АМАЛЬГАМА

Руководство пользователя программного обеспечения для разработки имитационных моделей и систем поддержки принятия решений Amalgama Platform

Ульяновск, 2022

Содержание

1. Введение

1.1. Назначение документа

Настоящий документ – концепция программного обеспечения (ПО) Amalgama Platform – набора инструментов для разработки имитационных моделей и систем поддержки принятия решений на языке программирования Java.

Концепция содержит:

- описание функциональных характеристик и состава ПО Amalgama Platform
- информацию, необходимую для установки и эксплуатации ПО Amalgama Platform.

Подробное описание всех элементов ПО Amalgama Platform, а также обучающие материалы для работы с ПО Amalgama Platform размещены в свободном доступе на интернет-сайте ***www.platform.amalgamasimulation.com***.

1.2. Цель создания ПО Amalgama Platform

Amalgama Platform создана для разработки имитационных моделей и систем поддержки принятия решений с использованием преимуществ стандартных подходов к разработке ПО, таких как IDE, системы контроля версий, рефакторинг кода и модульное тестирование. ПО Amalgama Platform предлагает набор библиотек Java и подключаемых модулей Eclipse, которые позволяют создавать быстрые и современные имитационные модели путем написания кода.

2. Функциональные характеристики

2.1. Назначение и функциональность программы

ПО Amalgama Platform представляет собой набор инструментов для разработки имитационных моделей и систем поддержки принятия решений на языке программирования Java.

Amalgama Platform разработана таким образом, чтобы обеспечивать полное разделение логики, данных и их представления. Поэтому она предоставляет средства для разработки всех типов решений:

- автономных кроссплатформенных имитационных моделей
- имитационных моделей с общим хранилищем сценариев и веб-представлением результатов
- интегрированных имитационных моделей с инструментами тестирования и проверки в виде десктоп-приложений.

ПО Amalgama Platform состоит из:

- событийного движка для построения имитационных моделей (simulation engine)
- прикладного программного интерфейса (Java API);
- библиотек для моделирования сложных процессов, таких как потоки жидкостей, движение кранов, поездов и др.
- готовых визуальных компонентов: таблиц, графиков, диаграмм.

Платформа содержит инструменты для создания кроссплатформенных десктоп-приложений с графическим интерфейсом.

Имитационная модель, созданная с помощью ПО Amalgama Platform, – это программа на языке Java с минимумом внешних зависимостей. Таким образом, ее можно легко встроить в консольные или веб-приложения (включая приложения Spring).

2.2. Особенности программы

- Свобода в выборе метода имитационного моделирования и прикладного программного интерфейса
- Решения основаны на структурированных моделях данных, без использования упрощенных визуальных подходов (типа point-and-click), генерирующих вспомогательный код
- Имеет встроенные алгоритмы и модули для построения логики планирования любой сложности

- Позволяет создавать кроссплатформенные приложения для моделирования предметной области, такие как Mine-Twin и Plant-Twin.
- Совместима с технологиями Java
- Содержит функциональность для модульного тестирования имитационных моделей.

2.3. Состав программы

ПО Amalgama Platform содержит несколько модулей:

- Simulation engine (событийный движок для построения имитационных моделей) – Java-класс, который обеспечивает API (прикладной программный интерфейс) для планирования событий в конкретный момент во время моделирования. Этот класс также содержит внутренний таймер, который позволяет пользователю выполнять моделирование на любом временном горизонте.
- State machine (класс создания стейт-чартов) – абстракция, часто используемая в имитационном моделировании. Она состоит из состояний и переходов между ними. Переходы определяют направления возможных изменений состояний.
- Monitored value – класс, рассчитывающий значения величин, которые изменяются линейно с определенной скоростью в процессе имитационного моделирования, и генерирует сообщение, когда достигнуто некоторое пороговое значение величины.
- Rate planner – класс, рассчитывающий кусочно-линейную функцию с шагами. Значения функции могут быть ограничены некоторыми нижними и верхними границами. Как правило, этот класс возвращает ожидаемое изменение некоторой величины с течением времени.
- Graph Agent library - инструмент, с помощью которого ПО Amalgama Platform моделирует движение объектов в некоторой транспортной сети или на графе.
- Discrete Rate library - инструмент, позволяющий моделировать поведение систем взаимосвязанных элементов потока, которые могут принимать, хранить или отправлять непрерывное количество какого-либо материала или смеси материалов.

3. Установка и эксплуатация ПО Amalgama Platform

3.1. Подготовка к работе с ПО Amalgama Platform

ПО Amalgama Platform – это набор Java-библиотек, которые помогают создавать имитационные модели в виде Java-приложений.

Для создания имитационных моделей с помощью ПО Amalgama Platform необходимо выполнить следующие действия:

1. Получить доступ к библиотекам ***ПО Amalgama Platform***
2. Установить на персональном компьютере (ПК) ***Eclipse IDE***
3. Установить ***Java SDK 11***
4. Установить ***JavaFX.***

3.1.1. Получение доступа к библиотекам ПО Amalgama Platform

Библиотеки ПО Amalgama Platform поставляются в двух форматах:

1. P2-репозиторий, который нужно добавить к .target-файлу проекта пользователя в Eclipse
2. Набор jar-файлов, которые должны быть включены в проект в Eclipse в качестве внешней зависимости.

Для получения логина и пароля к Nexus-репозиторию с библиотеками ПО Amalgama Platform нужно отправить запрос по адресу электронной почты *platform@am-sim.com*.

После получения параметров доступа нужно использовать их для получения библиотек:

1. Адрес P2-репозитория: *https://nexus.am-sim.com/repository/platform-p2/*
2. Jar-файлы могут быть скачаны напрямую из Nexus. Для чего нужно авторизоваться на *https://nexus.am-sim.com/* и ввести название репозитория 'platform-jars'.

3.1.2. Установка Eclipse IDE

Для установки Eclipse IDE на ПК пользователя следует пройти по ссылке *https://www.eclipse.org/downloads/packages/release/2021-06/r/eclipse-modeling-tools* и скачать Eclipse Modeling Tools 2021-06.

Далее выбрать Start Eclipse, перейти в 'Help' → 'Eclipse Marketplace' и установить 'e(fx)clipse'.

3.1.3. Установка Java SDK 11

Для установки Java SDK 11 на ПК пользователя следует пройти по ссылке https://www.oracle.com/java/technologies/downloads/archive/ и скачать Java SE 11.

3.1.4. Установка JavaFX

Для установки JavaFX на ПК пользователя следует пройти по ссылке JavaFX (openjfx.io).

3.2. Создание консольного приложения

3.2.1. Введение

Имитационная модель в виде консольного приложения может использоваться для выполнения сценарного анализа, когда важнее быстро получить результаты моделирования, чем визуально отслеживать процесс моделирования.

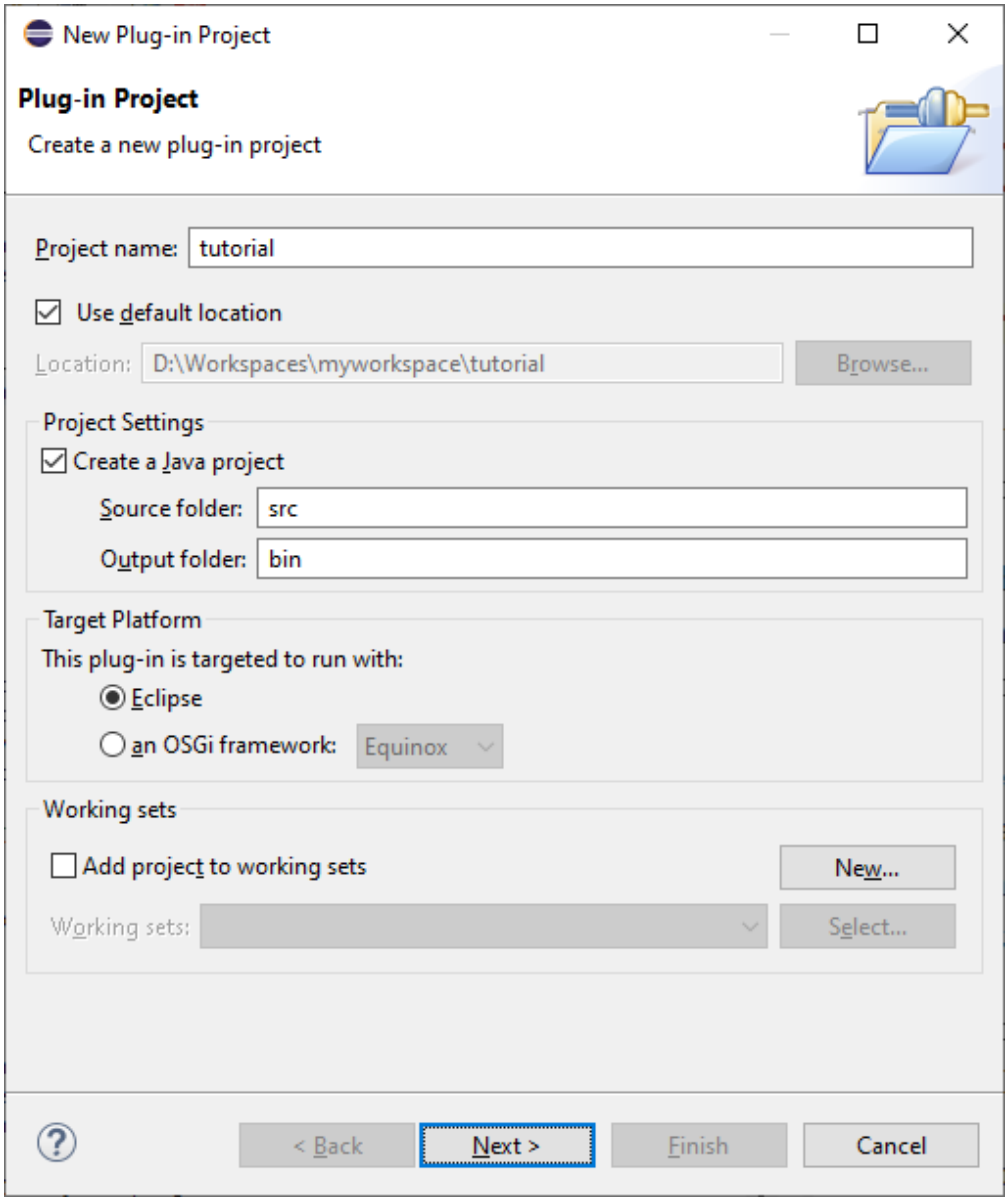
Консольное приложение может быть затем преобразовано в web-сервис для того, чтобы предоставить возможность нескольким пользователям выполнять моделирование удалённо.

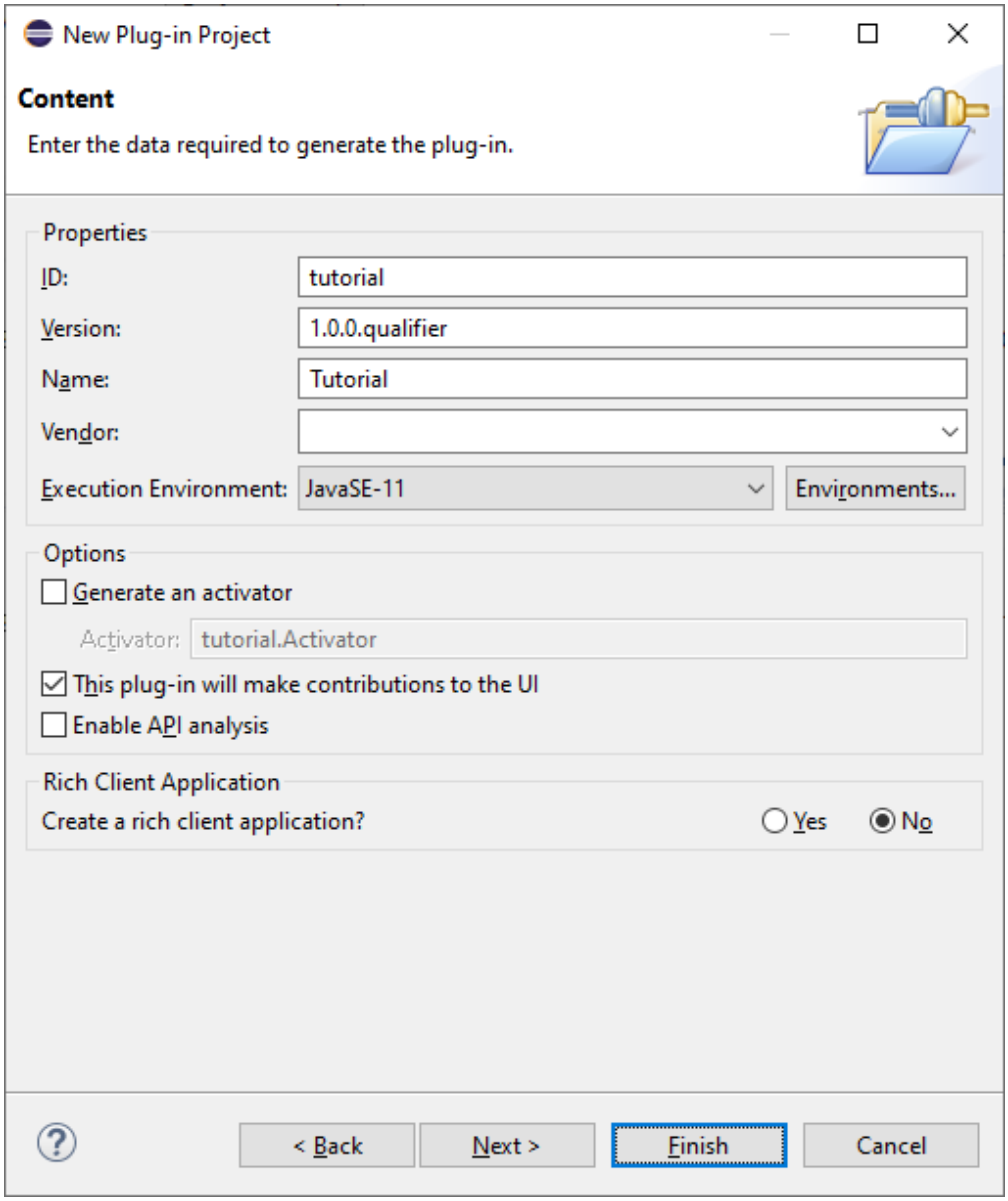
3.2.2. Создание проекта в IDE

В настоящем документе демонстрируется создание консольного приложения с использованием Eclipse IDE. Выбор IDE произволен, можно использовать любую IDE, поддерживающую написание Java-кода.

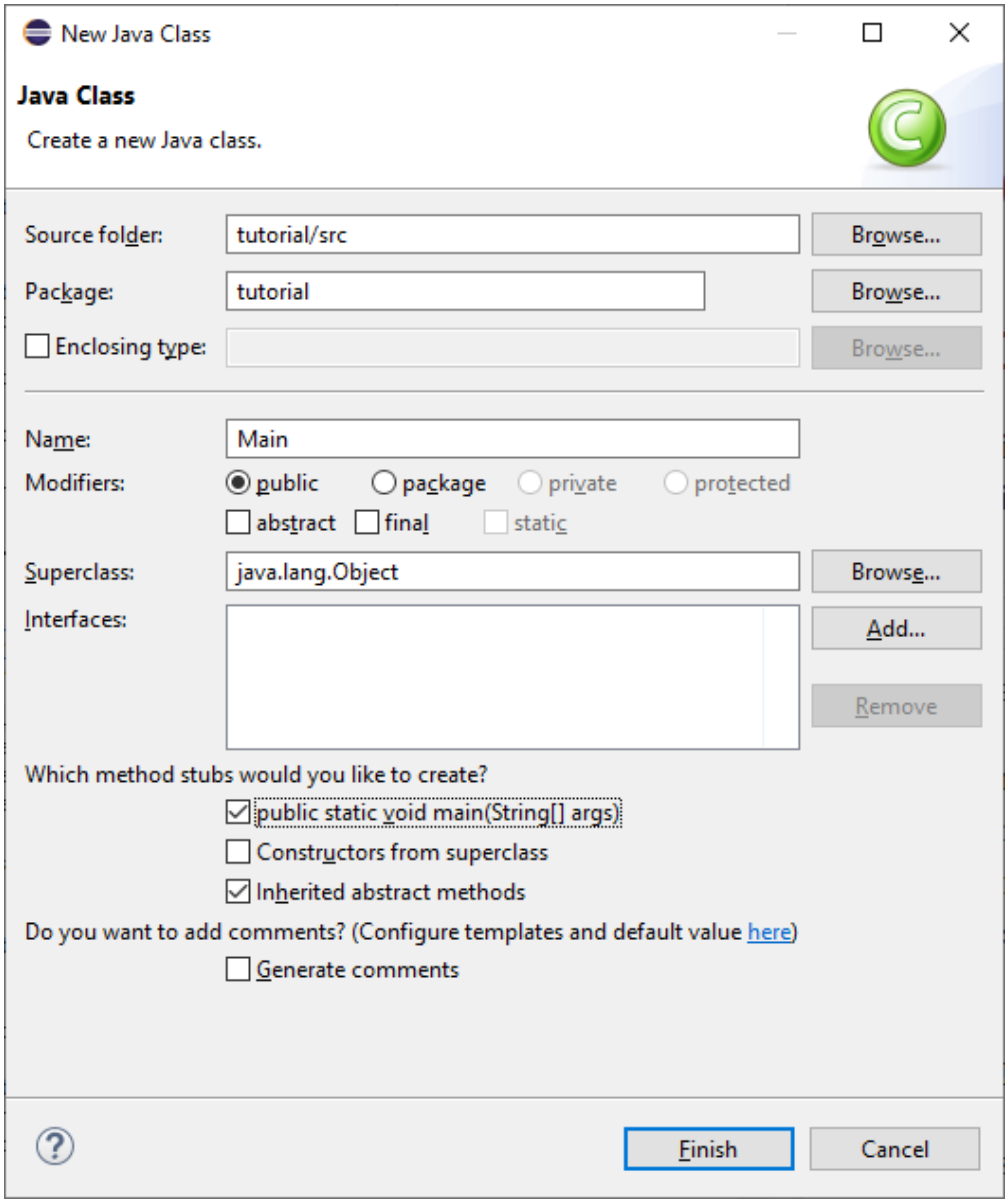
В первую очередь следует установить Eclipse (п. 3.1.2).

Далее нужно запустить Eclipse и создать новый Plug-in Project. Для примера назовём его 'tutorial'.


New Plug-in Project
Plug-in Project
Create a new plug-in project
Project name: tutorial
Use default location
Location: D:\Workspaces\myworkspace\tutorial
Browse...
Project Settings
Create a Java project
Source folder: src
Output folder: bin
Target Platform
This plug-in is targeted to run with:
Eclipse
an OSGi framework: Equinox
Working sets
Add project to working sets
New...
Working sets:
Select...
< Back
Next >
Finish
Cancel



После создания проекта нужно добавить пакет 'tutorial' и внутри пакета добавить новый класс 'Main' с методом `main`.



В контекстном меню класса в файле Main.java нужно выбрать 'Run As' → 'Java Application'. Приложение должно запускаться и завершаться без ошибок.

3.2.3. Внешние библиотеки

Далее нужно добавить в проект несколько внешних библиотек. Добавим их напрямую как jar-файлы.

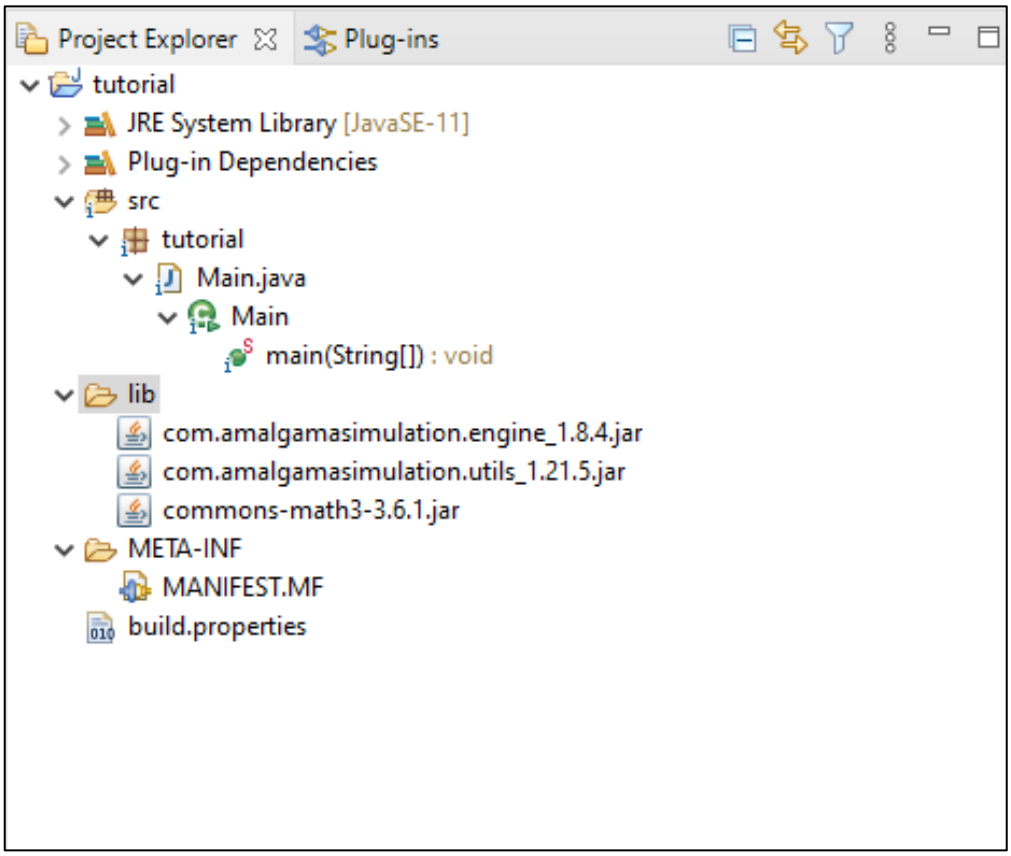
Для авторизации на сайте https://nexus.am-sim.com/ нужно использовать полученные параметры доступа (логин и пароль) (п. 3.1.1). Далее перейти в репозиторий 'platform-jars' и скачать последние версии следующих библиотек:

1. com.amalgamasimulation.engine.jar

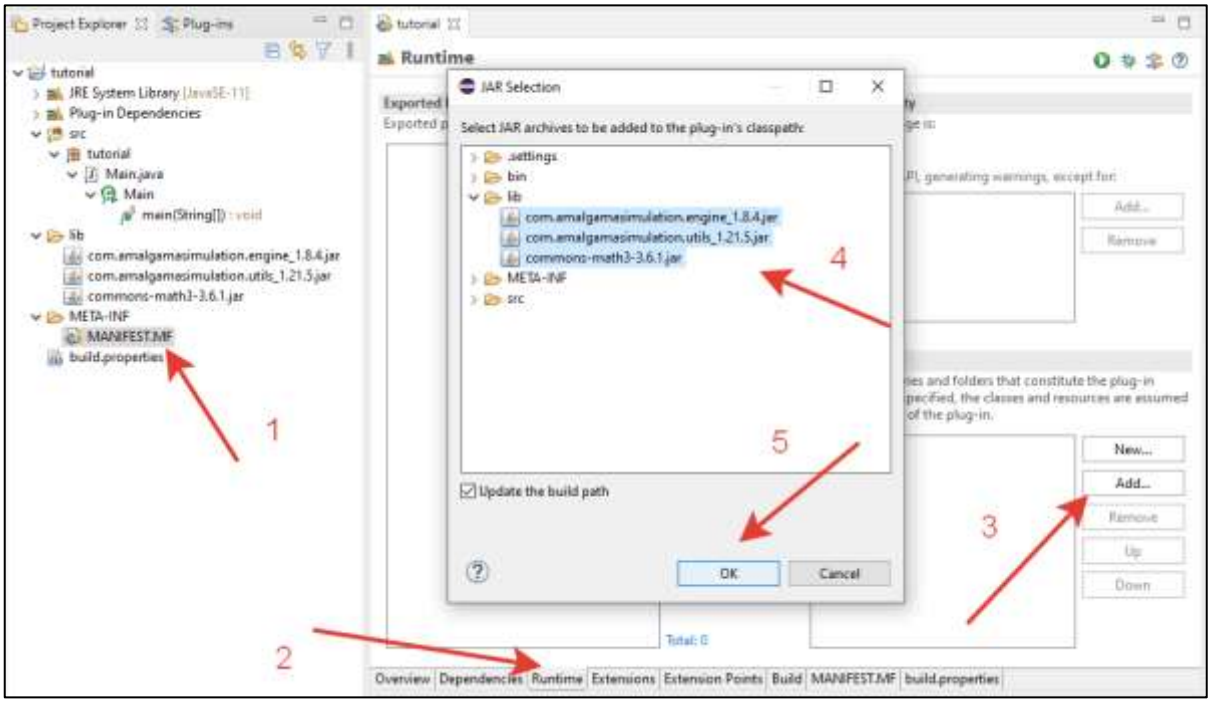
2. com.amalgamasimulation.utils.jar

Также понадобится сторонняя общедоступная математическая библиотека. Для ее добавления нужно перейти по ссылке https://commons.apache.org/proper/commons-math/download_math.cgi и скачать 'commons-math3-3.6.1-bin.zip'-файл, затем извлечь 'commons-math-3.6.1.jar' из .zip-архива.

Нужно создать новую 'lib' папку в корневой папке проекта и поместить в неё com.amalgamasimulation.engine.jar, com.amalgamasimulation.utils.jar и commons-math-3.6.1.jar библиотеки.



Затем следует перейти в файл MANIFEST.MF окна редактирования, перейти в режим исполнения программы 'Runtime' и в секции 'Classpath' нажать кнопку 'Add'. В открывшемся диалоговом окне выбрать все три библиотеки в папке 'lib' и нажать 'OK'.



Далее необходимо сохранить файл MANIFEST.MF.

После того, как библиотеки добавлены, добавим простой фрагмент кода, чтобы проверить их. Перепишем класс `Main`, чтобы получить следующий результат:

Класс Main, использующий Engine в первый раз

```
package tutorial;

import org.apache.commons.math3.distribution.TriangularDistribution;

import com.amalgamasimulation.engine.Engine;

public class Main {
    public static void main(String[] args) {
        TriangularDistribution d = new TriangularDistribution(1, 2, 3);

        Engine engine = new Engine();
        engine.scheduleAbsolute(5, () -> System.out.println("engine time is " +
engine.time()));
        engine.scheduleStop(10, "Engine stop");
        engine.setFastMode(true);
        engine.run(true);
    }
}
```

Запустите метод main(). Он должен скомпилироваться и после завершения напечатать в консоль:

```
engine time is 5.0
```

Это простое консольное приложение может быть расширено и использовано в качестве образца для сложных имитационных моделей.

3.2.4. Обзор кода созданного приложения

Посмотрим на код созданного приложения.

Мы создали объект `TriangularDistribution`, чтобы посмотреть, успешно ли подключена библиотека Apache math3.

После этого мы создали новый экземпляр класса `Engine`.

В следующей строке мы планируем событие, которое должно произойти в модельное время 5. По умолчанию модельное время измеряется в секундах. Мы также можем использовать метод `setTemporal()` экземпляра `Engine`, чтобы изменить это поведение. В этом случае мы используем 'абсолютное планирование': на пятой секунде модельного времени выполнится некоторое событие. `Engine` также поддерживает 'относительное планирование' в случае, когда мы хотим запланировать некоторое событие через какой-то промежуток времени от текущего модельного времени, например, 'через 2 часа от настоящего момента'.

Действие, которое мы запланировали, напечатает в консоли короткое сообщение с текущим модельным временем (метод `Engine.time()`). Значение, возвращаемое методом `time()`, - (неотрицательное) число единиц измерения времени (по умолчанию секунд), которые прошли с начала моделирования.

Следующая строка кода планирует остановку модели в модельное время 10 (т.е. на 10-ой секунде). Таким образом, общая длительность периода моделирования будет 10 секунд.

Нужно ли ждать 10 реальных секунд, пока модель остановится? Не обязательно. Вместо этого мы настраиваем моделирование на работу в максимальном режиме, т.е. настолько быстро, насколько позволяет аппаратное обеспечение ПК.

Последняя строчка кода запускает моделирование. Единственному параметру метода `run()` задано значение 'true', что означает, что моделирование запускается в синхронном режиме: вызов метода будет заблокирован, когда моделирование завершится.

3.3. Создание десктоп-приложения

3.3.1. Введение

Если требуется полнофункциональное приложение с графическим пользовательским интерфейсом, то имитационная модель создаётся в виде десктоп-приложения с помощью приложения wizard ПО Amalgama Platform.

В большинстве созданных нами имитационных моделях содержится несколько общих шаблонов:

- Несколько типичных 'перспектив' или режимов: редактирование, моделирование, планирование, сценарный анализ и т.д.
- Каждая 'перспектива' имеет собственное удобное расположение элементов пользовательского интерфейса
- Почти каждое приложение использует набор внешних зависимостей.

Для экономии времени на выполнение рутинных действий мы создали приложение 'Wizard', которое создаёт стартовое приложение с графическим интерфейсом всего за несколько кликов.

Для этого необходимо выполнение следующих условий:

1. На ПК пользователя должен быть установлен Eclipse 2021-06 (п. 3.1.2)
2. Должен быть установлен Java SDK 11 (п. 3.1.3)
3. Должен быть установлен JavaFX (п. 3.1.4)
4. Должна быть настроена переменная окружения «JAVAFX_HOME», чтобы обозначить путь к папке с установленными файлами JavaFX.

3.3.2. Установка приложения wizard

Приложение wizard поставляется в виде jar-файла.

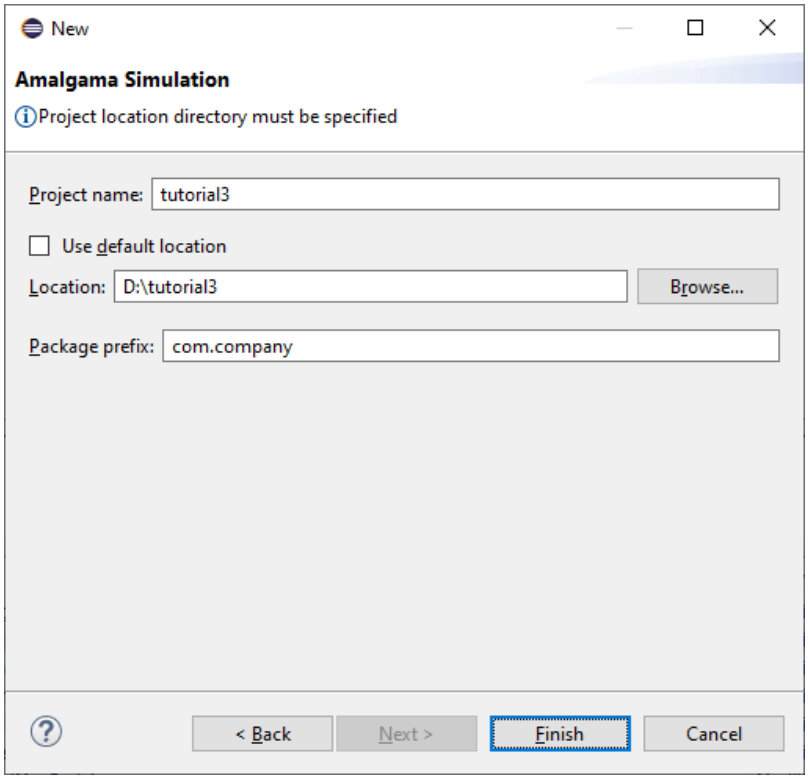
Для его получения необходимо перейти на сайт https://nexus.am-sim.com, используя полученные параметры доступа (логин и пароль) (п. 3.1.1)

На сайте нужно перейти в репозиторий 'platform-jars' и скачать последнюю версию файла wizard из папки 'wizard'. Скачанный файл нужно поместить в папку 'dropins' в локальной папке установки Eclipse и перезапустить Eclipse.

3.3.3. Создание приложения

- Запустите новый экземпляр Eclipse, используя новую рабочую область (Workspace)
- Перейдите в 'Window' → 'Preferences' → 'JavaFX' и в поле 'JavaFX 11+ SDK' выберите папку 'lib' вашего локального JavaFX
- Выберите 'File' → 'New' → 'Other…' и затем выберите 'Amalgama Simulation / Template' wizard. Нажмите Next.

- Введите название проекта, расположение проекта (его нужно запомнить) и префикс пакета.



Нажмите 'Finish'. Новый проект создан.

Далее потребуются некоторые дополнительные действия по очистке:

- Нажмите кнопку 'Open perspective' (в правом верхнем углу окна Eclipse) и выберите 'Plug-in Development'. Откроется вкладка 'Project Explorer'.
- Щелкните правой кнопкой мыши по проекту на вкладке 'Project explorer' и выберите 'Delete'. После открытия диалогового окна поставьте флажок 'Delete 8 nested projects'. НЕ ставьте первый флажок, иначе проект будет удален с диска.

Теперь можно использовать созданный образец приложения для создания собственных полнофункциональных имитационных моделей.

3.3.4. Обзор созданного приложения

3.3.4.1 Запуск нового приложения

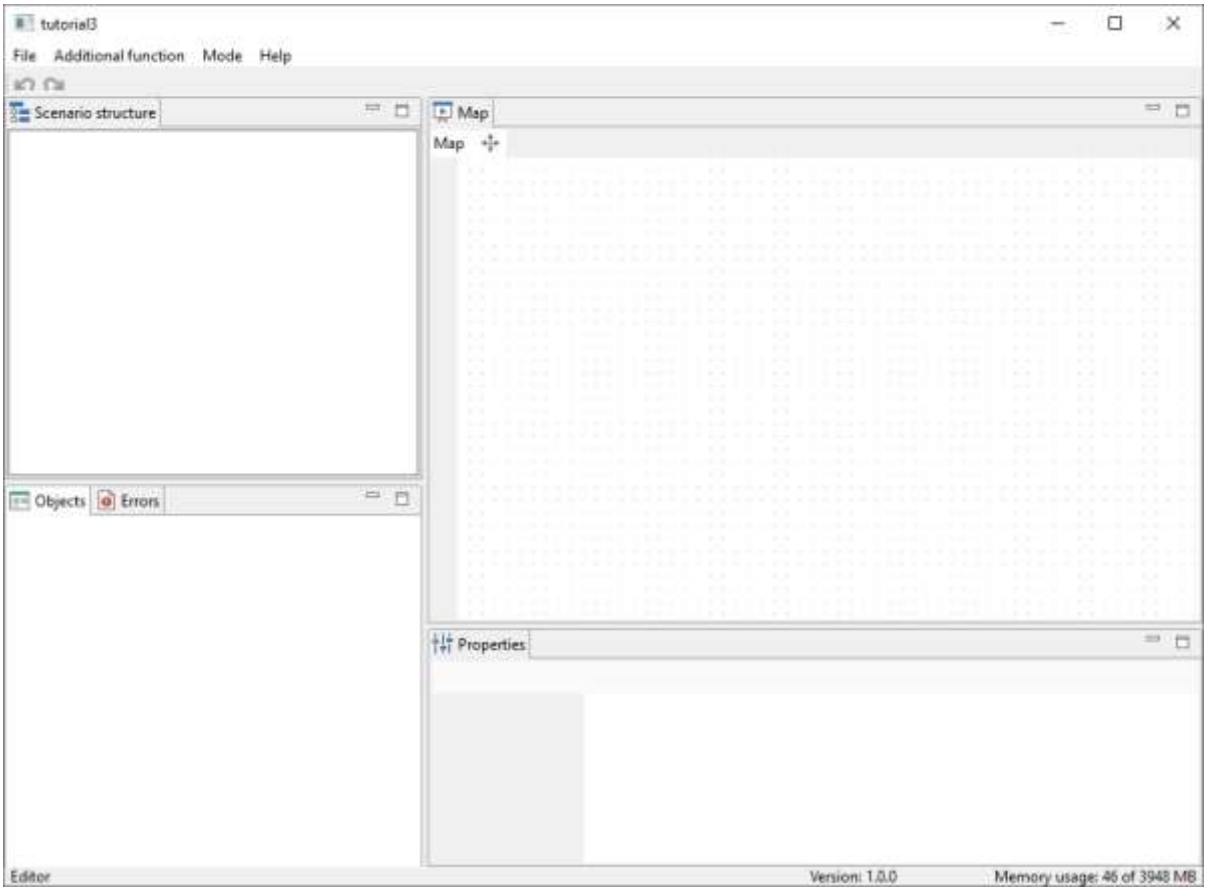
Импортируйте проект, который только что был создан с помощью приложения wizard.

В окне 'Project Explorer перейдите в папку 'releng/*target' вашего проекта и откройте файл *target. Нажмите на голубую ссылку 'Set as Active Target Platform'

(в правом верхнем углу окна редактирования файла 'target). Подождите, пока загрузятся все зависимости, это займёт некоторое время, но, к счастью, это одноразовое действие для каждой рабочей области (Workspace).

После этого перейдите в папку 'releng/*product', откройте файл '*product' и нажмите кнопку 'Launch an Eclipse application' (в правом верхнем углу).

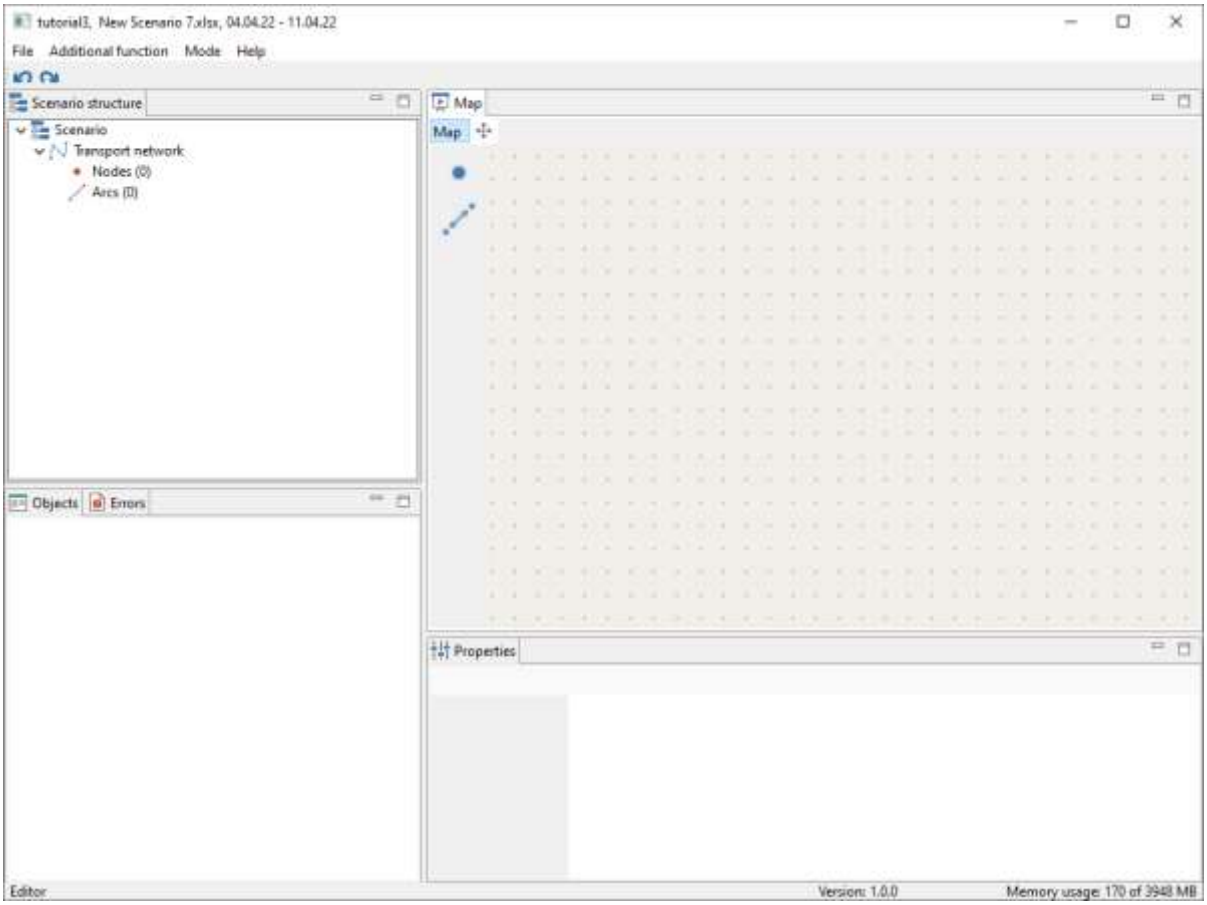
Созданное приложение должно успешно запуститься:



3.3.4.2 Режим редактирования

После запуска приложение откроется в режиме редактирования. Этот режим или 'перспектива' служит для создания и редактирования сценария, т.е. задания исходных данных для моделирования.

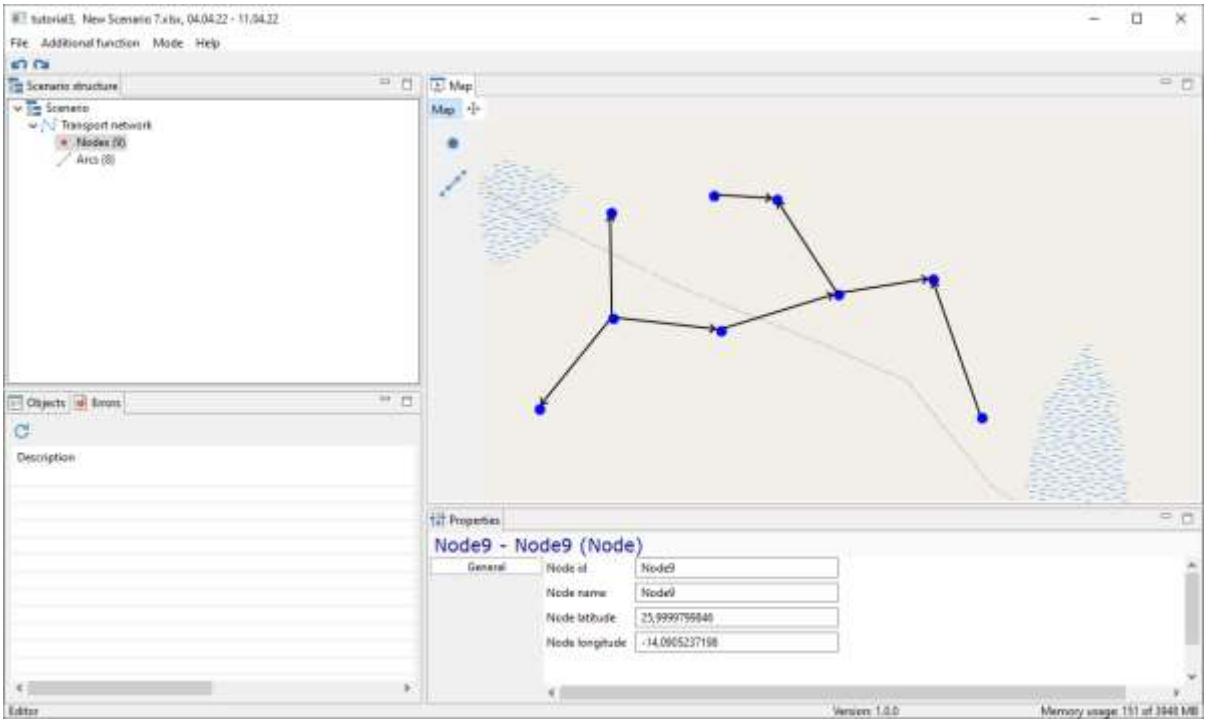
Для создания нового (пустого) сценария выберите команду 'File' → 'New'. Новый сценарий будет создан, в окне 'Scenario structure' появится структура сценария.



Модель данных сценария в созданном приложении уже содержит классы транспортной сети. Их легко добавить напрямую на карту. Для создания нового узла перейдите в окно 'Map'. С левой стороны окно содержит панель инструментов с кнопками, которые используются для добавления узлов и рёбер на холст. Для добавления ребра создайте как минимум два узла, затем нажмите кнопку 'Arc'и последовательно кликните на узлах, которые хотите соединит ребром.

Нажмите кнопку 'Map' в левом верхнем углу окна 'Map для отображения географической карты под узлами и рёбрами. Кнопка 'Center' рядом с кнопкой 'Map' центрирует холст так, чтобы все узлы и рёбра были видны пользователю.

Нажмите и удерживайте правую кнопку мыши для перемещения карты. Нажмите кнопку 'Ctrl' и используйте колесо мыши для приближения и отдаления карты.



В левой нижней части перспективы редактирования находится окно 'Objects'. Каждый раз, когда вы выбираете узел или ребро на карте в окне 'Map', в окне 'Objects' отображаются объекты соответствующего типа. То же самое происходит, когда вы выделяете узлы ('Nodes') или рёбра ('Arcs') в окне Scenario 'structure'.

Окна 'Objects' и 'Map' синхронизированы: каждый раз, когда вы выбираете объект в одном окне, – он также выделяется и в другом окне.

Свойства выделенных объектов отображаются в окне 'Properties', в котором вы можете их редактировать. Например, вы можете вручную изменить географическую широту узла, и он соответствующим образом изменит своё расположение на карте.

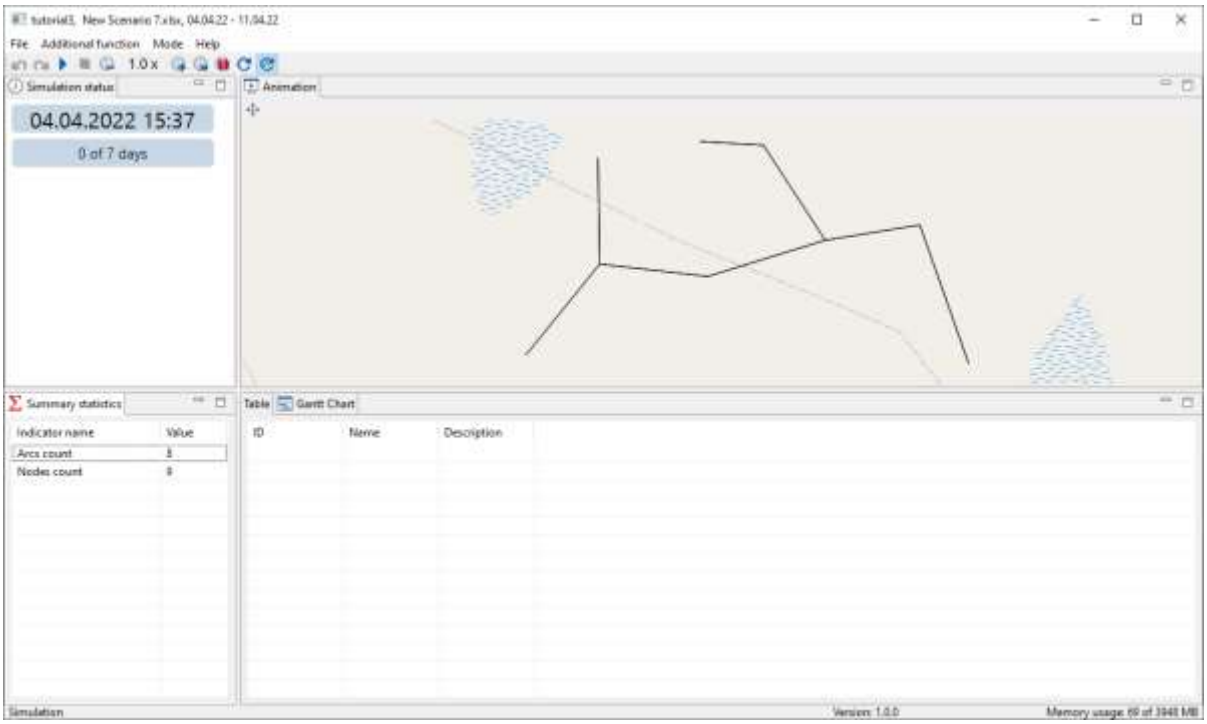
Сценарий может быть сохранен в файл и загружен из файла. Для этого в режиме редактирования используйте кнопки меню 'File' → 'Save' (или 'Save as') и 'File' → 'Open'.

Сценарий сохраняется в Excel-файл, структура которого отражает структуру модели данных: каждый класс модели данных соответствует листу Excel-файла. Если ваша модель данных содержит зависимости вида «многие-ко-многим», создаётся отдельный лист для разбиения зависимости «многие-ко-многим» на две зависимости «один-ко-многим».

Названия листов и колонок можно настроить (см. классы `ObjectType` и `EMFExcelTransformFactory` в подпроекте 'application'). Это удобно при ручном редактировании файла сценария Excel вне приложения.

3.3.4.3 Режим моделирования

Выберите 'Mode' → 'Simulation' для перехода в режим имитационного моделирования. Это режим, где вы можете запускать выполнение моделирования и наблюдать его текущее состояние и результаты моделирования.



Для управления процессом моделирования используется панель инструментов в верхней части окна перспективы моделирования.

Вы можете использовать кнопки 'Run' и 'Stop' для запуска и остановки моделирования. Несмотря на название, кнопка 'Stop' в действительности означает 'Pause': после остановки моделирования вы можете нажать кнопку 'Run' снова для продолжения моделирования.

Скорость моделирования может быть увеличена или уменьшена. Вы также можете включить режим максимальной скорости моделирования.

Диапазон периода моделирования задаётся в настройках сценария. Для его изменения нужно вернуться в режим редактирования, выбрать объект 'Scenario' в окне 'Scenario structure' и отредактировать даты начала и окончания сценария ('Begin date' и 'End date') в окне свойств 'Properties'.

Окно 'Simulation status' в левой верхней части перспективы моделирования отображает краткую статистику выполнения моделирования.

Окно 'Summary statistics' в левой нижней части перспективы моделирования отображает выходную статистику результатов моделирования. При создании нового приложения отображается количество узлов и рёбер. Реальные примеры

выходной статистики: общая дистанция, пройденная единицами техники, общая сумма затрат и т.д.

В окне 'Animation' отображаются объекты, работа которых будет моделироваться, и карта местности на заднем плане с заданным графом дорог.

3.3.4.4 Обзор исходного кода созданного приложения

Созданное приложение содержит 5 подпроектов (bundles):

1. 'application': основной подпроект, который создаёт графический интерфейс и зависит от других четырёх подпроектов
2. 'common': локализация и классы, используемые несколькими подпроектами
3. 'datamodel': модель данных начального состояния (т.е., модель данных сценария)
4. 'graphicaleditor': этот подпроект отвечает за окно карты ('Map') в перспективе редактирования
5. 'simulation': здесь содержится вся логика моделирования.