HOpeR – Tuesday afternoon workshop session

An Agent Based Model for Health Screening

This worksheet was prepared by Prof Stephan Onggo & Dr Steffen Bayer, based on an assignment developed by Sally Brailsford.

Health screening means testing a healthy (or seemingly healthy) person to see whether they have a disease. An example for this would be testing people for early signs of cancer or early signs of diabetic eye disease so they can be treated to prevent the cancer or eye disease developing further.

Simulation is widely used to compare and evaluate different screening programmes. In order to evaluate the costs and benefits of screening, a baseline simulation model is built to represent the untreated disease process in a population. Different scenarios are then run to show the costs and benefits of screening using different tests, or at different time intervals, or for different population subgroups. Simulation can show the effects of screening policies which would take many years to test in the real world.

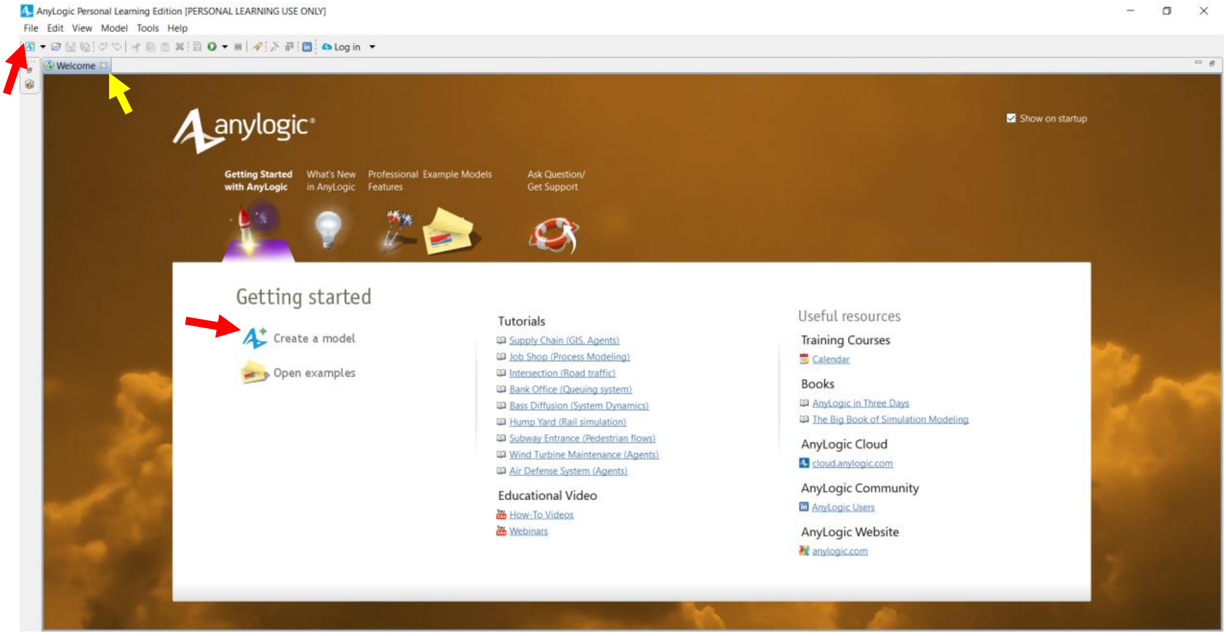
In this workshop you will build a simple simulation model to evaluate health screening for an imaginary disease called simulitis which can lead to blindness.

Simulitis disease process:

- There are 3 disease states: Healthy, Mild and Blind
- You can't go backwards (i.e. get better) and you can't go straight from Healthy to Blind in the same year
- If you are in state Mild, you don't have any symptoms and so you don't know you have the disease
- Disease progression
    - If you are in state Healthy, there is a 30% chance each year that you might you move to Mild (and therefore, a 70% chance that you stay healthy)
    - If you are in state Mild, there is a 20% chance each year that you might you move to Blind, and an 80% chance that you stay in Mild
    - Once you get to state Blind, there is no cure

A. Introduction to AnyLogic 8 User Interface

1. Start AnyLogic and the Welcome page below will be shown. The Welcome page shows a number of useful links that can help you learn how to use AnyLogic such as tutorials, YouTube videos, books, publicly available models in the cloud (good for examples) and AnyLogic user community forum (good for seeking help from other users).

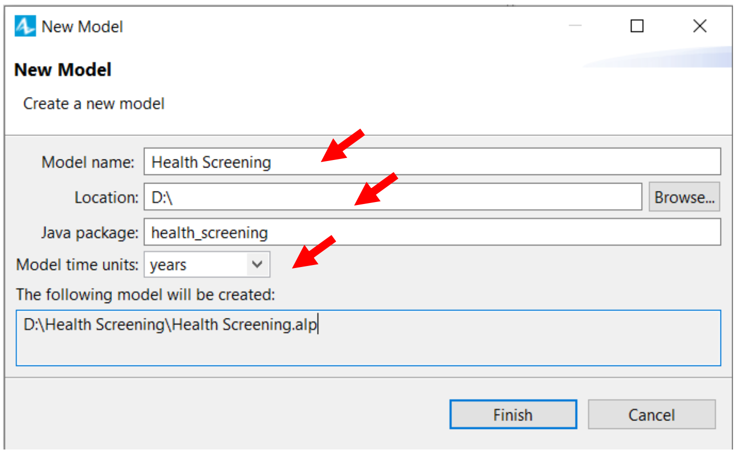


2. You can close the Welcome page by clicking the x icon next to the text “Welcome”.

3. To create a new model, select **File > New > Model** from AnyLogic's main menu or click 

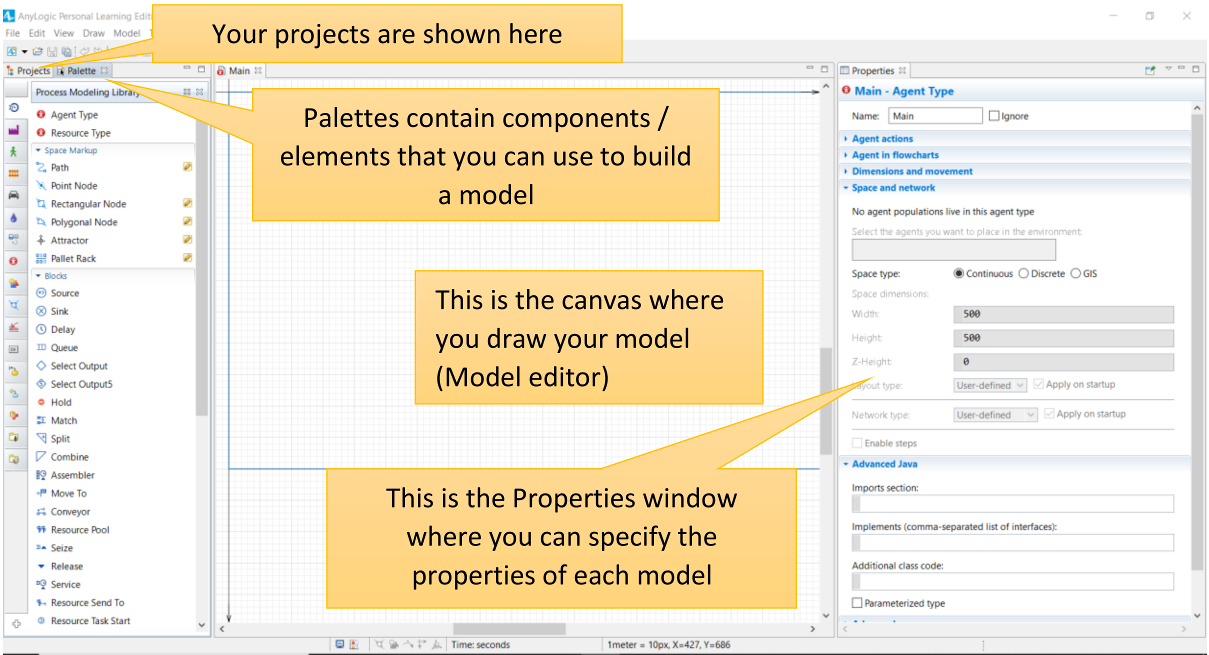
. The New Model wizard will open.

4. Enter the model name as “Health Screening” and specify where you want to store the model (e.g. flash drive, network drive). Set the model time units to years. Click **Finish** to create the blank model.

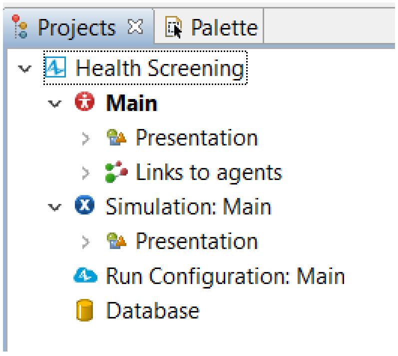


AnyLogic models have the extension .alp.

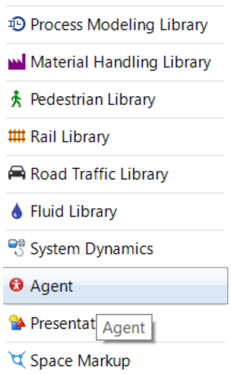
5. The User Interface is as follows.



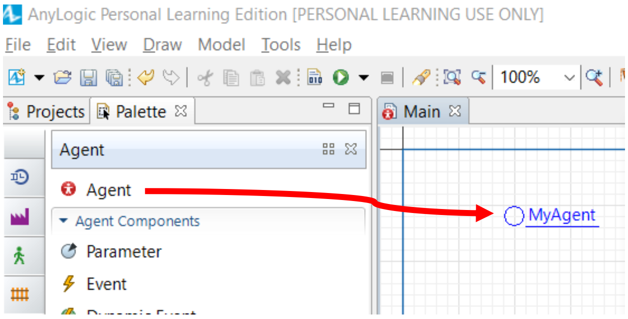
6. Click on **Projects** tab to see the structure of your model. It is organised in a tree-like structure. Currently, you have model Health Screening open. By default, the model comprises four parts (Main, Simulation, Run configuration and Database). Main is the part that will be used by AnyLogic to initialise the simulation (if you know Java/C/C++, this is the same as main()). Simulation and Run Configuration is where you specify the simulation parameters and configuration for running the simulation. Database is used to link your model to a database.



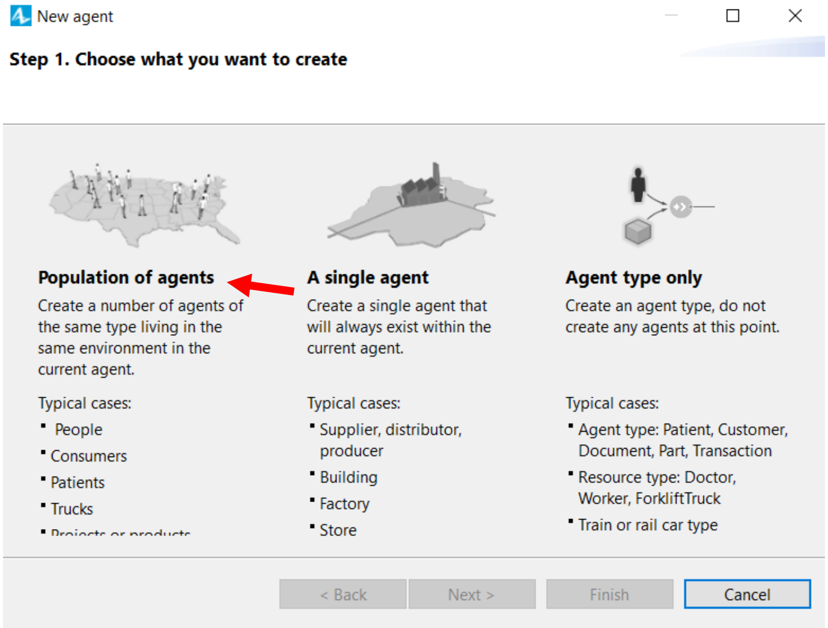
7. Let us start building the model. To do this we need the list of components in the palette. The components are grouped into Libraries in the palette. For example, Process Modelling Library contains elements for Discrete-Event Simulation, System Dynamics Library for the System Dynamics Simulation and Agent Library for Agent-Based Simulation. The elements from different libraries can be combined to form a Hybrid Simulation model.



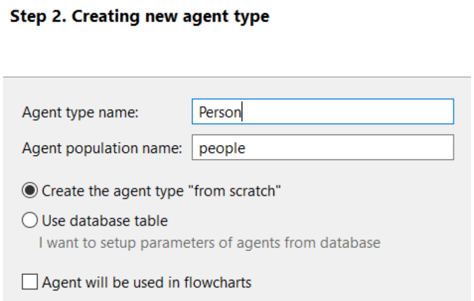
8. In this lab session you will model the Health Screening case. We want to create a population of individuals and simulate the state transition of each individual. Select Agent library on the Palette and drag **Agent** from the palette to the canvas (see the Figure below). This is the method to create one or more agents in the model.



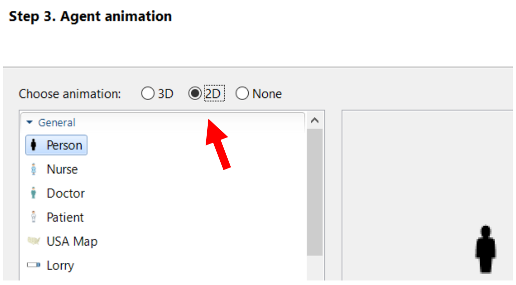
9. In the New Agent wizard, choose “Population of agents” (Step 1) to create multiple agents of the same type.



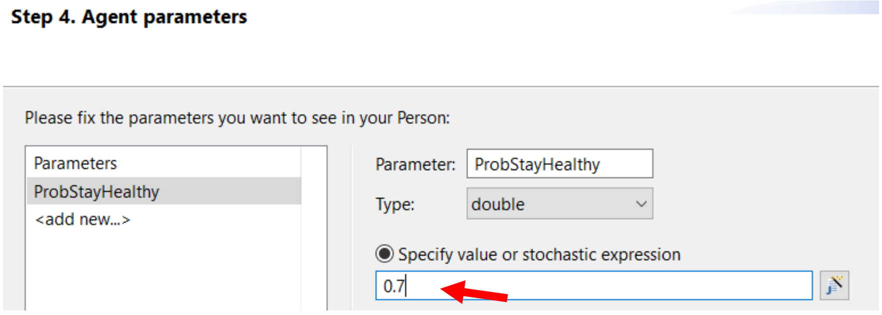
10. Enter “Person” as the agent type name; the agent population name is automatically given.



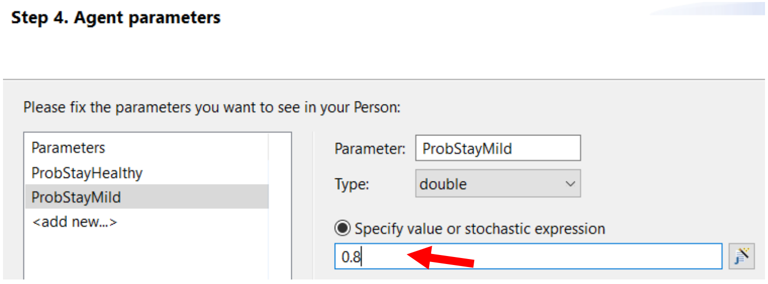
11. Select the visualisation of each agent. Let us choose 2D animation and select person.



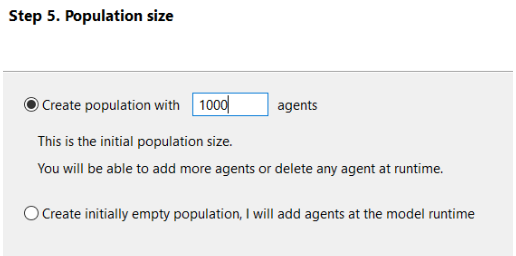
12. Click **<add new …>** to add a new parameter "ProbStayHealthy", type "double" and default value "0.7". This parameter defines the probability that a person stays healthy. By setting this to a constant, we assume that all individuals have the same probability.



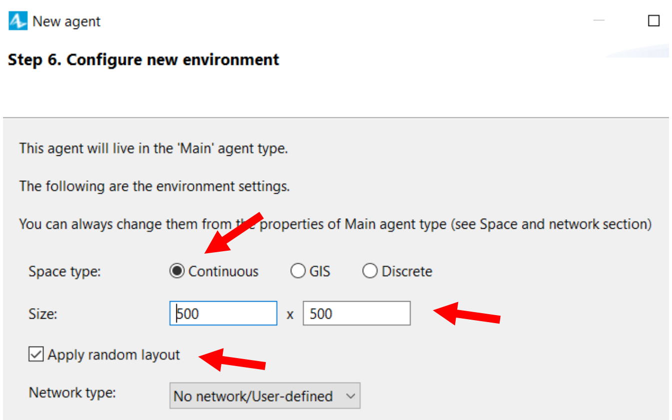
13. Click **<add new …>** again to add another parameter "ProbStayMild", type "double" and default value "0.8". This parameter defines the probability that a person stays in state Mild. By setting this to a constant, we assume that all individuals have the same probability. Click **Next**.



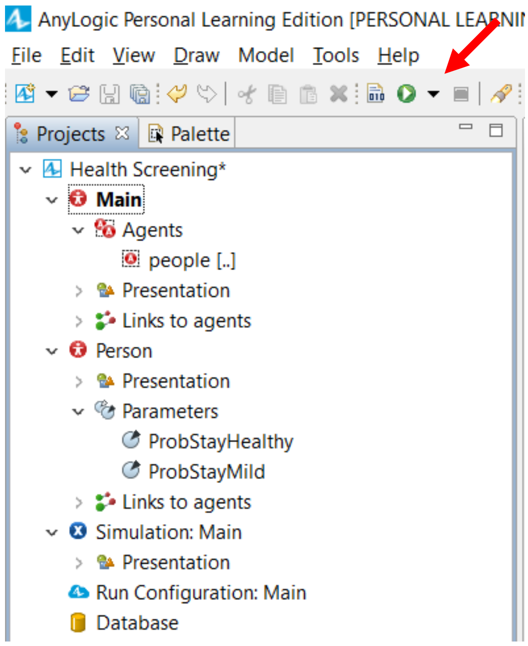
14. Let us now define the number of people. Enter 1000 to create 1000 agents. Click **Next**.



15. At this step, we will define the environment where the agents are located. In this case, we will create a **Continuous** space (i.e. each agent will be located in an (x, y) coordinate). The size of the space is 500 pixels x 500 pixels. Check **Apply random layout** to place the agents at random points in the Euclidean space. Click Finish.



16. We can now run the model. From the Ribbon, click on the small black triangle next to the Run button (see the Figure below). This should list the model(s) that are open. Click on the Health Screening to run the simulation.



You can see the hierarchy of your model here.

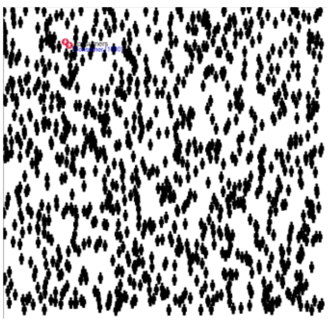
We have created an agent type called Person. Inside each person we have defined two parameters (ProbStayHealthy and ProbStayMild).

When the simulation is run (Main is executed), a collection of agents called people will be created. Each agent is created based on the agent type Person.

Analogy: Agent type = car blueprint, agent = car manufactured using the blueprint; agents = all cars manufactured using the blueprint

17. A new window should be open. You can run the simulation by clicking the play button. Click the play button.

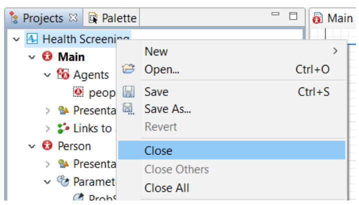
18. You should see 1000 people positioned at random locations within the 500 x 500 grid. At the moment, the agents do not do anything since we have not defined their behaviour.



If you don't see the people in the grid, you must have forgotten to check Apply random layout in step 15. To fix this, double click on the Main canvas. Expand the **Space and network** section in the **Properties** window and select "Random" for Layout Type.

19. Save your file. You can use menu **File > Save**, click the save button in the Ribbon, or press Control-S.

20. Try to close your project. You can use menu **File > Close**, or right click on Health Screening on the Projects tab to open the context menu and choose Close.



Let us recap what you have learned. Up to this stage, you have learned:

- How to create a new model in AnyLogic
- That model elements are grouped into libraries in the palette
- How to add an element into a model
- How to run a model
- How to save your model

B. Introduction to Java for AnyLogic

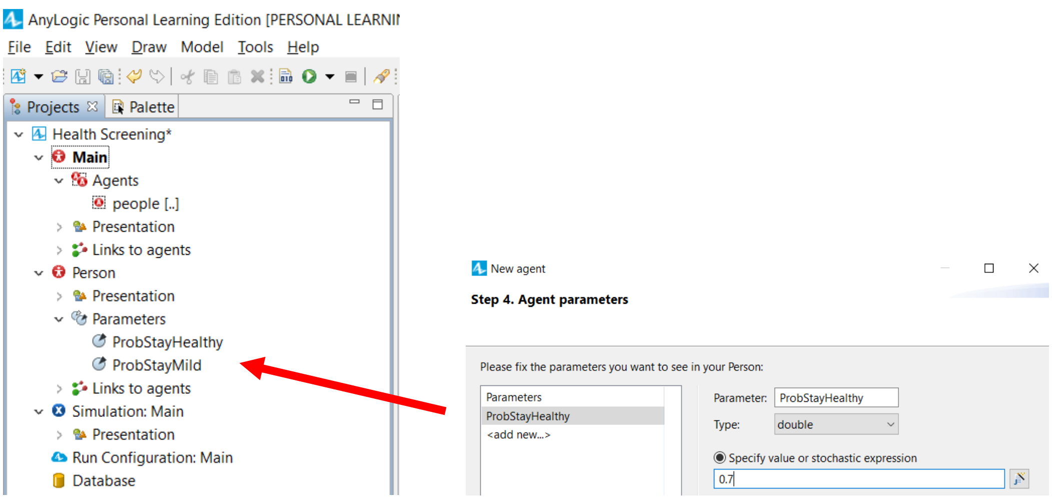
AnyLogic is written in Java. A model developed in AnyLogic is fully mapped into Java code. Hence, if you have a Professional license, you can produce a completely independent standalone Java application of your model. Fortunately, you do not need to write an extensive Java program in AnyLogic. However, except for VERY simple models, you will need to write some Java commands in some parts of the model. Hence, a basic understanding of Java syntax is needed.

B.1 Variables, Parameters and Data Types

As in Mathematics, a variable or parameter is used to store data of a specific data type. Hence, when we declare a variable or parameter, we need to specify the data type. In general the term 'parameter' is used to denote constants, although technically the value stored in a parameter can change. Common data types include:

Data type	Description	Example
Double	Real number	0.1 -123.45 1234578.99
Int	Integer number	256 -127 0 12345
Boolean	Boolean value	true false
String	Text string	"Bogota is great" "simulation 101" "0238059311"

In our example, we have declared two parameters: `ProbStayHealthy` and `ProbStayMild`. Both of them can be used to store double data type. The screen below should remind you of how you declared them earlier.



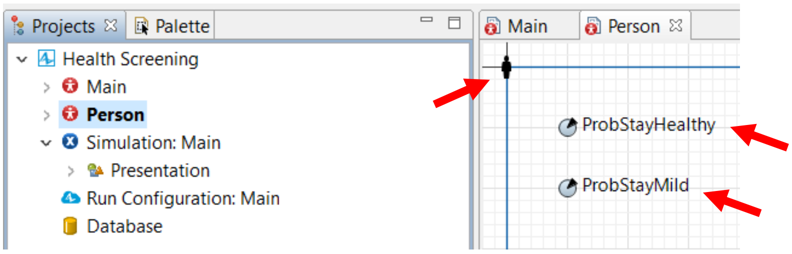
In this case, we declare the two parameters for each agent Person. Hence, we can assign different values to different agents to create a heterogenous population (instead of assigning the same value such as 0.7 to all agents). You will learn how to create a heterogenous population later on.

(We could declare a parameter globally by adding a parameter to Main. All parameters declared in Main are shared by all model components including all agents (hence, they are called global parameters). You will learn this later.)

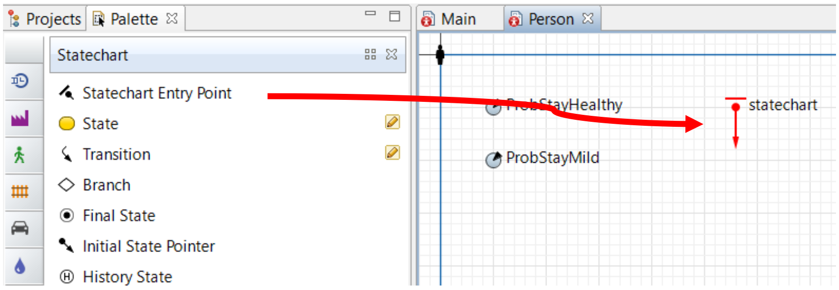
B.2 Functions and Methods

21. Let us open the Health Screening model by selecting menu **File > Open**, pressing **Control O** or clicking the **Open Model** button on the ribbon.

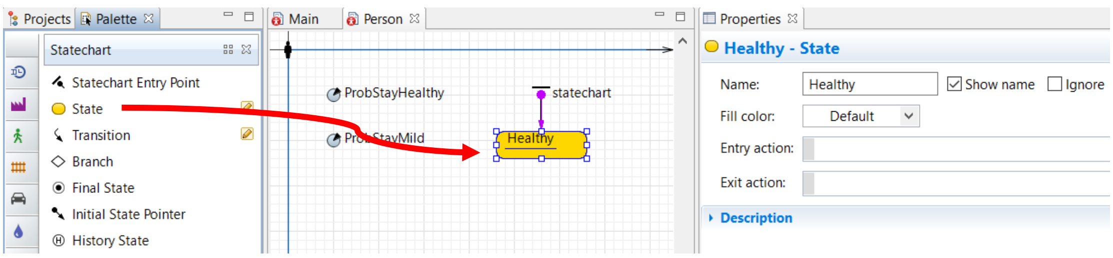
22. On the **Projects** tab, double click on the **Person**. This will open agent type **Person** in the canvas so that we can edit it. Remember that we have added two parameters earlier. These parameters should be shown in the canvas. It should also show the visual representation of the agent that you have chosen earlier.



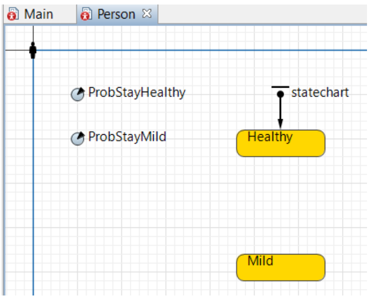
23. Let us define the behaviour of agent type **Person**. On the **StateChart** Palette, drag **Statechart Entry Point** to the canvas for **Person** (NOT **Main**!!!).



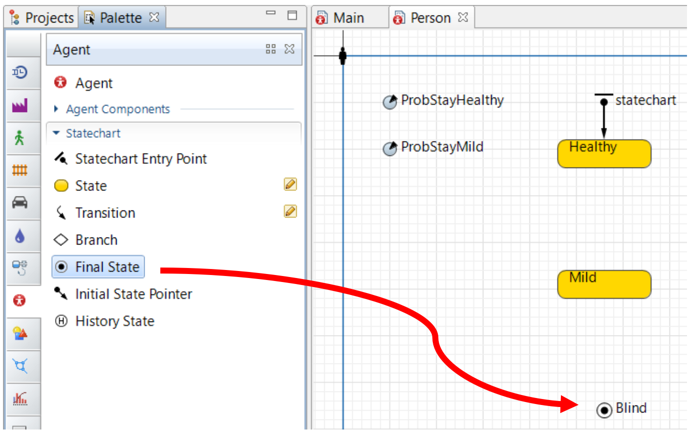
24. Drag **State** to the canvas and make sure it connects to the statechart entry point that you drew earlier. Rename the state as Healthy. The combination of statechart entry point and state will define the initial state of the agent when it is created during the simulation (i.e. all people are healthy at the start of the simulation).



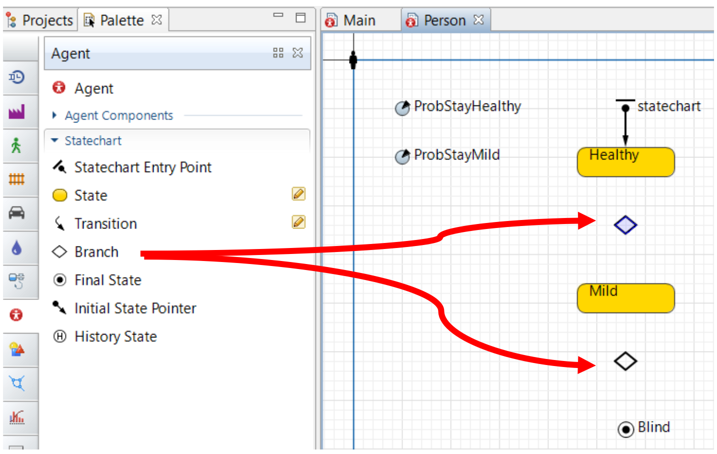
25. Drag another **State** to the canvas and name it Mild.



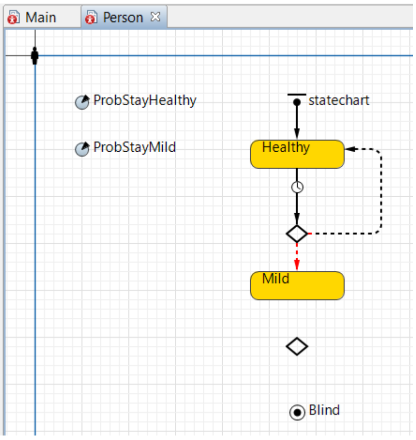
26. Drag a **Final State** to the canvas and name it Blind. A final state is a state in which once an agent enters it, the agent cannot move to another state.



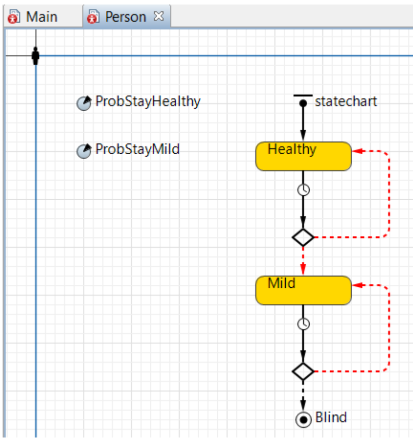
27. Drag two **Branch**es to the canvas.



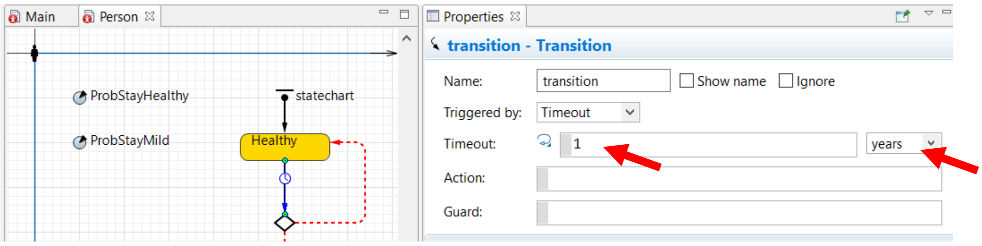
28. Let us define the state transition, as mentioned in the case description.
    a. Double click on Transition to enable drawing (the pencil button next to it should disappear) and then click on Healthy, followed by a click on the upper branch. You can move the ends of the transition to suit your taste but make sure they are still connected.
    b. Double click on Transition again, draw a transition between the upper branch and Mild.
    c. Double click on Transition again, draw a transition between the upper branch to Healthy. To create a bend on the line, you need to double click on the line. A small circle will appear so that you can drag it to bend the line.



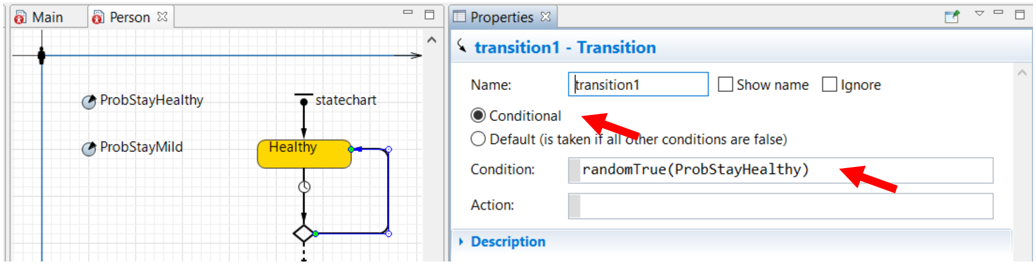
29. Repeat the previous step for the transitions between Mild and Blind. The complete state diagram should look like below.



30. Click on the transition between Healthy and the upper branch. By default, the trigger for the transition is Timeout (you will learn all transition types later). Let us specify the duration of the timeout as one year. This means that after one year the Person will move to one of the outgoing transitions (i.e. stay Healthy or move to Mild).

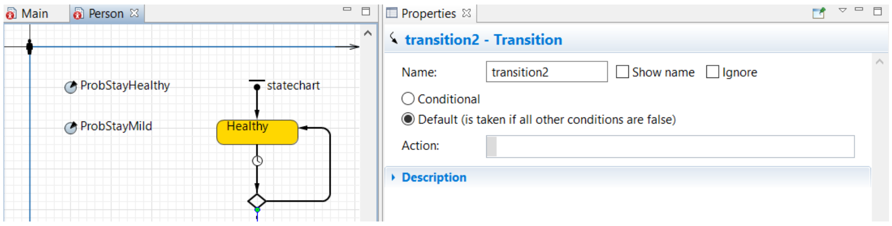


Click on the transition between the branch and Healthy. Keep the default name, i.e. do NOT change whatever name that is given to you (the name depends on the sequence when you draw them; hence it may not be the same as mine). Select Conditional and specify the Condition as `randomTrue(ProbStayHealthy)`. This means that this transition will be taken when the condition is true which depends on the probability to stay in Healthy state.

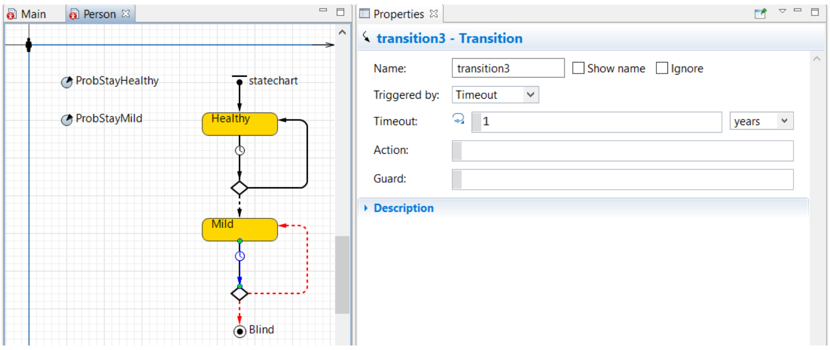


One of the built-in functions in AnyLogic is *randomTrue(x)* which returns a Boolean value "true" or "false". When the function is called, the computer will pick a value between 0 and 1 at random and then check whether this number is smaller than *x*. If so, it returns "true" otherwise "false".

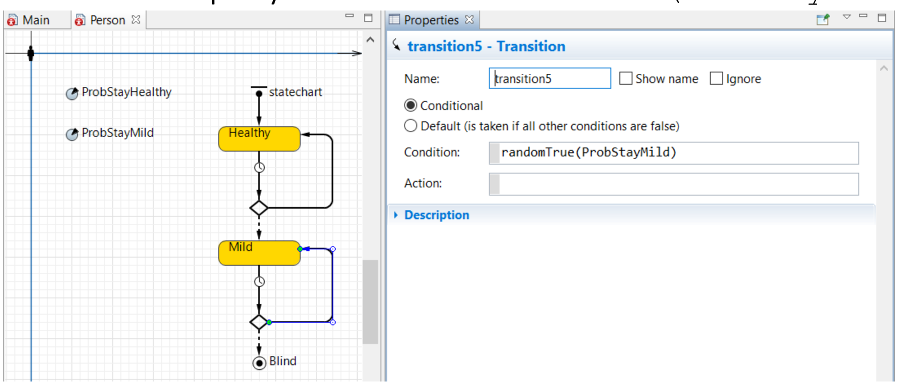
31. Click on the transition from the upper branch to Mild. Keep the default name. We can keep the default setting, i.e. this transition will only be taken if the other transition is not taken.



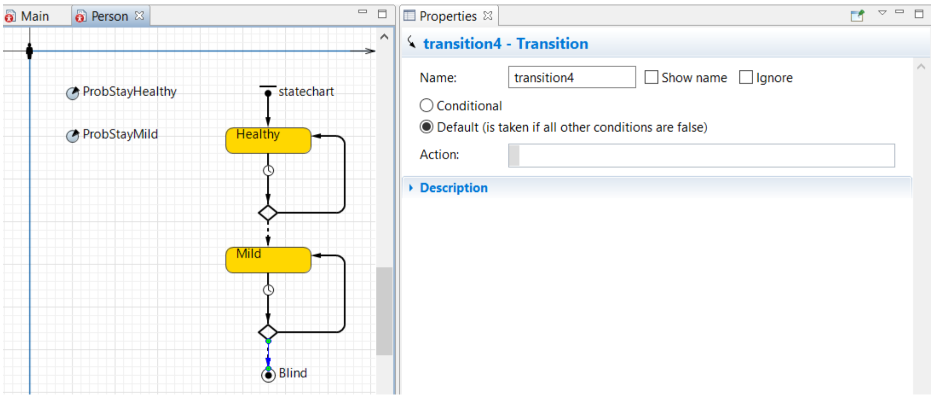
32. Click on the transition between Mild and the lower branch. Let us specify the duration of the timeout as one year. This means that after one year the Person will move to one of the outgoing transitions (i.e. stay Mild or move to Blind).



33. Click on the transition between the lower branch and Mild. Keep the default name. Select Conditional and specify the Condition as `randomTrue(ProbStayMild)`.

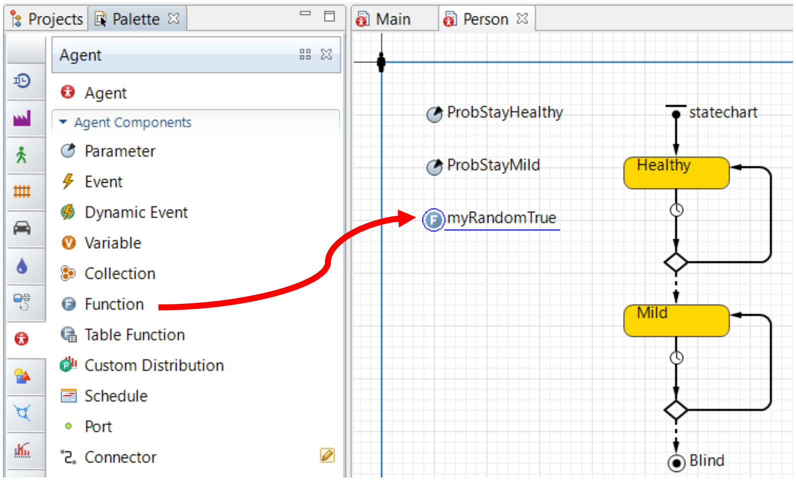


34. Click on the transition from the lower branch to Blind. Keep the default name. We can keep the default setting, i.e. this transition will only be taken if the other transition is not taken.

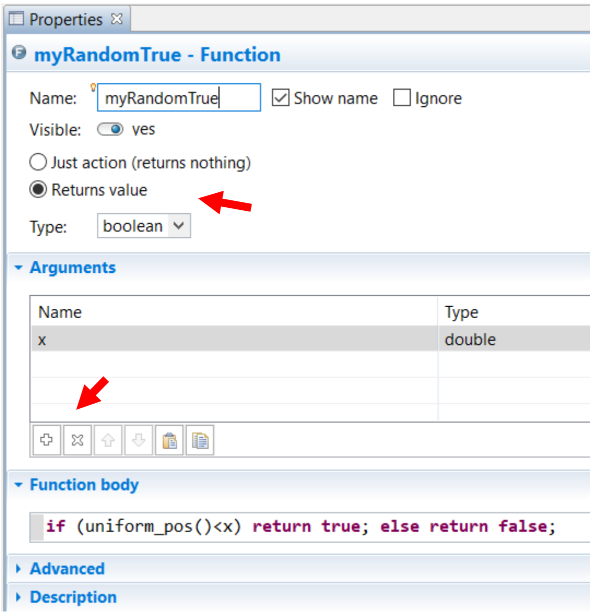


35. OPTIONAL: If you are interested to know more about creating your own function, you can try this step; otherwise, you can go directly to the next step.

    a. We can create our own function. Let say we want to replicate the built-in function randomTrue. To do this, drag **Function** from **Agent** Palette to the **Person** canvas.

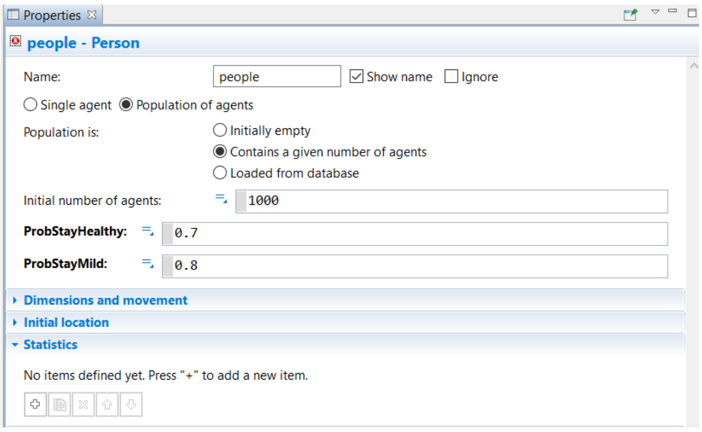


    b. Name the function **myRandomTrue**. We want the function to return a Boolean value (a function that does not return a value is called Method).
    c. Expand the **Arguments** section. Click the + button to add a parameter x and the type is double.
    d. Expand the **Function body** section. Type command `if (uniform_pos() < x) return true; else return false;`
    e. The function property should look like the figure at the top of p13. Here, we create a function myRandomTrue that accepts one parameter and returns true if the built-in function `uniform_pos()` return a value that is less than x, otherwise, it returns false. Function uniform_pos returns a random value between 0 and 1. *Please note that java requires a semicolon (;) to mark the end of a statement*. The most common mistake among AnyLogic learners is not to know when to use semicolon and when not to use it.

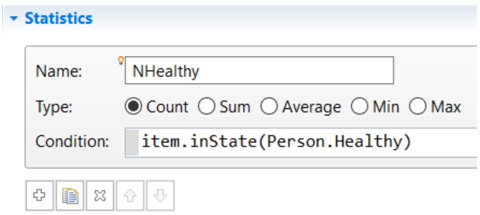


f. Now, you can either use randomTrue or myRandomTrue in your model. Both are identical. In practice, you will only create your own function if it is not available. Here, we simply want to show how create your own function.

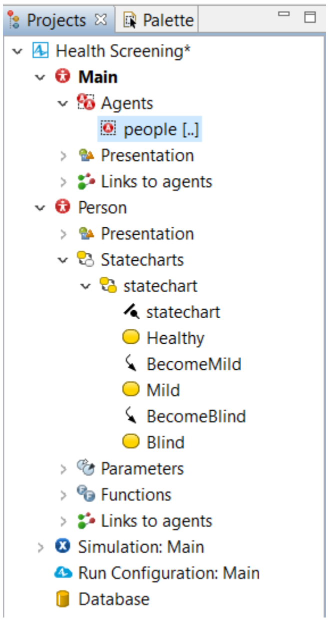
36. Now that we have defined the complete state transition diagram of agent type Person, we need a way to calculate the number of agents in various states. Click on **Main** tab and click on **people** in the canvas. You should see the following form in the **Properties** window.



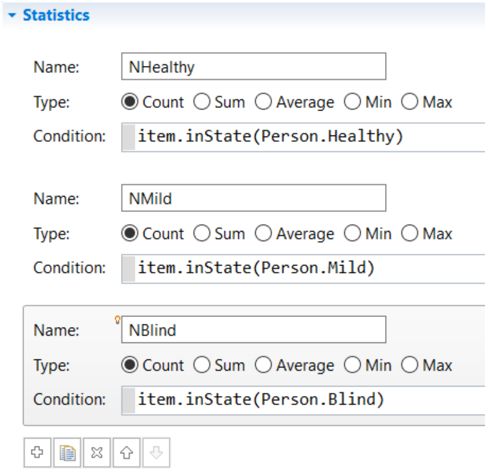
37. Expand the Statistics section, click + button to add new metric called NHealthy. This metric is calculated by counting the number of people in state Healthy. This is done by entering the detail below.



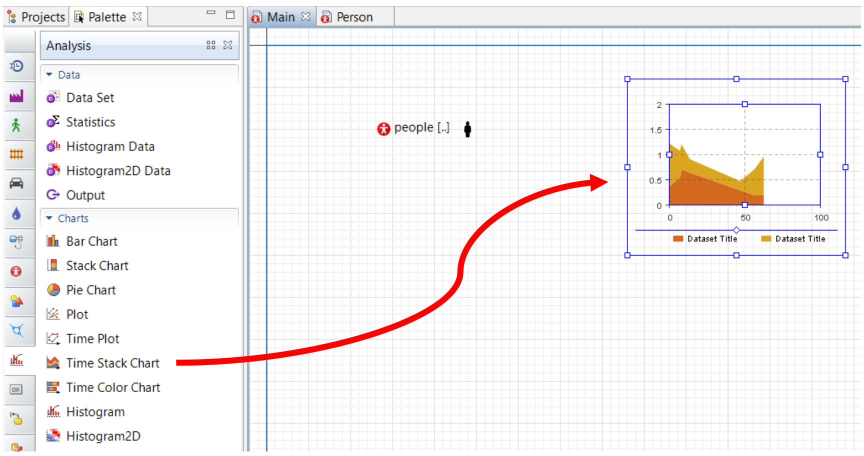
To understand how it works, we can see the model hierarchy in the **Projects** tab. Remember we define the statistics for people inside Main (see the figure on the right). The word `item` refers to a person in the population. The function `inState(S)` returns true if the item is in state S. Remember that we define three states in agent type Person (see the figure on the right): Healthy, Mild and Blind. Hence, we can replace S with `Person.Healthy` to refer to state Healthy. We use `Person` because the state Healthy is defined inside the agent type Person. Likewise, previously we used `item` because instate is a built-in function defined inside a built-in class `item`. The type **Count** is used to count the number of agents in the condition we specify in **Condition**. Note that we do not use semicolon because it is only a condition (not a statement).

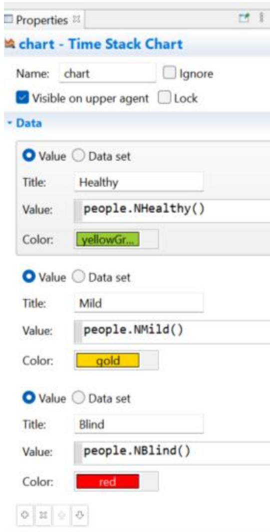


38. Add two more statistics for NMild and NBlind. The complete statistics are shown below.



39. Having defined the statistics, we can now show the statistics. One way of doing it is by visualising it. Let us drag **Time Stack Chart** from **Analysis** palette to the **Main** canvas.



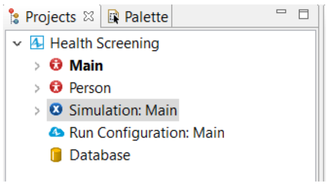


40. While the chart is selected, expand the **Data** section in the **Properties** window. This is where we define the timeseries data to be displayed. In this case, we want to display statistics NHealthy, NMild and NBlind that we have specified earlier. The complete Data section should look like the figure on the right. Use + button to add more data to display. You can change the colours according to your preference.

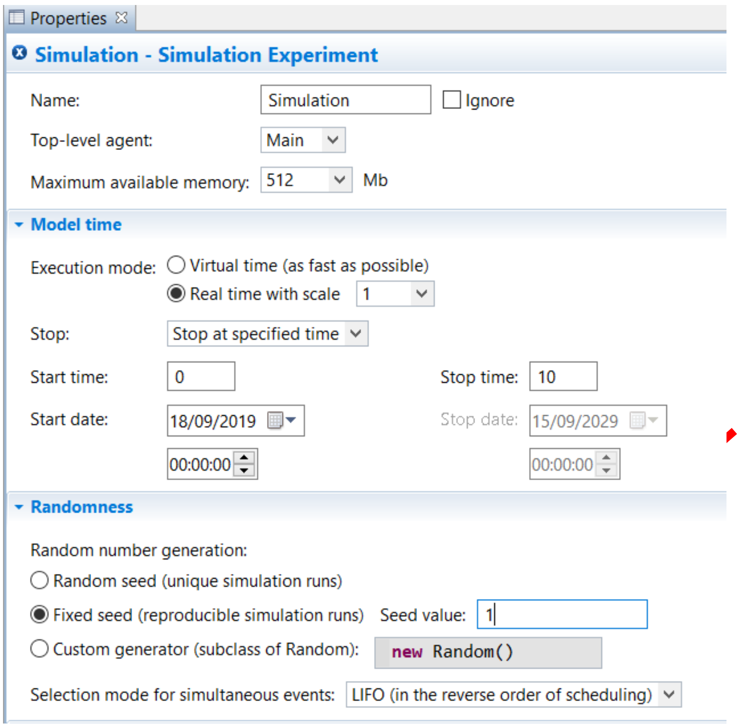
41. Expand the **Data Update** section. By default, the chart will show up to 100 latest samples and the data are updated every year from year 0. We need to change so that the chart will show up to 11 latest samples.

42. Expand the **Scale** section. Change the **Time Window** to 10 model time units.

43. Now, let us define the stopping condition for our simulation Click on **Projects** tab. Select **Simulation: Main**.

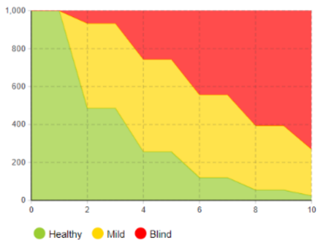


On the **Properties** window, expand the **Model time** section. Change **Stop** to **Stop at specified time**. Set the **Stop time** to 10. This means that we will run the simulation for 10 years.



'Random' numbers are generated by a deterministic algorithm. Hence, they are not truly random, but they appear to be: they pass statistical tests for randomness). The same seed value will generate exactly the same sequence of numbers. Hence it is common practice to set different seed values for each simulation run. You can try by changing the seed value to a different number before you run the simulation. It is also possible to do this automatically.

44. Run the model, and observe the proportion of healthy, mild and blind after 10 years. Please note that this only shows the result of one simulation run. You MUST NOT base your decision on one simulation run, just as you must not make a decision based on one sample.



Let us recap what you have learned. Up to this stage, you have learned:

- How to model behaviour using State Transition Diagram (or State Chart)
- How to create statistics
- How to add charts to visualise model outputs
- Basic Java concepts: variables, functions, and methods
- AnyLogic model hierarchy

Other resources:

To learn more about AnyLogic, down load the free book AnyLogic in Three Days. Please note that some features are not available in the PLE version.

To learn more about Java Programming for AnyLogic, see https://anylogic.help/advanced/code/general.html.

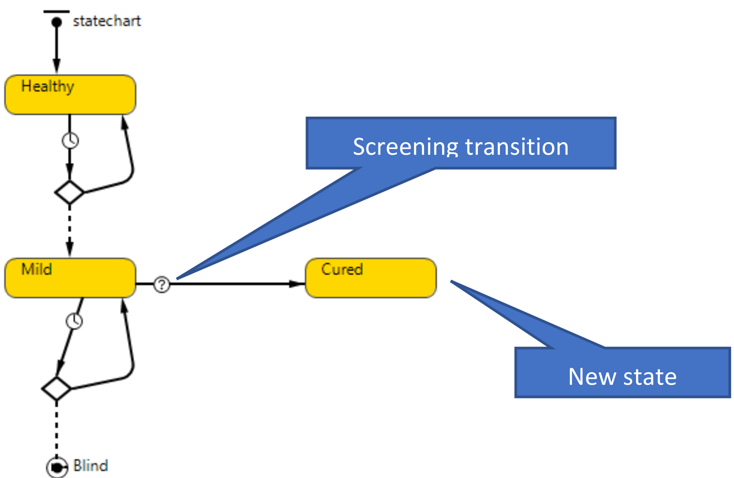
C. Including screening in the model

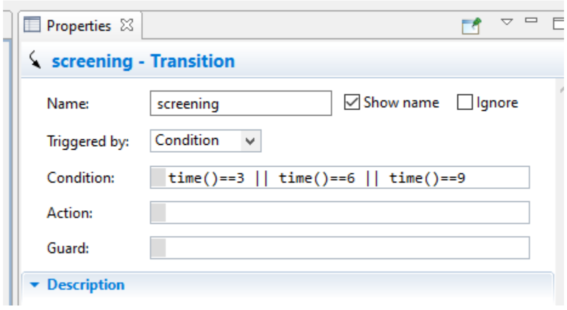
We now want to implement health screening in the model. We assume people get people get screened in years 3, 6, 9. If they are in state mild when they get screened, they will be permanently cured.

To implement this we need to make two changes to our model:

A. We need to introduce a new state "cured"

B. We need to define the transition to that new state (due to screening)





Here:

`time()` is a function which returns the current model time,

"==" means `equal`

"||" means `or`

Now you can run the model and see the effect of screening.

As an exercise, add code to add up the total number of cured people. You will also need to collect a new statistic (in people) and change the graph.

<u>And finally: Heterogeneity of agents</u>

So far we have assumed that all people are the same. Let's now consider the case that 20% of people are controlling their risk factors (maybe through medication). If risk factors are controlled the probability of going from healthy to mild reduces, i.e. the probability to stay healthy increases.

To implement this we need to make four changes to the model:

A. Add a new parameter indicating whether or not risk factors are controlled

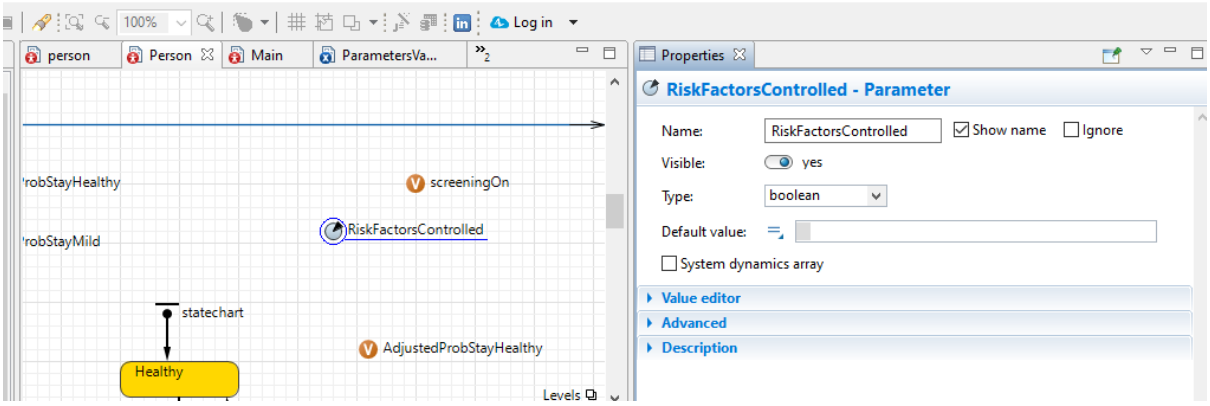
B. Assign the value "True" for risk factor control to 20% of people

C. Adjust Probability to Stay Healthy depending on risk factor control

D. Make the transition from Healthy back to Healthy depend on the adjusted probability

A. Add new parameter to person

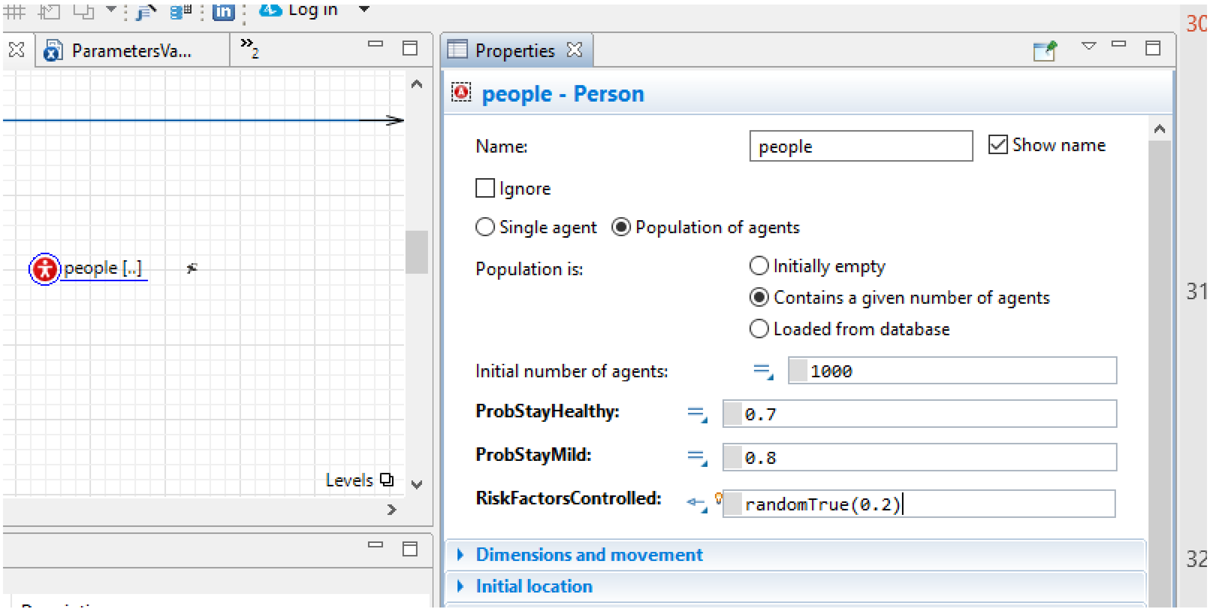
We add a new parameter RiskFactorsControlled to the person (make sure you do this on the person, not the Main level). We define this parameter as Boolean which means it can only take the values true or false.



B. Randomly assign values to new parameter

This has to be done in the properties of people (on Main).

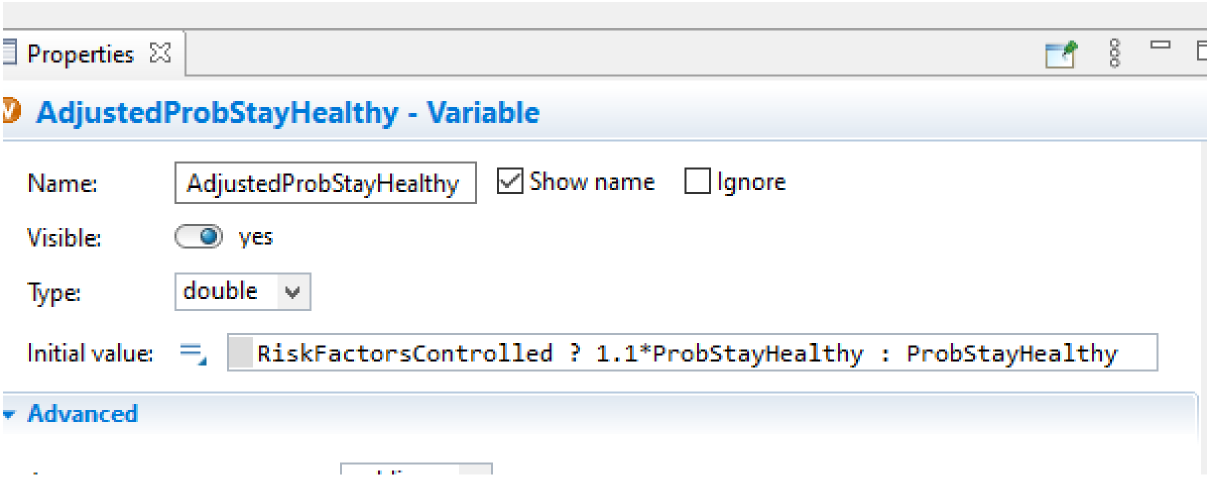
Use the function randomTrue to assign the value True for 20% of people and the value False for 80%.



C. Adjust probability depending on whether risk factors are controlled.

We need to define a new variable for a person to represent the adjusted probability of staying healthy, depending on whether or not the risk factors are controlled.

For our model we assume that the probability to stay healthy increases by 10% if risk factors are controlled.



Here we use the conditional operator "?".

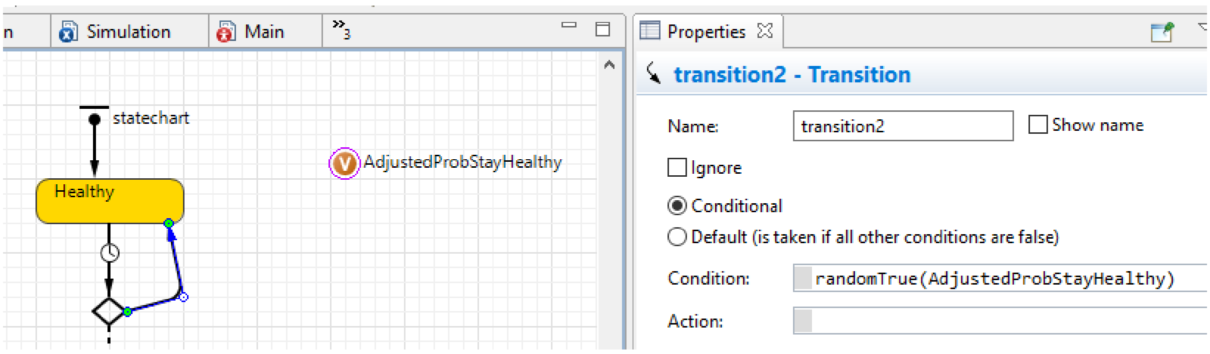
The conditional operator ? can be used to change the value of an expression, variable or parameter depending on a condition

<condition> ? <value if true> : <value if false>

In our model this is useful to change the probability of somebody staying healthy depending on a property of the person

D. Change transition condition

Finally, we can change the transition condition back to the state healthy so that it depends on this adjusted probability.



Now run the model and see the impact.

If you finish early and want to try to build the supply chain model from the lecture, go ahead! We will send you the completed models later on this week.