

## ПРИЛОЖЕНИЕ А

(обязательное)

### Процесс создания дискретно-событийной модели в пакете моделирования Anylogic

Наиболее полная информация по библиотеке моделирования процессов находится в справочной системе [help.anylogic.ru](http://help.anylogic.ru), раздел *Справочные руководства по библиотекам > Библиотека моделирования процессов*, а также *Справка AnyLogic > Основные принципы > Методы моделирования > Дискретно-событийное моделирование*.

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. Поскольку модель организована иерархически, то она отображается в виде дерева: сама модель образует верхний уровень дерева; эксперименты, классы активных объектов и Java-классы образуют следующий уровень; элементы, входящие в состав активных объектов, вложены в соответствующую подветвь дерева класса активного объекта и т.д.

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.

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 экспери-

мент модели. Один эксперимент, названный Simulation, создается автоматически. Выберите его щелчком мыши по элементу дерева и измените настройки модели в окне Свойства (рисунок А.1). Окно Свойства имеет вкладки: основные, дополнительные, модельное время, презентация, окно, параметры, описание.

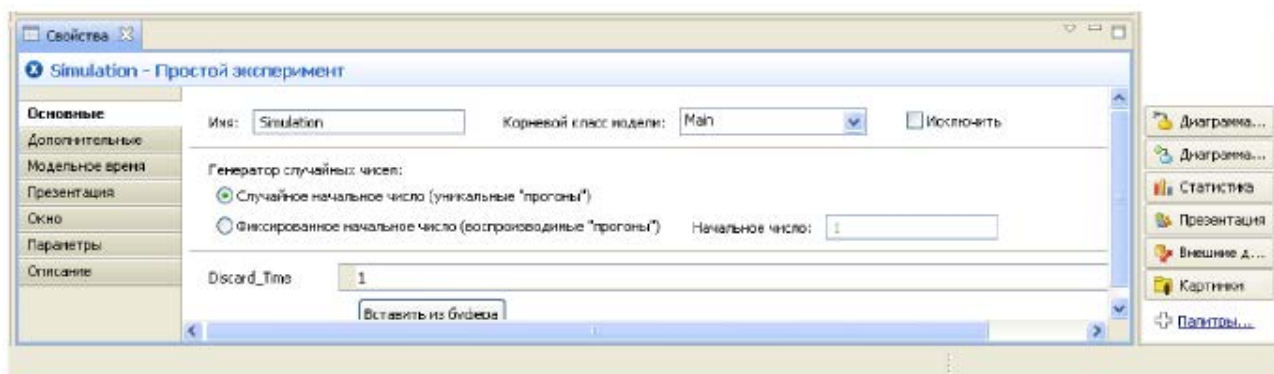


Рисунок А.1 – Настройки модели

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:

- 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;

- 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.

Построим с помощью элементов библиотеки Enterprise Library модель простой системы массового обслуживания – модель банковского отделения.

В банковском отделении находятся банкомат и стойки банковских кассиров, которые предназначены для быстрого и эффективного обслуживания посетителей банка. Операции с наличностью клиенты банка производят с помощью

банкомата, а более сложные операции, такие как оплата счетов, - с помощью кассиров.

Необходимо произвести оценку затрат операций и определить, сколько денег тратится на обслуживание одного клиента и какую часть этой суммы составляют накладные расходы на оплату работы персонала банка, а какую на обслуживание посетителей.

### 1. Создание нового проекта.

Создайте новую модель. Переименуйте класс Main в Model. В свойствах эксперимента Simulation задайте выполнение модели в режиме реального времени с выполнением одной единицы модельного времени в одну секунду. В этой модели под единицей модельного времени мы будем понимать одну минуту работы банковского отделения.

### 2. Создание блок-схемы.

Создайте блок-схему модели, которая пока будет состоять только из банкомата. Для этого перетащите в окно структуры элементы библиотеки Enterprise Library и соедините их так, как показано на рисунке A.2.

Объект source генерирует заявки (entities) определенного типа через заданный временной интервал. Заявки представляют собой объекты, которые производятся, обрабатываются, обслуживаются или еще каким-нибудь образом подвергаются воздействию моделируемого процесса: это могут быть клиенты в системе обслуживания, детали в модели производства, документы в модели документооборота и др. В нашем примере заявками будут посетители банка, а объект source будет моделировать их приход в банковское отделение.

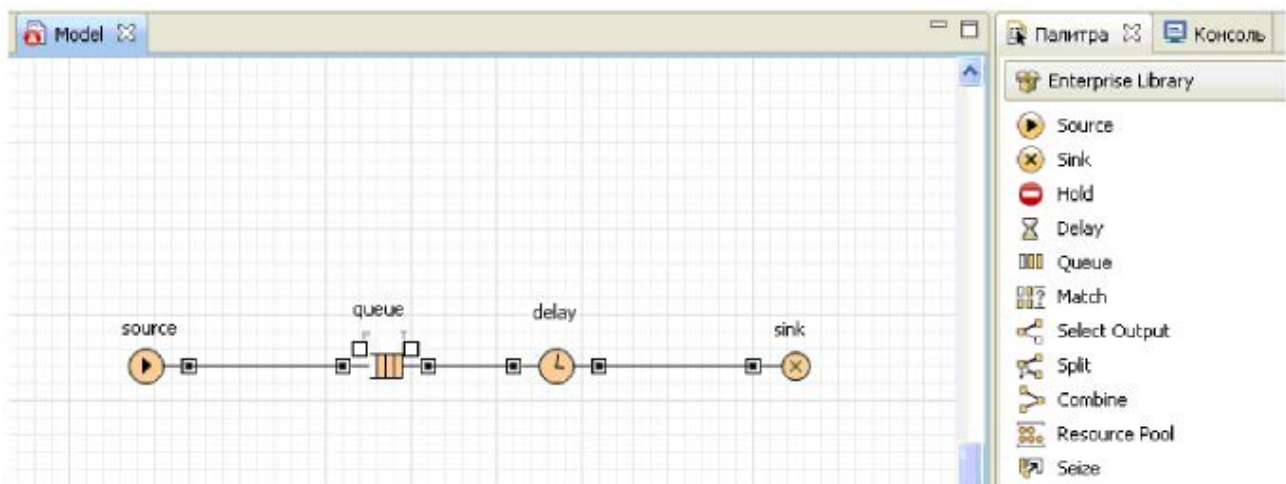


Рисунок A.2 – Структурная диаграмма модели банковского отделения

Объект queue моделирует очередь клиентов, ожидающих обслуживания. Объект delay моделирует задержку. В нашем примере он будет имитировать банкомат, тратящий определенное время на обслуживание клиента.

Объект sink обозначает конец блок-схемы.

### 3. Запуск модели.

Для каждой модели, созданной в Enterprise Library, автоматически создается блок-схема с наглядной визуализацией процесса, с помощью которой вы можете изучить текущее состояние модели, например, длину очереди, количество

обслуженных человек и т.д.

Для запуска модели (рисунок А.3) 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, нарисованную вами для главного класса активного объекта этого эксперимента (Main).

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.

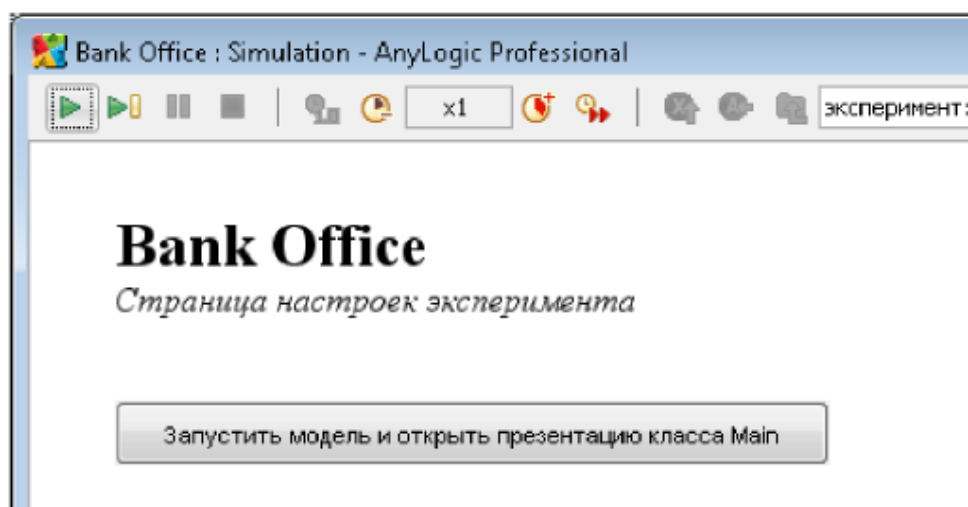


Рисунок А.3 – Запуск модели

На рис. 4 видно, что 4 человека стоят в очереди, а 23 человека покинули очередь (блок *queue*), из них 22 обслужили (блок *sink*), а один еще обслуживается у банкомата (блок *delay*).

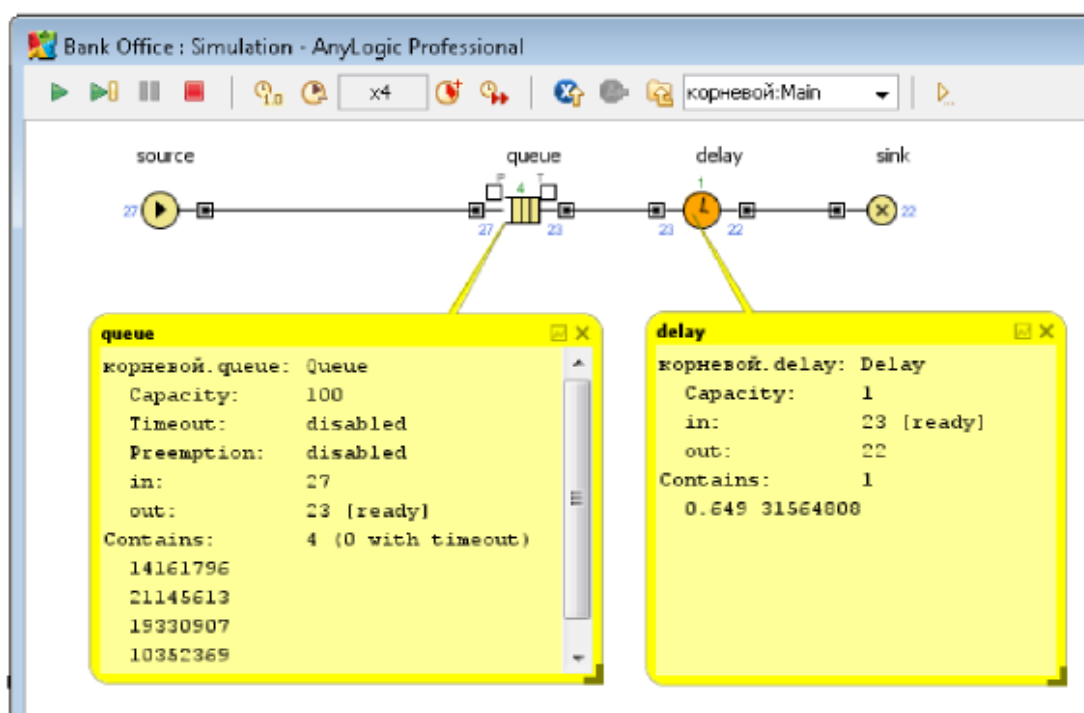


Рисунок А.4 – Процесс выполнения модели

С помощью кнопок панели инструментов «Замедлить» и «Ускорить» можно изменить скорость выполнения модели. Во время выполнения модели можно следить за состоянием любого блока диаграммы процесса с помощью окна инспекта этого объекта. Чтобы открыть окно инспекта, щелкните мышью по значку блока. В окне инспекта будет отображена базовая информация по выделенному блоку, например, для блока Queue будет отображена вместимость очереди, количество заявок, прошедшее через каждый порт объекта, и т.д.

#### 4. Изменение данных модели.

Задайте данные модели, изменяя свойства созданных объектов (рисунок А.5).

В свойстве `interarrivalTime` объекта `source` укажите, как часто в отделение приходят клиенты - `exponential(0.67)`.

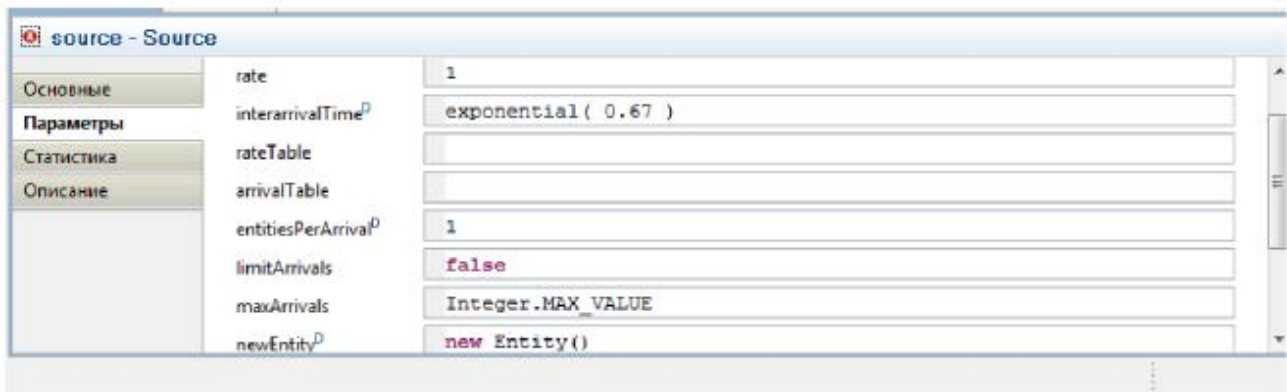


Рисунок А.5 – Свойства объекта source

Интервал между приходом клиентов распределен экспоненциально со средним значением, равным 1.5 единицы модельного времени. Заметьте, что аргумент функции `exponential()` равен 0.67, потому что в качестве аргумента задается интенсивность прихода клиентов.

Функция `exponential()` является стандартной функцией генератора случайных чисел AnyLogic. AnyLogic предоставляет функции и других случайных распределений, таких как нормальное, равномерное, треугольное и т.д. За детальным описанием функций и их параметров обращайтесь к руководству пользователя или справочнику классов (см. методы класса `Func`). Для вызова руководства пользователя, справочника классов AnyLogic выберите соответствующие пункты меню Справка.

В свойстве `capacity` объекта `queue` (рисунок А.6) задайте максимальную длину очереди - 15.

В свойстве *delay Time* объекта *delay* (рисунок А.7) задайте время задержки (время обслуживания) – *triangular(0.8, 1, 1.3)*.

Обслуживание одного клиента занимает примерно 1 минуту. Здесь время обслуживания распределено по треугольному закону со средним значением, равным 1 минуте, минимальным - 0.8 и максимальным - 1.3 минуты.

Запустите модель и проанализируйте ее работу.

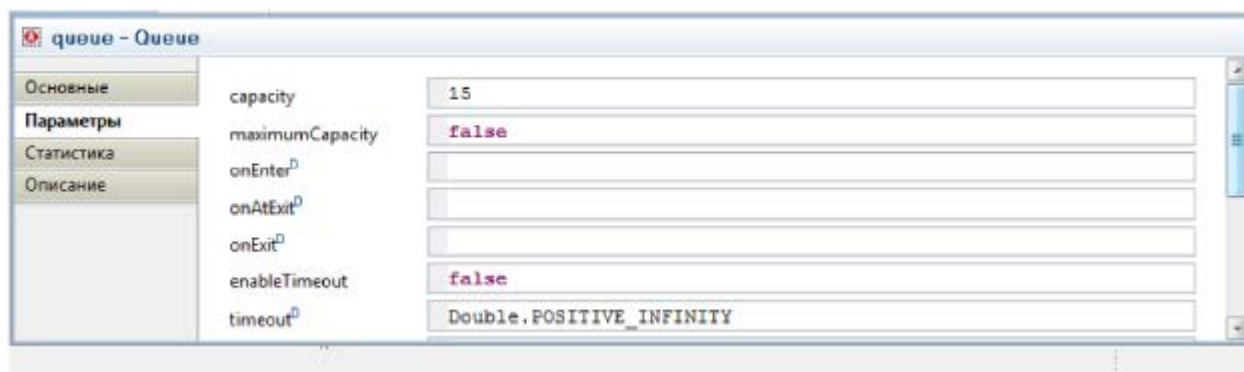


Рисунок А.6 – Свойства объекта queue

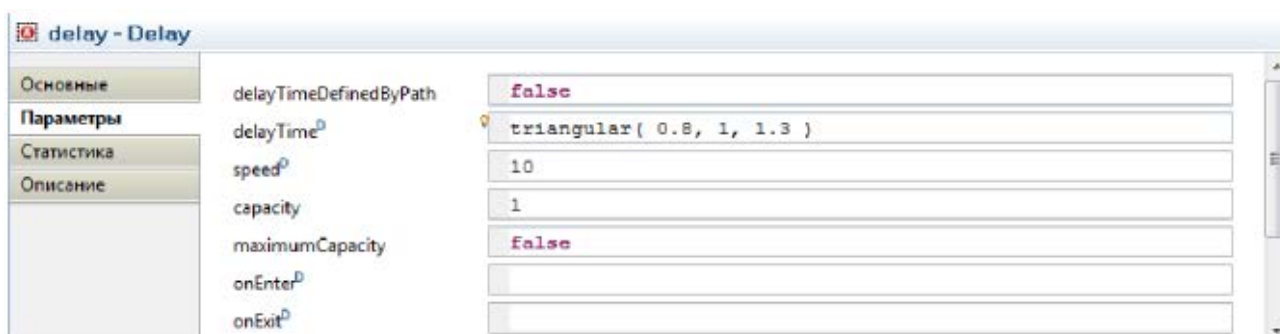


Рисунок 7 – Свойства объекта delay

## 5. Сбор статистики.

AnyLogic позволяет производить сбор сложной статистики. Для этого нужно лишь включить у объекта режим сбора статистики, поскольку по умолчанию он отключен для повышения скорости выполнения модели.

В системе собирается статистика по длине очереди для блока queue (length) и статистика по коэффициенту использования для блока delay (utilization). Чтобы включить сбор статистики для объекта, установите переключатель «Включить сбор статистики» на вкладке Основные свойств объекта.

Запустите модель и просмотрите в окне инспекта статистику для блоков queue и delay. Можно также просмотреть собранную статистику с помощью диаграмм и графиков или путем вывода числовых значений на анимацию.

## 6. Моделирование банковских кассиров.

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  $\text{uniform()} < 0.5$ , 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;

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 (рисунок А.8).

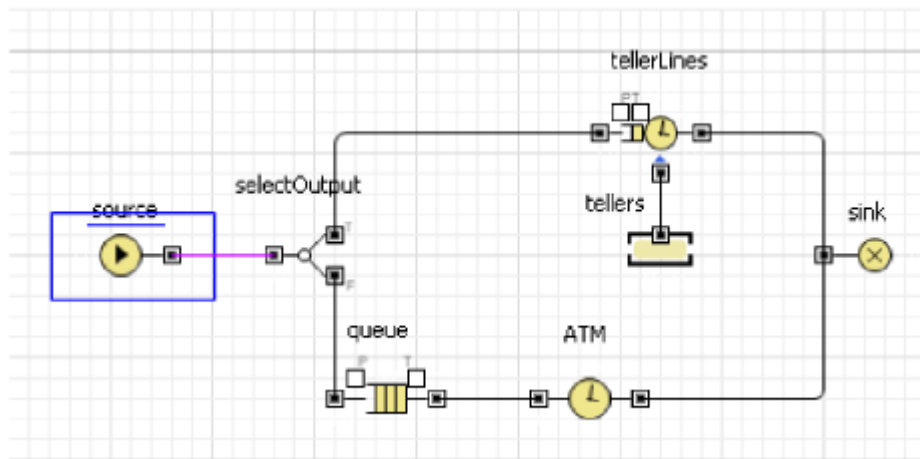


Рисунок А.8 – Структурная диаграмма банковского отделения

Запустите модель и изучите ее поведение.

#### 7. 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 (рисунок А.9);

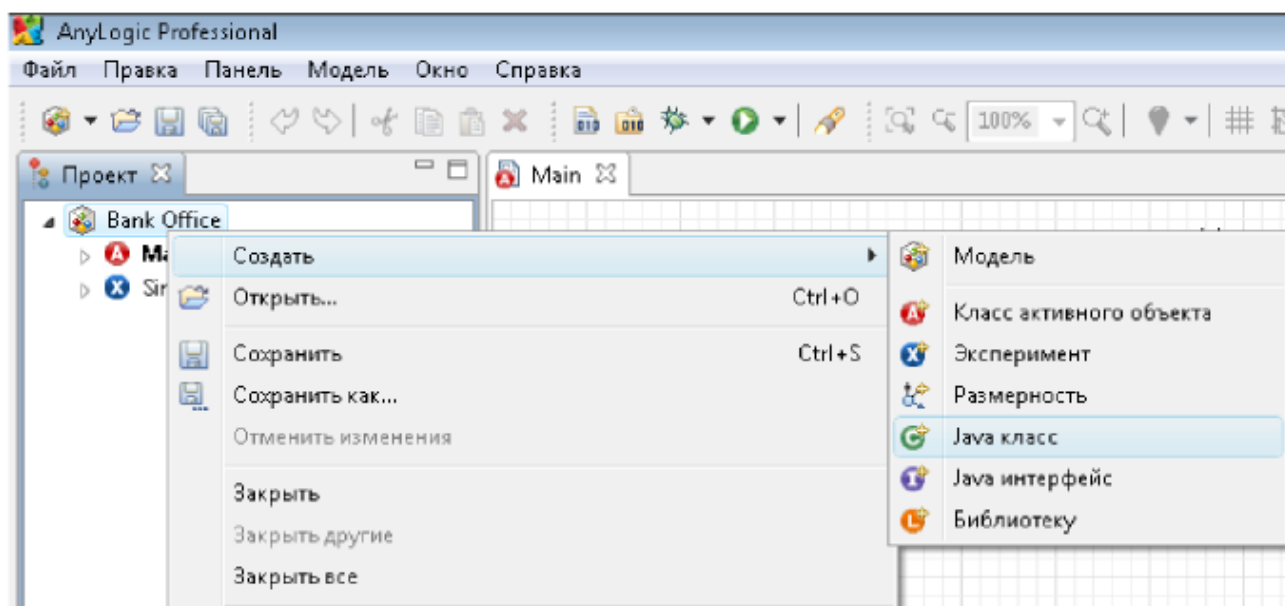


Рисунок А.9 – Создание собственного Java класса

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 (рисунок А.10): выберите из выпадающего списка Базовый класс полное имя данного класса: com.xj.anylogic.libraries.enterprise.Entity;

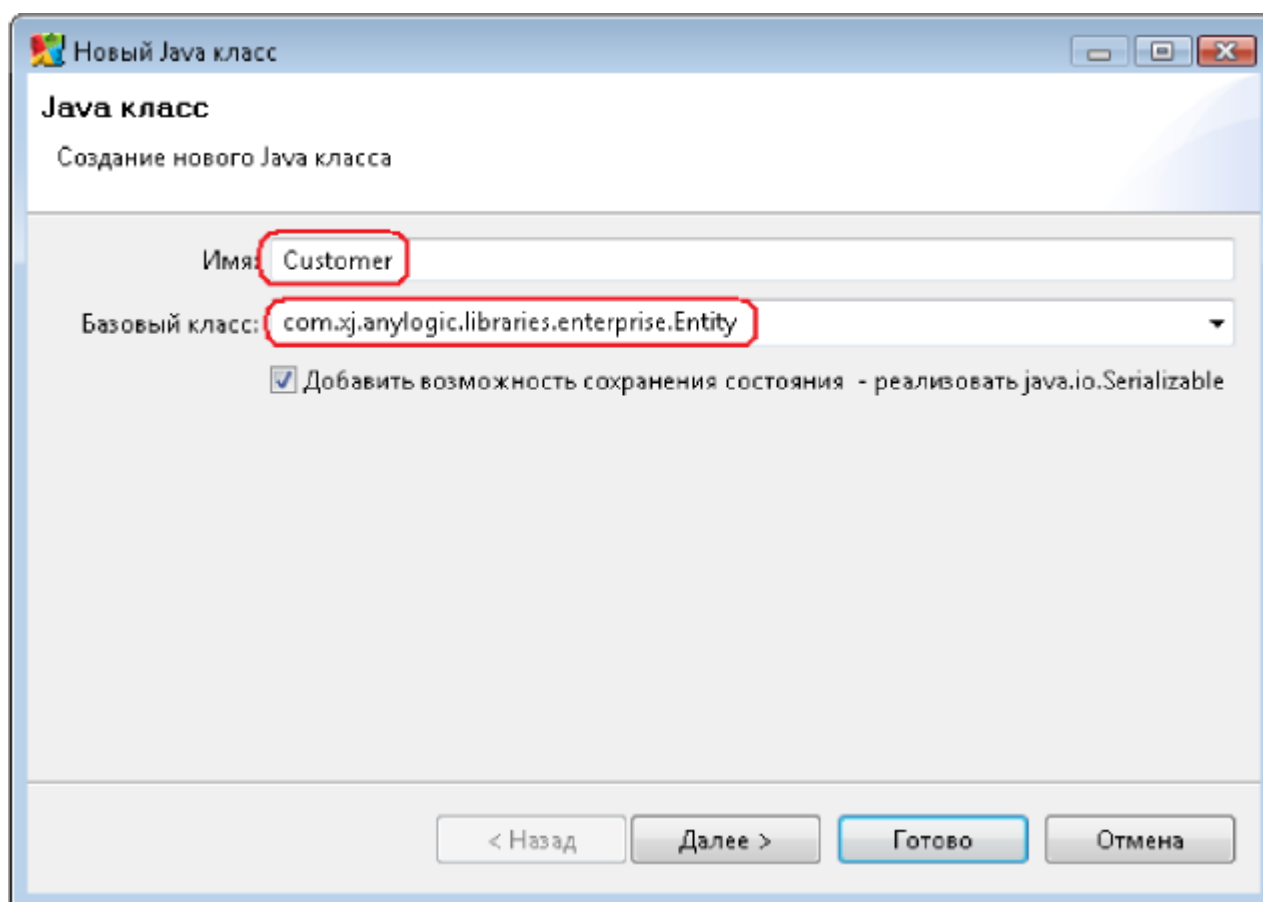


Рисунок А.10 – Создание собственного Java класса

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;

5) 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.

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 (рисунок А.11). Измените Имя на waitTimeDistr.

Измените Заголовок на Waiting time distribution.

Сделайте Кол-во интервалов равным 50. Задайте Начальный размер интервала: 0.01;

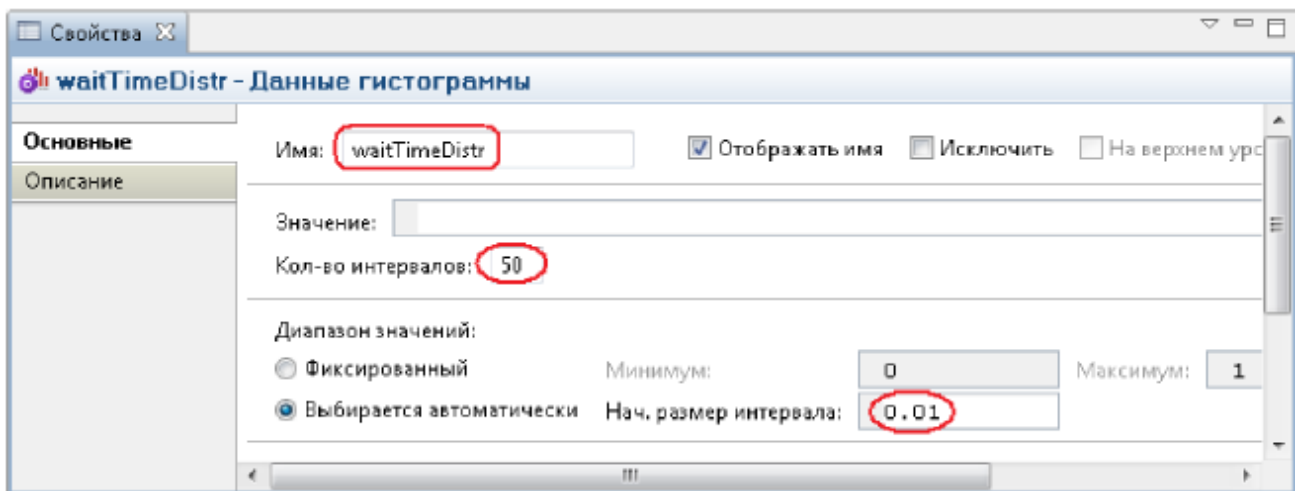


Рисунок А.11 – Свойства элемента сбора данных гистограммы

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 выходе. 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 графического редактора, 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.

## 8. Оценка затрат операций.

Enterprise Library предоставляет инструменты для проведения оценки затрат операций. Метод оценки затрат операций (*activity-based costing*, ABC-метод) оценивает процесс и эффективность операций, определяет стоимость обслуживания/производства и указывает возможности для усовершенствования продуктивности и эффективности процесса. С помощью этого метода производится количественная оценка стоимости и производительности операций, эффективности использования ресурсов и стоимости объектов.

Проведем учет затрат операций в нашем примере, чтобы понять, во сколько в среднем обходится обслуживание одного клиента и какие накладные расходы связаны с обслуживанием клиентов, ожидающих своей очереди.

Сначала необходимо написать вспомогательную функцию для пересчета почасовой зарплаты в поминутную.

Создайте математическую функцию (перетащите элемент Функция с палитры Основные на диаграмму активного класса), назовите ее toMinute. На странице свойств задайте тип, аргументы и выражение функции. Тип возвращаемого значения должен быть double. Аргумент функции perHour типа double. Напишите выражение функции (тело функции) - `return perHour / 60;`. Функция должна быть статической, поскольку эта функция не использует значения, специфичные для конкретного экземпляра класса Model. В противном случае нам пришлось бы вызывать ее со ссылкой на конкретный экземпляр класса Model.

Добавьте необходимые параметры в классе сообщений Customer:

- serviceCost будет хранить информацию о том, во сколько компании обходится обслуживание этого клиента. Тип - double. Значение по умолчанию - 0;
- waitCost - затраты на ожидание клиента в очереди. Тип - double. Значение по умолчанию - 0. Заметим, что банк несет издержки, связанные с обслуживанием: клиентов, ожидающих своей очереди;
- existenceCostPerHour задает, во сколько обходится компании пребывание клиента в банке. Тип - double. Значение по умолчанию - `Main.toMinute(1.5)`.

Теперь необходимо добавить код в свойства блоков:

1) блок tellerLines (блок Service) - добавьте код в Действие при выходе:

```
waitTimeDistr.add(time() - entity.startWaiting);
Costwait.add(entity.existenceCostPerHour*(time() - entity.startWaiting));
Costservice.add ((time()-entity.enteredSystem)*(entity.existenceCostPerHour +
Model.busyCostRate));
```

2) блок queue добавьте код в Действие при выходе:

```
waitTimeDistr.add(time() - entity.startWaiting); Costwait.add(
entity.existenceCostPerHour*(time()- entity.startWaiting));
Costservice.add((time()-entity.enteredSystem)* (entity.existenceCostPerHour +
Model.busyCostRate));
```

Добавьте в модель вспомогательные элементы, собирающие статистику затрат компании, для чего необходимо создать:

1) переменные, задающие заработную плату кассиров. Для этого перетащите элемент Простая переменная с палитры Основные на диаграмму активного класса:

- busyCostRate, тип double, значение по умолчанию toMinute (6.5), глобальный;
- idleCostRate, тип double, значение по умолчанию toMinute (4.0), глобальный.

Мы платим кассиру 6.5 дол. в час, если он был занят обслуживанием клиентов и 4.0 дол., если он был свободен;

2) простую переменную, задающую расходы, которые связаны с работой банкомата:

ATMCostPerUse, тип - double, значение по умолчанию - 0.3, глобальный. Одна операция банкомата обходится компании в 0.30 дол.;

3) переменную timeUpdateCosts. Создайте еще несколько переменных и назовите их так:

- tellersIdleTime;
- tellersBusyTime;
- tellersIdleCost;
- tellersBusyCost.

Эти переменные будут хранить информацию о том, сколько времени кассиры были заняты обслуживанием клиентов и сколько им требуется выплатить за работу;

5) два набора данных гистограммы (палитра Статистика) и назовите их так:

- Costwait;
- Costservice.

Данные наборы данных будут хранить статистику затрат компании на обслуживание клиентов;

6) функцию, которая будет обновлять статистику (палитра Основные). Назовите ее UpdateCosts.

```
double dt = time()-timeUpdateCosts; tellersIdleTime += tellers.idle()*dt;
tellersBusyTime+=(tellers.capacity-tellers.idle() )*dt;
tellersIdleCost += tellersIdleTime*idleCostRate;
tellersBusyCost += tellersBusyTime*busyCostRate;
timeUpdateCosts = time();
return 0 ;
```

Сосчитайте затраты на обслуживание клиента:

1) измените свойства объекта tellers (рисунок А.12). Добавьте следующий код:

```
onSeize - UpdateCosts ();
onRelease - UpdateCosts ();
```

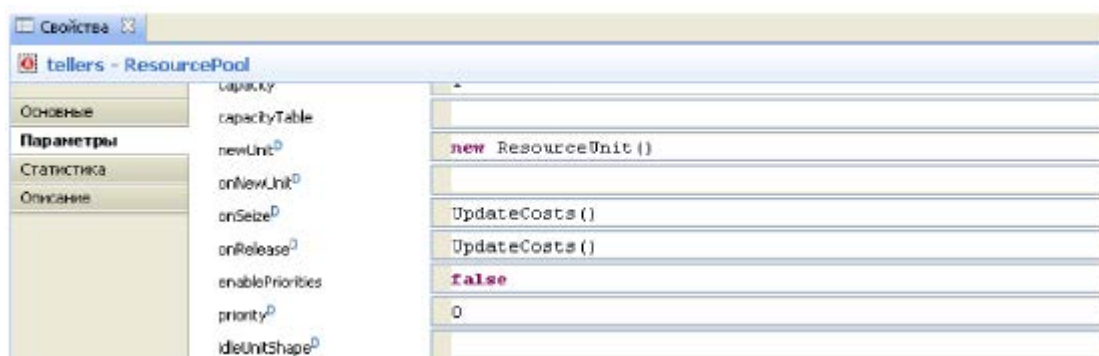


Рисунок А.12 – Свойства объекта tellers

2) измените свойства объекта АТМ . Добавьте строки, чтобы учесть затраты на обслуживание клиента:

```
Costservice.add((time()-entity.enteredSystem)* (entity.existenceCostPerHour
+ Model.ATMCostPerUse));
```

3) запустите модель. Посмотрите статистику затрат компании на обслуживание клиентов. Покажите, во сколько в среднем обходится обслуживание одного клиента и какие накладные расходы связаны с обслуживанием клиентов, ожидающих своей очереди.