

**Автономная некоммерческая организация профессионального образования
«ПЕРМСКИЙ ГУМАНИТАРНО-ТЕХНОЛОГИЧЕСКИЙ КОЛЛЕДЖ»
(АНО ПО «ПГТК»)**

**МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ
ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ
МДК 02.01. РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ**

для специальности

**09.02.11 Разработка и управление программным обеспечением
(код и наименование специальности)**

Квалификация выпускника

Программист

Форма обучения

Очная

Пермь 2026

Методические рекомендации по выполнению практических работ
МДК 02.01. РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ составлен в соответствии с требованиями Федерального государственного образовательного стандарта среднего профессионального образования по специальности 09.02.11 Разработка и управление программным обеспечением

Данные методические рекомендации помогут организовать самостоятельную деятельность студентов на основе деятельного и компетентного подходов к обучению, что соответствует ФГОС СПО по специальности 09.02.11 Разработка и управление программным обеспечением.

Автор – составитель: Могильникова Н.С., старший преподаватель.

Методические рекомендации по выполнению практических работ предназначен для оценивания достижений запланированных результатов по дисциплине МДК 02.01. РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ. Методические рекомендации по выполнению практических работ представляет собой комплект материалов для проведения практических занятий (в форме практической подготовке) и осуществления контроля за выполнением работ.

Методические рекомендации по выполнению практических работ позволяет оценивать:

Код ОК, ПК	Уметь	Знать	Владеть навыками
ОК 01 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам;	распознавать задачу и/или проблему в профессиональном и/или социальном контексте, анализировать и выделять её составные части определять этапы решения задачи, составлять план действия, реализовывать составленный план, определять необходимые ресурсы выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы владеть актуальными методами работы в профессиональной и смежных сферах оценивать результат и последствия своих действий (самостоятельно или с помощью наставника)	актуальный профессиональный и социальный контекст, в котором приходится работать и жить структура плана для решения задач, алгоритмы выполнения работ в профессиональной и смежных областях основные источники информации и ресурсы для решения задач и/или проблем в профессиональном и/или социальном контексте методы работы в профессиональной и смежных сферах порядок оценки результатов решения задач профессиональной деятельности	
ОК 02 Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности;	определять задачи для поиска информации, планировать процесс поиска, выбирать необходимые источники информации выделять наиболее значимое в перечне информации, структурировать получаемую информацию, оформлять результаты поиска оценивать практическую значимость результатов поиска	номенклатура информационных источников, применяемых в профессиональной деятельности приемы структурирования информации формат оформления результатов поиска информации современные средства и устройства информатизации, порядок их применения программное обеспечение в профессиональной деятельности, в том числе цифровые средства	

	применять средства информационных технологий для решения профессиональных задач использовать современное программное обеспечение в профессиональной деятельности использовать различные цифровые средства для решения профессиональных задач	психологические основы деятельности коллектива	
ОК 05 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста;	грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке проявлять толерантность в рабочем коллективе	правила построения устных сообщений особенности социального и культурного контекста	
ОК 09 Пользоваться профессиональной документацией на государственном и иностранном языках.	понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы участвовать в диалогах на знакомые общие и профессиональные темы строить простые высказывания о себе и о своей профессиональной деятельности кратко обосновывать и объяснять свои действия (текущие и планируемые) писать простые связные сообщения на знакомые или интересующие профессиональные темы	правила построения простых и сложных предложений на профессиональные темы основные общеупотребительные глаголы (бытовая и профессиональная лексика) лексический минимум, относящийся к описанию предметов, средств и процессов профессиональной деятельности особенности произношения правила чтения текстов профессиональной направленности	
ПК. 2.1 Проектировать модули	проектировать модули, соответствующие бизнес-задачам.	основные принципы проектирования модулей программного обеспечения	проектирования модулей ПО с

программного обеспечения.	создавать архитектурные диаграммы и документацию. определять структуру и интерфейсы модулей анализировать требования к модулю и определять его функциональность проектировать архитектуру модуля, включая выбор подходящих паттернов проектирования и структуры данных создавать диаграммы классов, последовательностей и прочих диаграмм для визуализации проектируемого модуля выбирать подходящие языки программирования и технологии для реализации модуля проектировать интерфейсы программного обеспечения для взаимодействия с другими модулями и системами учитывать требования к масштабируемости, производительности и безопасности при проектировании модуля проводить анализ и оптимизацию проектируемого модуля для повышения его эффективности и качества	языки программирования и технологии для реализации модулей паттерны проектирования и структуры данных для создания эффективных и масштабируемых модулей методы анализа требований и способов определения функциональности модуля принципы создания интерфейсов для взаимодействия с другими модулями и системами принципы обеспечения безопасности, производительности и масштабируемости при проектировании модулей методы анализа и оптимизации проектируемых модулей для повышения их эффективности и качества	учетом требований заказчика. создания архитектурных диаграмм и спецификаций модулей. определения интерфейсов и взаимодействия модулей в системе.
ПК. 2.2 Разрабатывать модули программного обеспечения.	разрабатывать модули программного обеспечения с использованием различных языков программирования и технологий применять паттерны проектирования и структуры данных для создания эффективных	язык программирования, основные конструкции, синтаксис паттерны проектирования структуры данных принципы создания интерфейсов для взаимодействия с другими модулями и системами, таких как REST API, SOAP работа с инструментальным	создание модулей программного обеспечения на различных языках программирования отладки и тестирования разработанных модулей применение структурного и объектно-

	и масштабируемых модулей анализировать требования и определять функциональность модуля создавать интерфейсы для взаимодействия с другими модулями и системами обеспечивать безопасность, производительность и масштабируемость при разработке модулей оптимизировать проектируемые модули для повышения их эффективности и качества работать с системой контроля версий улучшать производительность модулей, выявляя и устраняя узкие места проводить анализ и мониторинг производительности приложений применять инструменты для рефакторинга и оптимизации программного кода	программным обеспечением методы оптимизации кода и алгоритмов эффективные алгоритмы и структуры данных для повышения производительности многопоточность в программных модулях методы оптимизации сетевых протоколов для ускорения обмена данными кэширование данных управление памятью техники повышения производительности программного обеспечения	ориентированного программирования оптимизации кода и алгоритмов программных модулей для увеличения производительности мониторинга и анализа производительности приложений
ПК. 2.3 Выполнять интеграцию модулей и компонентов программного обеспечения.	интегрировать модули и компоненты, обеспечивая их взаимодействие работать с API и устанавливать соединения между компонентами отслеживать и устранять конфликты и ошибки интеграции анализировать и определять зависимости между модулями и компонентами работать с различными форматами данных и протоколами передачи данных	общих принципов функционирования аппаратных, программных и программно-аппаратных средств администрируемой информационно-коммуникационной системы международных стандартов локальных вычислительных сетей методы и подходы к интеграции модулей и компонентов принципы версионирования и управления изменениями при интеграции принципы безопасности при интеграции модулей и компонентов	интеграции программных модулей и компонентов в единое программное решение работы с API и веб-сервисами для взаимодействия между модулями работы с интеграционными платформами и инструментами обеспечения совместимости и стабильности системы

<p>ПК. 2.4</p> <p>Выполнять тестирование и отладку программного обеспечения.</p>	<p>анализировать требования к программному обеспечению и составлять планы тестирования. создавать тестовые сценарии и тест-кейсы для проверки функциональности и соответствия требованиям. выполнять тестирование программного обеспечения вручную и автоматизировать процесс тестирования. анализировать результаты тестирования и документировать найденные ошибки. разрабатывать стратегии отладки и исправлять ошибки в программном обеспечении. выполнять модульные тесты с использованием инструментов тестирования, в том числе автоматизированного тестирования использовать системы контроля дефектов ПО составлять отчет о выполнении тестирования ПО</p>	<p>принципы и методы тестирования программного обеспечения. основы программирования и архитектуры программного обеспечения. основы баз данных и SQL-запросов. инструменты для автоматизации тестирования основы разработки и отладки программного обеспечения на разных языках программирования понятие дефекта программного обеспечения критерии качества ПО виды и типы тестирования ПО техники ручного тестирования техники автоматизированного тестирования жизненный цикл дефекта ПО принципы работы в системе контроля дефектов основные понятия о качестве ПО</p>	<p>отладки программного обеспечения на уровне программных модулей тестирования программного обеспечения формирования тестовых сценариев подготовки тестовых платформ (установка операционной системы, дополнительного ПО и другого по необходимости) оценки объема тестирования ПО с целью определения необходимых ресурсов для его выполнения настройки тестовой среды и аппаратных средств для выполнения тестирования ПО в соответствии с заданием на тестирование в пределах своей компетенции формирования и представления отчетности о подготовке к выполнению задания на тестирование ПО в соответствии с установленными регламентами выполнения тестовых процедур на тестовых данных</p>
<p>ПК. 2.5</p> <p>Осуществлять документирование программных модулей программного обеспечения.</p>	<p>описывать функциональность модулей в документации создавать диаграммы для иллюстрации работы модулей программировать с использованием комментариев для документирования кода</p>	<p>стандарты технической документации принципы документирования программного обеспечения инструменты для создания технической документации и комментирования кода</p>	<p>создания технической документации для модулей документирования кода, API и интерфейсов работы со специализированным ПО по документированию программного кода</p>

	<p>использовать специальные метки/теги для отметки важных частей кода в документации вести журнал изменений и фиксировать обновления программных модулей разбивать модули на логические блоки и описывать каждый блок отдельно включать в документацию особенности модулей, такие как ограничения, уязвимости или оптимальные настройки проводить регулярное обновление документации при изменении модулей или добавлении нового функционала.</p>		
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--	--

Перечень практических занятий.

Практическое занятие «Моделирование интеллектуальных систем»

Цель: знать виды и классификацию локальных сетей, физические среды передачи данных, научиться устанавливать и настраивать сетевой интерфейс.

Основные сведения:

I. Построение модели в среде AnyLogic.

Пакет AnyLogic – отечественный профессиональный инструмент нового поколения, который предназначен для разработки и исследования имитационных моделей. Разработчик продукта – компания «Экс Джей Текнолоджис» (XJ Technologies), г. Санкт-Петербург; электронный адрес: www.xjtek.ru.

Графическая среда моделирования поддерживает проектирование, разработку, документирование модели, выполнение компьютерных экспериментов, оптимизацию параметров относительно некоторого критерия.

При разработке модели можно использовать элементы визуальной графики: диаграммы состояний (стейтчарты), сигналы, события (таймеры), порты и т.д.; синхронное и асинхронное планирование событий; библиотеки активных объектов.

Пользовательский интерфейс

После запуска AnyLogic открывается рабочее окно, в котором для продолжения работы надо создать новый проект или открыть уже существующий.

Чтобы создать новый проект, щелкните по соответствующей кнопке на панели инструментов или выберите пункт меню **Файл | Создать проект** и затем из ниспадающего меню – **Модель**. Откроется диалоговое окно **Новая модель**, где задается имя и местоположение нового проекта. Далее следуйте указаниям *Мастера создания модели*. Можно создавать новую модель «с нуля» или использовать шаблон.

При открытии проекта (нового или существующего) AnyLogic всегда открывает среду разработки проекта – графический редактор модели (рис. 3.1). Рассмотрим основные составляющие этого редактора.

Окно проекта обеспечивает легкую навигацию по элементам проекта, таким как пакеты, классы и т.д. Поскольку проект организован иерархически, то он отображается в виде дерева: сам проект образует верхний уровень дерева рабочего проекта, пакеты – следующий уровень, классы активных объектов и сообщений – следующий и т.д. Можно копировать, перемещать и удалять любые элементы дерева объектов, легко управляя рабочим проектом.

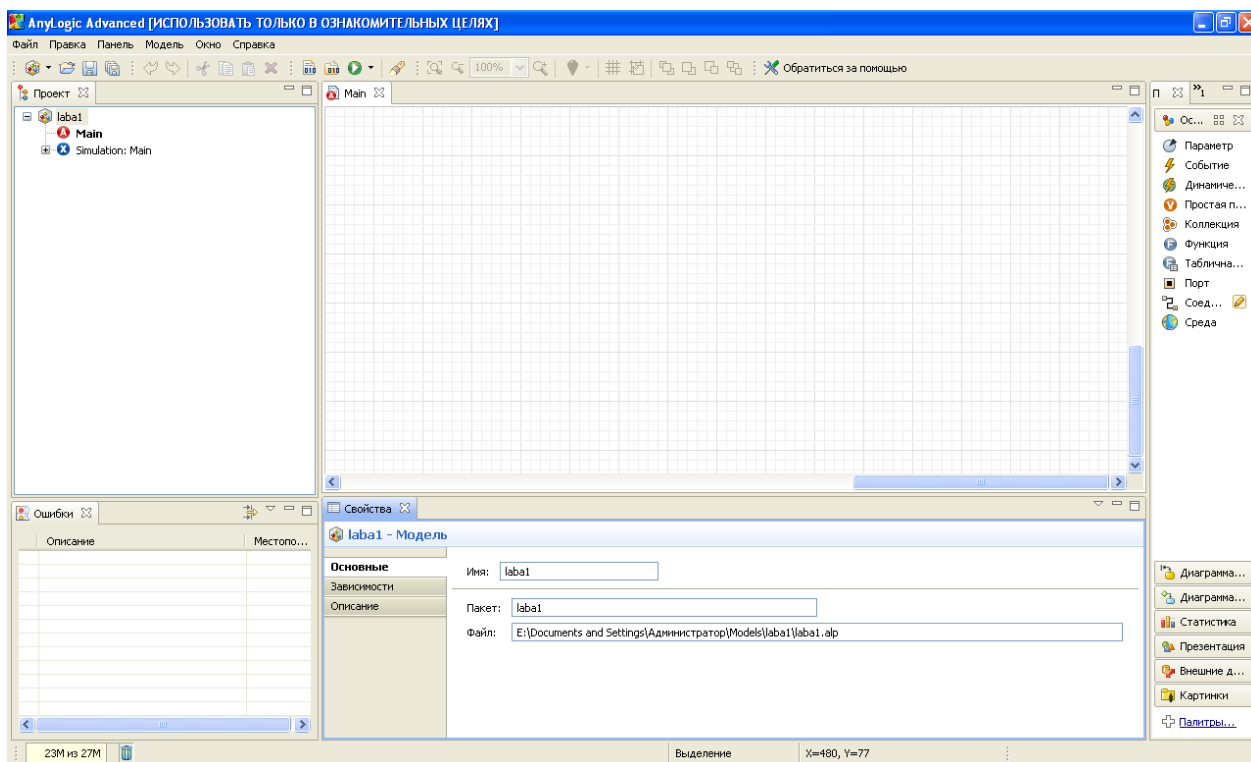


Рис. 3.1 Главное окно проекта.

Одна из ветвей в дереве проекта имеет название *Simulation:Main* (эксперимент). Эксперимент хранит набор настроек, с помощью которых конфигурируют работу модели. Один эксперимент создается автоматически при создании проекта. Это *простой эксперимент* с именем *Simulation*, позволяющий визуализировать модель с помощью анимации и поддерживающий инструменты для отладки модели. Простой эксперимент используется в большинстве случаев. Поддерживается ещё несколько типов экспериментов для различных задач моделирования.

Структурная диаграмма. При построении модели нужно задать ее структуру (т.е. описать, из каких частей состоит модель системы) и поведение отдельных объектов системы. В AnyLogic структурными элементами модели являются так называемые *активные объекты*. Активный объект имеет структуру и поведение. Элементы структуры – это другие активные объекты, включенные как составные элементы данного активного объекта, и связи, которые существуют между включенными активными объектами. Активные объекты могут содержать: события, стейтчарты, переменные, функции, уравнения, параметры.

Структура активного объекта задается графически на структурной диаграмме. Поведение задается с помощью стейтчарта и определяет реакции активного объекта на внешние события – логику его действий во времени.

Диаграмма состояний (или стейтчарт – *statechart*) – это модифицированные графы переходов конечного автомата. Стейтчарт позволяет графически задать пространство состояний алгоритма поведения объекта, события, которые являются причинами срабатывания переходов из

одних состояний в другие, и действия, происходящие при смене состояний. Стейтчарты в AnyLogic поддерживают следующие типы событий: *сигнал* – объект может послать сигнал другому объекту, чтобы уведомить его о чем-то; *таймаут* – в течение заданного промежутка времени в стейтчарте ничего не происходит; *событие* – событие, при котором значение булево выражения становится «истина».

Кроме того, в окне редактора для модели можно построить двумерное или трехмерное анимационное представление, которое помогает понять, что происходит с моделью во времени. Именно в этом окне визуально представляется имитация поведения моделируемой системы. Элементы анимационной картинки имеют свои параметры, которые могут быть связаны с переменными и параметрами модели. Изменение переменных модели во времени ведет к изменению графического образа, что позволяет пользователю наглядно представить динамику моделируемой системы с помощью динамически меняющейся графики.

Окно свойств. В редакторе AnyLogic для каждого выделенного элемента модели существует свое окно свойств, в котором указываются свойства (параметры) этого элемента. При выделении какого-либо элемента в окне редактора снизу появляется окно свойств, показывающее параметры данного выделенного элемента. Окно свойств содержит несколько вкладок. Каждая вкладка содержит элементы управления, такие как поля ввода, флажки, переключатели, кнопки и т.д., с помощью которых можно просматривать и изменять свойства элементов модели. Число вкладок и их внешний вид зависит от типа выбранного элемента.

Окно палитры. Содержит элементы (графические объекты), которые могут быть добавлены на структурную диаграмму. Элементы разбиты по группам, отображаемым на разных вкладках. Чтобы добавить объект палитры на диаграмму, щелкните сначала по элементу в палитре, а затем щелкните по диаграмме.

Параметры. Активный объект может иметь параметры. Параметры обычно используются для задания характеристик объекта. Вы можете задать различные значения параметров для разных объектов одного и того же класса, что требуется в тех случаях, когда объекты имеют одинаковое поведение, но их характеристики разные. Возможно описание параметров любых *Java*-классов.

Чтобы создать параметр класса активного объекта (рис. 3.2), в окне **Проект** щелкните мышью по классу активного объекта. В окне **Свойства** щелкните по кнопке **Новый параметр**. Задайте свойства параметра в открывшемся диалоговом окне **Параметр**.

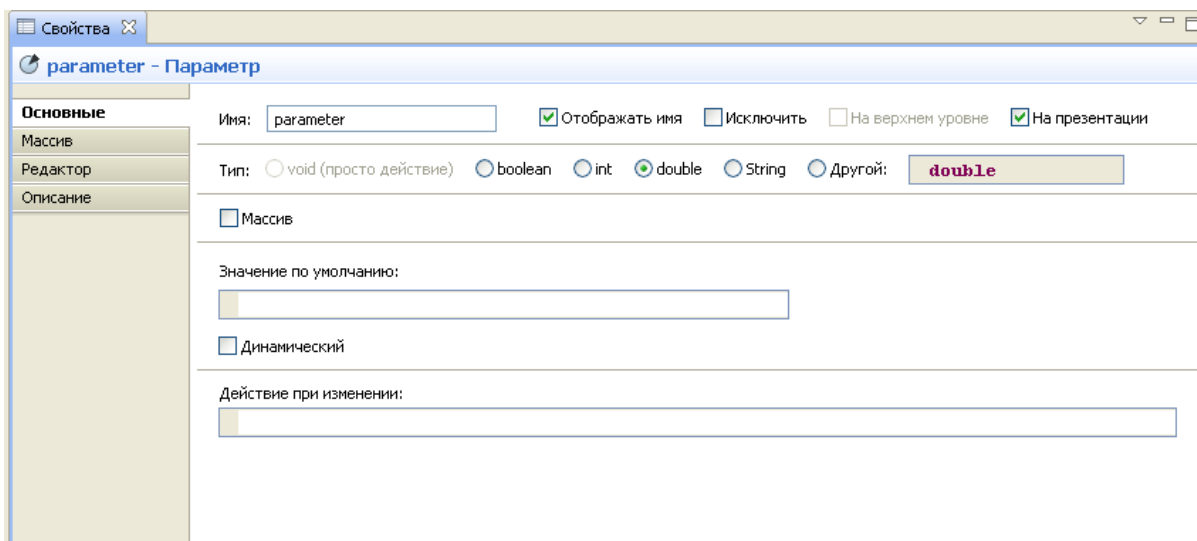


Рис. 3.2. Окно параметров.

Переменные. Активный объект может содержать переменные. Переменные могут быть либо внутренними, либо интерфейсными. Активный объект может иметь переменные, моделирующие, меняющиеся во времени величины. Переменные могут быть вынесены в интерфейс активного объекта и связаны с переменными других активных объектов. Тогда при изменении значения одной переменной будет немедленно меняться и значение связанной с ней зависимой переменной другого объекта. Этот механизм обеспечивает непрерывное и/или дискретное взаимодействие объектов.

Передача сообщений. AnyLogic позволяет передавать информацию от одного объекта другому путем передачи специальных пакетов данных – сообщений. С помощью передачи сообщений можно реализовать механизм оповещения – сообщения будут представлять команды или сигналы, посылаемые системой управления. *Можно также смоделировать поток заявок*, в этом случае сообщения будут представлять собой заявки – объекты, которые производятся, обрабатываются, обслуживаются или еще каким-нибудь образом подвергаются воздействию моделируемого процесса (документы в модели бизнес-процесса, клиенты в модели системы массового обслуживания, детали в производственных моделях).

Сообщения принимаются и посылаются через специальные элементы активных объектов – *порты*. Обмен сообщениями возможен только между портами, соединенными соединителями – элементами, играющими роль путей движения сообщений.

Чтобы соединить порты вложенных объектов, выберите в палитре «Основная» инструмент **Соединитель**, перетащите его на диаграмму и подсоедините к портам. Также можно сделать двойной щелчок мышью по одному порту, а затем двойной щелчок мышью по второму порту.

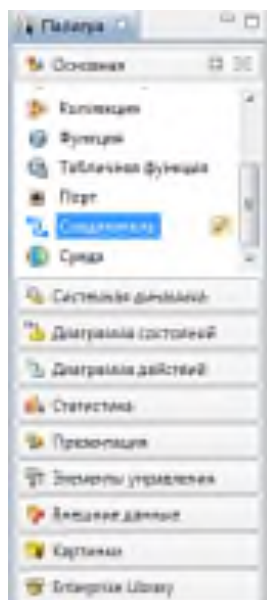
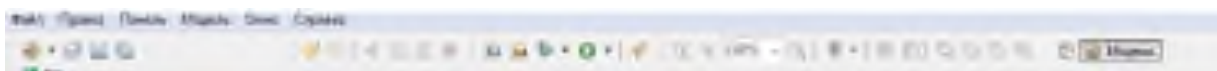


Рис.3.3. Выбор соединителя.

Запуск и просмотр модели. Запускать и отлаживать модель можно с помощью меню **Модель** и панели инструментов:



При исполнении модели запустится компилятор, который построит исполняемый код модели в языке *Java*, скомпилирует его и затем запустит модель на исполнение.

Для запуска модели щелкните по кнопке **Выполнить**, затем выберите эксперимент из выпадающего списка. После этого откроется окно презентации, отображающее созданную презентацию для запущенного эксперимента. Щелкните по кнопке, чтобы запустить модель и перейти на презентацию.

В окне презентации можно увидеть: анимированную диаграмму модели, окна инспекта элементов модели, ожившую анимацию, диаграммы состояний, графики статистики.

В ходе выполнения лабораторной работы необходимо научиться создавать дискретно-событийные модели с помощью библиотеки Enterprise Library пакета AnyLogic.

Общая информация о создании моделей в Enterprise Library

Для создания новой модели щелкните мышью по кнопке **Создать проект**. Появится диалоговое окно, в котором вы должны будете дать имя файлу вашей модели и выбрать каталог, где он будет храниться.

Рассмотрим рабочее окно AnyLogic. В левой части рабочей области находится панель «Проект». Панель «Проект» обеспечивает легкую навигацию по элементам моделей, открытых в текущий момент времени.

В правой рабочей области отображается панель «Палитра», а внизу – панель «Свойства».

Панель «Палитра» содержит разделенные по категориям элементы, которые могут быть добавлены на диаграмму класса активного объекта или эксперимента. Панель «Свойства» используется для просмотра и изменения свойств выбранного в данный момент элемента (или элементов) модели.

В центре рабочей области AnyLogic открывается графический редактор диаграммы класса активного объекта *Main*.

Чтобы добавить объект на блок-схему модели, щелкните по объекту в окне палитры **Enterprise Library** и перетащите его мышью на структурную диаграмму. При этом его свойства будут отображены на панели «Свойства». В этом окне вы можете изменять свойства элемента в соответствии с требованиями вашей модели. Позднее для изменения свойств элемента нужно будет сначала щелчком мыши выделить его на диаграмме или в дереве проекта.

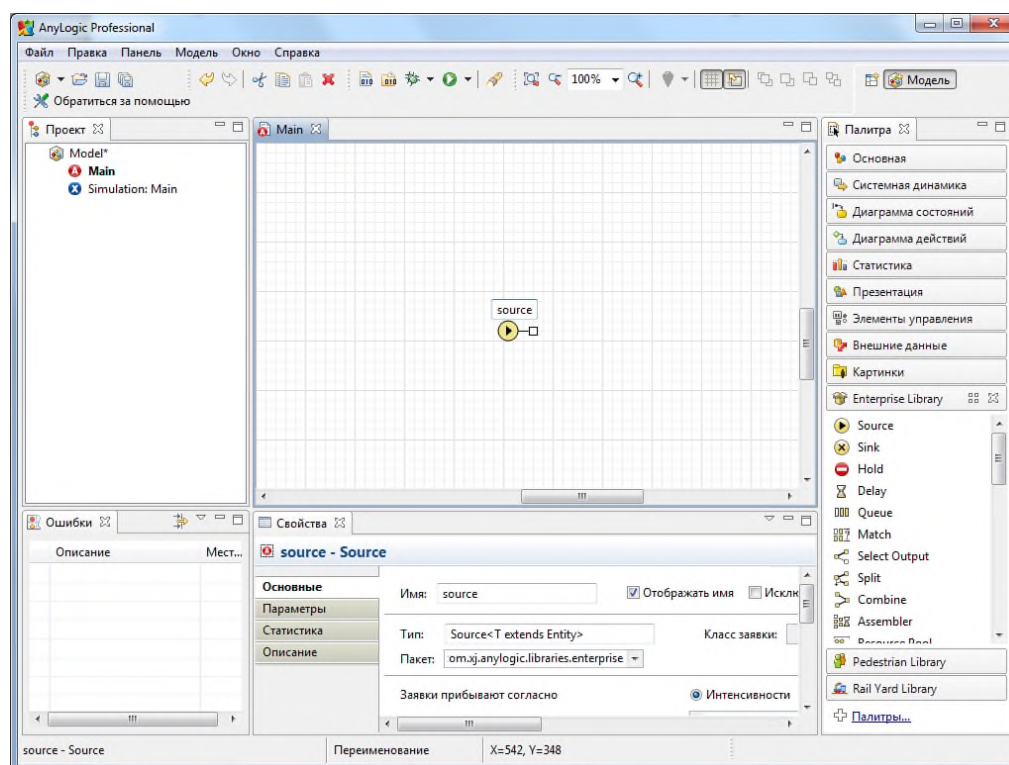


Рис.3.4. Развернутая библиотека Enterprise Library

и ее элемент, помещенный на схему.

Объекты должны взаимодействовать между собой, поэтому вы должны будете соединять их друг с другом. Можно соединять объекты с помощью мыши, перетаскиванием порта одного объекта на порт другого или с помощью специального средства «Соединитель». Чтобы соединить порты объектов, щелкните мышью по кнопке панели инструментов **Соединитель**, а затем щелкните мышью поочередно по обоим портам. Для добавления точки изгиба щелкните мышью по кнопке панели инструментов **Редактировать точки**.

Модель выполняется в соответствии с набором конфигурационных установок, называемым *экспериментом*. Вы можете создать несколько экспериментов и изменять рабочую конфигурацию модели, просто меняя текущий эксперимент модели. Один эксперимент,

названный *Simulation*, создается автоматически. Выберите его щелчком мыши по элементу дерева и измените настройки модели в окне **Свойства** (рис. 3.5). Окно **Свойства** имеет вкладки: основные, дополнительные, модельное время, презентация, окно, параметры, описание.

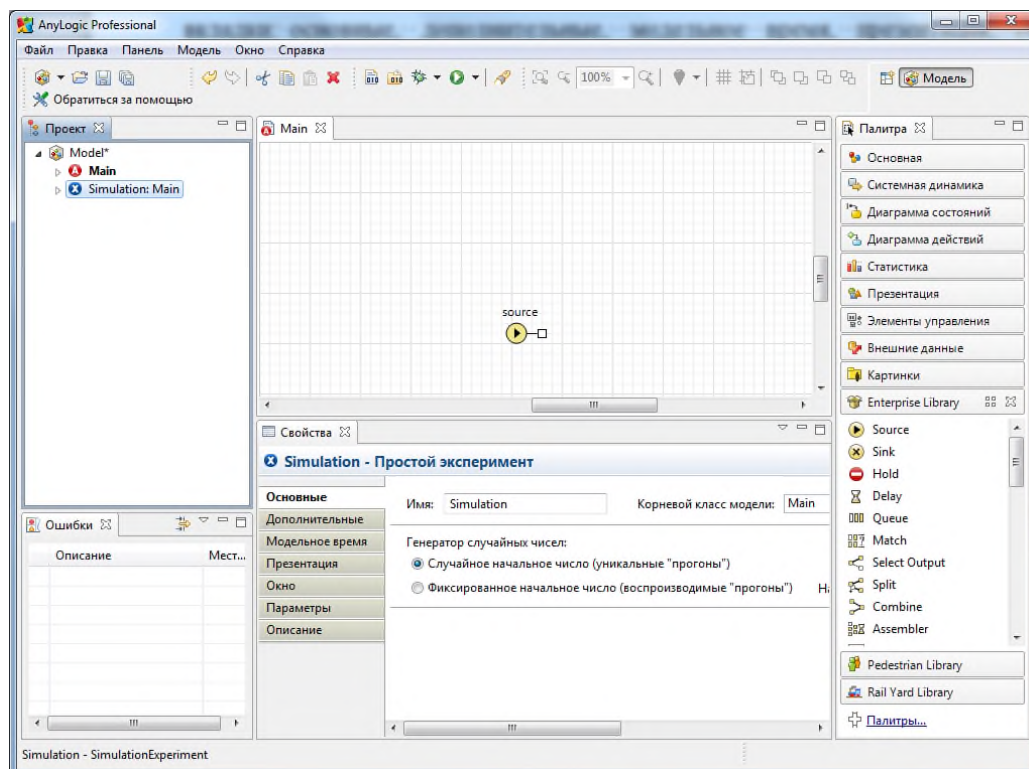


Рис. 3.5 Задание свойств эксперимента.

На вкладке **Основные** можно выбрать класс, который будет запущен при запуске модели. По умолчанию в качестве корневого объекта выбран объект класса *Main*, автоматически создаваемого в каждой модели. Вы можете переименовывать классы модели. Для этого нужно выделить класс щелчком мыши по значку класса в дереве модели и затем изменить его имя в окне **Свойства**.

На вкладке **Модельное время** можно:

1. задать режим моделирования. В режиме реального времени задается связь модельного времени с физическим, т.е. задается количество единиц модельного времени, выполняемых в одну секунду. Режим реального времени лучше всего подходит для показа анимации. В режиме виртуального времени модель выполняется без привязки к физическому времени – она выполняется так быстро, как это возможно. Данный режим лучше всего подходит, когда требуется моделировать работу системы в течение достаточно длительного периода времени;
2. запустить модель так, чтобы она работала бесконечно, но можно и остановить ее в заданный момент времени. Вы можете остановить модель по достижении переменной заданного значения или по выполнении какого-нибудь определенного условия.

Дополнительные свойства эксперимента (вкладка **Дополнительные**) позволяют управлять

выполнением модели. Можно задать действие перед и после запуска модели, а также задать численные методы для прогона и точность получаемых значений.

На вкладке **Презентация** можно определить вид и скорость выполнения прогона.

Моделирование одноканальной СМО с очередью.

Постановка задачи. В банковский офис обращаются клиенты. Офис представляет собой автоматизированный пункт обслуживания, в котором установлен банкомат. Банкомат обслуживает одновременно одного клиента. Клиенты прибывают по экспоненциальному закону с интенсивностью $\lambda=0,67$. Одновременно в офисе может находиться не более 15 клиентов. Интервал времени работы банкомата подчиняется треугольному закону распределения с параметрами $x_{\min}=0.8$, $x_{\max}=1,3$ предпочтительное значение 1.

Построение модели. Модель строится с «нуля». Банковский офис представляет собой систему массового обслуживания (СМО). Построение модели такой системы выполняется с помощью элементов библиотеки Enterprise Library. Для построения СМО используются элементы:

- Source – источник заявок.
- Queue – очередь ожидающих обслуживания заявок.
- Delay – Элемент моделирующий узел обслуживания.
- Sink – Элемент принимающий отработанные заявки.

Общий вид модели СМО банковского офиса показан на рисунке 3.6.

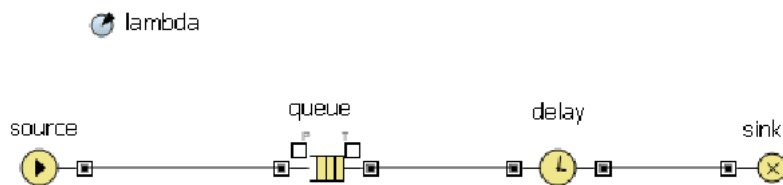


Рис. 3.6. Модель офиса

Источник заявок

Заявки – клиенты офиса пребывают с интенсивностью $\lambda=0.67$.

Источник заявок обладает следующими настройками:

- Заявки пребывают согласно интенсивности.
- Интенсивность прибытия равна λ . **Lambda** – параметр (панель «Основная» пункт **параметр**). Значение соответствует интенсивности потока клиентов, и равно 0,67.

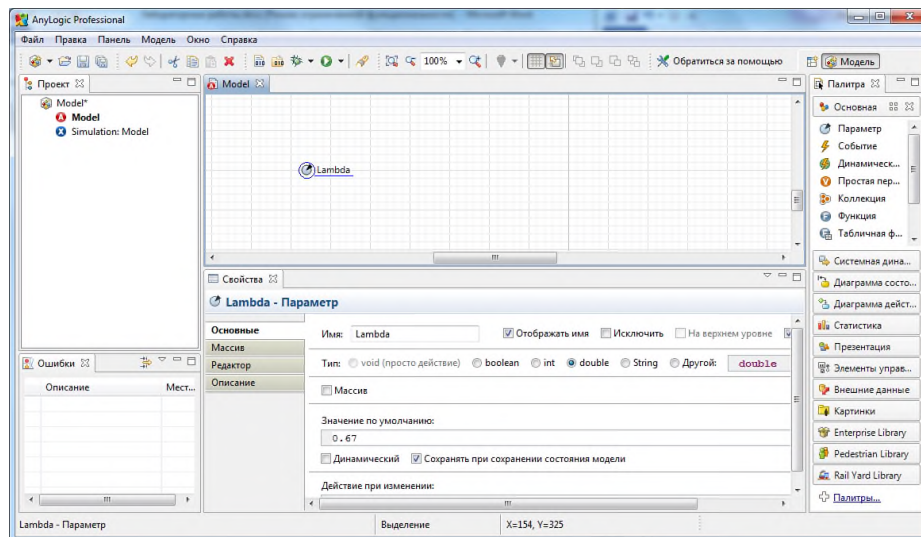


Рис. 3.7. Задание параметра интенсивности заявок.

Закон распределения потока заявок можно задать в свойстве *interarrivalTime* на вкладке *Параметры* для объекта *source*. По умолчанию распределение случайного потока заявок подчиняется экспоненциальному закону. –*exponential()*.

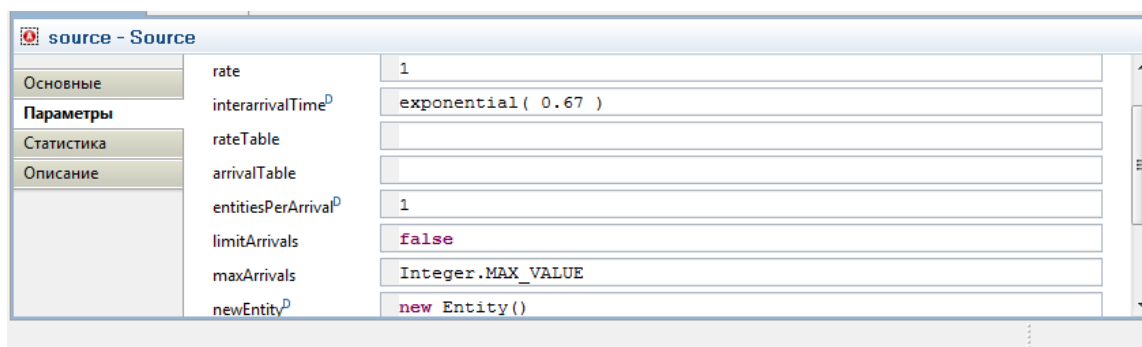


Рис. 3.8. Задание закона распределения потока заявок.

AnyLogic предоставляет функции и других случайных распределений, таких как:

- нормальное с дисперсией σ и мат. ожиданием m – $\text{normal}(\sigma, m)$;
- равномерное на отрезке $[a, b]$ – $\text{uniform}(a, b)$;
- треугольное с минимальным значением a , средним значением b и максимальным- c – $\text{triangular}(a, b, c)$;

и т.д.

- Количество заявок пребывающих за один раз равно единице.

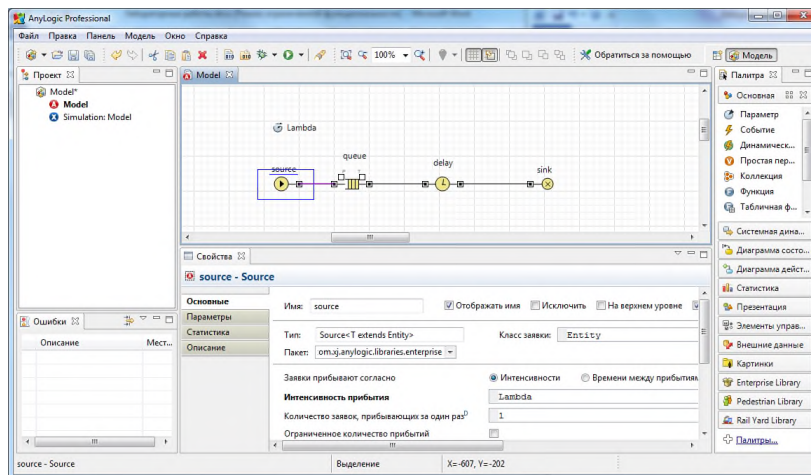


Рис. 3.9. Параметры источника заявок.

Очередь

Этот элемент характеризуется параметрами:

- Вместимость очереди равна 15.
- Включить сбор статистики – да.

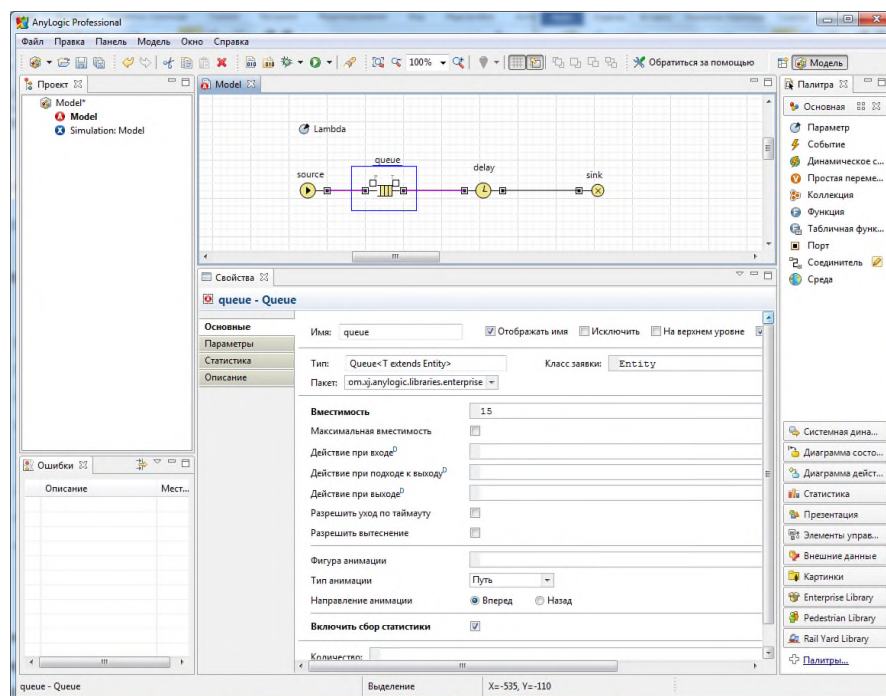


Рис. 3.10. Параметры очереди.

Узел обслуживания

Параметры элемента:

- Задержка задается явно.
- Время задержки равно: $\text{triangular}(0.8, 1.3, 1)$.
- Вместимость узла – один клиент.
- Включить сбор статистики- да.

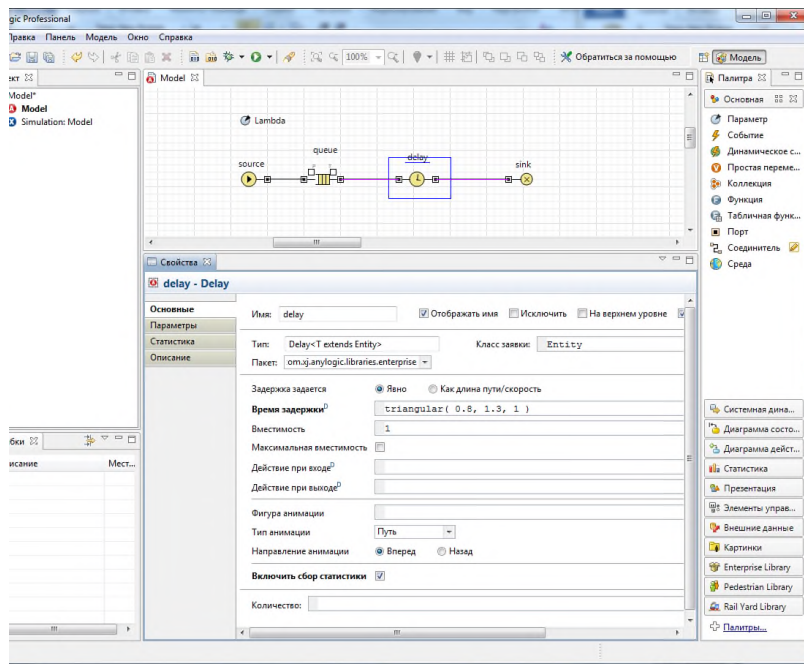


Рис. 3.11. Параметры узла обслуживания.

Элемент, принимающий заявки обладает параметрами настройки по умолчанию.

Настройте эксперимент модели:

- Модельное время – минуты.
- Время остановки модели не задано.
- Задайте Режим выполнения со скоростью 8.

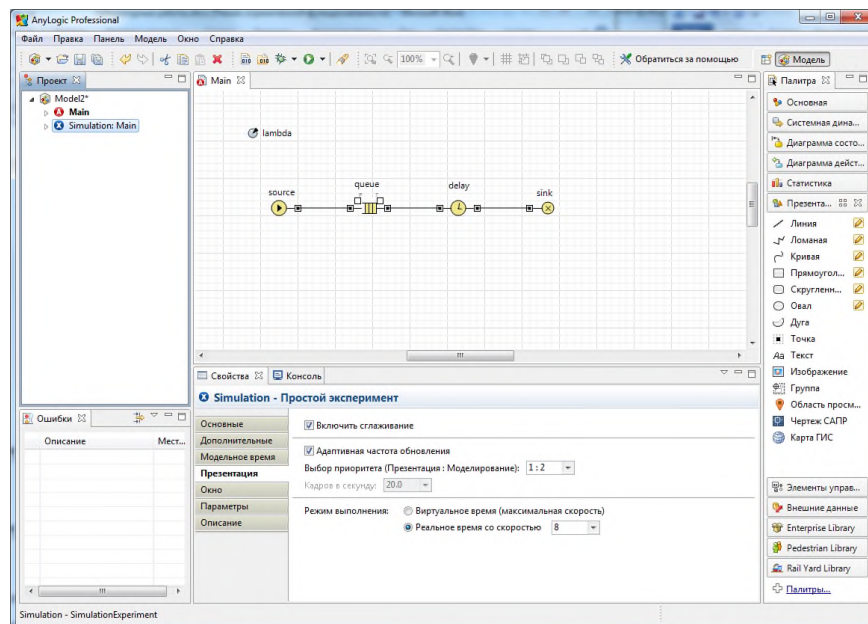


Рис. 3.12. Настройка параметров эксперимента.

Для запуска модели щелкните мышью по кнопке **Запустить**. Откроется окно с презентацией запущенного эксперимента. AnyLogic автоматически помещает на презентацию каждого простого эксперимента заголовок и кнопку, позволяющую запустить модель и перейти на презентацию.

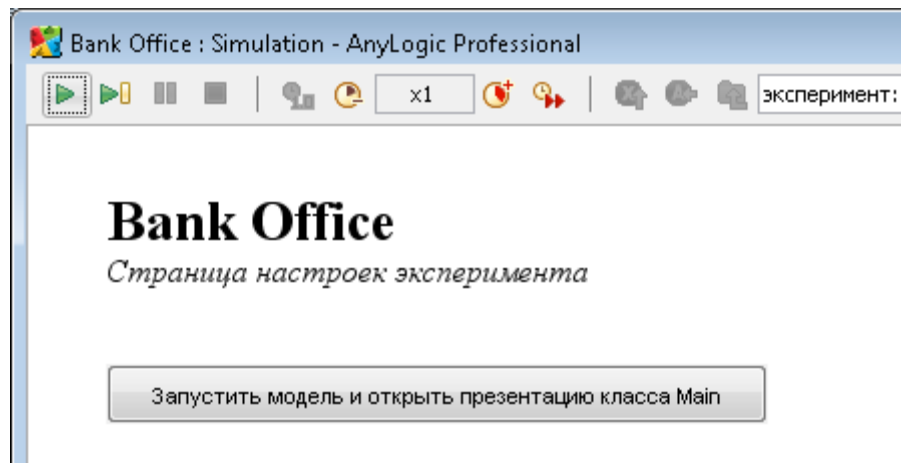


Рис. 3.13. Запуск эксперимента.

Щелкните по этой кнопке. AnyLogic переключится в режим работы модели. С помощью визуализированной блок-схемы вы можете проследить, сколько человек находится в очереди, сколько человек в данный момент обслуживается и т.д.

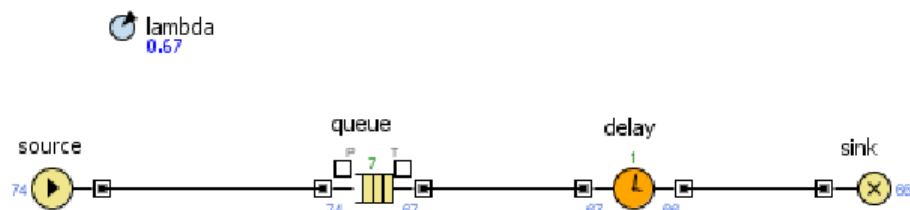


Рис.3.14. Вид работающей модели

Анимация модели

Покажем процесс обслуживания клиентов в виде анимации очереди, ведущей к банкомату, так как это показано на рисунке 3.15.

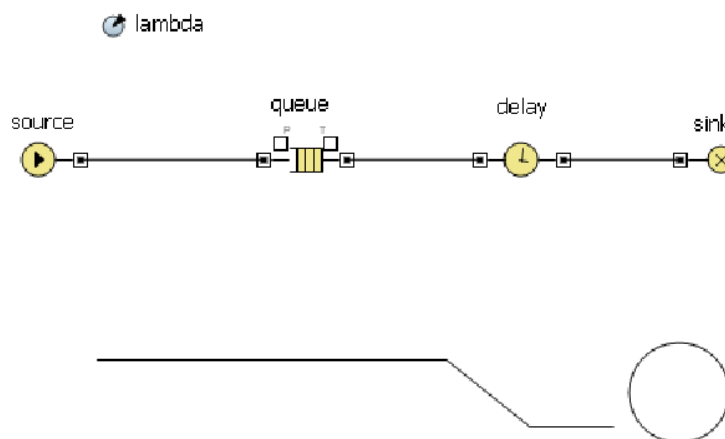


Рис.3.15. Анимация очереди

Банкомат представим в виде окружности. Когда клиент находится в банкомате, окружность будет окрашена в красный цвет, при свободном банкомате окружность закрашивается в зеленый цвет.

С помощью элемента «Овал» палитры «Презентация» разместите окружность и присвойте

ей имя ServicePoint. Цвет заливки должен изменяться динамически:

```
delay.size()>0 ? Color.red: Color.green
```

Здесь size() – метод объекта delay, который возвращает количество заявок-клиентов в приборе обслуживания.

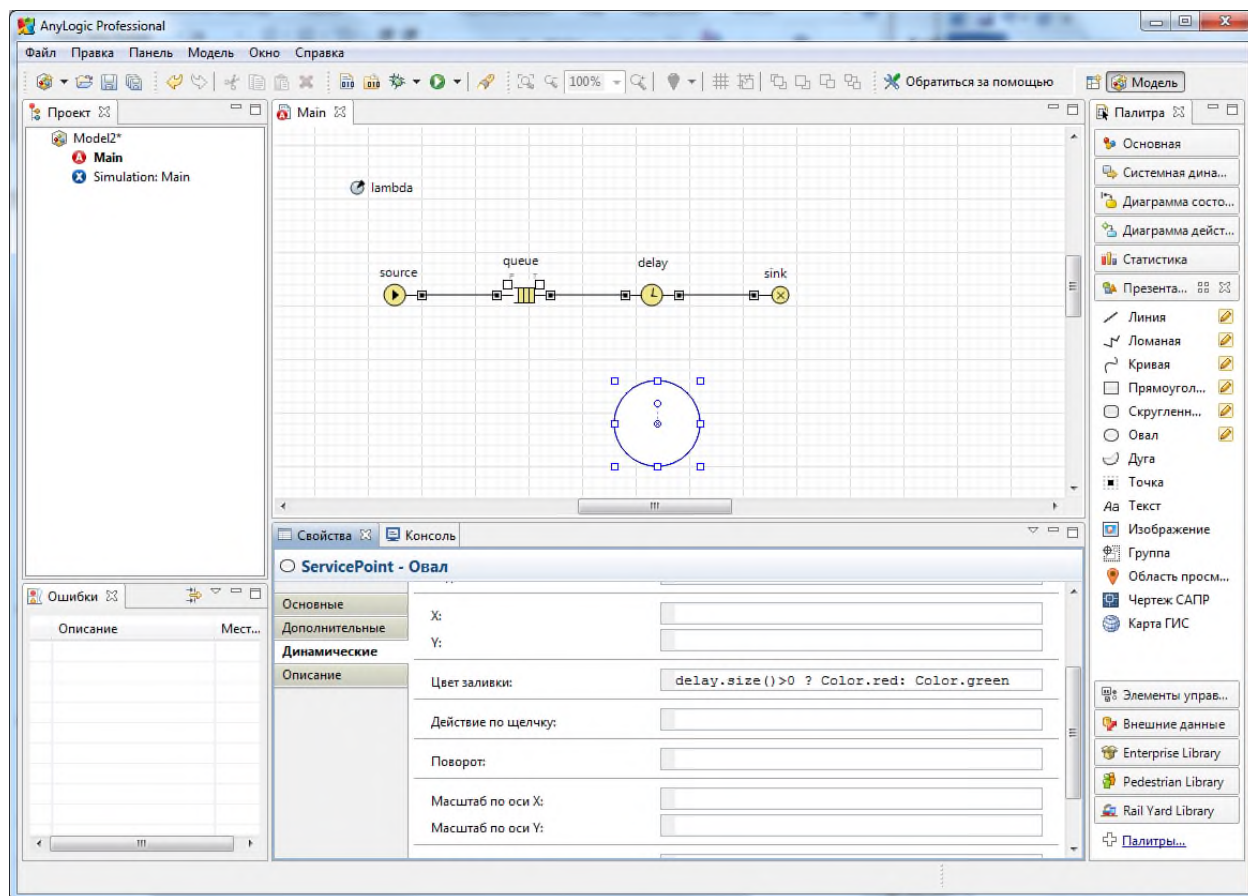



Рис.3.16. Задание свойств окружности – банкомата.

Для отображения очереди следует нарисовать ломаную линию (см. рисунок 3.15), используя элемент «Ломаная» из палитры «Презентация». Режим рисования включается после выполнения двойного щелчка по пиктограмме .

Рисование ломаной нужно выполнять по направлению движения клиентов к банкомату: слева на право. Ломаной присвойте имя GoToService.

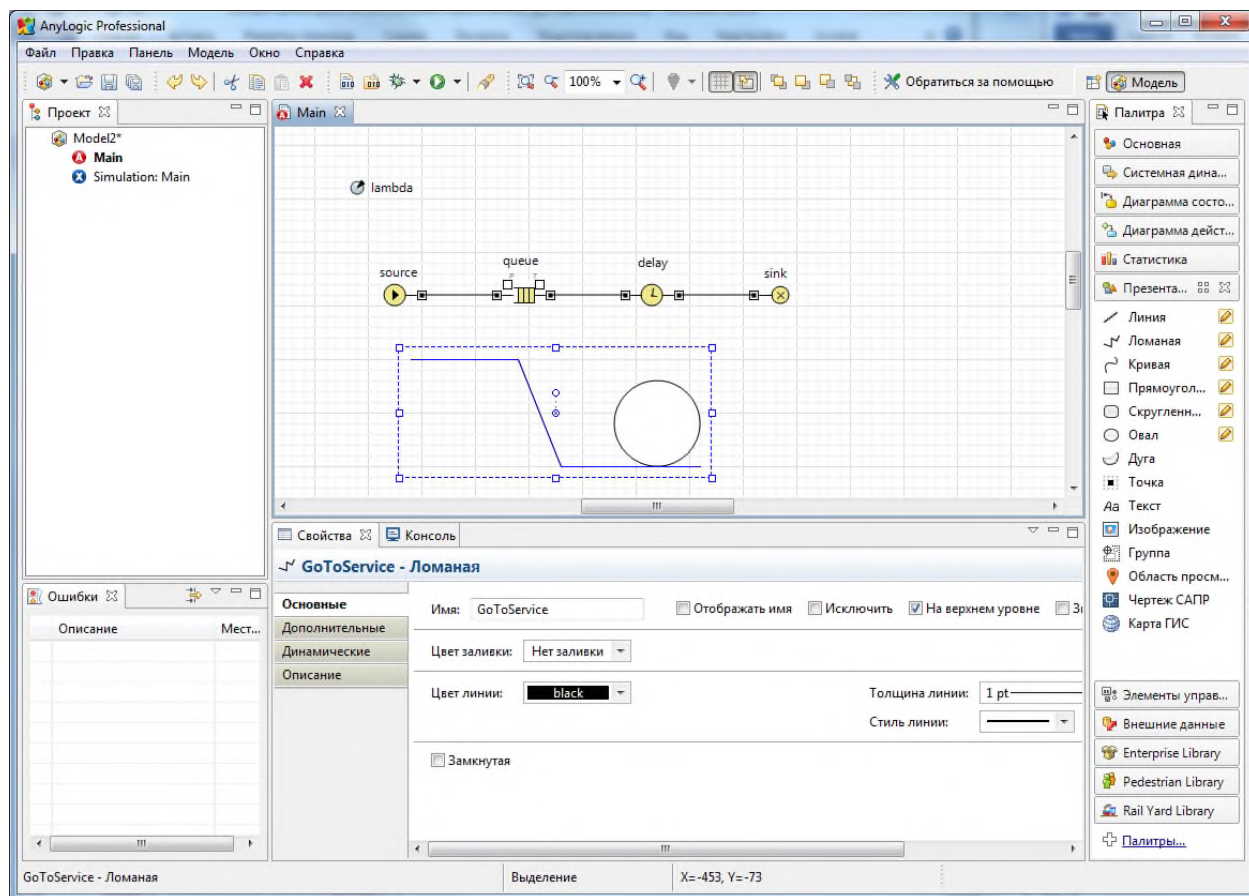


Рис.3.17. Задание свойств ломаной – очереди.

После создания элементов презентации нужно выполнить ряд настроек модели для связи графических элементов с объектами схемы.

Откройте окно свойств элемента очередь (queue) и на вкладке «Основные» задайте настройки так, как это показано на рисунке 3.18.

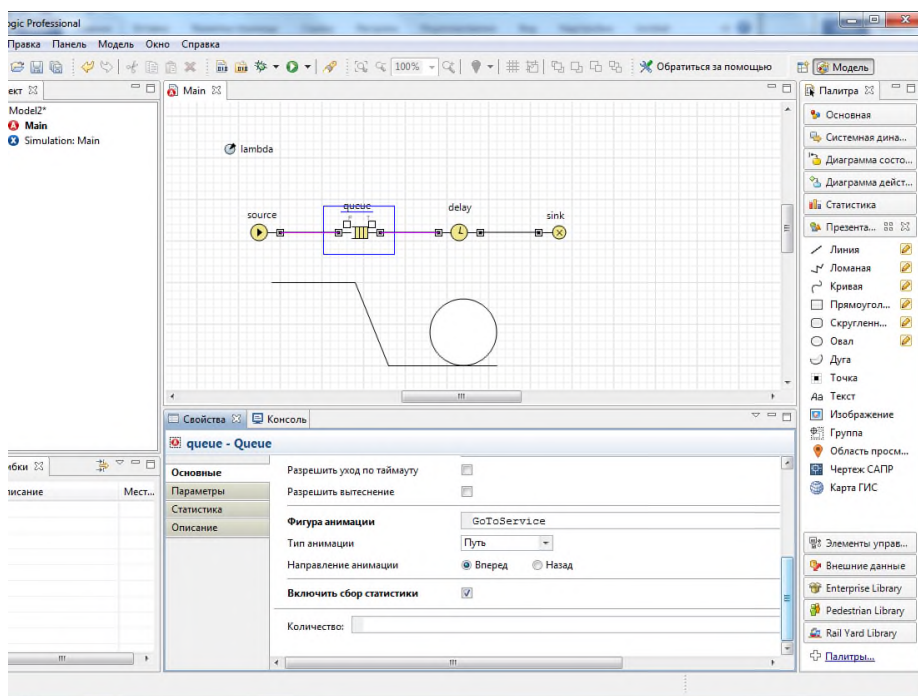


Рис.3.18. Настройка очереди

Откройте прибор обслуживания – элемент delay, и настройте на вкладке «Основные», свойства анимации:

- Фигура анимации: ServicePoint
- Тип анимации: Одиночная

Установите режим скорости исполнения равным 4 и протестируйте модель. На рисунке 3.19. Показан вид работающей модели.

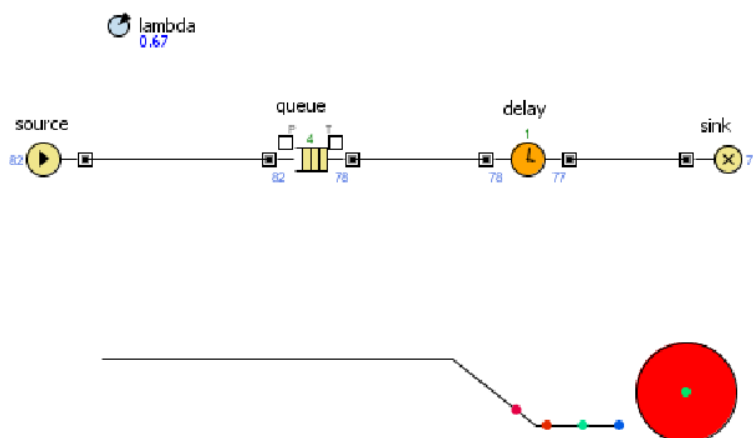


Рис.3.19. Модель с анимацией

Размещение датчиков.

Чтобы представить процесс загрузки прибора обслуживания и очереди разместим два датчика – столбчатые диаграммы. Первая диаграмма отображает среднее значение клиентов в очереди, а вторая - среднее значение числа обслуженных клиентов в банкомате (приборе обслуживания).

Для размещения диаграмм нужно использовать палитру «Статистика» и элемент «Столбиковая диаграмма». Разместите две диаграммы (см. рисунок 3.20).

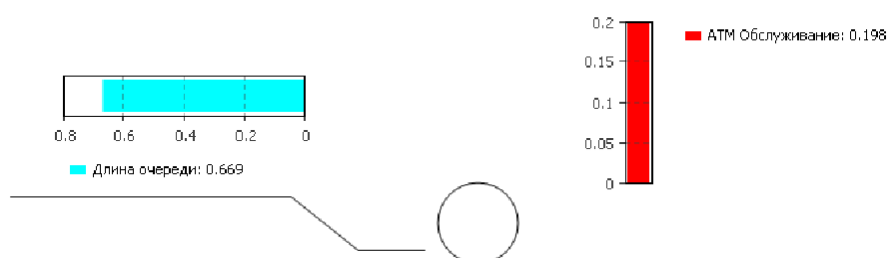


Рис.3.20. Диаграммы загрузки

Первую поместите над очередью. Добавьте элемент данных. Задайте подпись «Длина очереди». Выберите цвет, а затем в качестве значения задайте выражение:

```
queue.statsSize.mean()
```

Здесь метод `mean()` – возвращает среднюю длину очереди.

При вводе выражения можно использовать помощник AnyLogic. Для этого следует нажать комбинацию клавиш CTRL + SPACE.

После ввода выражения нужно сменить ориентацию диаграммы на горизонтальное. Для этого необходимо открыть вкладку «Внешний вид» и изменить направление столбцов (см. рисунок 3.21).



Рис.3.21. Направление столбцов диаграммы

Вторую диаграмму расположите рядом с изображением банкомата. Назовите диаграмму «АТМ Обслуживание», а в качестве значения задайте выражение:

`delay.statsUtilization.mean()`

задающее среднее время обслуживания заявки в процессоре. Направление столбцов вертикальное.

Вид работающей модели показан на рисунке 3.22.

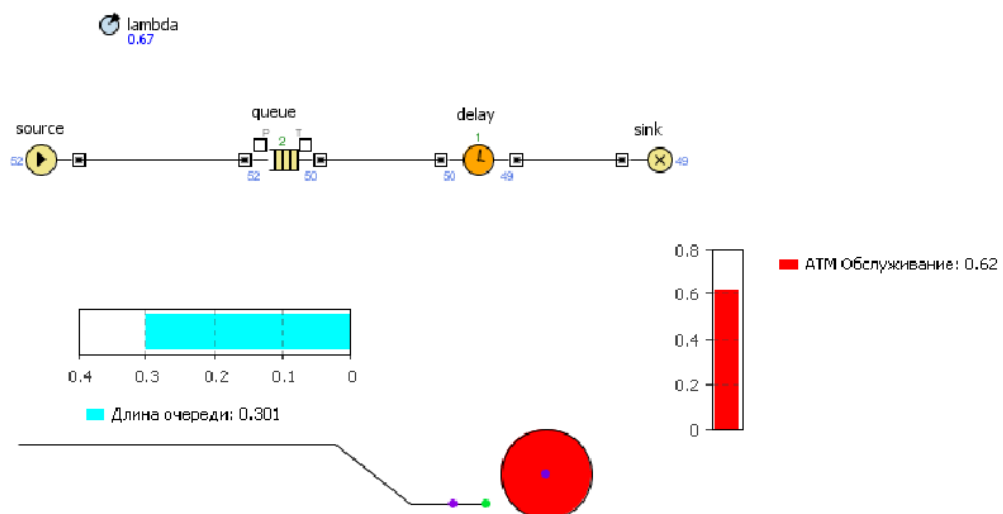


Рис.3.22. Модель с диаграммами

Моделирование многоканальной смо с очередью.

Усложним модель, добавив в нее банковских кассиров. Можно моделировать число кассиров, как и банкомат, с помощью объектов *delay*. Но куда более удобным представляется моделирование числа кассиров с помощью ресурсов. **Ресурс** – это специальный объект Enterprise Library, который может потребоваться заявке для выполнения какой-то задачи. В нашем примере посетителям банковского отделения (заявкам) необходимо получить помощь у банковских служащих (ресурсов).

Добавьте на диаграмму следующие объекты:

1. *selectOutput* – является блоком принятия решения. В зависимости от заданного вами условия, заявка, поступившая в этот объект, будет поступать на один из двух выходов объекта. Оставьте свойство *selectCondition* – *uniform()* < 0.5, тогда к кассирам и банкомату будет приходить примерно равное количество клиентов;

2. *Service* – моделирует занятие заявкой ресурса на определенное время. С помощью этого объекта мы промоделируем обслуживание клиента кассиром. Задайте следующие свойства объекта: назовите объект *tellerLines* (свойство **Имя**); укажите, что в очереди к кассирам может находиться до 20 человек (свойство *queueCapacity*); задайте время обслуживания (свойство *delayTime*). Будем полагать, что время обслуживания имеет треугольное распределение с минимальным средним значением 2.5, средним – 6 и максимальным – 11 минут;

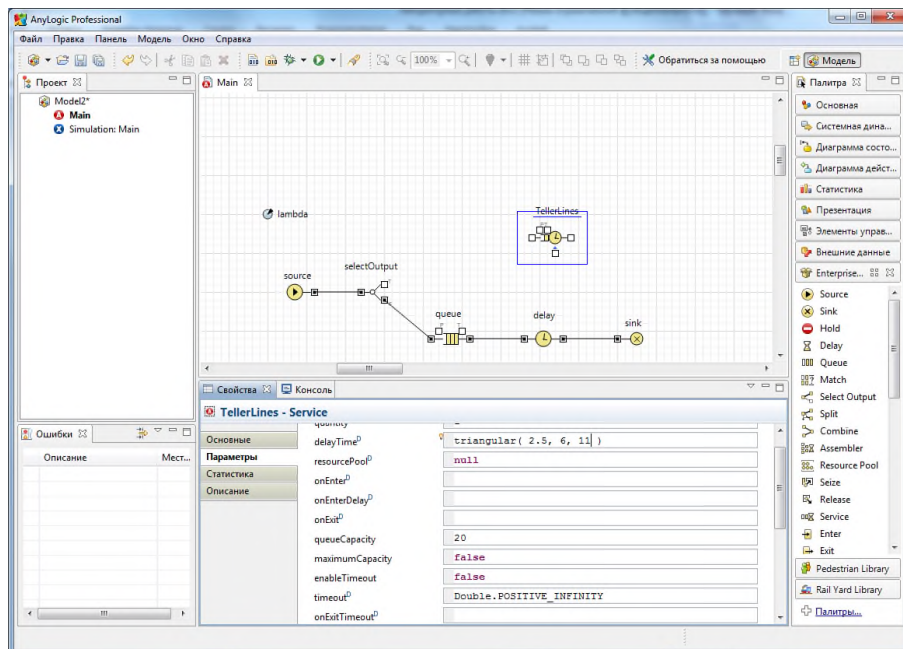


Рис. 3.23. Задание свойств линии касс.

3. *ResourcePool* – задает ресурсы определенного типа. Он должен быть подсоединен к объектам, моделирующим занятие и освобождение ресурсов (в нашем случае это объект *Service*). Задайте следующие свойства объекта: назовите объект *tellers*; задайте число кассиров (свойство *capacity*) – 4.

Измените имя объекта *delay* на *ATM* (банкомат). Соедините объекты соответствующим образом (рис. 3.24).

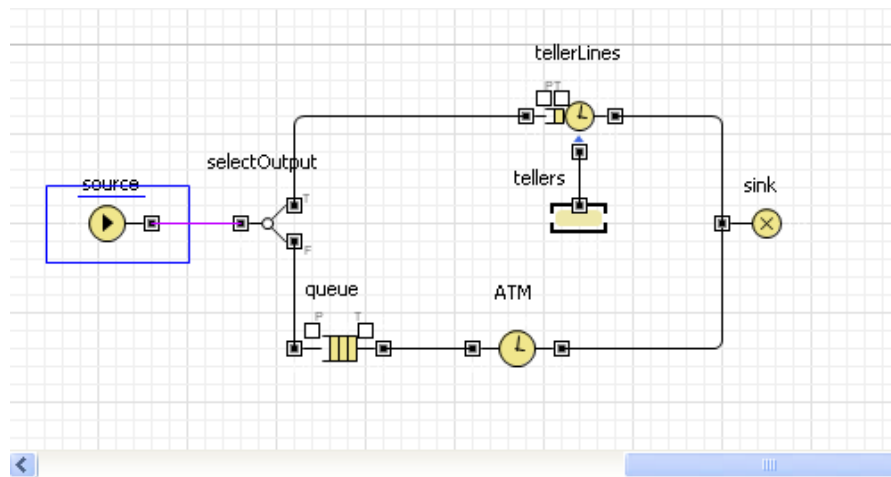


Рис. 3.24. Вид двухканальной СМО.

Запустите модель и изучите ее поведение.

Сбор статистики о времени обслуживания клиента.

Необходимо определить, сколько времени клиент проводит в банковском отделении и сколько времени он теряет, ожидая своей очереди. Соберем эту статистику с помощью специальных объектов сбора данных и отобразим собранную статистику распределения времени обслуживания клиентов с помощью гистограмм.

Создадим класс сообщения *Customer*. Сообщения этого класса будут представлять клиентов банковского отделения. Выберите базовый класс *Entity*(сообщения), добавьте параметры для хранения информации о проведенном времени:

1. в панели **Проект**, щелкните правой кнопкой мыши по элементу модели и выберите **Создать | Java класс** из контекстного меню (рис. 3.25);

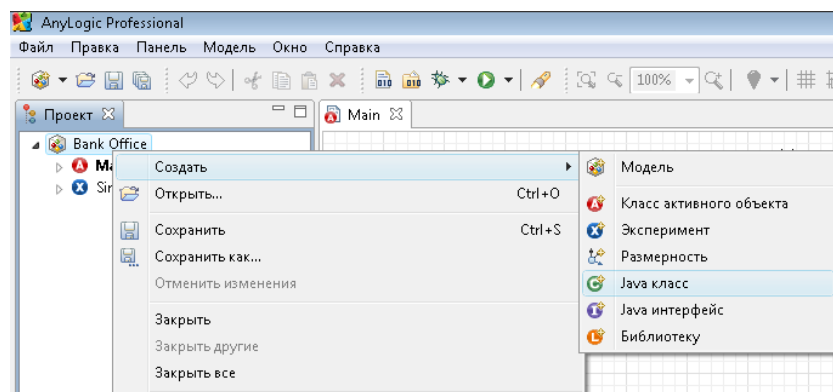


Рис. 3.25 Добавление Java- класса.

2. появится диалоговое окно **Новый Java класс**. В поле **Имя** введите имя нового класса *Customer*;

3. сделайте так, чтобы этот класс наследовался от базового класса заявки *Entity* (рис. 3.26): выберите из выпадающего списка **Базовый класс** полное имя данного класса: *com.xj.anylogic.libraries.enterprise.Entity*;

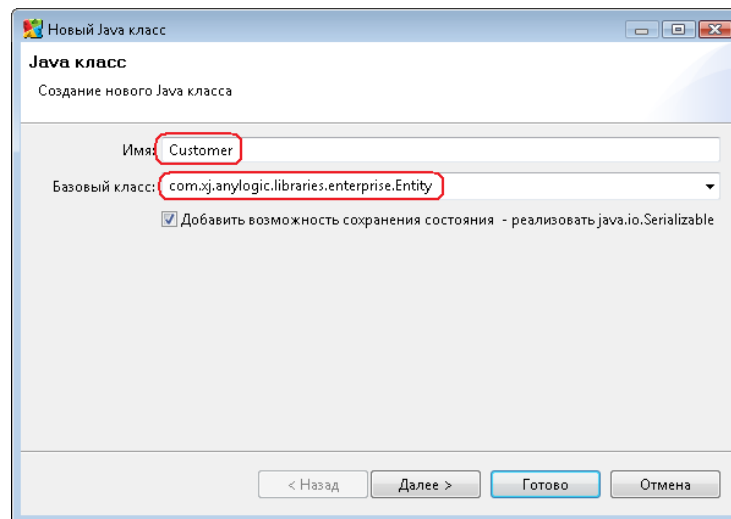


Рис. 3.26 Задание свойств Java- класса.

4. щелкните мышью по кнопке **Далее**. На второй странице Мастера вы можете задать параметры создаваемого Java-класса. Создайте параметры:

- *enteredSystem* типа *double* для сохранения момента времени, когда клиент пришел в банковское отделение;
- *startWaiting* типа *double* для сохранения момента времени, когда клиент встал в очередь к банкомату;

щелкните мышью по кнопке **Готово**. Вы увидите редактор кода созданного класса. Можете закрыть его, щелкнув мышью по крестику в закладке с его названием.

Теперь вычислим время, которое тратится персоналом банка на обслуживание клиентов, и время, которое клиенты тратят на ожидание своей очереди.

Для этого добавьте элементы сбора статистики по времени ожидания клиентов и времени пребывания клиентов в системе. Эти элементы будут запоминать соответствующие значения времени для каждого клиента и предоставят пользователю стандартную статистическую информацию: среднее, минимальное, максимальное из измеренных значений, среднеквадратичное отклонение, доверительный интервал для среднего и т.п.:

1. чтобы добавить объект сбора данных гистограммы на диаграмму, перетащите элемент **Данные гистограммы** с палитры **Статистика** на диаграмму активного класса;
2. задайте свойства элемента (рис. 3.27).
 - Измените **Имя** на *waitTimeDistr*.
 - Измените **Заголовок** на *Waiting time distribution*.
 - Сделайте **Кол-во интервалов** равным 50.
 - Задайте **Начальный размер интервала**: 0.01;

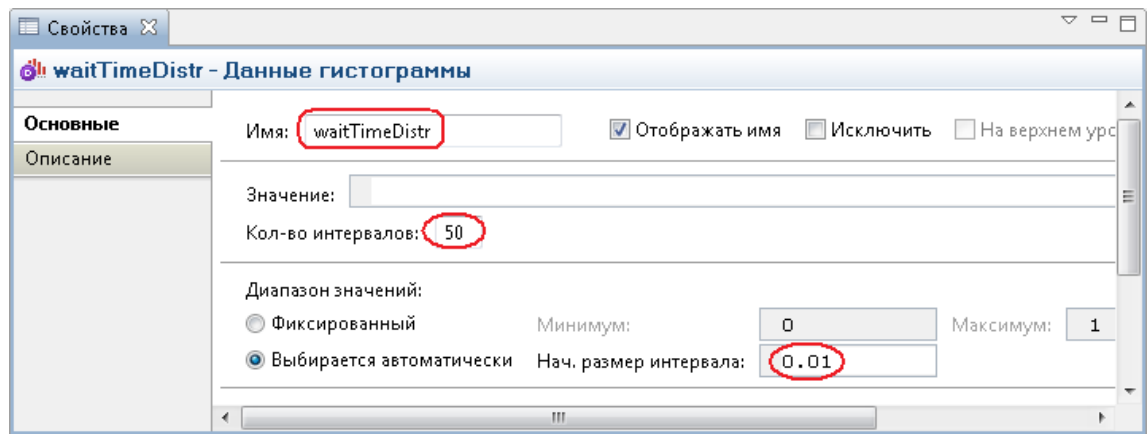


Рис. 3.27 Задание свойств гистограммы.

3. создайте еще один элемент сбора данных гистограммы (**Ctrl** + перетащите только что созданный объект данных гистограммы, чтобы создать его копию). Измените **Имя** этого элемента на *timeInSystemDistr*, а **Заголовок** на *Time in system distribution*.

Измените свойства блоков вашей диаграммы процесса. Задайте следующие свойства объектов диаграммы:

1. блок *source*, свойство **Новая заявка** – введите `new Customer()`. Введите `Customer` в поле **Класс заявки**. Это позволит напрямую обращаться к полям класса заявки *Customer* в коде динамических параметров этого объекта. Введите `entity.enteredSystem = time();` в поле **Действие при входе**. Этот код будет сохранять время создания заявки-клиента в переменной *enteredSystem* нашего класса заявки *Customer*. Функция `time()` возвращает текущее значение модельного времени;

2. блок *tellerLines* (блок *Service*) – введите `Customer` в поле **Класс заявки**. Добавьте код в поля:

Действие при входе: `entity.startWaiting = time();`

Действие при выходе: `waitTimeDistr.add(time() - entity.startWaiting);`

3. блок *queue* – введите `Customer` в поле **Класс заявки**. Добавьте код в поля: **Действие при входе:** `entity.startWaiting = time();`

Действие при выходе: `waitTimeDistr.add(time()-entity.startWaiting)`

Данный код добавляет время, в течение которого клиент ожидал обслуживания в объект сбора данных *waitTimeDistr*;

4. блок *ATM* (блок *delay*) – введите `Customer` в поле **Класс заявки**;

5. блок *sink* – введите `Customer` в поле **Класс заявки**. Напишите следующий код, чтобы сохранить в наборах данных данные о клиенте, покидающем банковское отделение (**Действие при входе**):

`timeInSystemDistr.add(time()-entity.enteredSystem);`

Данный код добавляет полное время пребывания клиента в банковском отделении в объект сбора данных гистограммы *timeInSystemDistr*.

Добавьте две гистограммы для отображения распределений времен ожидания клиента и пребывания клиента в системе.

Чтобы добавить гистограмму на диаграмму класса активного объекта, перетащите элемент **Гистограмма** из палитры **Статистика** в то место, куда вы хотите ее поместить. Укажите, какой элемент сбора данных хранит данные, которые хотите отображать на гистограмме: щелкните мышью по кнопке **Добавить данные** и введите в поле **Данные** имя соответствующего элемента – *waitTimeDistr*.

Аналогичным образом добавьте еще одну гистограмму и расположите ее под ранее добавленной. В поле **Данные** введите *timeInSystemDistr*. Измените заголовки отображаемых данных.

Запустите модель. Включите режим виртуального времени и посмотрите, какой вид примет распределение времени ожидания и времени пребывания клиента в системе.

Индивидуальные варианты заданий.

Внесите изменения в модель банковского отделения согласно варианту (параметры законов распределения задайте произвольно).

Вариант	Распределение вероятности прихода клиентов в банк	Вероятность обращения к кассиру/к банкомату	Время обслуживания клиента кассиром	Количество кассиров
1	Экспоненциальное	1/1	4±2	1
2	Экспоненциальное	1/2	6±2	2
3	Экспоненциальное	2/1	8±2	3
4	Экспоненциальное	1/3	7±2	4
5	Экспоненциальное	3/1	9±2	5
6	Нормальное	1/1	8±2	1
7	Нормальное	1/2	7±2	2
8	Нормальное	2/1	9±2	3
9	Нормальное	1/3	4±2	4

10	Нормальное	3/1	6±2	5
11	Треугольное	1/1	2±2	1
12	Треугольное	1/2	7±2	2
13	Треугольное	2/1	9±2	3
14	Треугольное	1/3	4±2	4
15	Треугольное	3/1	6±2	5
16	Равномерное	1/1	4±2	1
17	Равномерное	1/2	6±2	2
18	Равномерное	2/1	7±2	3
19	Равномерное	1/3	8±2	4
20	Равномерное	3/1	9±2	5

Проанализируйте поведение модели. Постройте графики и диаграммы.

Контрольные вопросы.

1. Виды интеллектуальных систем и области их применения.
2. Основные модели интеллектуальных систем
3. Архитектура интеллектуальных информационных систем.
4. Типовая схема функционирования интеллектуальной системы.
5. Примеры интеллектуальных систем
6. Понятие модели представления знаний (МПЗ).
7. Основные МПЗ, их особенности и области применения.
8. Понятие вывода на знаниях.
9. Методы представления знаний в базах данных информационных систем.
10. Формальная грамматика как способ представления знаний в продукционной МПЗ.
11. Понятие и форма записи правил продукции.
12. Синтаксические деревья, задачи разбора и вывода.
13. Составные части экспертной системы: база знаний, механизм вывода, механизмы приобретения и объяснения знаний, интеллектуальный интерфейс.
14. Ограничения, присущие экспертным системам.

15. Особенности экспертных систем экономического анализа.
16. Статические и динамические экспертные системы.
17. Организация процесса приобретения и формализации знаний.
18. Эксперт и инженер по знаниям: формы и порядок взаимодействия.
19. Проблемы неопределенности в экспертных системах.
20. Классификация методов обработки неопределенности знаний.
21. Теория субъективных вероятностей.
22. Байесовское оценивание.
23. Теорема Байеса как основа управления неопределенностью.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

№ п.п.	Содержание изменения	Дата, номер протокола заседания кафедры, подпись зав.кафедрой
1	2	3
1		
2		
3		
4		