



AnyLogic[®] 6

Enterprise Library Tutorial

© 1992–2007 XJ Technologies Company. All rights reserved
AnyLogic and XJ Technologies are registered trademarks of XJ Technologies Company.

Copyright © 1992-2007 XJ Technologies. All rights reserved.

XJ Technologies Company Ltd

AnyLogic@xjtek.com

<http://www.xjtek.com/anylogic>

Contents

CONTENTS	3
BANK MODEL	4
STEP 1. CREATING A NEW MODEL	5
STEP 2. CREATING A SIMPLE MODEL	7
Creating the model flowchart.....	7
Configuring the model	11
Running the model	12
STEP 3. CREATING A MODEL ANIMATION	14
STEP 4. COLLECTING UTILIZATION STATISTICS	19
STEP 5. ADDING CUSTOM ANIMATIONS FOR CLIENTS	23
STEP 6. ADDING TELLERS.....	26
STEP 7. COLLECTING CUSTOMER TIME STATISTICS	36

Bank Model

AnyLogic provides the Enterprise Library, a discrete-event simulation library containing objects you can use to rapidly simulate complex discrete-events systems like:

- Manufacturing processes with detailed shop floor layout
- Simple and complex service systems (e.g. banks, airports, etc.)
- Business processes with activity based costing
- Logistics and supply chain models

The Enterprise Library allows you to create flexible models, collect basic and advanced statistics, and effectively visualize the process you are modeling to validate and present your model.


In this tutorial you will learn how to create models with the Enterprise Library in the fields of manufacturing and business processes.

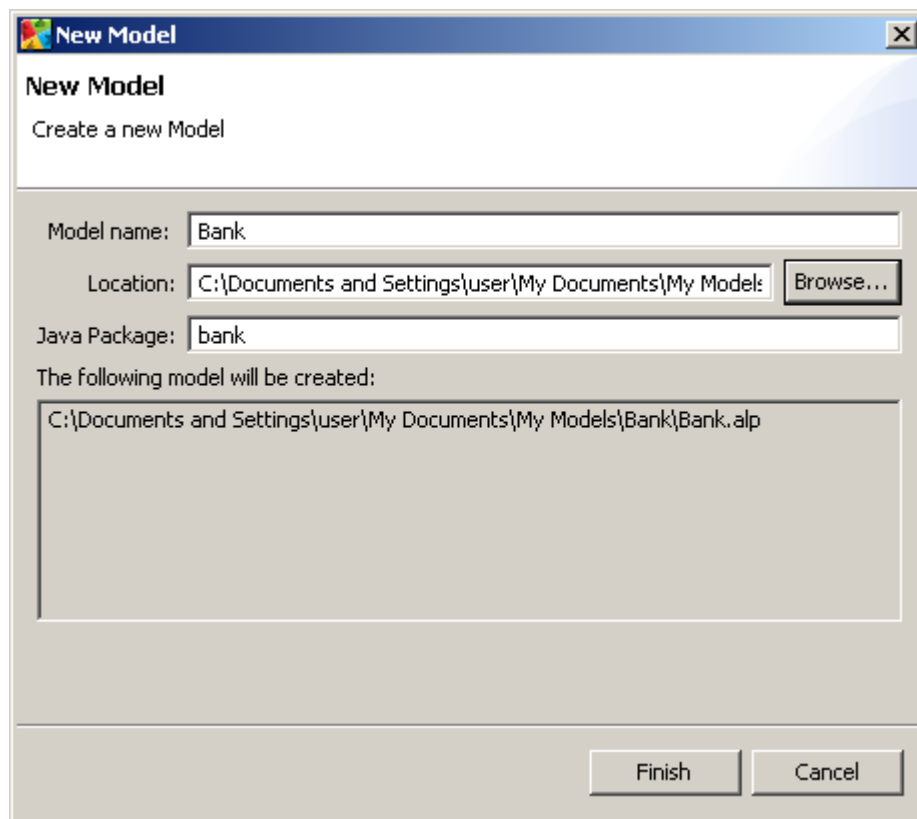
We will create a simple service system of a bank department, consisting of an automatic teller machine and teller lines. ATM provides people with a quick self-service for cash. More complex transactions, e.g. paying bills, are completed by tellers, allowing customers more time without inconveniencing those customers looking for quick cash.

Step 1. Creating a New Model

First, we will create a new model.

Create a new model

1. Click the **New Model**  toolbar button. The **New Model** dialog box is displayed.
2. Specify the name of the model. In the **Model Name** edit box, type `Bank`.

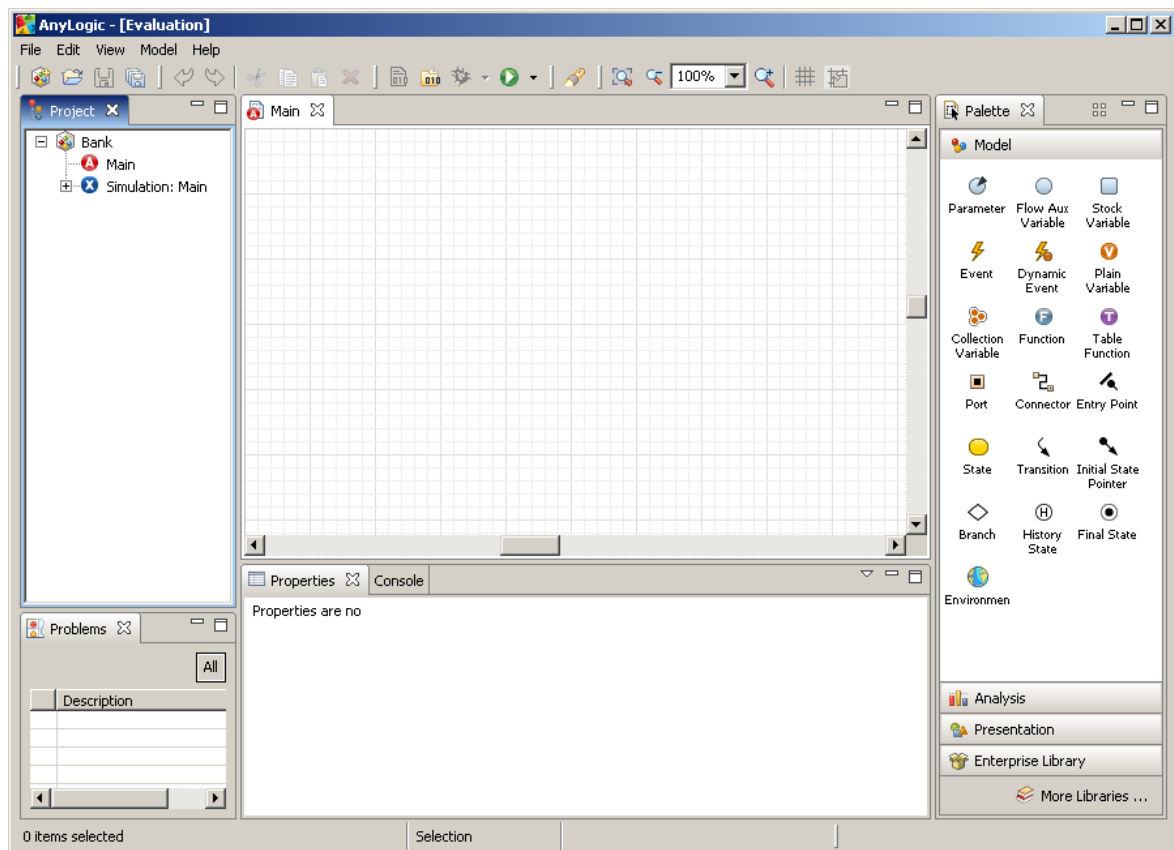


3. Specify the location where you want to store your model file. Browse for the existing folder using the **Browse** button, or type the name of the folder you want to create in the **Location** edit box.
4. Click **Finish**.

New model is created. It already has one active object class called *Main* and experiment called *Simulation*.

Active objects are the main building blocks of AnyLogic model. They can be used to model very diverse objects of the real world.


In the center of the workspace you will see the graphical editor. It shows the diagram of the *Main* class. By default it does not contain any elements.



To the left of the graphical editor you can see the **Project** view. The **Project** view provides access to AnyLogic models currently opened in the workspace. The workspace tree provides easy navigation throughout the models.

On the right side of the workspace you can see the **Palette** view, and at the bottom - the **Properties** view. The **Palette** view simplifies the process of drawing diagrams, providing the list of model elements grouped by categories in a number of stencils.

The **Properties** view is used to view and modify the properties of a currently selected model item(s). When you select something – e.g., in the **Project** view or in a graphical editor – the **Properties** view displays the properties of the selection.

When working with a model, do not forget to save it by clicking **Save** .

Step 2. Creating a Simple Model

Now we will create the simplest model consisting of the ATM only.

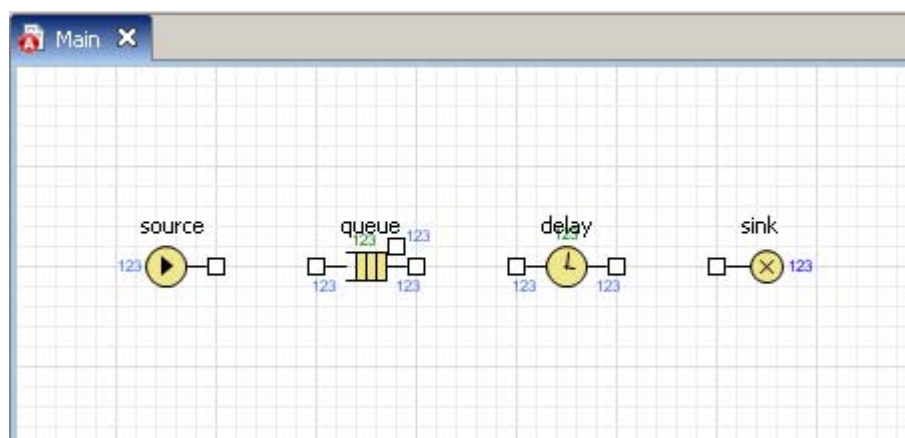
First, we will create a flowchart describing the system.

Creating the model flowchart

In AnyLogic you create flowcharts by adding the objects from the library stencil onto the class diagram, setting custom properties for the objects, and connecting objects together.

Add flowchart objects

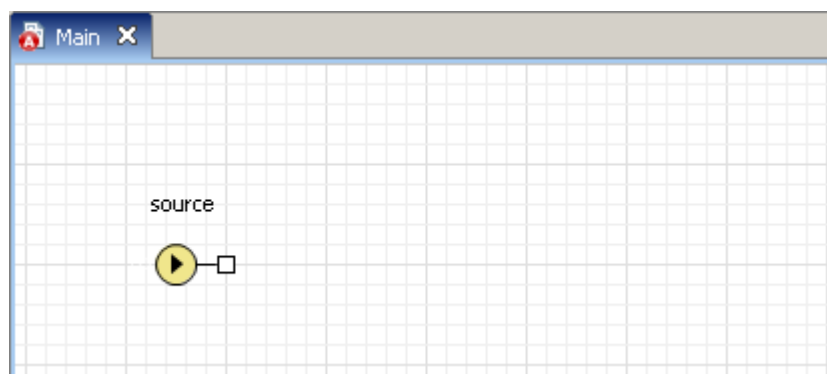
1. Add flowchart objects onto the diagram as shown below:



To add an object of the library on the diagram, first open the library stencil in the **Palette** view, then click on the object in the library stencil and finally, click in the graphical editor where you want to place the object.

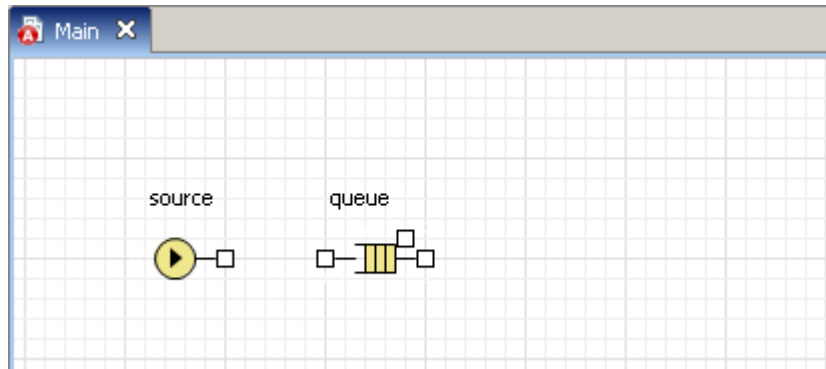
Enterprise Library stencil shows all the objects of the library.

2. Add **Source** object. **Source** object generates entities with the specified interarrival time. In our example, it models customer arrival.

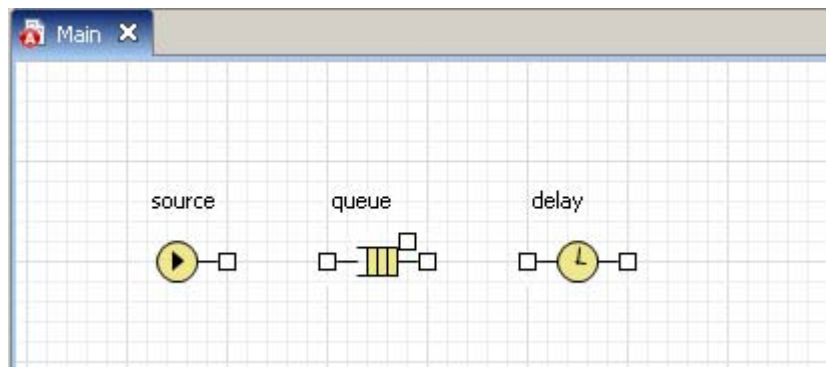


Please refer to *Enterprise Library Reference Guide* for the description of all the *Enterprise Library* objects. You can find there all object functions and their parameters. To open *Enterprise Library Reference Guide*, choose the corresponding item from the *AnyLogic Help*. To invoke *AnyLogic Help*, choose **Help|Help Contents** from AnyLogic menu.

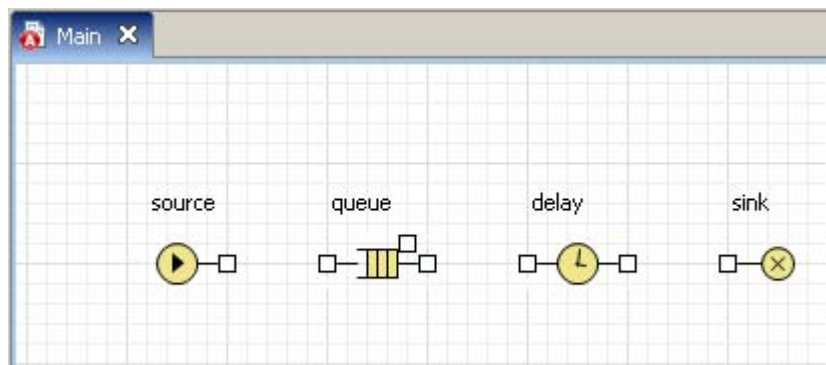
3. Add **Queue** object to model a queue of customers waiting for the moment they can be served.



4. Add **Delay** object. The **Delay** object models the ATM that is spending some time serving the customer.

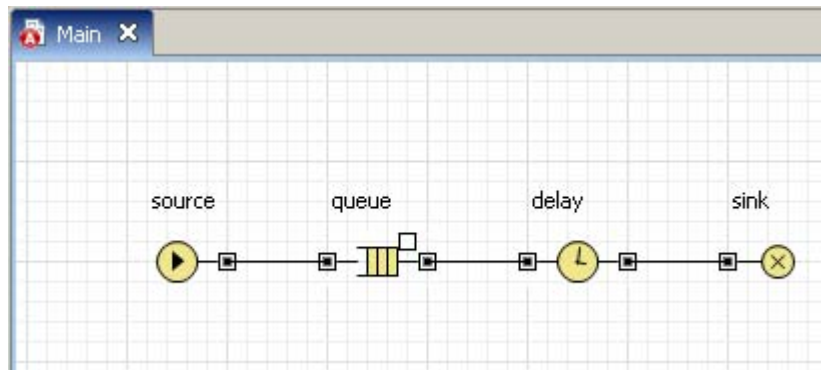


5. Add **Sink** object. This object indicates the end of the flowchart.



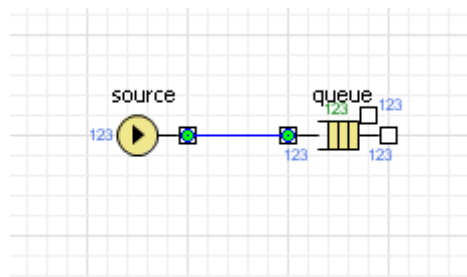
Connect the flowchart objects

1. Connect the objects as shown in the figure:



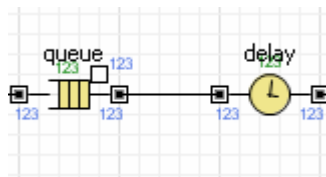
First, connect the port of *source* with the left port of *queue* (this port plays the role of the input port). Generally, input ports are placed on the left side of the object, while output ports - on the right.

To connect ports of objects, double-click the first port and then click the second port. The connector linking two ports appears.

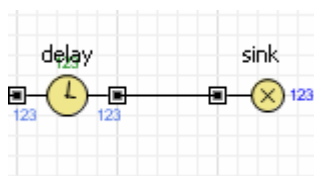


Cyan points inside ports indicate the correct connection. In the case you do not see cyan circles, it probably means that you put connector's point close by a port and you need to move it onto it.

2. Connect the output port of *queue* with the input port of *delay*.



3. Connect the output port of *delay* with the port of *sink*.



Now we will define the data of our model, adjusting the properties of the flowchart blocks.

To modify properties of some model element, first select it by clicking on it in the graphical editor or in the **Project** view. This opens the properties of this element in the **Properties** view.

Set up the properties of the flowchart objects

1. Modify the properties of the *source*.

- Specify how often customers arrive. Type 0.67 for **Interarrival rate**.

The screenshot shows the 'Properties' window for a 'source - Embedded Object'. The 'General' tab is selected. The 'Name' is 'source'. The 'Type' is 'Source<T extends Entity>'. The 'Arrival type' is 'Rate'. The 'Arrival rate*' is '0.67'. The 'Entities per arrival' is '1'. The 'New entity' is 'new Entity()'. The 'Show Name' checkbox is checked.

2. Modify the properties of the *queue*.

- Set queue capacity to 15 entities. At most 15 customers will wait in a queue.
- Select the **Enable statistics** check box to turn statistics collecting for this object on.

The screenshot shows the 'Properties' window for a 'queue - Embedded Object'. The 'General' tab is selected. The 'Name' is 'queue'. The 'Type' is 'Queue<T extends Entity>'. The 'Capacity*' is '15'. The 'Maximum capacity' checkbox is unchecked. The 'On enter', 'On at exit', and 'On exit' fields are empty. The 'Enable exit on timeout' checkbox is unchecked. The 'Animation guide shape' field is empty. The 'Animation type' is 'Path'. The 'Animation direction' has 'Forward' selected. The 'Enable statistics*' checkbox is checked.

3. Modify the properties of the *delay*.

- Name the object *ATM*.

- Specify the processing time. Assume that processing time is triangularly distributed with mean value of 1, min of 0.8 and max value of 1.3 minutes.

The `triangular()` function is the standard AnyLogic random number generator. AnyLogic provides also other random number distributions, like normal, poisson, exponential etc. Please refer to *AnyLogic Help* for the description of all the random number generators (see *AnyLogic Functions* topic).

- Select the **Enable statistics** check box.

The screenshot shows the 'Properties' window for an 'ATM - Embedded Object'. The 'General' tab is active. The 'Name' field is 'ATM'. The 'Type' is 'Delay<T extends Entity>'. The 'Delay time*' field contains the formula 'triangular(0.8, 1, 1.3)'. The 'Capacity' is '1'. The 'Maximum capacity' checkbox is unchecked. The 'On enter' and 'On exit' fields are empty. The 'Animation guide shape' field is empty. The 'Animation type*' dropdown is set to 'Single'. The 'Enable statistics*' checkbox is checked.

Configuring the model


Model simulation has a set of specific settings. A group of model settings is called an experiment, and experiments are displayed at the bottom of the model branch in the workspace tree. You can create several experiments for the same model with alternative model settings. One experiment is created by default and named *Simulation*.

If we start the model, it will work for 100 time units and then will stop. We want to study the process over a long period of time, therefore configure the model to work infinitely.

Set the model to stop at time 8


1. In the **Project** view, click the *Simulation:Main* experiment item.
2. On the **Model Time** tab of the **Properties** window, clear the **Stop model at specified time** check box.

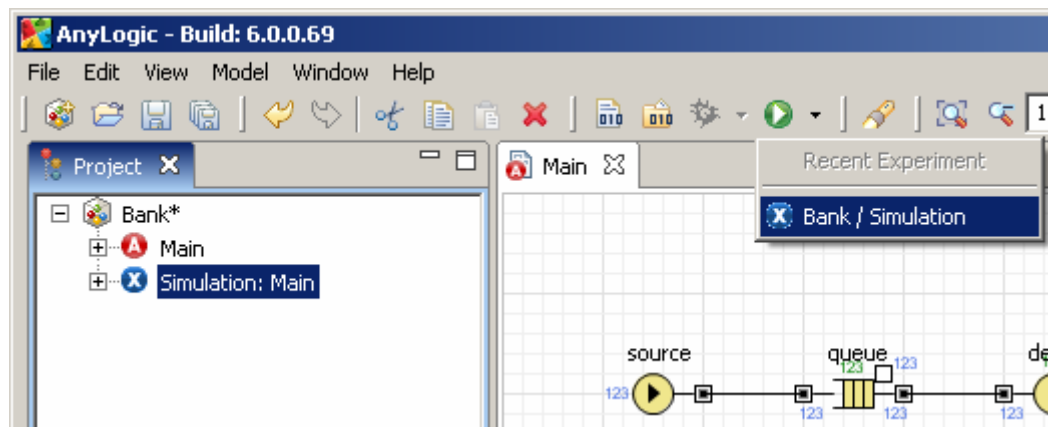
Running the model

We are ready to run the model created. First, build your model by clicking the **Build Model**  toolbar button. If there are some errors in your model, the building fails and the **Problems** view appears listing all the errors found in your model. Double-click an error in the list to open the location of the error and fix it.

After the model is successfully built, you can start it. Running the simulation, you automatically bring the current model up to date.

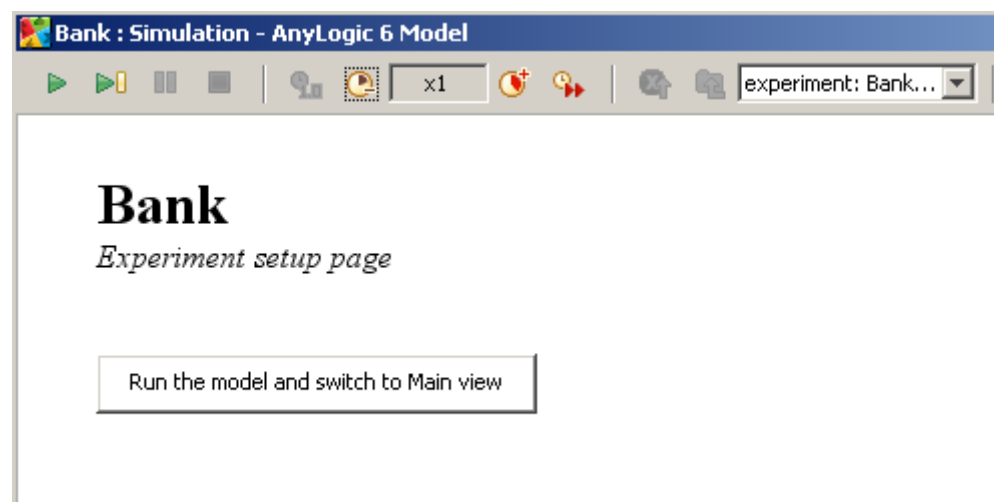
Start the model

1. Click the **Run**  toolbar button and choose the experiment you want to run from the drop-down list. Your simulation experiment is called *Bank/Simulation*.



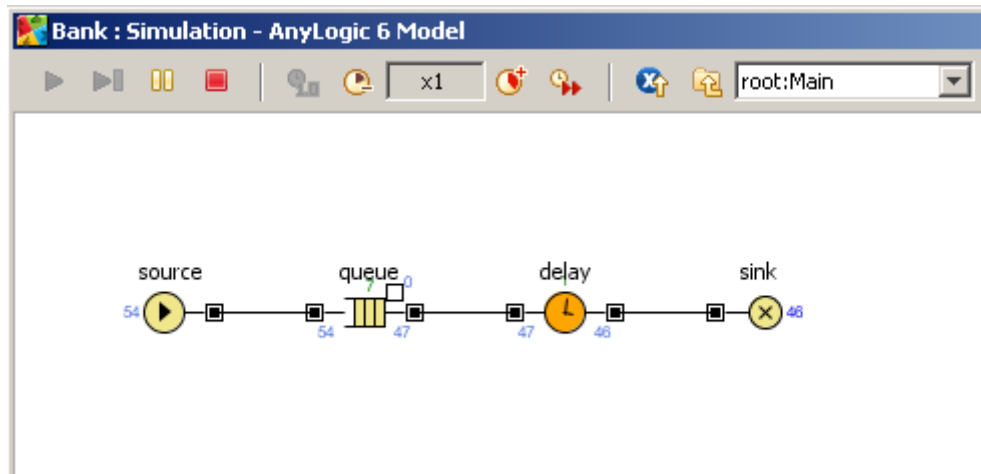
In the case this model is the only one opened in the workspace at the moment you will be prompted to run this particular experiment. Later on this button will start the previously run experiment. To run any other experiment, right-click the experiment in the **Project** view and choose **Run** from the context menu.

Having started the model, you will see the presentation window. It displays the presentation designed for your simulation experiment.



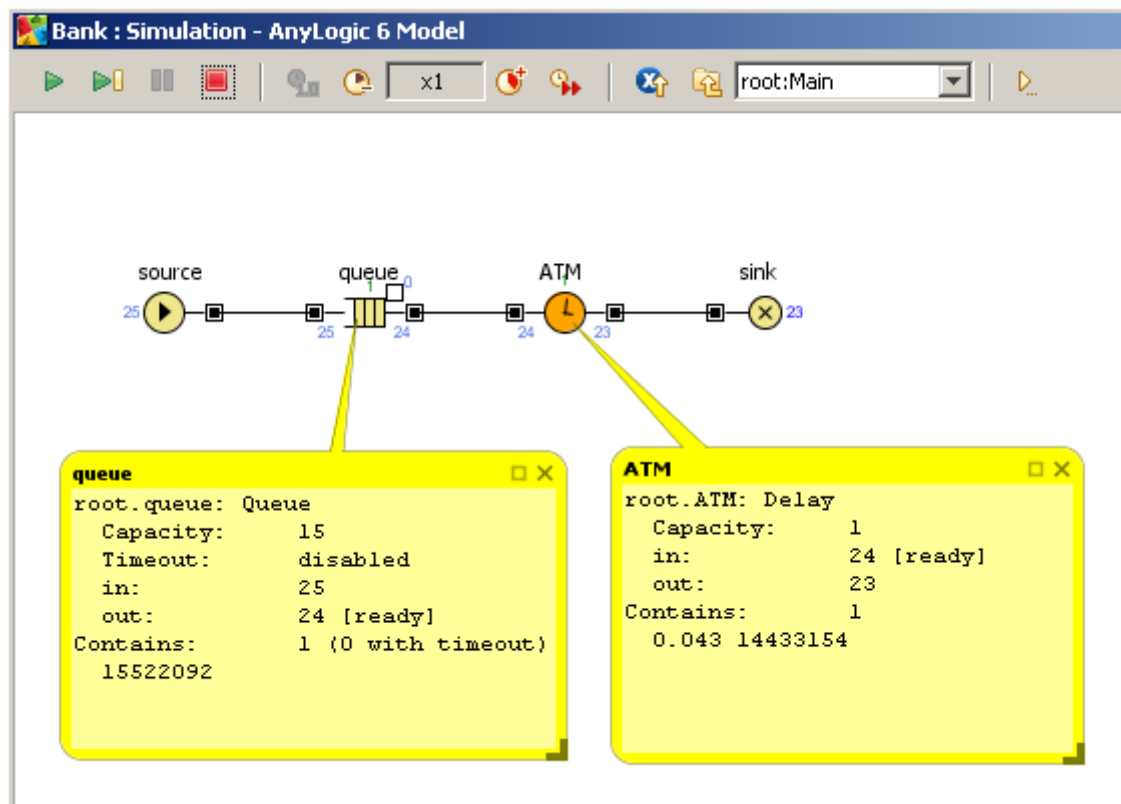
Click the button to **Run the model and switch to Main view**. and observe the process dynamics. You will see animated flowchart. Each model created with Enterprise Library

instantly has animated flowchart where you can see detailed current object status, for example queue size, number of entities left and so on – completely in graphics!



If needed, adjust the execution speed to your needs using **Slow down** and **Speed up** toolbar buttons.

You can inspect flowchart objects to get the detailed information on their current state. Click on the object to open its inspect window. Inspect window show statistics on the object, e.g. **Queue** object's inspect shows the queue capacity, the number of entities passed through either port of the object and also whether the timeout option is enabled for this queue. **Contains** string displays the number of entities currently being in the object along with IDs of these entities.



Step 3. Creating a Model Animation


Although the flowchart is animated, you may want to see the actual bank department layout animated. That is also possible! For each model you can create an animation to visually represent your model. You can create any animation you want. Now we will draw the layout consisting of the ATM and a queue. Then we will animate clients standing in the queue and using the ATM. We also want to visualize the current status of the ATM.

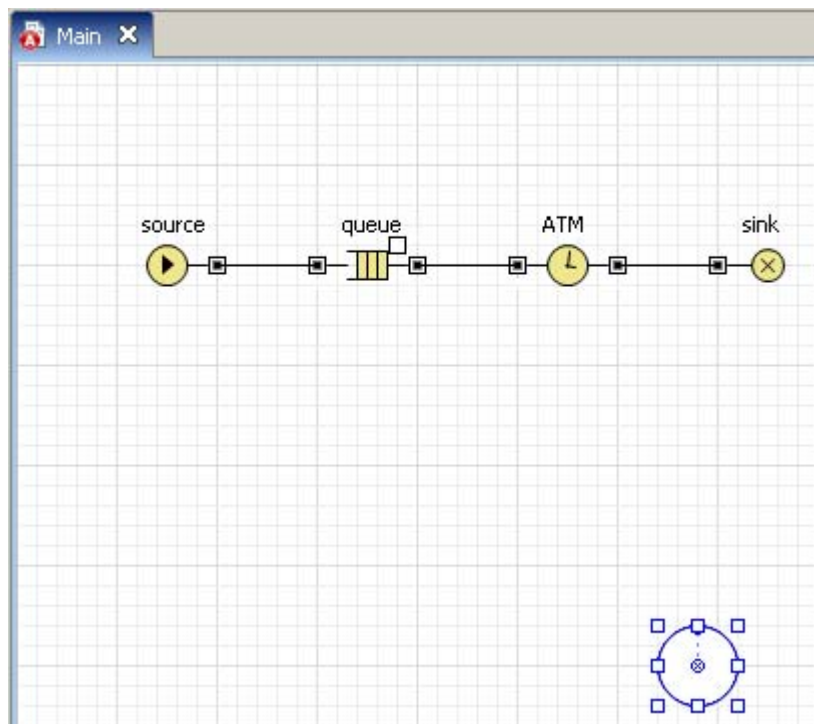
Now we will draw the layout of our bank. You draw the layout on the same diagram where you draw a flowchart. However, if you have existing image of the layout, you can simply import this picture as the bank layout instead of drawing it by yourself.

Draw an oval to depict the ATM


1. Draw ATM as an oval. First, open the **Presentation** stencil of the **Palette** view. This stencil contains shapes and controls you can add on your presentation.

To open some stencil of the **Palette** view, just click on the corresponding tab of the view.

3. Select the **Oval**  element in the **Presentation** stencil.
4. Click in the graphical editor to draw the oval on the diagram. Place it in the place shown in the figure below.

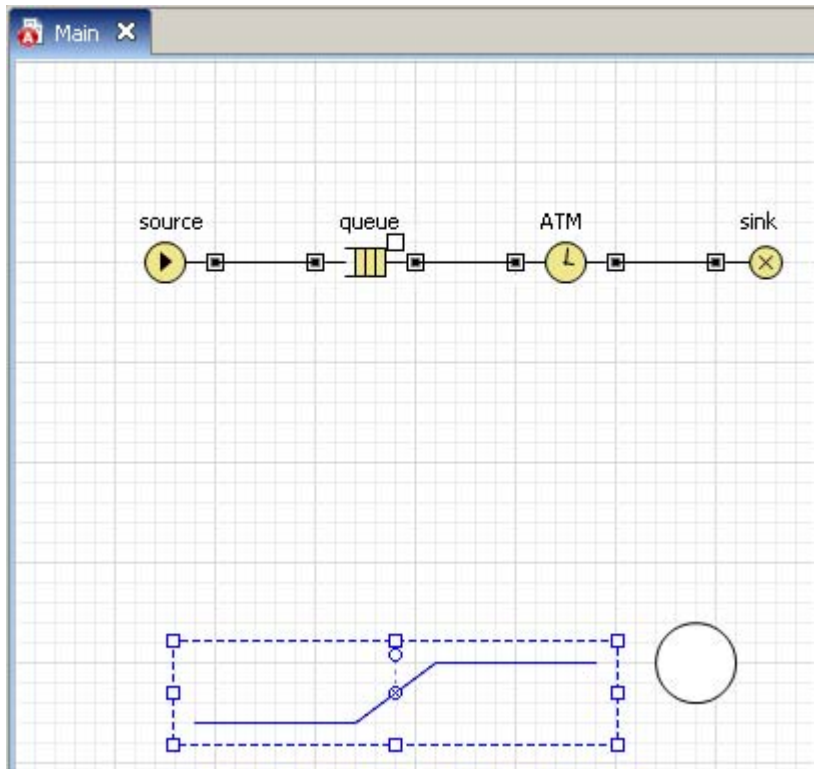


Draw a polyline to depict a queue to ATM


1. Select the **Polyline**  element in the **Presentation** stencil.
2. Successively click at each polyline point on the diagram and finally double-click where you want to place the end point of the polyline to finish drawing.

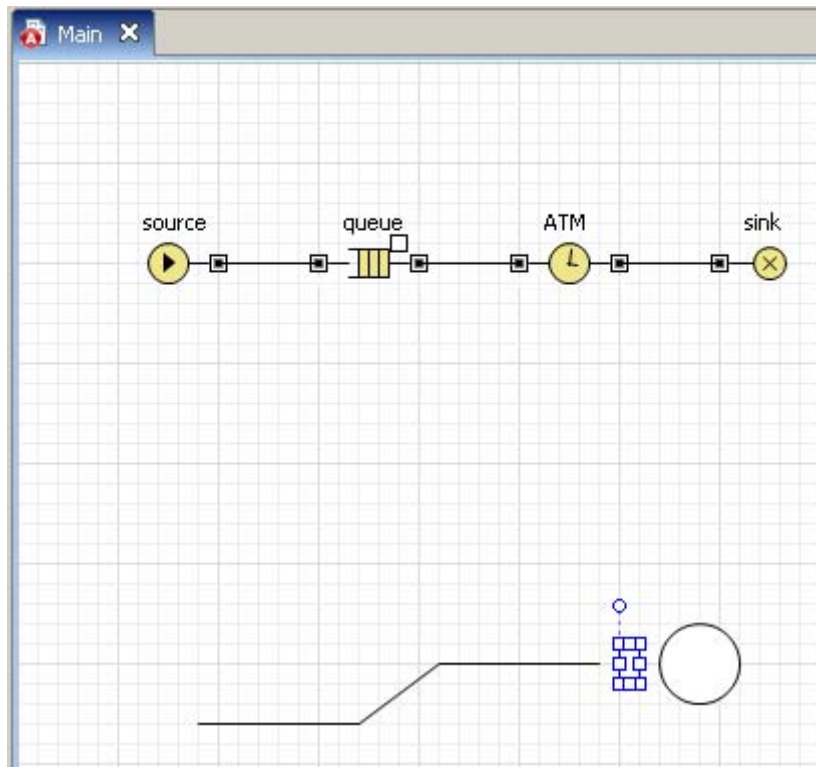
Draw queue path as shown in the figure below. Start drawing it from left to right and place the final polyline's point near the ATM oval. The starting point—that is, the point

you click first to draw a polyline—is important. By default, entities will be moving from the point you draw first to the point you draw last.

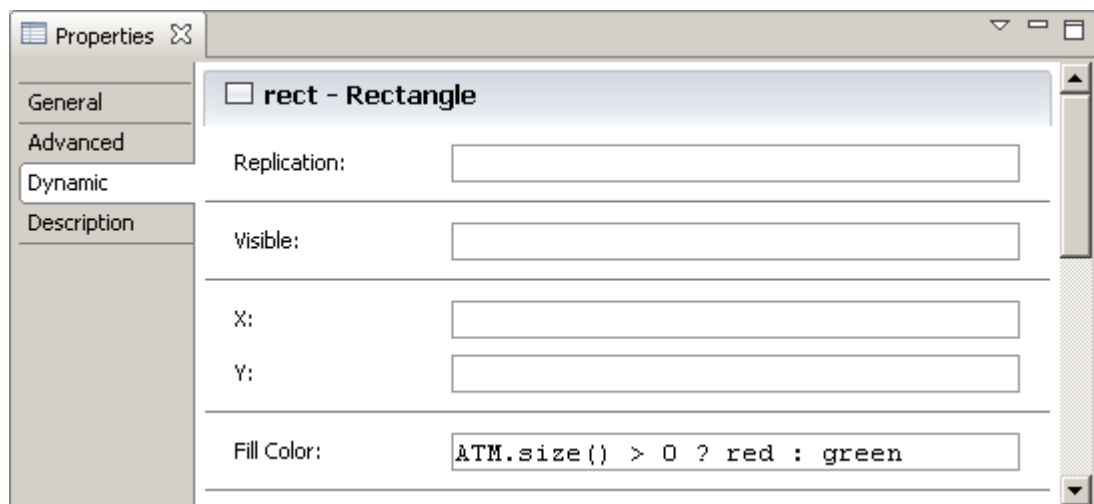


Draw a rectangle to indicate a status of the ATM

1. Select the **Rectangle**  element in the **Presentation** stencil.
2. Drag the rectangle in the graphical editor. Place it between the queue path and the ATM oval.



Set up some properties for the ATM drawn. Click the **Dynamic** tab of the **Properties** view to open dynamic properties of the rectangle. Type run-time color expression for the shape in the rectangle's **Fill Color** dynamic property: `ATM.size() > 0 ? red : green`



- Note that ATM here is the name of the **Delay** object we created. The expression determines the rectangle color at run time. The `size()` function returns the number of entities currently being processed. The color will be red, if a customer is served at this time, and green otherwise.

Now we are ready to animate the layout. This is done by setting up the animation properties for the logic flowchart objects.

To modify properties of the object, first open its properties in the **Properties** view by clicking on the object in the graphical editor or in the **Project** view.

1.2 Set up animation for queue

1. Set up *queue* animation. To animate a queue, just specify the queue path. Therefore, set *polyline* as the **Animation guide shape**.

Properties

queue - Embedded Object

General

Parameters

Description

Name: queue ☒ Show Name ☐ Ignore ☐ Pi

Type: Queue<T extends Entity> Generic parameters: Entity

Capacity* 15

Maximum capacity ☐

On enter

On at exit

On exit

Enable exit on timeout ☐

Animation guide shape* polyline

Animation type Path

1.2 Set up animation for ATM

1. Set *oval* as the **Animation guide shape**.
2. Change **Animation type** to *Single*. Many objects of the Enterprise Library support several animation styles. For example, a queue can show its contents like a line of items, a disordered heap of items, arranged items, and so on. For more information on animation styles please refer to *Enterprise Library Reference Guide*, **Animating Enterprise Library Objects** help topic. In this case, the ATM will be animated by a single animation of a customer being served.

Properties

ATM - Embedded Object

General
Parameters
Description

Name: ☒ Show Name ☐ Ignore ☐ P

Type: Generic parameters:

Delay time*

Capacity

Maximum capacity ☐

On enter

On exit

Animation guide shape*

Animation type*

Now you can run the model and observe its behavior. If you want to speed up the simulation significantly, switch to virtual time mode by clicking the **Toggle real/virtual time mode** toolbar button. Switching to virtual time mode allows you to view simulation run at its maximum speed. Therefore, you can simulate a long period of time.


Note that when the ATM station is serving a customer, it becomes red, and when it is idle, it is green.

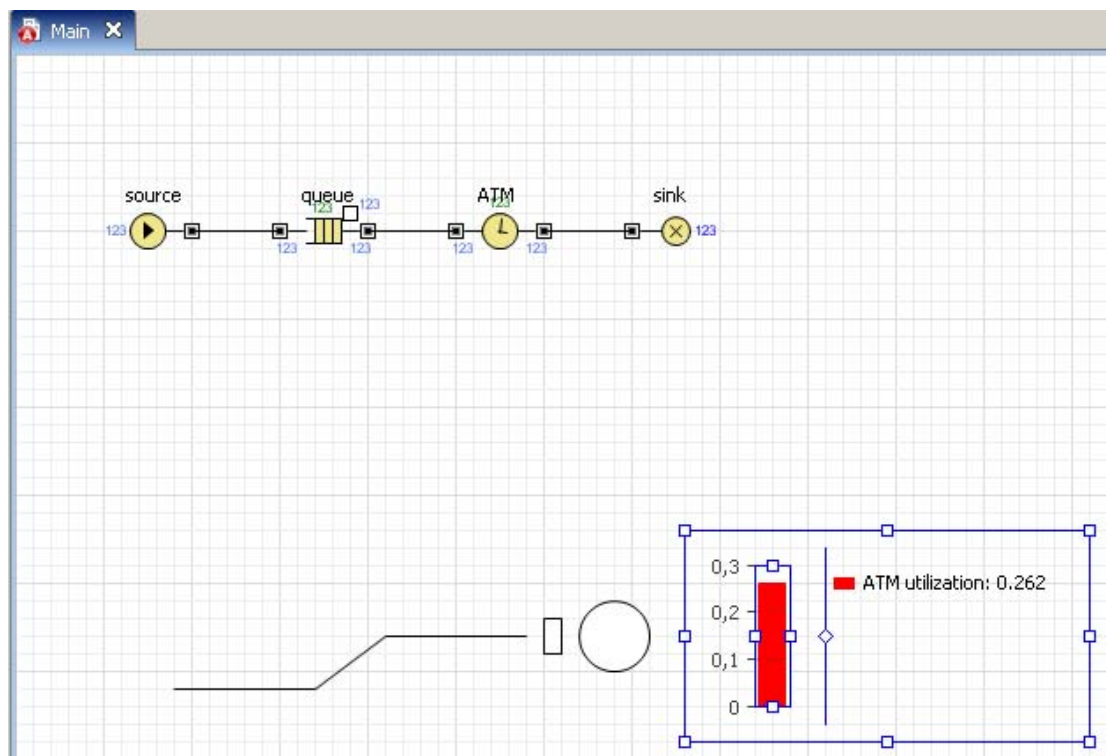
Step 4. Collecting Utilization Statistics

With AnyLogic, you can collect complicated statistics whenever you need them. The objects of the Enterprise Library are already capable of collecting the basic statistics. All you need is to turn the statistics collection for the object on, as it is disabled by default to speed up the model execution. Since we have already done this earlier, now we are ready to view the statistics collected for the flowchart objects with charts.

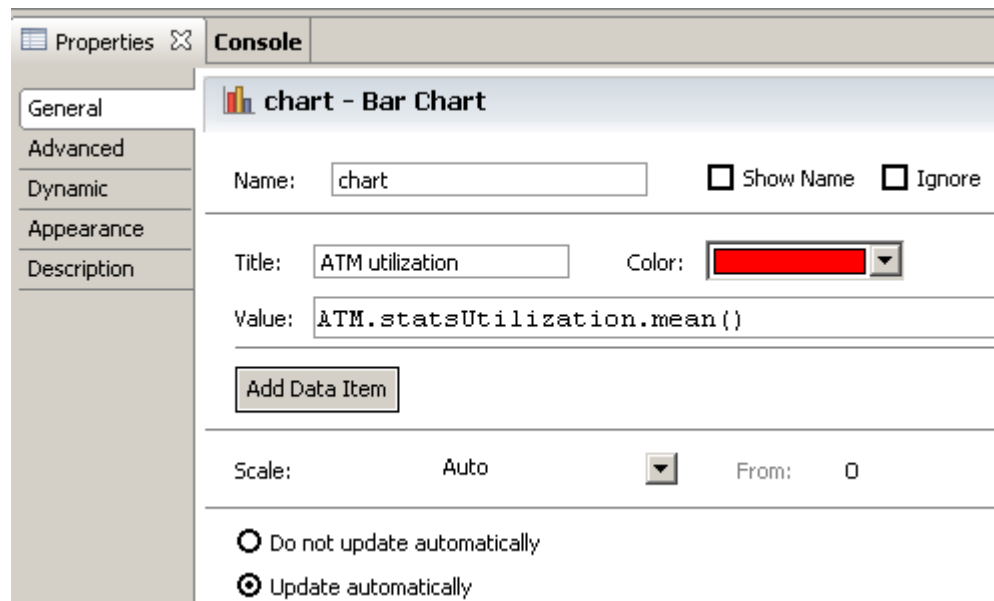
We want to observe how mean ATM utilization and mean queue length change with time.

Add a bar chart to indicate mean ATM utilization

1. Open the **Analysis** stencil of the **Palette** view. This stencil contains charts and data objects used for collecting data and performing various statistical analysis on them. Select the **Bar Chart**  element in the stencil.
2. Click the place in the graphical editor where you want to place the chart and then resize it.

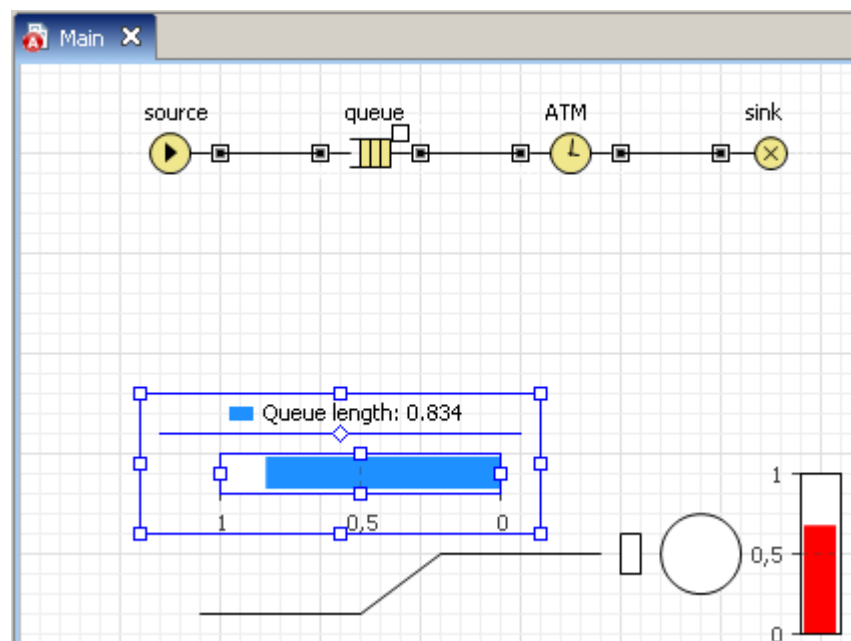


4. Go to the **Properties** of the chart. Click **Add Data Item** to add data item to be displayed by this chart.
5. Modify the data item's **Title**: *ATM utilization*.
6. Type `ATM.statsUtilization.mean()` as the **Value** of the data item. Here `ATM` is the name of the **Delay** object we created. Each **Delay** object has `statsUtilization` data set that collects statistics on the object utilization. The `mean()` is the function that returns the mean value measured. You can use other methods to get statistical values, such as `min()` and `max()`. You can find the list of all the methods on the **StatisticsContinuous** page.

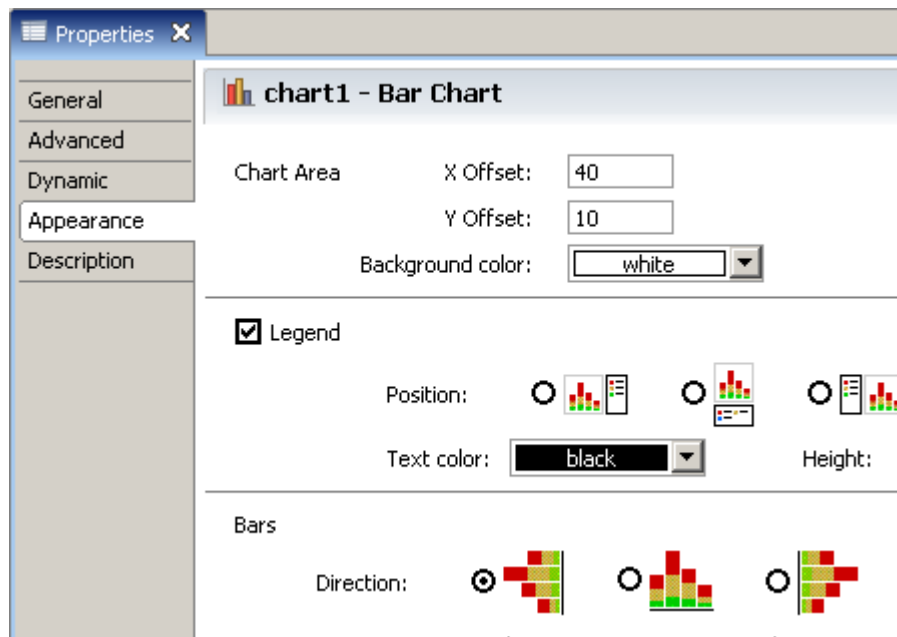


Add a bar chart to indicate mean queue length

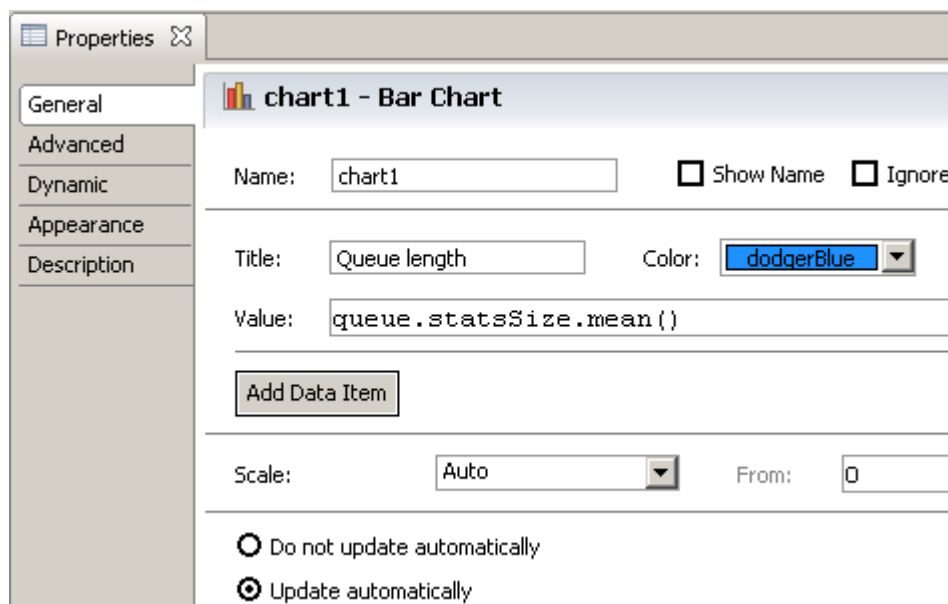
1. Add one more bar chart in the same way. Resize it to look like the one in the figure.



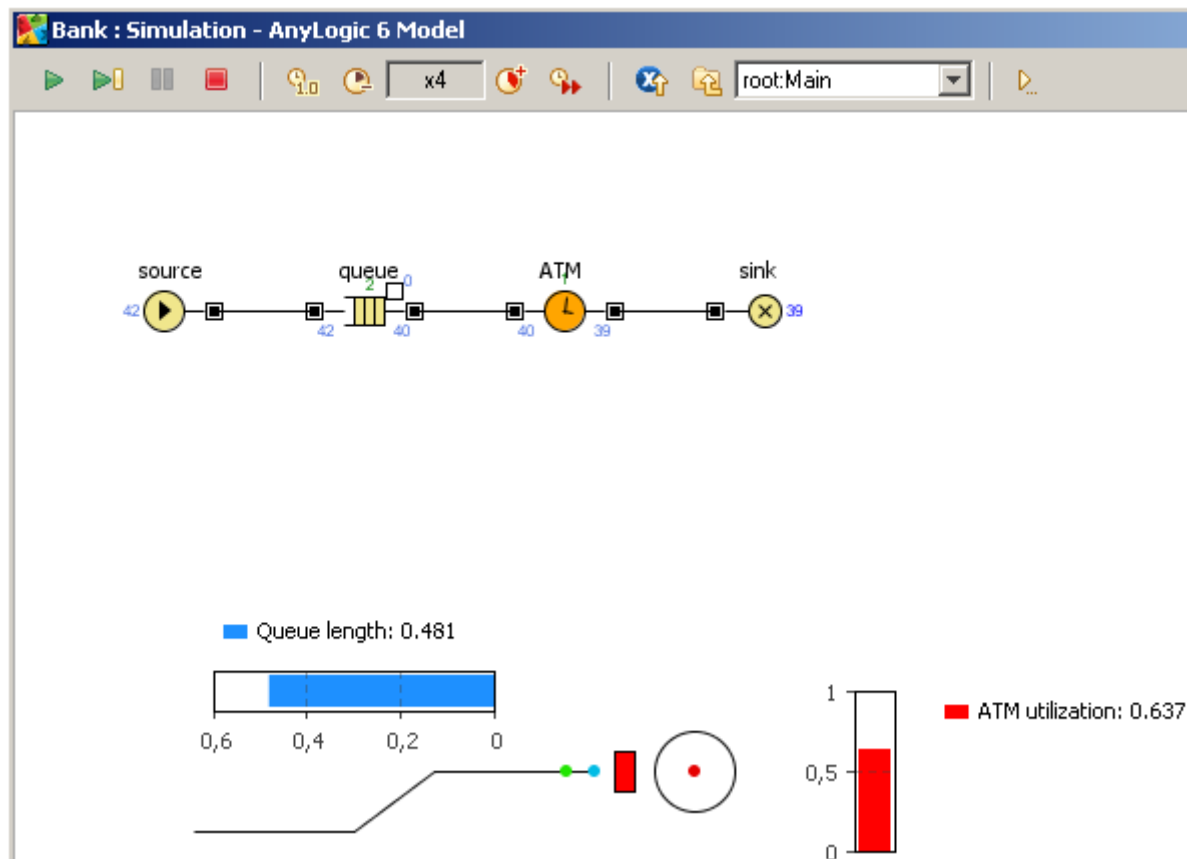
2. Open the **Appearance** tab of the **Properties** view and choose the first option from the **Bars Direction** section to make bars grow to the left.



3. Add a data item to be displayed by the chart. Set **Title**: *Queue length* and **Value**: `queue.statsSize.mean()` Here `statsSize` is the data set of type **StatisticsContinuous** that collects the statistics on the **Queue** size.



Run the model and observe the ATM utilization and mean queue length with just created charts.

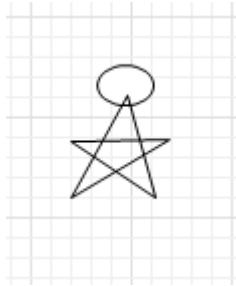


Step 5. Adding Custom Animations for Clients

At the moment bank clients are displayed on the animation as small rectangles. Now we want to define some custom animation for bank clients. You can use for that any existing image or draw some sophisticated animation using primitive AnyLogic shapes.

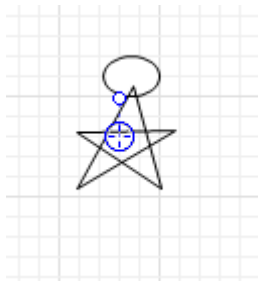
1 Draw a group of shapes to represent a bank client

1. Draw animation for customer as shown in the following figure. Draw it using polyline and oval.

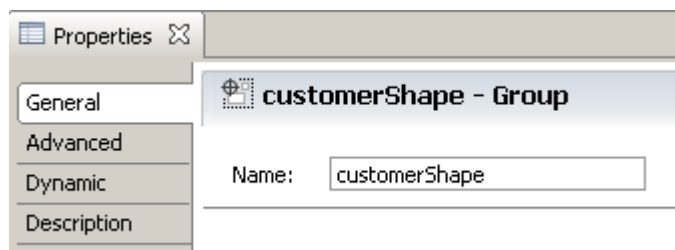


2 Add just drawn shapes to the group

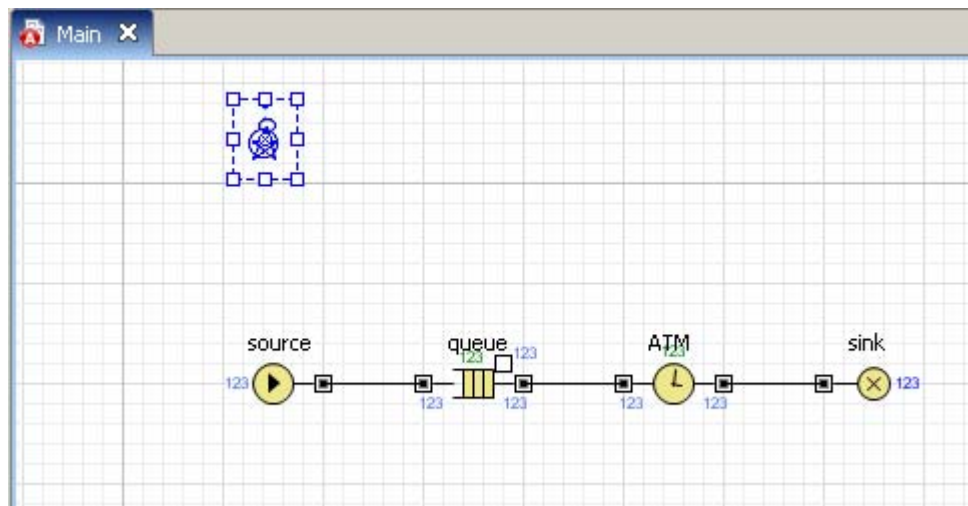
1. Select the oval and the polyline.
2. Right-click the selection and choose **Grouping|Create a group** from the popup menu.
3. You will see the group icon appeared in the centre of the selection.



4. Select the group and modify its properties. Name the group *customerShape*.



5. Select shapes and belittle them by dragging the handles.
6. We want to place the group outside the presentation area. Therefore, drag the diagram down with right mouse button pressed. Then drag the group of shapes above the flowchart as shown on the figure.



Successfully added shapes will move along with the group. If some shape stays where it is, it means it was not added to the group and you need to redo the previous steps.

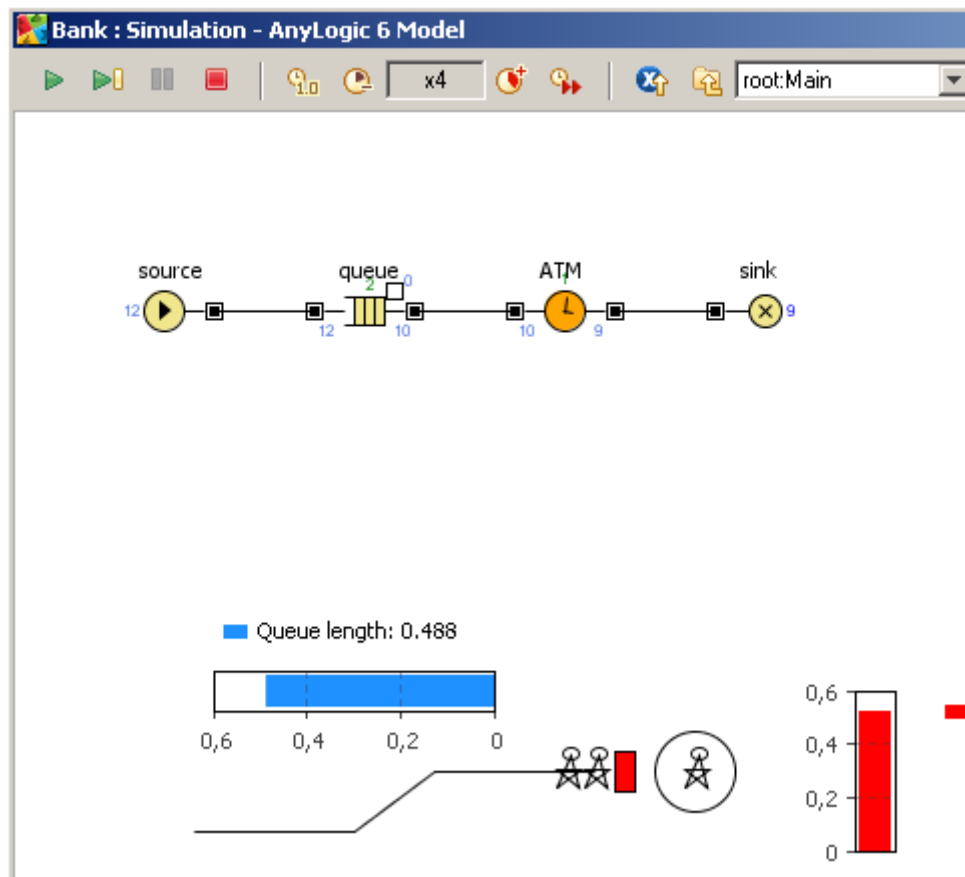
Now tell the source object that generated entities should be represented by this group of shapes.

1. Modify the source properties

1. Select the *source* object.
2. Type `customerShape` as **Entity animation shape**. You can use code completion assistant.

source - Embedded Object	
Name:	source <input checked="" type="checkbox"/> Show Name
Type:	Source<T extends Entity> Generic
Arrival type	Rate
Arrival rate*	0.67
Entities per arrival	1
New entity	new Entity()
On exit	
Entity animation shape*	customerShape

Run the model and view the animation.



You can see customers are represented by our custom shapes now.

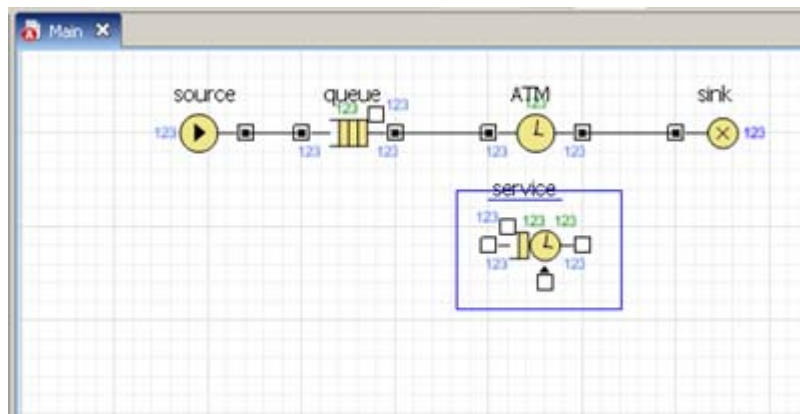
This model demonstrated the basics of the Enterprise Library. Now we are ready to create more advanced model.

Step 6. Adding Tellers

Now we will create another part of the system by adding tellers that are working at the bank. Now some clients will come to see tellers, some – to access the ATM. We can model tellers using delays in the same way as we modeled ATM. However, modeling tellers using resources is much more convenient. *Resource* is a special unit that can be possessed by entity. Only one entity can possess a resource at a time; therefore entities compete for resources.

Modify the flowchart

1. Add **Service** object.



Service seizes resource units for the entity, delays the entity, and releases the seized units.

Modify the object properties:

- There is one queue for all tellers. Set up **Queue capacity** to be of 20 places.
- We assume that service time is triangularly distributed with the min value of 2.5, average value of 6, and the max value of 11 minutes. Set **Delay time**: `triangular(2.5, 6, 11)`

Properties ✕

service - Embedded Object

General

Parameters

Description

Name: ☒ Show Name ☐ Ignore ☐ P

Type: Generic parameters:

Resource quantity:

Delay time*:

ResourcePool object:

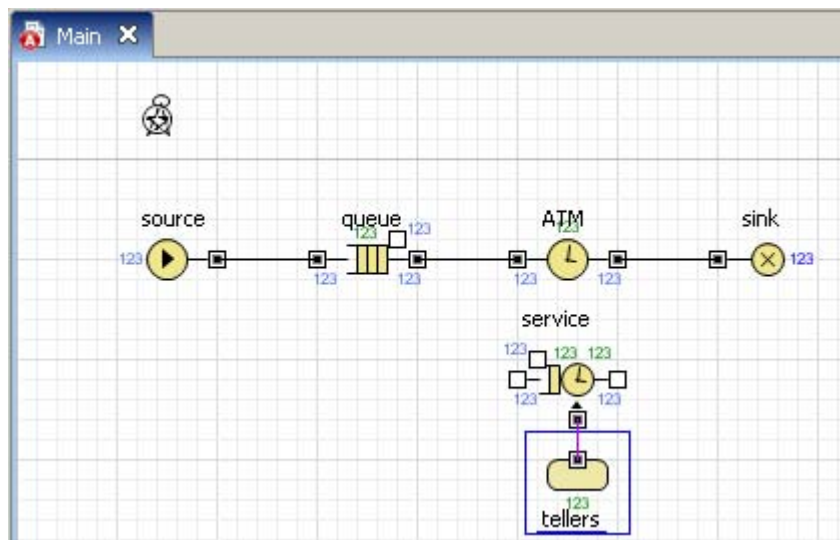
On enter:

On enter delay:

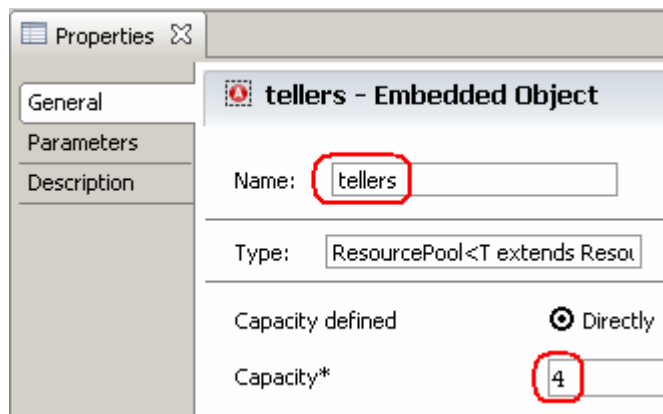
On exit:

Queue capacity*:

2. Add **ResourcePool** object. **ResourcePool** object is storage for resource units. It should be connected to resource seizing and releasing objects (**Service** in our case). So connect it to the lower port of the *service* object.



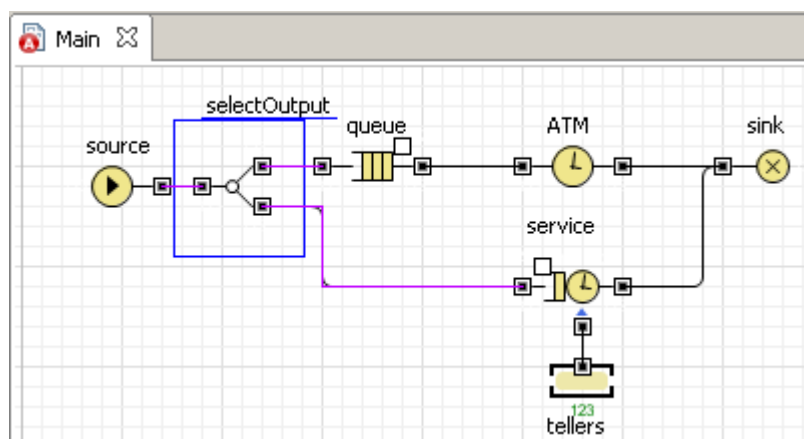
Name the object *tellers*. Specify that this resource object has only four resource units: define its **Capacity**: 4



- Place **SelectOutput** object. **SelectOutput** object is a decision making block. The entity arrived at the object is forwarded along one of two output ports depending on the user-defined condition.

Leave the default object properties. The entity routing condition `randomTrue(0.5)` defines that the number of customers competing for ATM and teller service will be approximately equal.

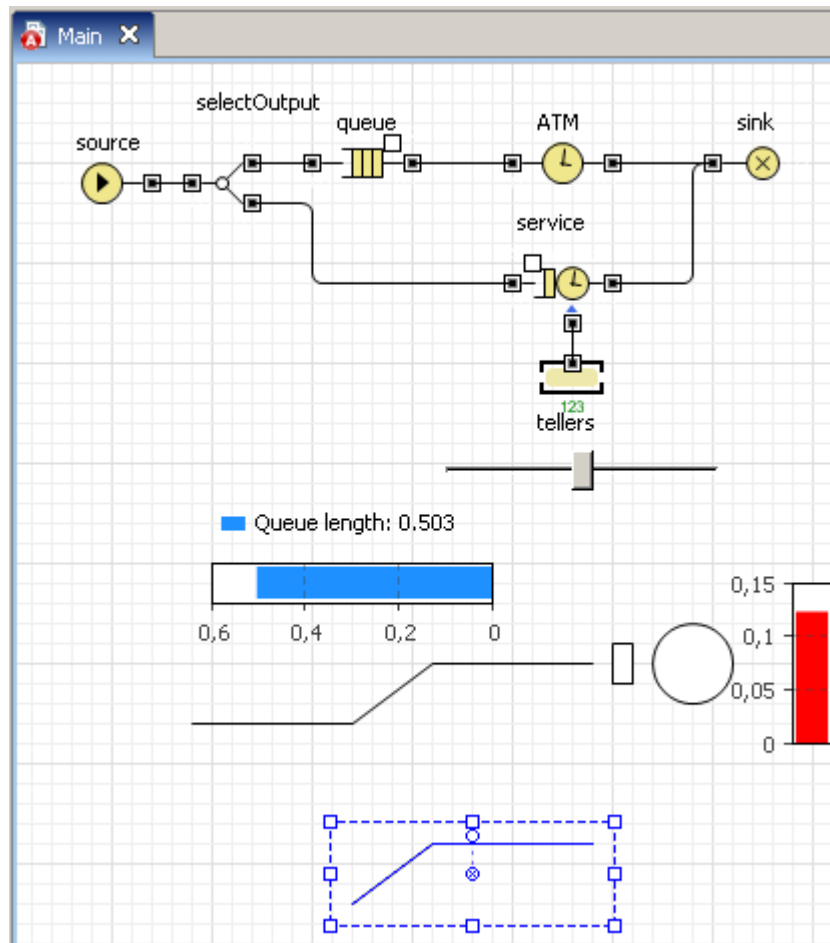
Connect *selectOutput* and *service* to other objects as shown in the figure:



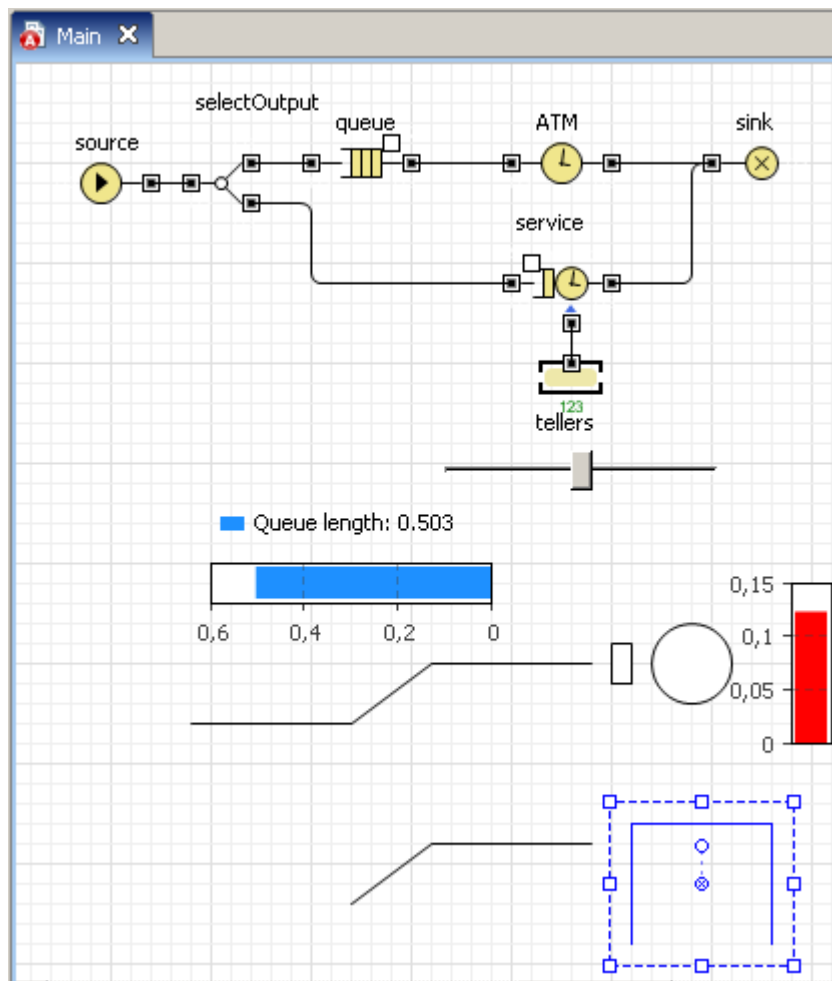
Since the model changed, we need to alter the model animation as well.

1. Modify the animation

- Draw a polyline to animate the waiting customers queue. Draw it from left to right. Name the polyline *queueBeforeTellers*.



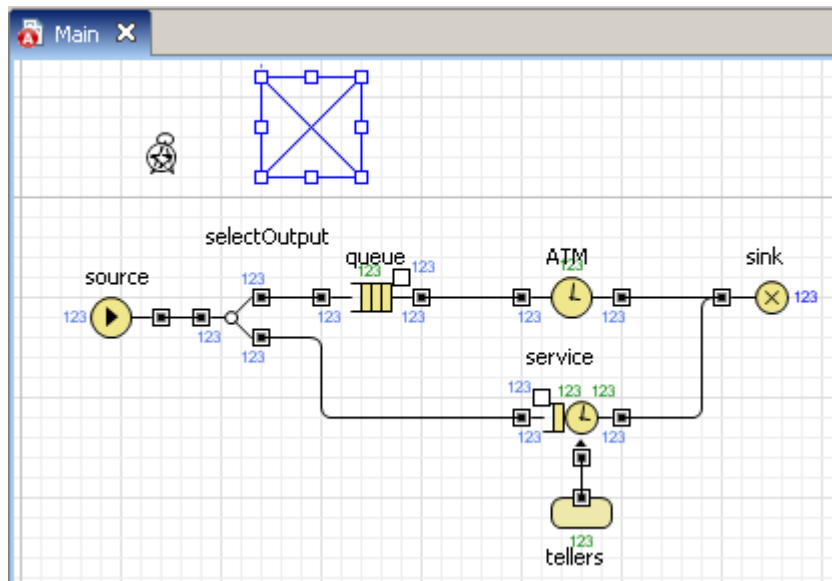
2. Draw a polyline with four points to the right of *queueBeforeTellers*. This polyline will indicate where teller animations are placed. Name it *tellerPlaces*.



Now we will animate our tellers by defining two images representing tellers being idle and busy. AnyLogic will automatically switch between these images when the resource unit becomes seized and released.

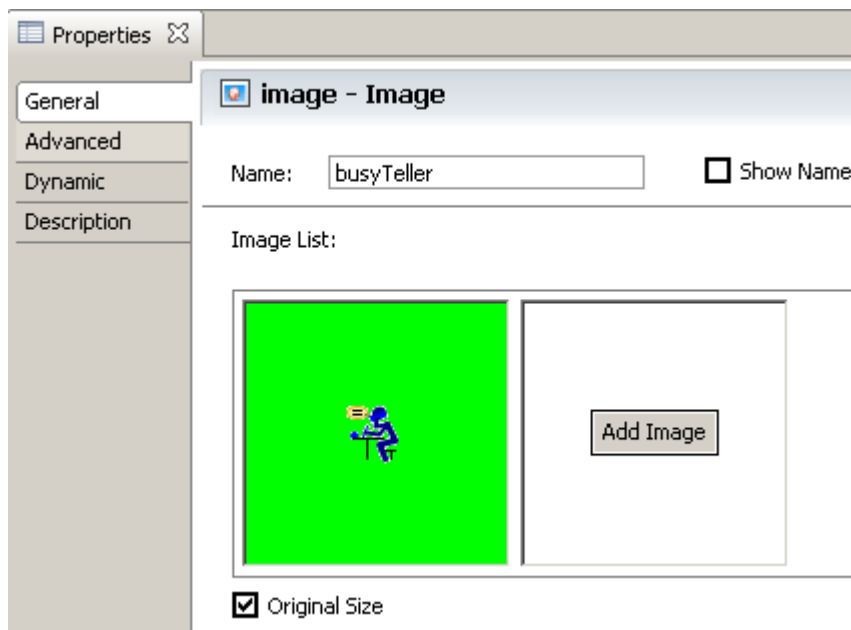
Animate tellers

1. Add **Image** from the **Presentation** stencil to the place shown in the figure with a blue crossed rectangle.



2. Go to the **Properties** of the image shape. Name the image *busyTeller*.
3. Define the image that this shape will display. Use **Add Image** button to add *Teller Busy.png*. You can find this file in the folder where *Billing Department* sample model is located (somewhat like *AnyLogic 6/examples/Billing Department*).

You will see the added image in the preview area.



4. Select the **Original Size** check box. If **Original size** option is not selected, the picture is shown to fit the rectangle of the image shape, and otherwise the picture is shown without any distortions.
5. Create one more image to show relaxing teller. Name it *idleTeller* and choose *Teller Idle.png* as the image displayed by this shape.

Now we will animate the flowchart by setting up animation properties of the flowchart objects.

1. Animate the model logic

1. Now we need to animate customers queue. Therefore, modify the *service* properties:
 - o Set *queueBeforeTellers* as **Animation guide shape (queue)**

The screenshot shows the 'Properties' window for an 'Embedded Object' named 'service'. The 'General' tab is selected. The 'Name' is 'service', 'Show Name' is checked, and 'Ignore' and 'F' are unchecked. The 'Type' is 'Service<T extends Entity>' and 'Generic parameters' is 'Entity'. The 'Resource quantity' is '1'. The 'Delay time*' is 'triangular(2.5, 6, 11)'. The 'ResourcePool object' is 'null'. The 'On enter', 'On enter delay', and 'On exit' fields are empty. The 'Queue capacity*' is '20'. The 'Enable exit on timeout' checkbox is unchecked. The 'Animation guide shape (queue)*' field is highlighted with a red rectangle and contains the text 'queueBeforeTellers'.

service - Embedded Object	
Name:	service <input checked="" type="checkbox"/> Show Name <input type="checkbox"/> Ignore <input type="checkbox"/> F
Type:	Service<T extends Entity> Generic parameters: Entity
Resource quantity	1
Delay time*	triangular(2.5, 6, 11)
ResourcePool object	null
On enter	
On enter delay	
On exit	
Queue capacity*	20
Enable exit on timeout	<input type="checkbox"/>
Animation guide shape (queue)*	queueBeforeTellers

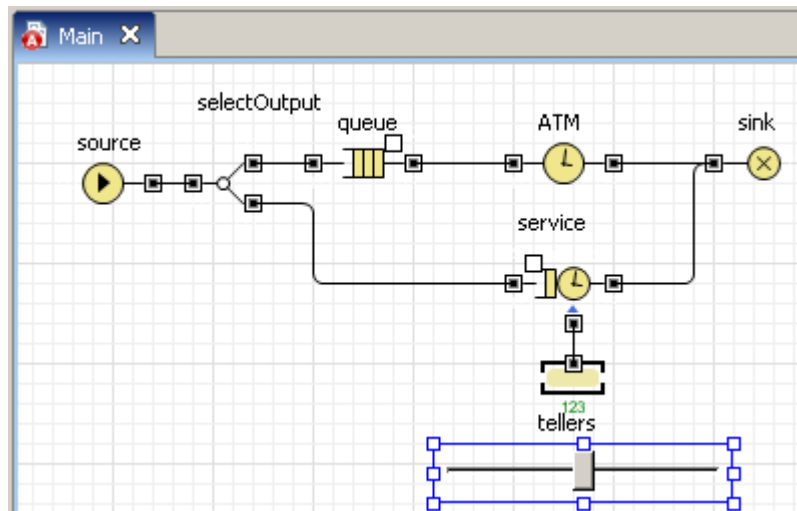
2. To animate tellers, modify the *tellers* properties:
 - o Specify *tellerPlaces* as **Animation guide shape**
 - o Choose **Set** as **Animation type**
 - o Set *idleTeller* as **Idle unit animation shape**
 - o Set *busyTeller* as **Busy unit animation shape**

The screenshot shows the 'Properties' window for an 'Embedded Object' named 'tellers'. The 'General' tab is selected. The 'Name' is 'tellers', with 'Show Name' checked and 'Ignore' unchecked. The 'Type' is 'ResourcePool<T extends Resol'. The 'Capacity defined' is 'Directly'. The 'Capacity*' is '4'. The 'New resource unit' is 'new ResourceUnit ()'. The 'On new unit', 'On seize', and 'On release' fields are empty. The 'Idle unit animation shape*' is 'idleTeller', the 'Busy unit animation shape*' is 'busyTeller', and the 'Enable rotation' checkbox is checked. The 'Animation guide shape*' is 'tellerPlaces' and the 'Animation type*' is 'Set'. The fields for 'Idle unit animation shape*', 'Busy unit animation shape*', 'Animation guide shape*', and 'Animation type*' are highlighted with red rectangles.

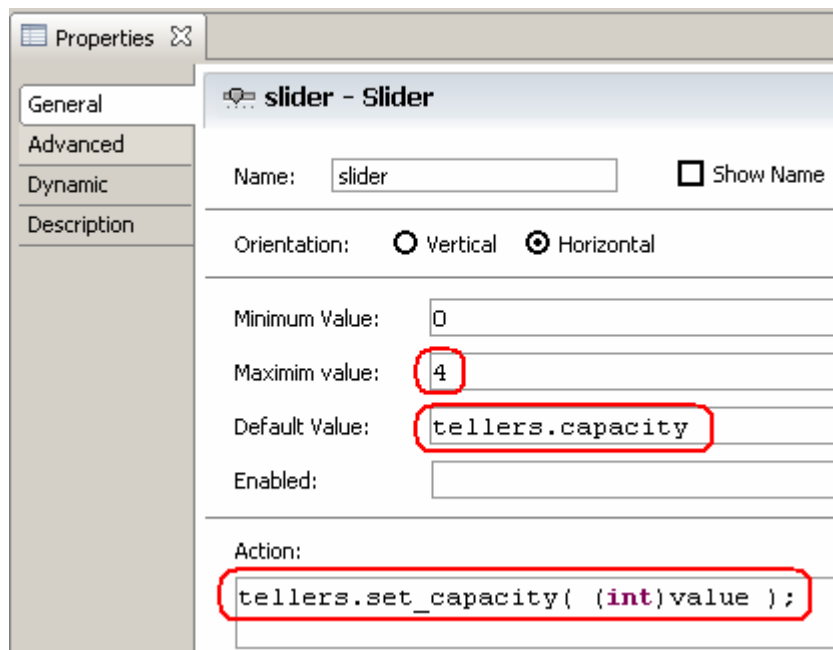
In the current model the number of tellers is fixed. However, we may need to vary it dynamically at the model run time. AnyLogic provides a set of controls enabling changing the model parameters. Now we will add a slider to vary the number of tellers at run time.

Add a slider to vary the number of tellers

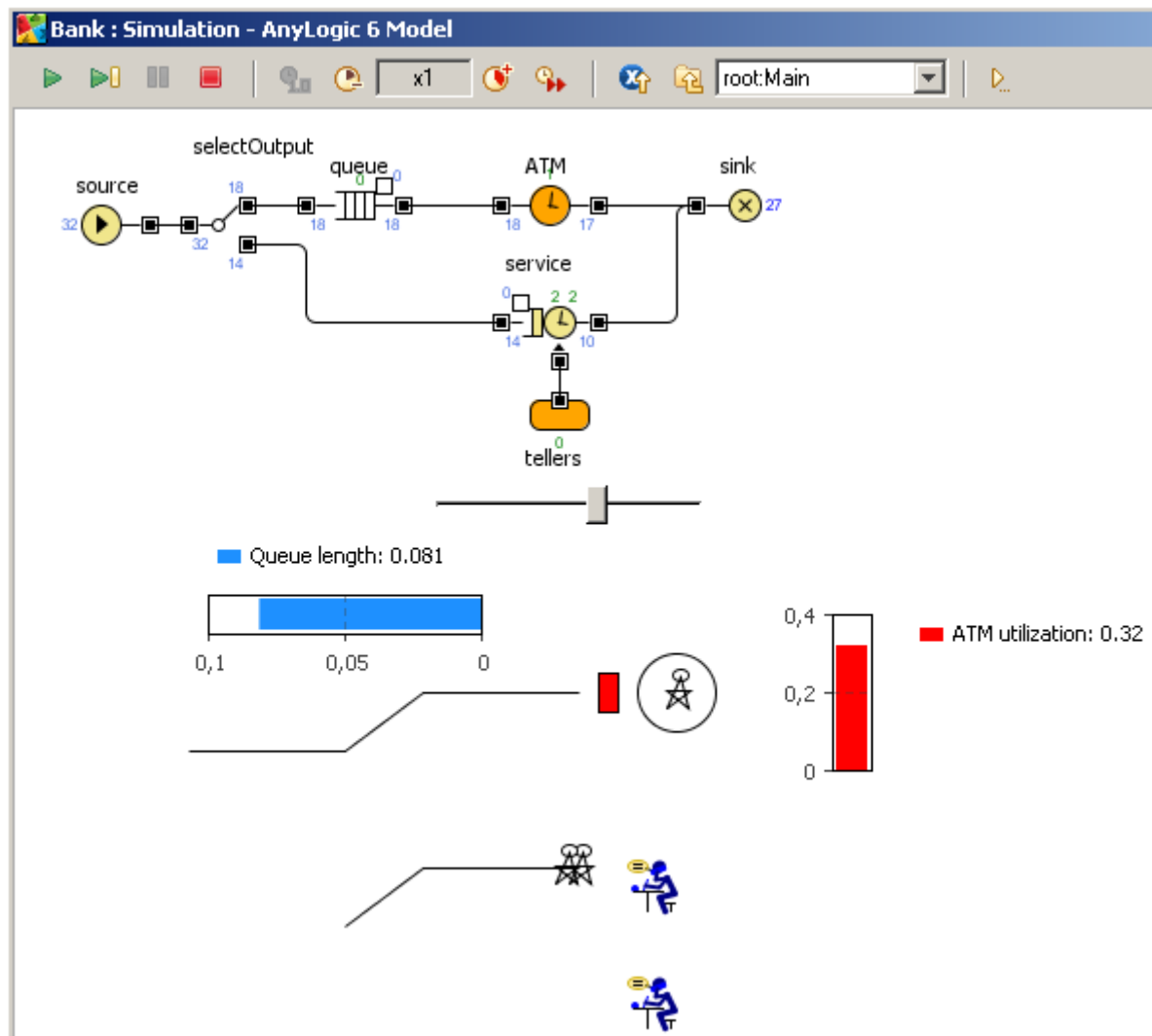
1. Add a slider. Controls are drawn in the same way as animation shapes. Open the **Presentation** stencil of the **Palette**, select the **Slider** element there and then click in the graphical editor where you want to place the slider. Place it below our **ResourcePool** object to let the user know that this slider will vary the capacity of this particular object.



2. We want to vary the number of tellers in the system from 0 to 4. Therefore, set 4 as slider's **Maximum value**.
3. Type `tellers.capacity` as the slider's **Default value**.
4. Specify `tellers.set_capacity((int)value);` as slider's **Action**.



Run the model. You can see tellers being animated with our images. Now you can change the number of tellers with the slider. Therefore, you can see how many tellers you need for the specific customer arrival rate.



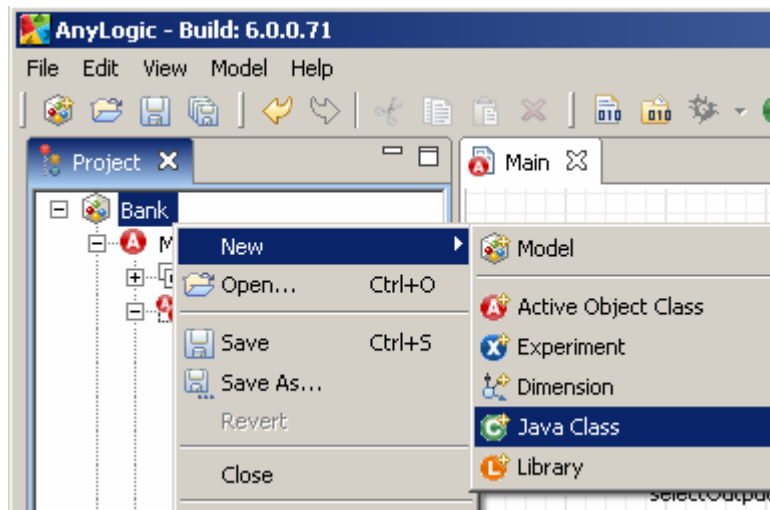
Step 7. Collecting Customer Time Statistics

We want to know how much time customer spends waiting in ATM queue and the whole time he spends in the bank. We will collect time statistics using AnyLogic analysis data objects and observe the resulting time distributions using histograms.

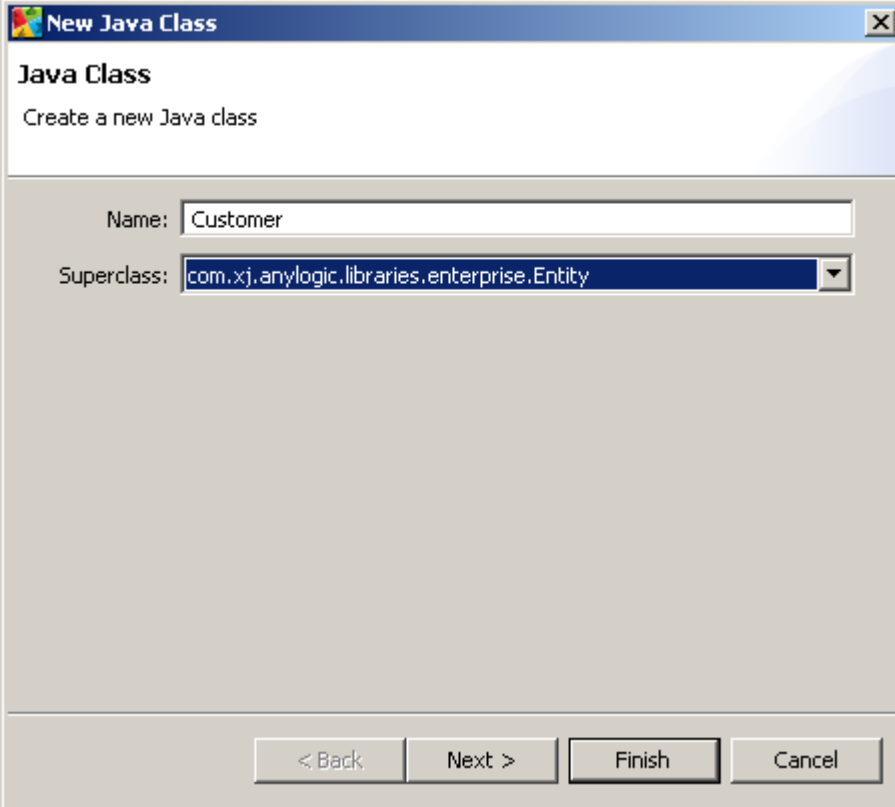
First, we will create new entity class *Customer* to represent customers in our model. This class will have parameters to carry the required time information.

Define *Customer* Java class

1. Right-click model item and choose **New|Java Class** from the popup menu.



2. This opens **New Java Class** dialog. Type the name of the new Java class: *Customer*. Specify that *Customer* is derived from *Entity*. Therefore, select *com.xj.anylogic.enterprise.Entity* from the **Superclass** drop-down list.



New Java Class

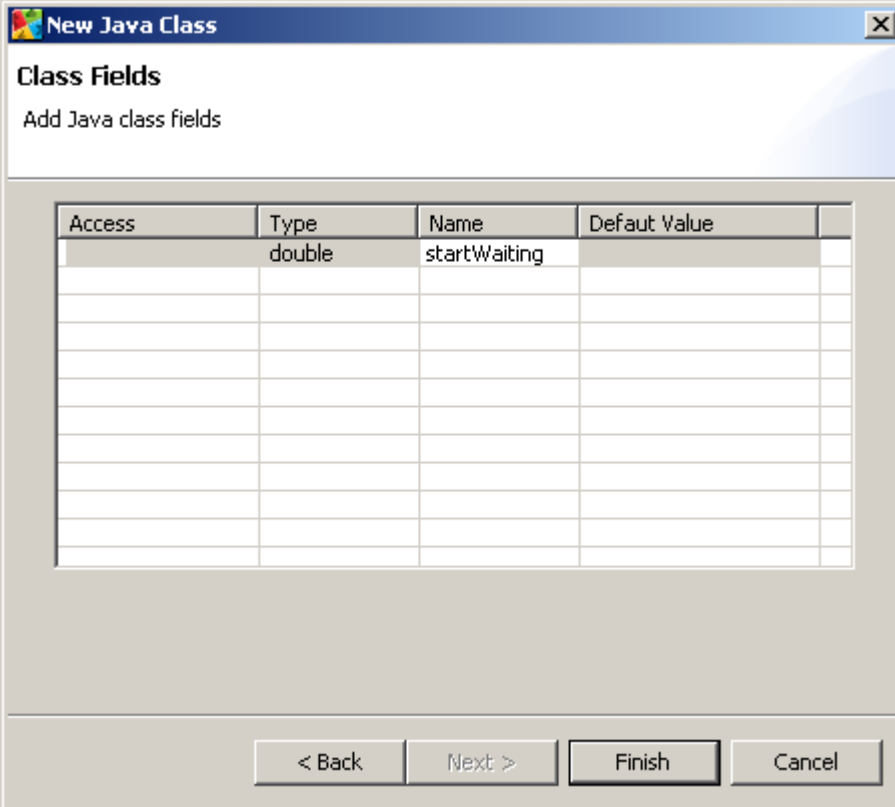
Java Class
Create a new Java class

Name:

Superclass:

< Back Next > Finish Cancel

- Click **Next** to open the next page of the wizard. On the second page of the wizard, define the parameters of the class. Create *startWaiting* parameter of type *double* to store the time when the customer starts waiting in the ATM queue.



New Java Class

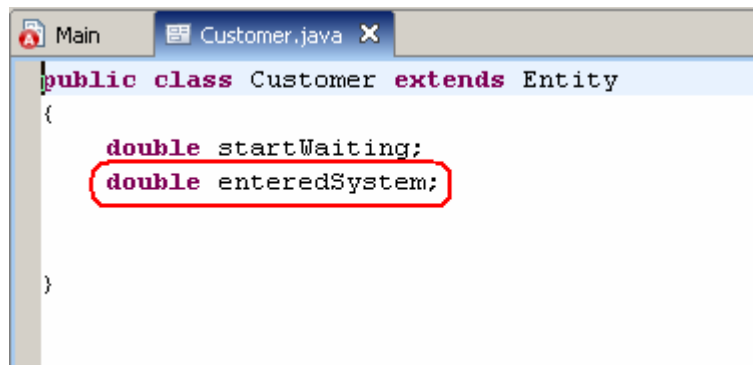
Class Fields
Add Java class fields

Access	Type	Name	Default Value
	double	startWaiting	

< Back Next > Finish Cancel

- Click **Finish**. You will see the code editor for the created class opened.

Type `double enteredSystem;` in the class body to define one more class parameter. This parameter will store the time when the customer enters the bank department.

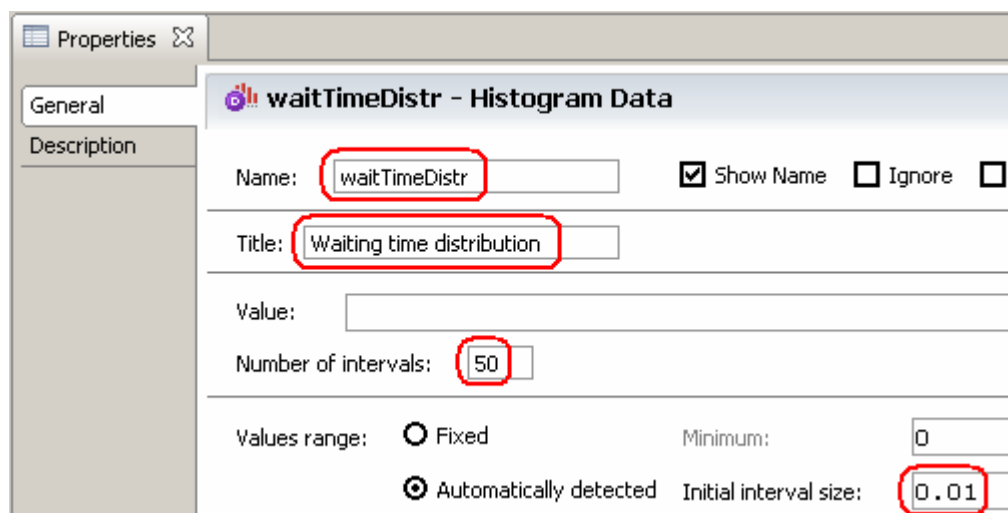


- When finished, close the code editor by clicking on the “cross” button in the upper right corner of the editor.

Add histogram data objects to store statistics on customer's waiting time and time in system. Histogram data objects support standard statistical analysis on the data values being added (calculate mean, minimum, maximum, deviation, variance and mean confidence interval).

Add histogram data objects to collect time statistics

- To add histogram data object on the diagram, select **Histogram Data** in the **Analysis** stencil of the **Palette** and then click on the class diagram.
- Set up the properties of the element.
 - Change the **Name** to *waitTimeDistr*.
 - Change the **Title** to *Waiting time distribution*.
 - Set the **Number of intervals** equal to 50.
 - Set the **Initial interval size**: 0.01.



- Create one more histogram data object. **Ctrl+drag** histogram data object to create its copy. Change the **Name** to *timeInSystemDistr*. Change the **Title** to *Time in system distribution*.

Properties

General

Description

timeInSystemDistr - Histogram Data

Name: ☒ Show Name ☐ Ignore ☐

Title:

Value:

Number of intervals:

Values range: ☐ Fixed Minimum:
☒ Automatically detected Initial interval size:

Now we will modify properties of our flowchart objects.

Modify the properties of the flowchart objects

1. Modify *source* properties:

- Specify new `Customer()` as **New entity**. Now this object will generate entities of our *Customer* class. Type *Customer* in **Generic parameters** field. This enables direct access to the fields of the *Customer* entity in the dynamic parameters of this object.
- Type `entity.enteredSystem = time();` in **On exit**. This code stores the time when a customer was generated in the *Customer's* variable `enteredSystem`.

The `time()` function returns the current model time.

Properties

General

Parameters

Description

source - Embedded Object

Name: ☒ Show Name ☐ Ignore ☐ Public

Type: Generic parameters:

Arrival type:

Arrival rate*:

Entities per arrival:

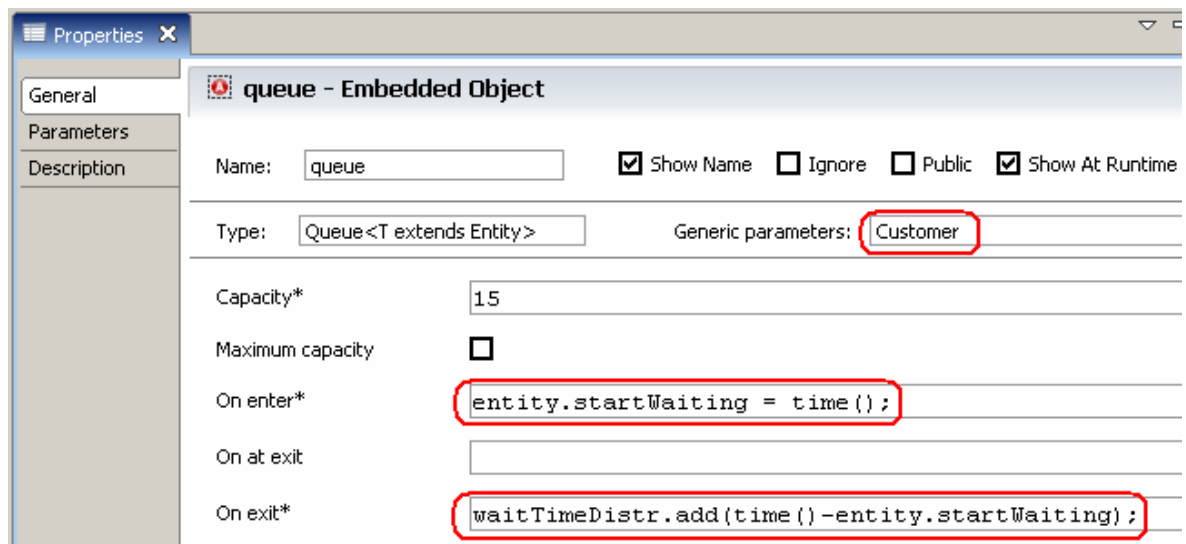
New entity*:

On exit*:

Entity animation shape*:

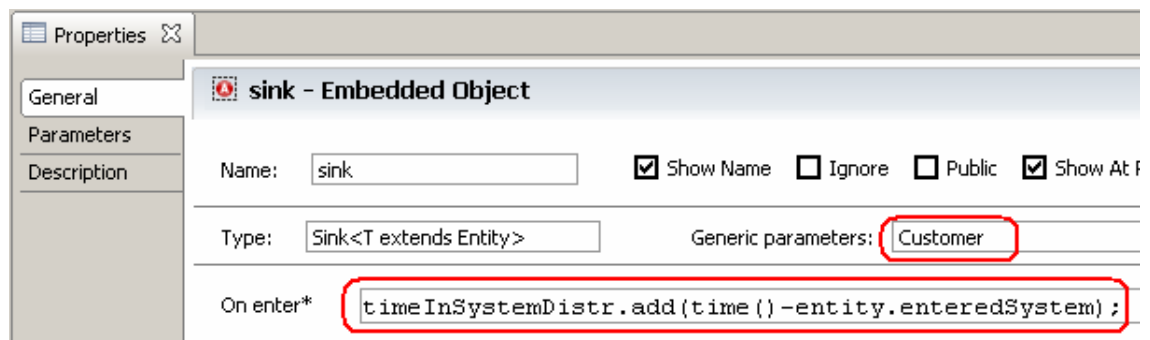
2. Modify *queue* properties:

- Type **Customer** in **Generic parameters** field.
- Type `entity.startWaiting = time();` in **On enter**. This code stores the time when a customer started waiting in the queue in the *Customer's* variable `startWaiting`.
- Type `waitTimeDistr.add(time() - entity.startWaiting);` in **On exit**. This code adds waiting time of the customer to the `waitTimeDistr` histogram data object.

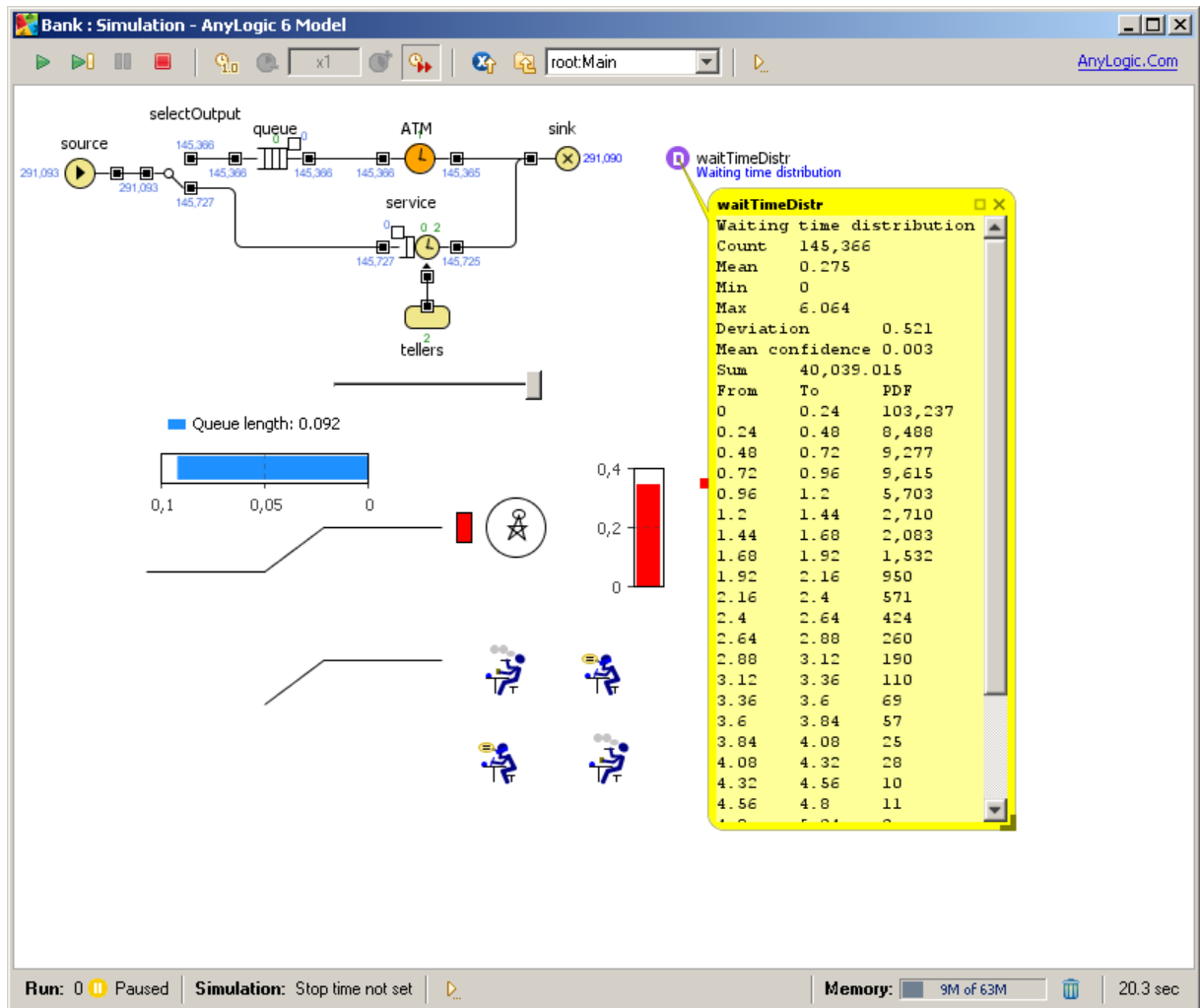


3. Modify *sink* properties:

- Type **Customer** in **Generic parameters** field
- Type `timeInSystemDistr.add(time() - entity.enteredSystem);` in **On enter**. This code adds the whole time the customer spent in the bank to the `timeInSystemDistr` histogram data object.



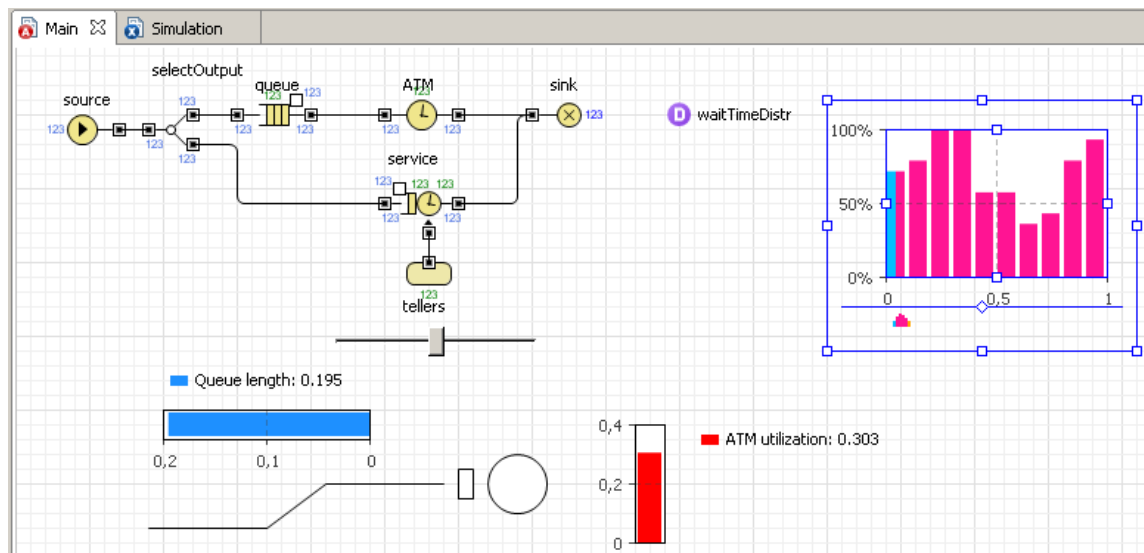
Run the model and view the statistics using inspect window of the data set. Open inspect window for data set by clicking on it. Here you can see standard statistical analysis on the data values being added to this data object.



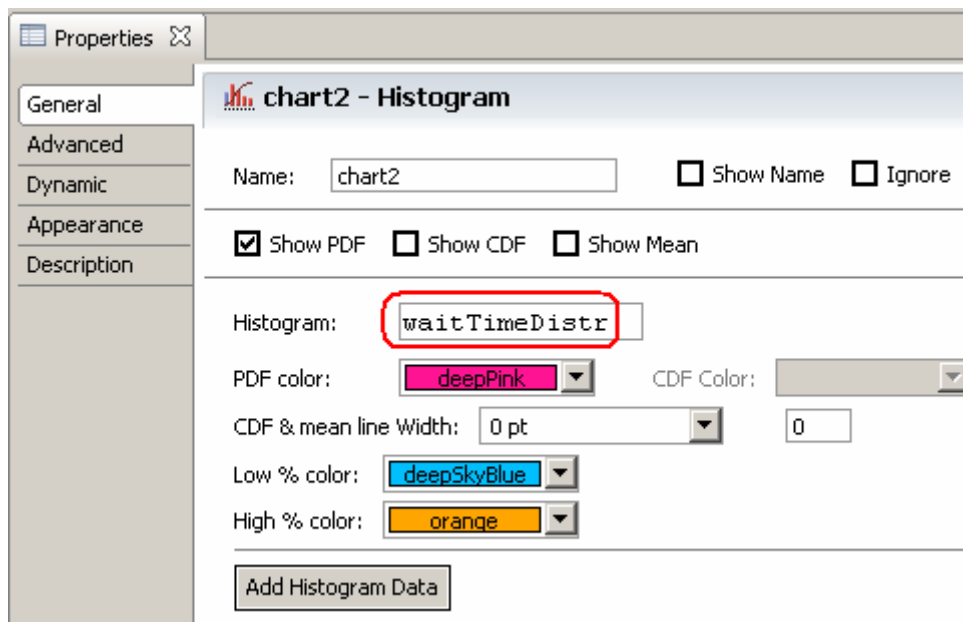
Now we want to display the collected statistics using standard histograms.

 **Add two histograms to display distributions of customer's waiting time and time in system**

1. To add histogram on the diagram, select **Histogram** in the **Analysis** stencil of the **Palette** and then click on the class diagram where you want to place the histogram.



2. Define the data object to be displayed by this histogram. Click the **Add Histogram Data** button and specify the **Histogram** to be displayed: *waitTimeDistr*.



3. Add one more histogram below the existing one.
4. Choose the data object to be displayed: *timeInSystemDistr*.
5. Change the **Title** of the displayed data to *Time in system distribution*.

Run the model. Set virtual time mode and observe distribution of customer's time in system and waiting time.

