FALSE	
GlobalSubstreamSeed	0.0	

Output
Entity
Name
ObjectType
SimTime
Parent
Simulation
Software Name
Software Version

```
Loading Simulation Environment ...
Simulation Environment Loaded
Found OpenGL version: 4.6 (Compat profi
Found GLSL: #version 460 compatibility

OpenGL Major: 4 Minor: 6 IsCore:false
Loading...
Configuration File Loaded
```

Figure 29: Screenshot of ssq.cfg when paused during simulation