

Debugging in AnyLogic

Nathaniel Osgood

CMPT 858

4-5-2011

Avoiding Debugging

- Defensive Programming
- Offensive Programming

Offensive Programming: Try to Get Broken Program to Fail Early, Hard

- Asserts: Actually quit the program
- Fill memory allocated with illegal values
- Fill object w/illegal data just before deletion
- Set buffers at end of heap, so that overwrites likely trigger page fault
- Setting default values to be illegal in enums
- We will talk about Assertions & Error Handling later this week

Assertion Goal: Fail Early!

- Alert programmer to misplaced assumptions as early as possible
- Benefits
 - Documents assumptions
 - Reduces likelihood that error will slip through
 - Helps discourage “lazy” handling of only common case
 - Forces developer to deal explicitly with bug before continuing
 - Reduces debugging time
 - Helps improve thoroughness of tests

Avoid Side Effects in Assertions

- Because assertions may be completely removed from the program, it is unsafe to rely on side effects occurring in them

~~assert ++i < max;~~

Arnold et al. The Java Programming Language, Fourth Edition. 2006.

Enabling Assertions in AnyLogic

The screenshot displays the AnyLogic Advanced interface. The main workspace shows a state transition diagram for a 'Person' agent. The 'PregnancyStatus' state is currently set to 'NonPregnant'. The diagram includes transitions for becoming pregnant and returning to a non-pregnant state. A list of methods is visible on the right, including 'FertilityRateAgeSexEthnicity', 'PerformBirth', and 'FinalizeDeath'.

The 'Properties' window at the bottom is open to the 'Simulation - Simulation Experiment' tab. Under the 'Advanced' section, the 'Java Machine Arguments' field is highlighted with a red oval and contains the text `-enableassertions`. Other fields include 'Maximum Available Memory' (1024 Mb) and 'Command-line Arguments'.

Enabling Assertions in Java

- 2 ways
 - Usual: Via java runtime command line
 - enableassertions/-ea[*descriptor*]
 - e.g.
 - enableassertions:com.acme.Plotter
 - enableassertions:com.acme...
 - disableassertions/-da[*descriptor*]
 - Less common: via reflection (ClassLoader)
 - public void **setDefaultAssertionStatus(boolean enabled)**
 - public void **setPackageAssertionStatus(String packageName, boolean enabled)**
 - public void **setClassAssertionStatus(String className, boolean enabled)**

Debugging AnyLogic

- Debugging is the process of locating the faults behind observed failures
- AnyLogic's education now contains a debugger
- You can attach to AnyLogic from debuggers such as eclipse
 - The key thing is to set anylogic to use a port

Debugging Options

- Using output for manual tracing & reporting
- Using AspectJ & tracing
- Using an external debugger (e.g. via eclipse)
- Using AnyLogic Professional/Research debugger

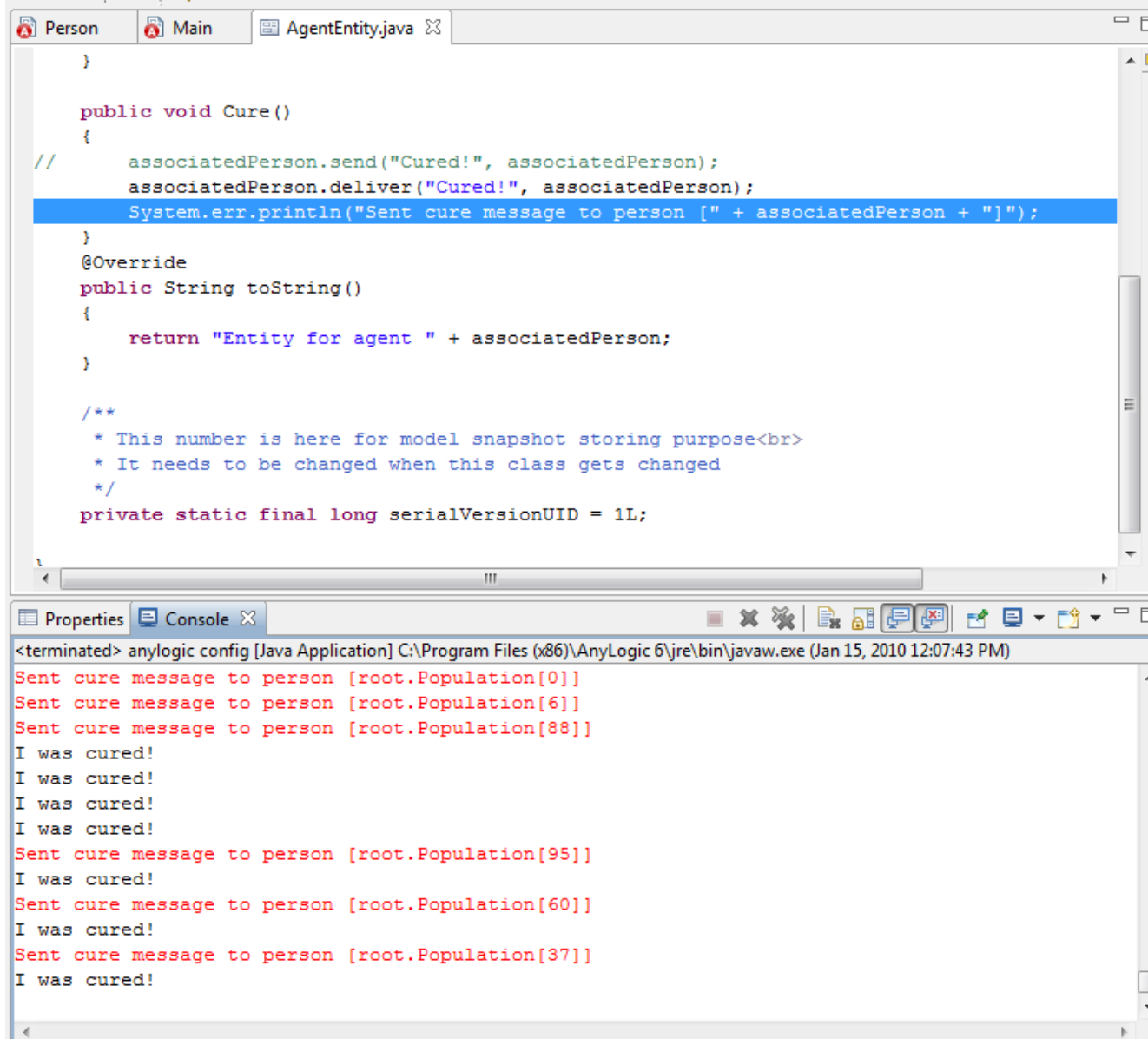
Using output for manual tracing & reporting

- Pros
 - Minimal learning curve
 - Flexible
 - Easily targeted
- Cons
 - Requires time-consuming manual
 - “markup”
 - de-markup
 - Can require many build/simulation iterations to localize problem
 - Limited capacity of console

Output to the Console: How To

- `System.err.println(String)`
 - `System.err.println("Sent cure message to person [" + associatedPerson + "]);`
- `println(String)`
- `System.out.println(String)`

Use in AnyLogic



The screenshot displays an IDE window with two tabs: 'Person' and 'Main'. The 'AgentEntity.java' file is open, showing the following code:

```
    }  
  
    public void Cure()  
    {  
        // associatedPerson.send("Cured!", associatedPerson);  
        associatedPerson.deliver("Cured!", associatedPerson);  
        System.err.println("Sent cure message to person [" + associatedPerson + "]);  
    }  
    @Override  
    public String toString()  
    {  
        return "Entity for agent " + associatedPerson;  
    }  
  
    /**  
     * This number is here for model snapshot storing purpose<br>  
     * It needs to be changed when this class gets changed  
     */  
    private static final long serialVersionUID = 1L;
```

The console output shows the following messages:

```
<terminated> anylogic config [Java Application] C:\Program Files (x86)\AnyLogic 6\jre\bin\javaw.exe (Jan 15, 2010 12:07:43 PM)  
Sent cure message to person [root.Population[0]]  
Sent cure message to person [root.Population[6]]  
Sent cure message to person [root.Population[88]]  
I was cured!  
I was cured!  
I was cured!  
I was cured!  
Sent cure message to person [root.Population[95]]  
I was cured!  
Sent cure message to person [root.Population[60]]  
I was cured!  
Sent cure message to person [root.Population[37]]  
I was cured!
```

AspectJ and Eclipse

- AspectJ is a language that allows for succinctly describing “cross cutting” functionality in programs – such as tracing or logging requests
- AspectJ can automatically insert tracing instrumentation into our code
 - This gives us many of the benefits of manual tracing program execution without the need for the markup & mark-down work

AnyLogic Built-in Debugger

Running the Debugger

The screenshot displays the AnyLogic Professional software interface. The main window shows a Java code editor with the following code:

```
package sir_agent_based_networks;  
import java.sql.Connection;  
import java.sql.SQLException;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.Calendar;  
import java.util.Collection;  
import java.util.Collections;  
import java.util.Comparator;  
import java.util.Currency;  
import java.util.Date;  
import java.util.Enumeration;  
import java.util.HashMap;  
import java.util.HashSet;  
import java.util.Hashtable;  
import java.util.Iterator;  
import java.util.LinkedList;  
import java.util.List;  
import java.util.ListIterator;  
import java.util.Locale;  
import java.util.Map;  
import java.util.Random;  
import java.util.Set;
```

The interface includes several panels:

- Projects:** A tree view showing the project structure, including "SIR Agent Based Networks" and sub-projects like "Main", "Parameters", "Functions", "networkTypeToString", "Environments", "Embedded Objects", "Analysis Data", "Presentation", "Person", "Simulation: Main", and "MonteCarlo: Main".
- Palette:** A panel on the right containing various modeling elements such as "General", "Parameter", "Event", "Dynamic Event", "Plain Variable", "Collection", "Function", "Table Function", "Port", "Connector", "Environment", "System Dynamics", "Statechart", "Actionchart", "Analysis", "Presentation", "3D", "Controls", "Connectivity", "Pictures", "3D Objects", "Enterprise Library", "Pedestrian Library", "Rail Yard Library", and "Palettes...".
- Console:** A window at the bottom showing the command prompt output: "<terminated> anylogic config [Java Application] C:\Program Files (x86)\AnyLogic 6 Professional\jre\bin\javaw.exe (Nov 1, 2010 1:19:13 AM)".
- Properties and Console:** A panel at the bottom left showing a table with columns "Description" and "Locat..".

The status bar at the bottom indicates "Read-Only", "Smart Insert", and "16 : 26".

Running the Models

The screenshot displays the AnyLogic Professional software interface. The main window is titled "SIR Agent Based Simulation - AnyLogic Professional [EVALUATION USE ONLY]". The central area shows the "SIR Agent Based Model of Disease Diffusion" configuration screen. On the left, a "Model parameters" table lists various settings, and on the right, a "Description" section provides details about the model's states and transitions. At the bottom, a status bar shows the simulation is in "Idle" mode with 0 EPS, 0 FPS, and 10M of 297M memory used.

Model parameters

Total population:	200
Fraction initially infected:	0.05
Contact rate:	5.0 contacts per day
Infectivity:	0.05
Average illness duration:	15.0 days
Layout type:	Ring
Network type:	Random
Links per agent:	10
Maximum link distance:	50.0
Percent of long distance links:	0.05
M:	2

Description

We distinguish between three different states of a person: Susceptible, Infectious and Recovered. To reflect the fact that some people are already infected at the beginning of the simulation, the agent statechart entry point has a branch InitiallyInfected. The choice of the initial state is probabilistic and is based on a global model parameter PercentInitiallyInfected.

The transition to the state Infectious models the event of disease being passed to the agent from a sick person. In the model terms the trigger of that transition is a message "infection". Once in the Infectious state, the agent is able to pass the disease to others, therefore we are interested in his contacts. We assume that ContactRate is constant while in the Infectious state, and we can use internal transition Contact that will be repeatedly taken until the agent recovers. In the action of that transition the agent chooses another agent from the people that he knows (this is defined by the social network), and sends him the message "infection". However, as not every contact results in infection being passed, message is sent with the probability InfectionProbability, which is yet another global parameter. If the message reaches an agent who is already infectious or recovered, it is ignored. Finally, the transition Recovery is a timeout that should model the illness duration.

This model allows you to explore the dynamics of disease diffusion in these types of network:

- Based on distance – people are linked if the (geographical) distance between them is not more than a given value.
- Random – a person is linked to a random subset of the population.
- Small World (Watts 1999) – most links are neighbors, but there is a certain percent of long-range links. To establish neighborhood people can be placed e.g. on a virtual ring
- Scale Free (Barabasi, Albert 1999) – some people are "hubs" with lots of connections and some are "hermits". This type of network is build using a preferential attachment algorithm where a probability of a new person links to the existing people is proportional to the number of links those people already have.
- Ring Lattice – each person is linked to a fixed number of his closest neighbors on a ring. This type of network is most distant to the fully connected network.

Run the model

anylogic config [Java]

Run: 0 | **Idle** | **Step:** - | **EPS:** 0 | **FPS:** 0.0 | **Memory:** 10M of 297M | 0.0 sec

Setting a Breakpoint

The screenshot displays the AnyLogic Professional software interface. The main window shows a grid-based workspace with the text "SIR Agent Ba" visible. The interface is divided into several panels:

- Debug**: Located on the left, it shows a list of threads including "Thread [AWT-Windows] (Ru)", "Thread [AWT-Shutdown] (Ru)", "Thread [AWT-EventQueue-0]", "Thread [AnyLogic simulation]", "Thread [DestroyJavaVM] (Ru)", and "Thread [AnyLogic presentati]".
- Breakpoints**: Located at the top left, it shows a breakpoint set on the "networkTypeToString : Function - Body [relative line: 1]".
- Variables** and **Expressions**: Located at the top right, these panels are currently empty.
- Palette**: Located on the right, it contains various components such as "General", "Parameter", "Event", "Dynamic Event", "Plain Variable", "Collection", "Function", "Table Function", "Port", "Connector", "Environment", "System Dynamics", "Statechart", "Actionchart", "Analysis", "Presentation", "3D", "Controls", "Connectivity", "Pictures", "3D Objects", "Enterprise Library", "Pedestrian Library", "Rail Yard Library", and "Palettes...".
- Properties** and **Console**: Located at the bottom left, they show the properties and console output for the selected function.
- Code**: Located at the bottom right, it displays the source code for the "networkTypeToString - Function".

```
Function body:  
switch( type ) {  
  case Environment.NETWORK_USER_DEFINED:  
    return "Custom";  
  case Environment.NETWORK_RANDOM:  
    return "Random";  
  case Environment.NETWORK_ALL_IN_RANGE:  
    return "Based on distance";  
  case Environment.NETWORK_RING_LATTICE:  
    return "Ring lattice";  
  case Environment.NETWORK_SMALL_WORLD:  
    return "Small world";  
  case Environment.NETWORK_SCALE_FREE:  
    return "Scale free";  
  default: return "Unknown";  
}
```

The status bar at the bottom indicates "Selection" and "X=-489, Y=308".

When we Hit the Breakpoint...

The screenshot displays the AnyLogic Professional software interface during a debugging session. The main window shows the source code of the `networkTypeToString` function in `Main.java`. A breakpoint is set at the beginning of the `switch` statement, and the execution has stopped at this point. The `Variables` window shows the current state of the function, with `this` pointing to the `Main` object and `type` set to `1`. The `Properties` window at the bottom shows the configuration for the `networkTypeToString` function, including its name, access level, and return type (`String`).

Breakpoints

- networkTypeToString : Function - Body [relative line: 1]

Variables

- this (sir_agent_based_networks.Main) : (id=46)
- type (int) : 1

```
String
networkTypeToString( int type ) {
switch( type ) {
case Environment.NETWORK_USER_DEFINED:
return "Custom";
case Environment.NETWORK_RANDOM:
return "Random";
case Environment.NETWORK_ALL_IN_RANGE:
return "Based on distance";
case Environment.NETWORK_RING_LATTICE:
return "Ring lattice";
case Environment.NETWORK_SMALL_WORLD:
return "Small world";
case Environment.NETWORK_SCALE_FREE:
return "Scale free";
}
```

Properties - networkTypeToString - Function

General

Name: networkTypeToString Show name Ignore Public Show at runtime

Code

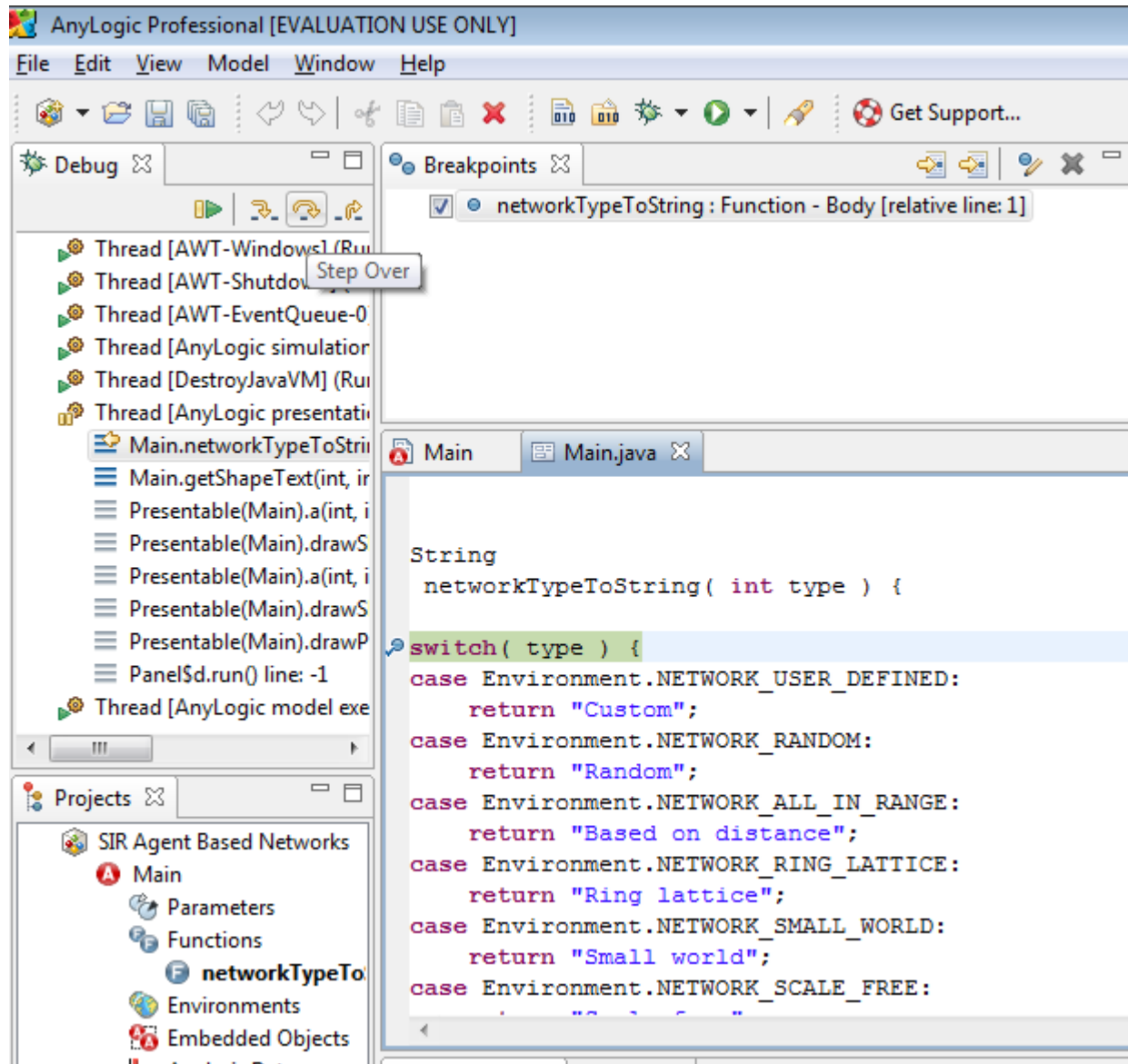
Access: default Static

Return type: void boolean int double String Other: String

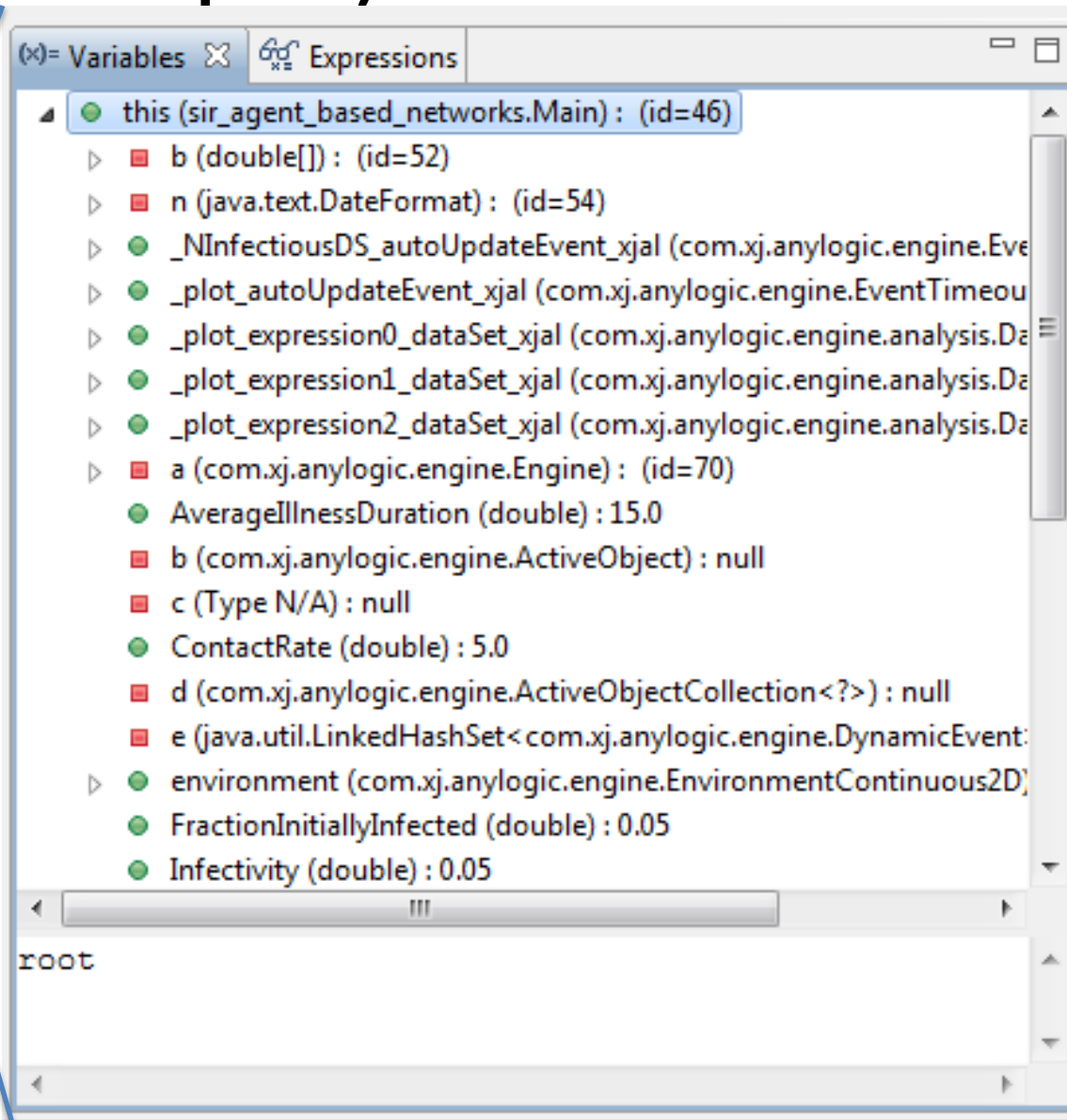
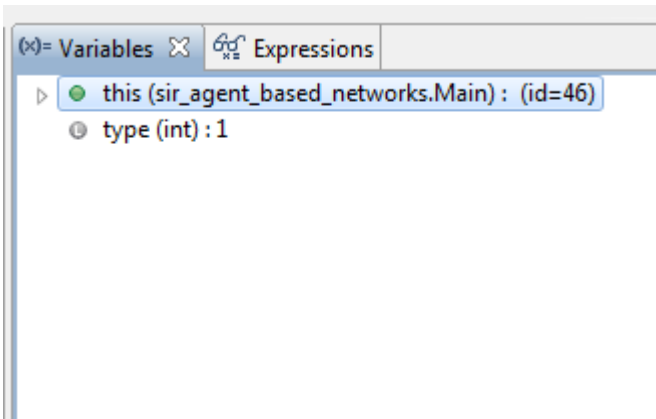
Function arguments:

Read-Only Smart Insert 559 : 17

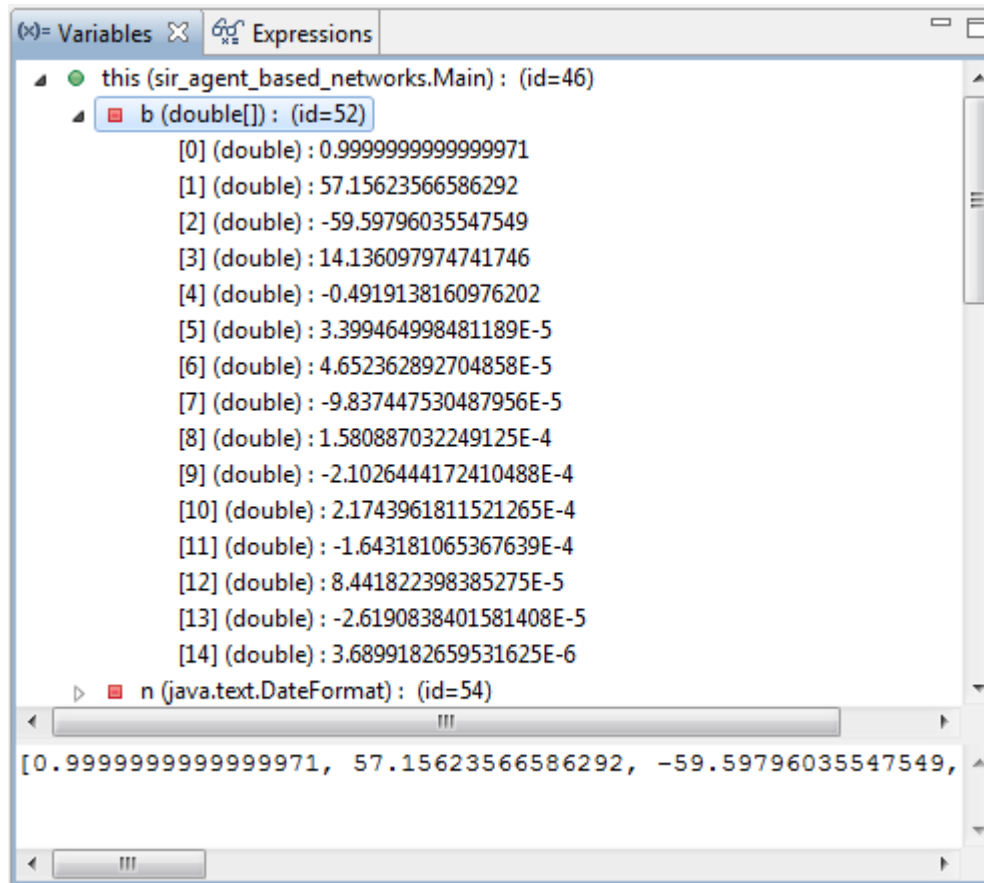
Components to Direct Execution



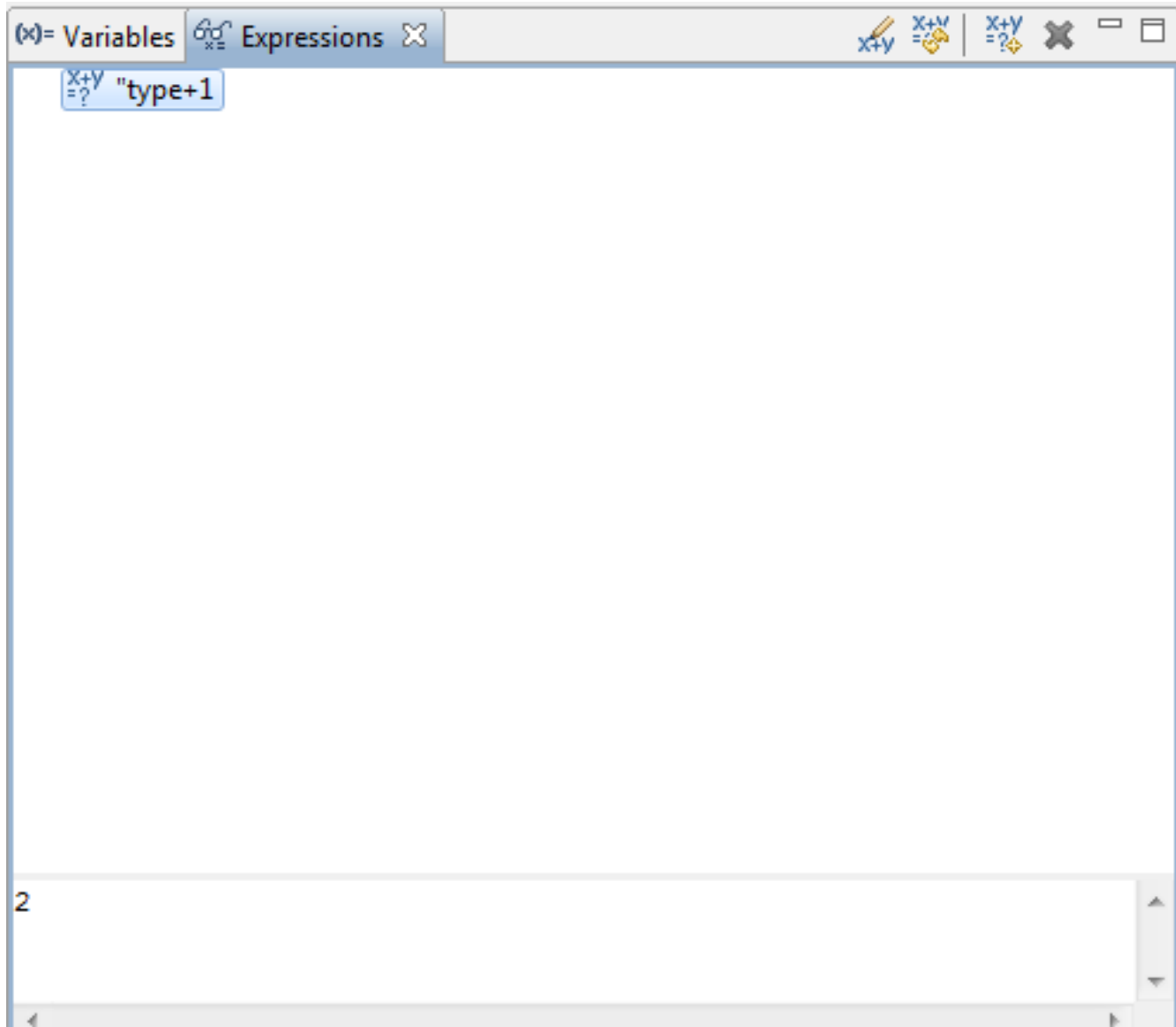
Visible (“In-Scope”) Variables



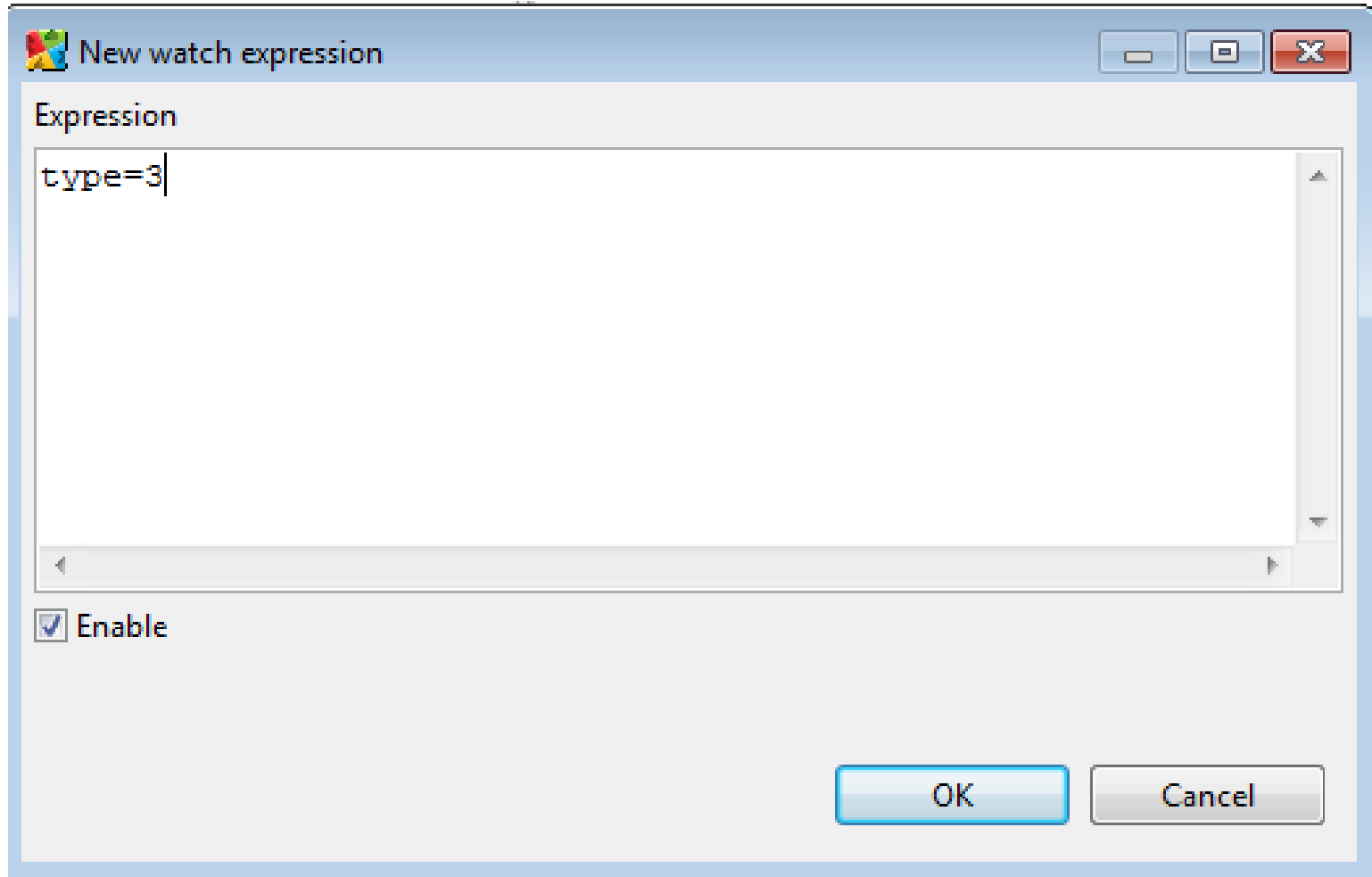
Exploring Composite Variable Values in the Debugger



Inspecting Composite Variables



Changing Variable Values During Debugging



Stepping into Auto-Generated Code

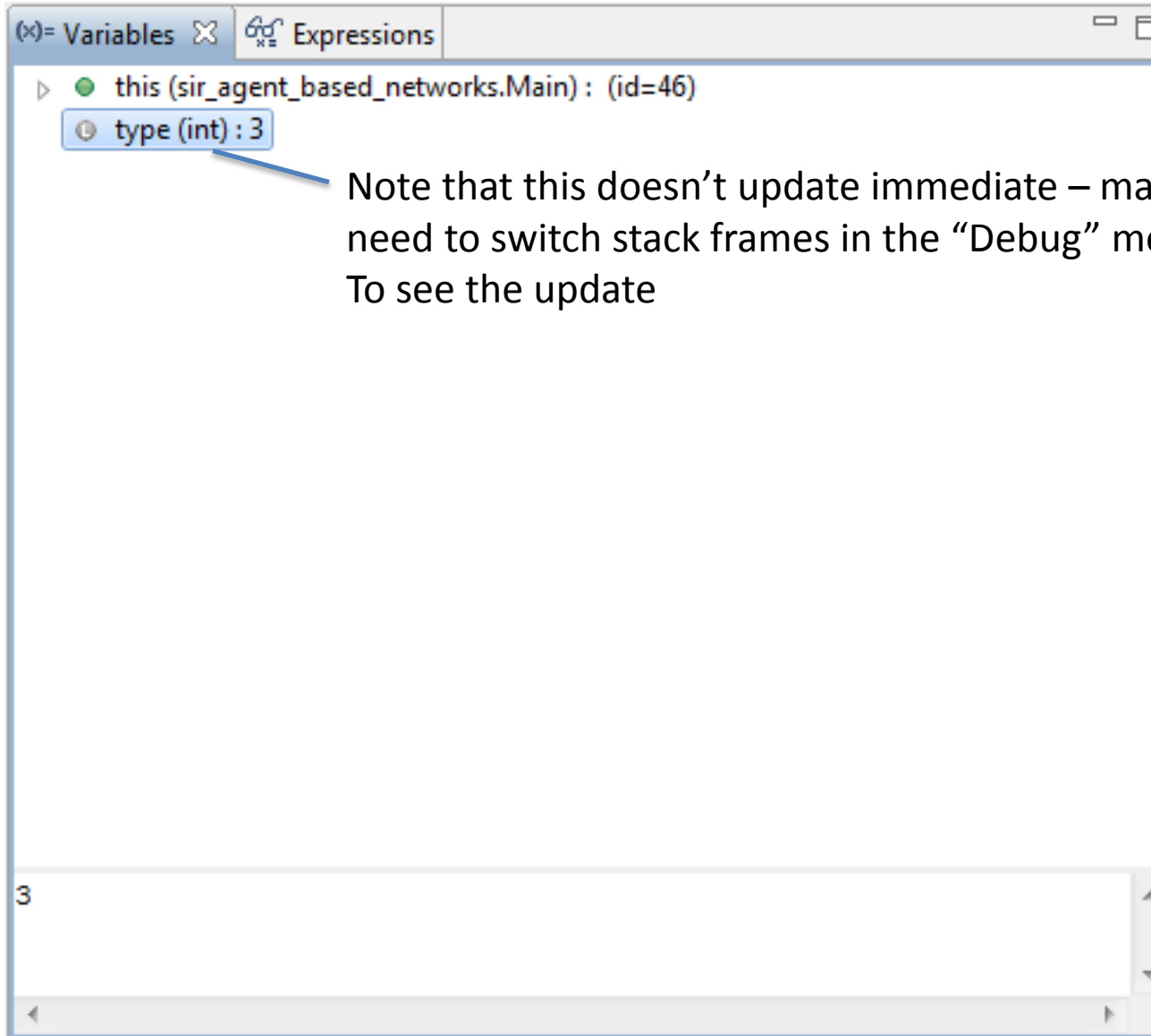
The screenshot displays the AnyLogic Professional software interface during a debugging session. The main window shows the source code of a Java function named `networkTypeToString` in `Main.java`. A breakpoint is set on the line `networkTypeToString(environment.getNetworkType())`. The `Variables` window shows the current state of variables: `this (sir_agent_based_networks.Main) : (id=46)`, `_shape (int) : 15`, and `index (int) : 0`. The `Properties` window for the `networkTypeToString - Function` shows the following configuration:

- Name:** `networkTypeToString`
- Show name:**
- Ignore:**
- Public:**
- Show at runtime:**
- Access:** `default`
- Static:**
- Return type:** `String`

The `Code` tab in the properties window shows the following Java code:

```
case text12: return "Infectivity:";
case text13: return
format( AverageIllnessDuration ) + " days"
;
case text14: return "Average illness duration:";
case text15: return
networkTypeToString( environment.getNetworkType() )
;
case text16: return "Network type:";
case text17: return
environment.getConnectionsPerAgent ()
;
case text18: return "Links per agent:";
case text19: return "Maximum link distance: ";
case text20: return
environment.getConnectionRange ()
;
case text21: return "Percent of long distance links:";
```


Seeing Result of Expression Evaluation



The screenshot shows a debugger window with two tabs: "Variables" and "Expressions". The "Expressions" tab is active, displaying a tree view of expressions. The root expression is "this (sir_agent_based_networks.Main) : (id=46)", which is expanded to show a child expression "type (int) : 3". A blue callout box points to the "type (int) : 3" expression with the text: "Note that this doesn't update immediate – may need to switch stack frames in the 'Debug' method To see the update". At the bottom of the window, a text box contains the value "3".

Variables Expressions

▶ this (sir_agent_based_networks.Main) : (id=46)

 type (int) : 3

Note that this doesn't update immediate – may need to switch stack frames in the "Debug" method To see the update

3

External Debugging in Eclipse

- The “Eclipse” editor is one of the most popular extant software development tools
- Eclipse offers plug-ins of many sorts
 - Debuggers
 - Profilers
 - Visualization tools
 - Version control of models
- Eclipse can be used to debug AnyLogic models at the Java source-code level

Steps Required for Eclipse Debugging

- One time set-up for a particular model
 - Set up AnyLogic to allow debugging connections
 - Set up Eclipse to know
 - How to connect to AnyLogic
 - Where to look for source code files
- Every time want to debug
 - Go to Eclipse
 - Tell debugger to connect to AnyLogic process
 - Interrupt process
 - Set breakpoints, etc.

Setup In AnyLogic

- `-Xdebug -Xnoagent -Djava.compiler=NONE -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=8321`
- These go under the "Advanced" tab of the simulation run to use

Set up: Creating a Debugging Configuration in Eclipse

Debug Configurations

Create, manage, and run configurations

Attach to a Java virtual machine accepting debug connections

Name: Anylogic Application

Connect Source Common

Project: Browse...

Connection Type: Standard (Socket Attach)

Connection Properties:

Host: localhost

Port: 8321

Allow termination of remote VM

- Eclipse Application
 - edu.usask.cs.silverRCP.product
- Java Applet
- Java Application
 - HTMLinksToFiles
 - Main
 - Run main class
 - TestJavaDecisionTree4
- JUnit
- JUnit Plug-in Test
- OSGi Framework
- Remote Java Application
 - Anylogic Application**
- Task Context Plug-in Test
- Task Context Test

Setting Up Source Code Folders

The screenshot shows the Eclipse IDE's 'Debug Configurations' dialog box. The title bar reads 'Debug Configurations' and the subtitle is 'Create, manage, and run configurations'. Below the subtitle, it says 'Attach to a Java virtual machine accepting debug connections'. A green bug icon is in the top right corner.

The left sidebar contains a tree view of configurations. Under 'Remote Java Application', the 'Anylogic Application' configuration is selected and highlighted. Other configurations include 'Eclipse Application', 'Java Applet', 'Java Application', 'JUnit', and 'Task Context Test'.

The main area of the dialog is titled 'Name: Anylogic Application'. It has three tabs: 'Connect', 'Source', and 'Common'. The 'Source' tab is active, showing the 'Source Lookup Path' section. This section contains a list of folders:

- src.generated - C:\Users\Nate\AnyLogicWorkspace\EclipseDebuggingExample_BUILD
- anqiV2 - C:\Users\osgood\AnyLogicWorkspace\AnqiV2_BUILD\classes
- anqiV2 - C:\Users\osgood\AnyLogicWorkspace\AnqiV2_BUILD\src.generated
- Default

To the right of the list are several buttons: 'Add...', 'Edit...', 'Remove', 'Up', 'Down', and 'Restore Default'.

Add Source Folder

The screenshot illustrates the process of adding a source folder to a debug configuration in Eclipse. The main window is titled "Debug Configurations" and shows a configuration named "Anylogic Application". The "Source" tab is active, displaying a list of source lookup paths:

- src.generated - C:\Users\Nate\AnyLogicWorkspace\EclipseDebuggingExample_BUILD
- anqiV2 - C:\Users\osgood\AnyLogicWorkspace\AnqiV2_BUILD\classes
- anqiV2 - C:\Users\osgood\AnyLogicWorkspace\AnqiV2_BUILD\src.generated
- Default

Three dialog boxes are overlaid on the main window:

- Add Source**: A dialog titled "Select the type of source to add to the source lookup path" with the subtitle "A directory in the local file system". It lists several options, with "File System Directory" selected.
- Directory Selection**: A dialog titled "Choose directory to add:" showing a tree view of the file system. The "src.generated" folder is selected.
- Add File System Directory**: A dialog titled "File system folder" with the subtitle "Specify folder and whether subfolders should be searched". The "Directory:" field contains "c:\users\Nate\anylogicworkspace\" and the "Search subfolders" checkbox is checked.

At the bottom of the screen, a snippet of Java code is visible, showing a loop that iterates over an array:

```
Can only iterate over an array or an instance of java.lang.Iterable
Can only iterate over an array or an instance of java.lang.Iterable
controlDefault cannot be resolved
controlDefault cannot be resolved
DataSet cannot be resolved to a type
DataSet cannot be resolved to a type
Engine cannot be resolved to a type
if (MessageDialog
```

Once Set up, Can...

- Set breakpoints
- See the variables, with symbolic information
- Suggestions
 - Set a breakpoint on a thrown runtime exception (regardless of whether caught)
 - Throw a caught runtime exception from model startup code
 - When catch this in Eclipse, can then use to set breakpoints (including in other files)

Example Setup: Set up Function to Trigger the Debugger

The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a model diagram with various components like 'populationSize', 'Population [..]', 'environment', 'offspringDistanceFromMother', 'initialPrevalenceOfInfection', 'immigrantsPerYear', 'ImmigrantArrival', 'prevalenceOfInfectionAmongImmigrants', and 'MeanLifespan'. A function named 'TriggerDebugger' is highlighted in the workspace.

The 'TriggerDebugger' function is defined in the Properties view, under the 'Code' tab. The function body is as follows:

```
try
{
    throw new RuntimeException("arbitrary");
}
catch (RuntimeException e)
{
    traceIn("Threw & caught exception");
}
```

The interface also shows a Project view on the left with a tree structure including 'EclipseDebuggingExample*', 'Main', 'Parameters', 'Functions', 'Events', 'Environments', 'Embedded Objects', 'Analysis Data', 'Presentation', 'Person', 'DebuggingSession: Main', 'ProfilingSimulation: Main', and 'Simulation: Main'. The bottom status bar shows 'Problems' and a search field.

In Startup Code for Model, Call Function

The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a model diagram with various components like 'populationSize', 'Population [...]', 'environment', 'datasetInfective', 'offspringDistanceFromMother', 'initialPrevalenceOfInfection', 'immigrantsPerYear', 'ImmigrantArrival', 'prevalenceOfInfectionAmongImmigrants', 'MeanLifespan', and 'TriggerDebugger'.

The Properties window is open, showing the 'Main - Active Object Class' configuration. The 'Startup Code' field contains the following code:

```
environment.deliverToRandom("Infect!");  
TriggerDebugger();
```

The 'Destroy Code' field is currently empty.

The interface also shows a Project Explorer on the left with a tree view of the model structure, including 'Main', 'Parameters', 'Functions', 'Events', 'Environments', 'Embedded Objects', 'Analysis Data', 'Presentation', 'Person', 'DebuggingSession: Main', 'ProfilingSimulation: Main', and 'Simulation: Main'. The right sidebar contains a palette of model components such as 'Model', 'Parameter', 'Flow Aux...', 'Stock Vari...', 'Event', 'Dynamic...', 'Plain Vari...', 'Collectio...', 'Function', 'Table Fun...', 'Port', 'Connector', 'Entry Point', 'State', 'Transition', 'Initial Stat...', 'Branch', 'History St...', 'Final State', and 'Environ...'. At the bottom right, there are buttons for 'Action', 'Analysis', 'Presentati...', 'Connectivi...', and 'Enterprise...', along with a 'More Libraries...' link.

In Eclipse, Open “Debug” Perspective

The screenshot shows the Eclipse IDE in the 'Debug' perspective. The 'Person.java' file is open in the editor, and the 'Problems' view shows 841 errors. A context menu is open over the 'Debug' button in the toolbar.

```
void PerformBirth() {  
    Person mother = this;  
    Person offspring = get_Main().add_Population((double) 1.0, "A baby has been born! Baby's id is " +  
    // establish connections of infant  
    EstablishOffspringConnectionsBasedOnMothersConnections(mother, offspring);  
    // now position the baby to be close to the mother  
    EstablishOffspringLocationBasedOnMothersLocation(mother, offspring);  
    // ...  
    // ...  
    // ...  
    // Finally, establish a link between the baby and the mother  
    // (we do this last so we don't have to worry about the mother's connections is to this offspring)  
    offspring.connectTo(this);  
}
```

The 'Problems' view shows 841 errors, 281 warnings, and 0 others. The errors are listed in a table:

Description	Resource
Errors (100 of 841 items)	
ClassMatcher cannot be resolved to a type	GUIPrimePath...
ClassMatcher cannot be resolved to a type	GridPanel_Inte...
ClassMatcher cannot be resolved to a type	GridPanel_Inte...
ClassMatcher cannot be resolved to a type	SandboxMain...
ClassMatcher cannot be resolved to a type	SandboxMain...
ClassMatcher cannot be resolved to a type	ValidationTest...
ClassMatcher cannot be resolved to a type	ValidationTest...
ComponentNotFoundException cannot be resolved to a type	AddArtifactGU...
ComponentNotFoundException cannot be resolved to a type	AddArtifactGU...

Request Creation of Exception Breakpoint

The screenshot displays the Eclipse IDE interface with the 'Run' menu open. The 'Add Java Exception Breakpoint...' option is highlighted. The background shows the 'Person.java' file in the editor, with a line of code selected: `is " + offspring + " while the mother is " + this);`. The 'Outline' view on the right lists the methods of the `new ShapeGroup() { ...}` class, including `CurrentAge() : double`, `drawModelElements(Panels, Graphics2D, ...)`, and `PerformBirth() : void`. The status bar at the bottom indicates 'Writable', 'Smart Insert', '520 : 1', and 'Build Project'.

Run menu options:

- Resume
- Suspend
- Terminate
- Step Into
- Step Over
- Step Return
- Run to Line
- Use Step Filters (Shift+F5)
- Run (Ctrl+F11)
- Debug (F11)
- Profile
- Profile History
- Profile As
- Profile Configurations...
- Run History
- Run As
- Run Configurations...
- Debug History
- Debug As
- Debug Configurations...
- Toggle Breakpoint (Ctrl+Shift+B)
- Toggle Line Breakpoint
- Toggle Method Breakpoint
- Toggle Watchpoint
- Skip All Breakpoints
- Remove All Breakpoints
- Add Java Exception Breakpoint...**
- Add Class Load Breakpoint...
- All References...
- All Instances... (Ctrl+Shift+N)
- Watch
- Inspect (Ctrl+Shift+I)

Code in Person.java:

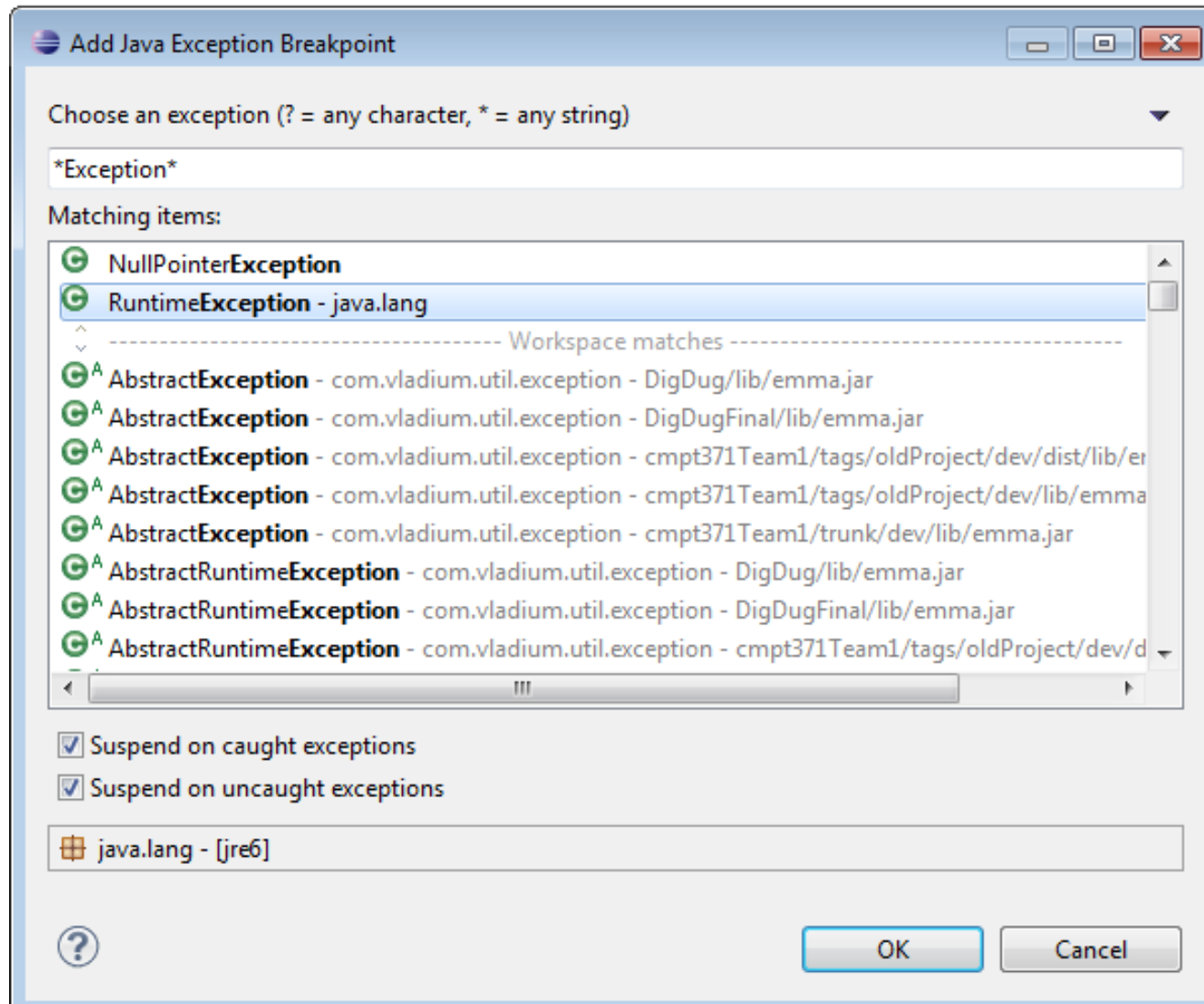
```
on((double) 0, ethnicity, RandomSex(), this.IsInfected());  
is " + offspring + " while the mother is " + this);  
  
Connections (offspring, mother);  
le mother (otherwise leads to stretching of mother's connections ac  
ocation (offspring, mother);  
  
MothersConnections ( Person offspring, Person mother ) {  
d all of the mother's connections
```

Outline view:

- new ShapeGroup() { ...}
- CurrentAge() : double
- drawModelElements(Panels, Graphics2D, ...)
- enterState(short, boolean) : void
- EstablishOffspringConnectionsBasedOn...
- EstablishOffspringLocationBasedOnMot...
- evaluateRateOf(TransitionRate) : double
- evaluateTimeoutOf(TransitionTimeout) :
- executeActionOf(Statechart) : void
- executeActionOf(TransitionMessage, Ob...
- executeActionOf(TransitionRate) : void
- executeActionOf(TransitionTimeout) : v...
- exitState(short, Transition, boolean, State...
- FertilityRateAgeSexEthnicity(double, Sex, ...)
- FinalizeDeath() : void
- get_Main() : Main
- getNameOf(Statechart) : String
- getNameOf(TransitionMessage) : String
- getNameOf(TransitionRate) : String
- getNameOf(TransitionTimeout) : String
- getNameOfState(short) : String
- getPersistentShape(int) : Object
- getStatechartOf(TransitionMessage) : Sta...
- getStatechartOf(TransitionRate) : Statech...
- getStatechartOf(TransitionTimeout) : Sta...
- IsInfected() : boolean
- isInReproductiveYears(double) : boolean
- onChange() : void
- onChange_ethnicity() : void
- onChange_initialAge() : void
- onChange_sex() : void
- onClickModelAt(Panels, double, double, i...
- onDestroy() : void
- onReceive(Object, Agent) : void
- PerformBirth() : void

Status bar: Writable | Smart Insert | 520 : 1 | Build Project

Request as Breakpoint Regardless of Handling



Should Now be in List of Enabled Breakpoints

The screenshot shows the Eclipse IDE interface. The top toolbar includes icons for File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The main editor area displays the source code for `Person.java`. The `PerformBirth` method is highlighted, and a breakpoint is set at line 29. The `EstablishOffspringConnectionsBasedOnMothersConnections` method is also visible. The `Outline` view on the right shows the class structure, including methods like `PerformBirth`, `get_Main`, `get_NameOf`, `get_PersistentShape`, `get_StatechartOf`, `is_Infected`, `is_InReproductiveYears`, `on_Change`, `on_ClickModelAt`, `on_Destroy`, `on_Receive`, and `PerformBirth`. The `Build Project` button is visible at the bottom right.

Debug - C:\Users\Nate\AnyLogicWorkspace\EclipseDebuggingExample_BUILD\src.generated\abmmmodelwithbirthdeath\Person.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Debug <terminated> Anylogic Application [Remote Java Application]
<disconnected> Java HotSpot(TM) Client VM[localhost:8321]

Variables Breakpoints Expressions

- RuntimeException
- NullPointerException: caught and uncaught
- RuntimeException: caught and uncaught
- CreateScenarioDialog [line: 436] - new Anonymous
- HTMLLinksToFiles [line: 27] - main(String[])
- HTMLLinksToFiles [line: 29] - main(String[])

MainClass.java DefaultTracingFilter MainClass.java DefaultTracingFilter Main.java Person.java

```
void PerformBirth (...) {  
    Person mother = this;  
    Person offspring = get_Main().add_Population((double) 0, ethnicity, RandomSex(), this.IsInfected());  
    println("A baby has been born! Baby's id is " + offspring + " while the mother is " + this);  
    // establish connections of infant  
    EstablishOffspringConnectionsBasedOnMothersConnections(offspring, mother);  
    // now position the baby to be close to the mother (otherwise leads to stretching of mother's connections ac  
    EstablishOffspringLocationBasedOnMothersLocation(offspring, mother);  
}  
  
void EstablishOffspringConnectionsBasedOnMothersConnections ( Person offspring, Person mother ) {  
    // now establish links between the baby and all of the mother's connections
```

new ShapeGroup() {...}
CurrentAge() : double
drawModelElements(Panel, Graphics2D,
enterState(short, boolean) : void
EstablishOffspringConnectionsBasedOn
EstablishOffspringLocationBasedOnMotl
evaluateRateOf(TransitionRate) : double
evaluateTimeoutOf(TransitionTimeout) :
executeActionOf(Statechart) : void
executeActionOf(TransitionMessage, Ob
executeActionOf(TransitionRate) : void
executeActionOf(TransitionTimeout) : vo
exitState(short, Transition, boolean, State
FertilityRateAgeSexEthnicity(double, Sex,
FinalizeDeath() : void
get_Main() : Main
getNameOf(Statechart) : String
getNameOf(TransitionMessage) : String
getNameOf(TransitionRate) : String
getNameOf(TransitionTimeout) : String
getNameOfState(short) : String
getPersistentShape(int) : Object
getStatechartOf(TransitionMessage) : Sta
getStatechartOf(TransitionRate) : Statech
getStatechartOf(TransitionTimeout) : Sta
IsInfected() : boolean
isInReproductiveYears(double) : boolean
onChange() : void
onChange_ethnicity() : void
onChange_initialAge() : void
onChange_isInitiallyInfected() : void
onChange_sex() : void
onClickModelAt(Panel, double, double, i
onDestroy() : void
onReceive(Object, Agent) : void
PerformBirth() : void

Build Project

Start AnyLogic Model (Experiment with Extra Debugging JVM Arguments)

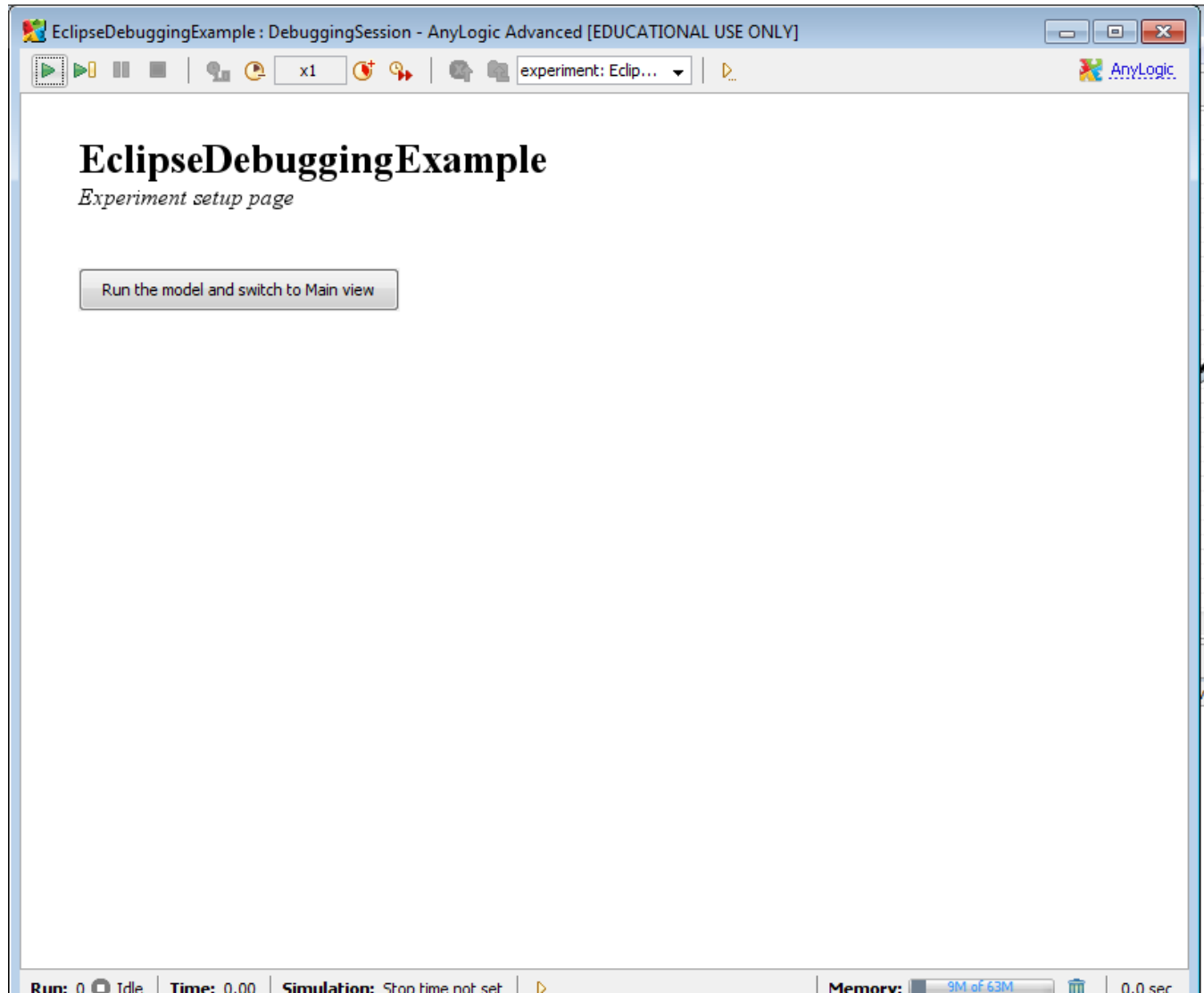
The screenshot displays the AnyLogic Advanced software interface, showing a simulation experiment setup for a debugging session. The main window is titled "DebuggingSession" and contains a diagram of the model structure. The diagram shows a hierarchy of objects: "populationSize", "Population [..]", "environment", "offspringDistanceFromMother", "initialPrevalenceOfInfection", "immigrantsPerYear", "MeanLifespan", "datasetInfective", "ImmigrantArrival", "prevalenceOfInfectionAmongImmigrants", and "TriggerDebugger".

The left sidebar shows the project structure, including "EclipseDebuggingExample", "Main", "Parameters", "Functions", "Events", "Environments", "Embedded Objects", "Analysis Data", "Presentation", "Person", and "DebuggingSession: Main". A context menu is open over "DebuggingSession: Main", showing options like "New", "Open with", "Open...", "Cut", "Copy", "Paste", "Delete", "Refresh", "Refactor", and "Run".

The bottom panel shows the "DebuggingSession - Simulation Experiment" configuration. The "General" tab is active, displaying the following settings:

- Name: DebuggingSession
- Main active object class (root): Main
- Ignore:
- Random number generation:
 - Random seed (unique simulation runs)
 - Fixed seed (reproducible simulation runs) Seed Value: 1
- offspringDistanceFromMother: 15 /* half a distance outside of perimeter */
- initialPrevalenceOfInfection: 0.01
- immigrantsPerYear: 100
- prevalenceOfInfectionAmongImmigrants: 0.10
- Mean lifespan: 80.0

Leave on Opening Screen for Now (So We can Set up Eclipse)



Go To Eclipse & Request AnyLogic Debugging Debug Configuration (previously set up)

The screenshot displays the Eclipse IDE interface during a debug session. The main window shows the 'Debug Configuration' dialog box, which is used to manage and select debug configurations. The configurations listed include:

- 1 Anylogic Application
- 2 AnylogicTracing2
- 3 AnyLogicTracing3MainClass
- 4 MainClass (6)
- 5 MainClass (5)
- 6 MainClass (4)
- 7 MainClass (3)
- 8 MainClass (2)
- 9 edu.usask.cs.silverRCP.product

The 'MainClass.java' editor is open, showing the following code snippet:

```
void PerformBirth(..) {  
    Person mother = this;  
    Person offspring = get_Main().add_Population((double) 0, ethnicity, RandomSex(), this.IsInfected());  
    println("A baby has been born! Baby's id is " + offspring + " while the mother is " + this);  
    // establish connections of infant  
    EstablishOffspringConnectionsBasedOnMothersConnections(offspring, mother);  
    // now position the baby to be close to the mother (otherwise leads to stretching of mother's connections a  
    EstablishOffspringLocationBasedOnMothersLocation(offspring, mother);  
}  
  
void EstablishOffspringConnectionsBasedOnMothersConnections(Person offspring, Person mother) {  
    // now establish links between the baby and all of the mother's connections
```

The 'Outline' view on the right shows the class structure of 'new ShapeGroup()' with various methods and attributes, including 'CurrentAge()', 'drawModelElements()', 'enterState()', 'EstablishOffspringConnectionsBasedOnMotl', 'evaluateRateOf(TransitionRate)', 'evaluateTimeoutOf(TransitionTimeout)', 'executeActionOf(Statechart)', 'executeActionOf(TransitionMessage, Ob', 'executeActionOf(TransitionRate)', 'executeActionOf(TransitionTimeout)', 'exitState(short, Transition, boolean, State', 'FertilityRateAgeSexEthnicity(double, Sex', 'FinalizeDeath()', 'get_Main(): Main', 'getNameOf(Statechart): String', 'getNameOf(TransitionMessage): String', 'getNameOf(TransitionRate): String', 'getNameOf(TransitionTimeout): String', 'getNameOfState(short): String', 'getPersistentShape(int): Object', 'getStatechartOf(TransitionMessage): Sta', 'getStatechartOf(TransitionRate): Statech', 'getStatechartOf(TransitionTimeout): Sta', 'IsInfected(): boolean', 'isInReproductiveYears(double): boolean', 'onChange(): void', 'onChange_ethnicity(): void', 'onChange_initialAge(): void', 'onChange_isInitiallyInfected(): void', 'onChange_sex(): void', 'onClickModelAt(Panel, double, double, i', 'onDestroy(): void', 'onReceive(Object, Agent): void', and 'PerformBirth(): void'.

Should Immediately See Something Like This

The screenshot displays the Eclipse IDE interface during a debug session. The title bar indicates the file being debugged is `Person.java` in the project `abmmmodelwithbirthdeath`. The menu bar includes `File`, `Edit`, `Source`, `Refactor`, `Navigate`, `Search`, `Project`, `Run`, `Window`, and `Help`.

The **Debug** panel shows the following threads:

- Anylogic Application [Remote Java Application]
- Java HotSpot(TM) Client VM[localhost:8321]
- Thread [DestroyJavaVM] (Running)
- Daemon Thread [AnyLogic presentation frame manager] (Running)
- Thread [AnyLogic simulation performance monitor] (Running)
- Thread [AWT-EventQueue-0] (Running)
- Thread [AWT-Shutdown] (Running)

The **Variables** panel shows the following variables:

- RuntimeException
- NullPointerException: caught and uncaught
- RuntimeException: caught and uncaught
- CreateScenarioDialog [line: 436] - new Anonymous
- HTMLLinksToFiles [line: 27] - main(String[])
- HTMLLinksToFiles [line: 29] - main(String[])

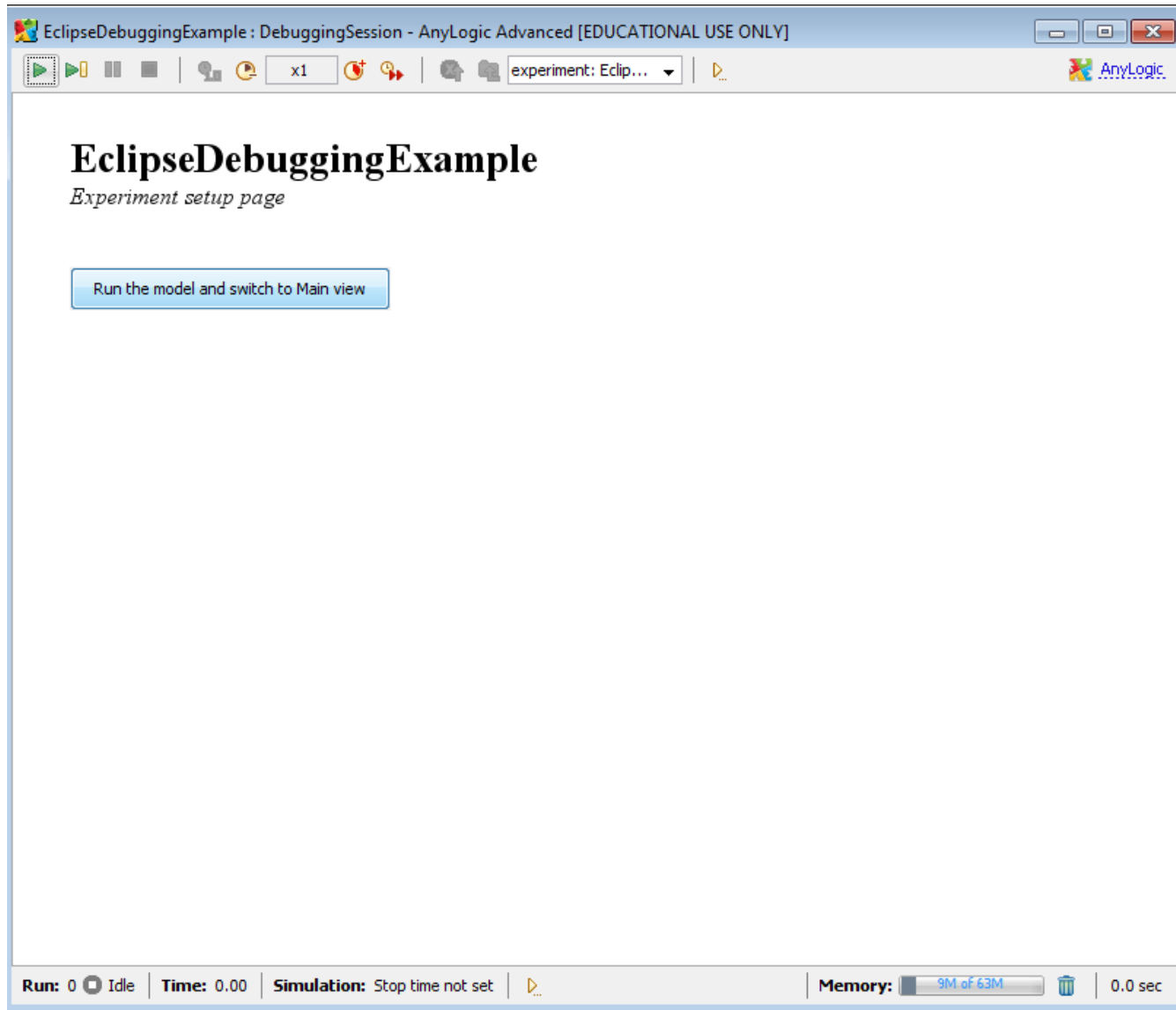
The **Outline** panel shows the class structure for `new ShapeGroup() { ... }`:

- `CurrentAge() : double`
- `drawModelElements(Panel, Graphics2D, ...)`
- `enterState(short, boolean) : void`
- `EstablishOffspringConnectionsBasedOnMot...`
- `EstablishOffspringLocationBasedOnMot...`
- `evaluateRateOf(TransitionRate) : double`
- `evaluateTimeoutOf(TransitionTimeout) :`
- `executeActionOf(Statechart) : void`
- `executeActionOf(TransitionMessage, Ob...`
- `executeActionOf(TransitionRate) : void`
- `executeActionOf(TransitionTimeout) : v...`
- `exitState(short, Transition, boolean, State...`
- `FertilityRateAgeSexEthnicity(double, Sex,`
- `FinalizeDeath() : void`
- `get_Main() : Main`
- `getNameOf(Statechart) : String`
- `getNameOf(TransitionMessage) : String`
- `getNameOf(TransitionRate) : String`
- `getNameOf(TransitionTimeout) : String`
- `getNameOfState(short) : String`
- `getPersistentShape(int) : Object`
- `getStatechartOf(TransitionMessage) : Sta...`
- `getStatechartOf(TransitionRate) : Statech...`
- `getStatechartOf(TransitionTimeout) : Sta...`
- `IsInfected() : boolean`
- `isInReproductiveYears(double) : boolean`
- `onChange() : void`
- `onChange_ethnicity() : void`
- `onChange_initialAge() : void`
- `onChange_isInitiallyInfected() : void`
- `onChange_sex() : void`
- `onClickModelAt(Panel, double, double, i...`
- `onDestroy() : void`
- `onReceive(Object, Agent) : void`
- `PerformBirth() : void`

The **Code Editor** shows the `Person.java` file with the following code:

```
void PerformBirth(..) {  
    Person mother = this;  
    Person offspring = get_Main().add_Population((double) 0, ethnicity, RandomSex(), this.IsInfected());  
    println("A baby has been born! Baby's id is " + offspring + " while the mother is " + this);  
    // establish connections of infant  
    EstablishOffspringConnectionsBasedOnMothersConnections(offspring, mother);  
    // now position the baby to be close to the mother (otherwise leads to stretching of mother's connections ac  
    EstablishOffspringLocationBasedOnMothersLocation(offspring, mother);  
}  
  
void EstablishOffspringConnectionsBasedOnMothersConnections(Person offspring, Person mother) {  
    // now establish links between the baby and all of the mother's connections;  
}
```

Return to AnyLogic & Start Simulation via Button Push



Back in Eclipse, the Debugger Should have been Triggered & at Exception Handler

(If not, close "Main.java" and double-click on topmost "stack frame" (Where Exception is triggered

The screenshot displays the Eclipse IDE interface during a debugging session. The top toolbar shows the 'Debug' button is active. The 'Debug' console on the left shows the current thread is 'Main.TriggerDebugger()' at line 441. The 'Variables' view below it shows a 'RuntimeException' exception is caught and uncaught. The 'Main.java' editor at the bottom shows the source code with a 'throw new RuntimeException("arbitrary");' statement highlighted. The 'Outline' view on the right lists the methods of the 'TriggerDebugger()' class, with 'TriggerDebugger()' at the bottom.

```
Debug - C:\Users\Nate\AnyLogicWorkspace\EclipseDebuggingExample_BUILD\src.generated\abmmmodelwithbirthdeath\Main.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help

Debug
Java HotSpot(TM) Client VM[localhost:8321]
  Thread [DestroyJavaVM] (Running)
  Daemon Thread [AnyLogic presentation frame manager] (Running)
  Thread [AnyLogic simulation performance monitor] (Running)
  Thread [AWT-EventQueue-0] (Running)
  Thread [AWT-Shutdown] (Running)
  Daemon Thread [AWT-Windows] (Running)
  Daemon Thread [AnyLogic ShapeControl action handler] (Suspended (exception RuntimeException))
  Main.TriggerDebugger() line: 441

Variables Breakpoints Expressions
[ ] RuntimeException
[ ] NullPointerException: caught and uncaught
[ ] RuntimeException: caught and uncaught
[ ] CreateScenarioDialog [line: 436] - new Anonymous
[ ] HTMLLinksToFiles [line: 27] - main(String[])
[ ] HTMLLinksToFiles [line: 29] - main(String[])
[ ] HTMLLinksToFiles [line: 36] - main(String[])
[ ] HTMLLinksToFiles [line: 38] - main(String[])

MainClass.java DefaultTracingFilter MainClass.java DefaultTracingFilter Person.java Main.java
--// User functions -----
void TriggerDebugger (...) {
    try
    {
        throw new RuntimeException("arbitrary");
    }
    catch (RuntimeException e)
    {
        traceIn("Threw & caught exception");
    }
}

--// Analysis Data Elements
```

Now Can Set Breakpoints in Main.java or Elsewhere (Here: Person.java)

The screenshot displays the Eclipse IDE interface during a debug session. The top toolbar shows the 'Debug' button is active. The 'Debug Console' at the top left shows the Java HotSpot(TM) Client VM [localhost:8321] and a list of running threads, including 'Main.TriggerDebugger()' at line 441. The 'Breakpoints' view in the middle left shows several breakpoints set in 'Main' and 'Person' classes. The 'Person.java' editor at the bottom left shows the 'PerformBirth()' method with a breakpoint set at line 518. The 'Outline' view on the right lists the methods of the 'Person' class, including 'PerformBirth()'. The status bar at the bottom indicates the current line is 516:1.

```
void PerformBirth (...) {  
    Person mother = this;  
    Person offspring = get_Main().add_Population((double) 0, ethnicity, RandomSex(), this.IsInfected());  
    println("A baby has been born! Baby's id is " + offspring + " while the mother is " + this);  
    // establish connections of infant  
    EstablishOffspringConnectionsBasedOnMothersConnections(offspring, mother);  
    // now position the baby to be close to the mother (otherwise leads to stretching of mother's connections ac  
    EstablishOffspringLocationBasedOnMothersLocation(offspring, mother);  
}
```

Warning: Breakpoints are Not
Shown in Source Window – Just in
“Breakpoints” area

Press “Resume” to Continue – Awaiting a Breakpoint

The screenshot shows the Eclipse IDE in a debug state. The main window displays the source code of `Person.java` with a breakpoint at line 441. The Debug console shows the JVM and various threads, with a "Resume (F8)" button highlighted. The Outline view on the right shows the class structure, and the Variables view at the bottom left shows the current state of the program.

Debug Console:

- Java HotSpot(TM) Client VM[localhost:8321]
- Thread [DestroyJavaVM] (Running)
- Daemon Thread [AnyLogic presentation frame manager] (Running)
- Thread [AnyLogic simulation performance monitor] (Running)
- Thread [AWT-EventQueue-0] (Running)
- Thread [AWT-Shutdown] (Running)
- Daemon Thread [AWT-Windows] (Running)
- Daemon Thread [AnyLogic ShapeControl action handler] (Suspended (exception RuntimeException))
- Main.TriggerDebugger() line: 441

Variables:

- Main [line: 73] - Main
- Main [line: 76] - Main
- Main [line: 323] - Main
- Main [line: 345] - Main
- Main [line: 2421] - Main
- MainClass [line: 12] - main(String[])
- Person [line: 518] - Person
- Person [line: 520] - Person

Source Code (Person.java):

```
void PerformBirth(..) {  
    Person mother = this;  
    Person offspring = get_Main().add_Population((double) 0, ethnicity, RandomSex(), this.IsInfected());  
    println("A baby has been born! Baby's id is " + offspring + " while the mother is " + this);  
    // establish connections of infant  
    EstablishOffspringConnectionsBasedOnMothersConnections(offspring, mother);  
    // now position the baby to be close to the mother (otherwise leads to stretching of mother's connections ac  
    EstablishOffspringLocationBasedOnMothersLocation(offspring, mother);  
}
```

Example Breakpoint in Main

The screenshot displays the Eclipse IDE in a debug state. The top toolbar shows the 'Debug' button is active. The 'Debug Console' on the left shows the execution flow, with a breakpoint at line 345 in Main.java. The 'Variables' view below it lists the current state of variables, including Main, Person, and MainClass. The 'Breakpoints' view shows a list of breakpoints, with the one at line 345 in Main.java selected. The 'Expressions' view is currently empty. The 'Outline' view on the right shows the class hierarchy and methods, with the 'add_Population' method highlighted. The 'Main.java' editor at the bottom shows the source code with the breakpoint set at line 345, which is the line where a new population object is instantiated.

```
Debug - C:\Users\Nate\AnyLogicWorkspace\EclipseDebuggingExample_BUILD\src.generated\abmmmodelwithbirthdeath\Main.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help

Debug Console
Thread [AWT-Shutdown] (Running)
Daemon Thread [AWT-Windows] (Running)
Daemon Thread [AnyLogic model execution thread] (Suspended (breakpoint at line 345 in Main))
Main.add_Population(double, Person$Ethnicity, Person$Sex, boolean) line: 345
Main.executeActionOf(EventRate) line: 289
EventRate.execute() line: not available
Engine.h() line: not available
Engine.a(Engine) line: not available
Engine$.run() line: not available

Variables
Main [line: 73] - Main
Main [line: 76] - Main
Main [line: 323] - Main
Main [line: 345] - Main
Main [line: 2421] - Main
MainClass [line: 12] - main(String[])
Person [line: 518] - Person
Person [line: 520] - Person

Breakpoints
Main [line: 73] - Main
Main [line: 76] - Main
Main [line: 323] - Main
Main [line: 345] - Main
Main [line: 2421] - Main
MainClass [line: 12] - main(String[])
Person [line: 518] - Person
Person [line: 520] - Person

Expressions

Outline
add_Population(double, Ethnicity, Sex, b
create() : void
create_Population_xjal(Person, int) : void
drawModelElements(Panel, Graphics2D,
evaluateRateOf(EventRate) : double
evaluateTimeoutOf(EventTimeout) : dou
executeActionOf(EventRate) : void
executeActionOf(EventTimeout) : void
getEmbeddedObjects() : List<Object>
getFirstOccurrenceTime(EventTimeout)
getModeOf(EventTimeout) : int
getNameOf(ActiveObject) : String
getNameOf(ActiveObjectCollection<?>)
getNameOf(EventRate) : String
getNameOf(EventTimeout) : String
getNameOfShape(int) : String
getPersistentShape(int) : Object
getShapeEmbeddedObject(int) : Object
getShapeReplication(int) : int
getShapeType(int) : int
getShapeX(int, int) : double
getShapeY(int, int) : double
instantiate_Population_xjal(int) : Person
onChange() : void
onChange_immigrantsPerYear() : void
onChange_initialPrevalenceOfInfection()
onChange_MeanLifespan() : void
onChange_offspringDistanceFromMothe
onChange_populationSize() : void
onChange_prevalenceOfInfectionAmong
onClickModelAt(Panel, double, double, i
onDestroy() : void
onStartup() : void
remove_Population(Person) : boolean
set_immigrantsPerYear(double) : void
set_initialPrevalenceOfInfection(double)

MainClass.java DefaultTracingFilter MainClass.java DefaultTracingFilter Person.java Main.java
... * @return newly created embedded object *
... */
public Person add_Population( double InitialAge, Person.Ethnicity ethnicity, Person.Sex sex, boolean isInit
int index = Population.size();
Person object = instantiate_Population_xjal( index );
// setup parameters:
object.InitialAge = InitialAge;
object.ethnicity = ethnicity;
object.sex = sex;
object.isInitiallyInfected = isInitiallyInfected;
// finish embedded object creation
create_Population_xjal( object, index );
object.start();
return object;
}
```


Example Breakpoint in Person

The screenshot displays the Eclipse IDE interface during a debug session. The top toolbar shows the 'Debug' button is active. The 'Debug Console' on the left shows the execution stack, with the current thread being 'Person.PerformBirth() line: 518'. The 'Variables' view below it lists several variables, including 'Person [line: 518] - Person'. The 'Outline' view on the right shows the class structure of 'Person', with 'PerformBirth()' highlighted. The main editor window shows the source code of 'Person.java', with the 'PerformBirth()' method selected. The code includes a breakpoint at line 518, which is currently active.

```
void PerformBirth() {  
    Person mother = this;  
    Person offspring = get_Main().add_Population((double) 0, ethnicity, RandomSex(), this.IsInfected());  
    println("A baby has been born! Baby's id is " + offspring + " while the mother is " + this);  
    // establish connections of infant  
    EstablishOffspringConnectionsBasedOnMothersConnections(offspring, mother);  
    // now position the baby to be close to the mother (otherwise leads to stretching of mother's connections ac  
    EstablishOffspringLocationBasedOnMothersLocation(offspring, mother);  
}
```

Once at Breakpoint, Can Look at Variables, Single Step, etc.

The screenshot shows the Eclipse IDE in debug mode. The main window displays the source code of `Person.java` with a breakpoint set at line 518 in the `PerformBirth()` method. The `Debug` console shows the current thread is `Person.PerformBirth() line: 518`. The `Variables` view shows the current state of the program, including the `Person` object at line 518. The `Outline` view shows the class structure, including the `PerformBirth()` method. The `Code` view shows the source code of the `PerformBirth()` method, with the current line highlighted.

```
void PerformBirth(...) {  
    Person mother = this;  
    Person offspring = get_Main().add_Population((double) 0, ethnicity, RandomSex(), this.IsInfected());  
    println("A baby has been born! Baby's id is " + offspring + " while the mother is " + this);  
    // establish connections of infant  
    EstablishOffspringConnectionsBasedOnMothersConnections(offspring, mother);  
    // now position the baby to be close to the mother (otherwise leads to stretching of mother's connections ac  
    EstablishOffspringLocationBasedOnMothersLocation(offspring, mother);  
}  
void EstablishOffspringConnectionsBasedOnMothersConnections(Person offspring, Person mother) {
```

Debug Console:

- Thread [AWT-Shutdown] (Running)
- Daemon Thread [AWT-Windows] (Running)
- Daemon Thread [AnyLogic model execution thread] (Suspended (breakpoint at line 518 in Person))
 - Person.PerformBirth() line: 518
 - Person.executeActionOf(TransitionTimeout) line: 333
 - TransitionTimeout.execute() line: not available
 - Engine.h() line: not available
 - Engine.a(Engine) line: not available
 - EngineSa.run() line: not available

Variables:

- Main [line: 73] - Main
- Main [line: 76] - Main
- Main [line: 323] - Main
- Main [line: 2421] - Main
- MainClass [line: 12] - main(String[])
- Person [line: 518] - Person
- Person [line: 520] - Person
- PodSchedule [line: 293] - PodSchedule

Outline:

- new ShapeGroup() {...}
- CurrentAge() : double
- drawModelElements(Panel, Graphics2D, ...)
- enterState(short, boolean) : void
- EstablishOffspringConnectionsBasedOnMothersConnections(Person, Person) : void
- EstablishOffspringLocationBasedOnMothersLocation(Person, Person) : void
- evaluateRateOf(TransitionRate) : double
- evaluateTimeoutOf(TransitionTimeout) : double
- executeActionOf(Statechart) : void
- executeActionOf(TransitionMessage, Object) : void
- executeActionOf(TransitionRate) : void
- executeActionOf(TransitionTimeout) : void
- exitState(short, Transition, boolean, Statechart) : void
- FertilityRateAgeSexEthnicity(double, Sex, Ethnicity) : void
- FinalizeDeath() : void
- get_Main() : Main
- getNameOf(Statechart) : String
- getNameOf(TransitionMessage) : String
- getNameOf(TransitionRate) : String
- getNameOf(TransitionTimeout) : String
- getNameOfState(short) : String
- getPersistentShape(int) : Object
- getStatechartOf(TransitionMessage) : Statechart
- getStatechartOf(TransitionRate) : Statechart
- getStatechartOf(TransitionTimeout) : Statechart
- IsInfected() : boolean
- isInReproductiveYears(double) : boolean
- onChange() : void
- onChange_ethnicity() : void
- onChange_initialAge() : void
- onChange_isInitiallyInfected() : void
- onChange_sex() : void
- onClickModelAt(Panel, double, double, double) : void
- onDestroy() : void
- onReceive(Object, Agent) : void
- PerformBirth() : void

Variables Displayed

The screenshot shows the Eclipse IDE in a debug state. The top toolbar includes icons for file operations, debugging (run, step over, step into, step out), and search. The main window is divided into several panes:

- Debug Console:** Shows the execution stack with the current thread being `Person.PerformBirth() line: 520`.
- Variables View:** A table showing the current state of variables:

Name	Value
this	Person (id=61)
mother	Person (id=61)
offspring	Person (id=64)
- Code Editor:** Displays the `Person.java` file with the `PerformBirth()` method selected. The current line is `Person offspring = get_Main().add_Population((double) 0, ethnicity, RandomSex(), this.IsInfected());`.
- Outline View:** Shows the class structure of `new ShapeGroup() {...}` with various methods like `CurrentAge() : double`, `drawModelElements()`, and `PerformBirth() : void`.

At the bottom of the IDE, the status bar shows `1 : 1` and `Build Project`.

Terminating Execution from AnyLogic Console

The screenshot shows the AnyLogic Advanced [EDUCATIONAL USE ONLY] interface. The main window displays a debugging session for a model named 'EclipseDebuggingExample'. The 'Main' component is selected, and the 'DebuggingSession: Main' is active. The console window shows the following output:

```
anylogic config [Java Application] C:\Program Files (x86)\AnyLogic 6\jre\bin\javaw.exe (N Terminate 2:17:09 PM)
Listening for transport dt_socket at address: 8321
Threw & caught exception
Population member root.Population[46] has died.
Population member root.Population[494] has died.
Population member root.Population[166] has died.
Population member root.Population[727] has died.
Population member root.Population[13] has died.
Population member root.Population[156] has died.
Population member root.Population[115] has died.
Population member root.Population[554] has died.
Population member root.Population[719] has died.
Population member root.Population[776] has died.
```

The 'Terminate' button is highlighted over the console output. The interface also shows a 'Properties' window and a 'Console' window. The 'Console' window is currently active, displaying the output of the simulation. The 'Properties' window is currently empty. The 'Console' window also shows a 'Terminate' button. The 'Console' window is currently active, displaying the output of the simulation.

Eclipse is Now Detached

The screenshot displays the Eclipse IDE interface during a debug session. The title bar indicates the current file is `Person.java` in the `src.generated\abmmmodelwithbirthdeath` package. The menu bar includes `File`, `Edit`, `Source`, `Refactor`, `Navigate`, `Search`, `Project`, `Run`, `Window`, and `Help`. The toolbar contains various icons for file operations, debugging, and navigation.

The **Debug** console shows the following output:

```
<terminated> Anylogic Application [Remote Java Application]
<disconnected> Java HotSpot(TM) Client VM[localhost:8321]
```

The **Outline** view on the right lists the methods of the `new ShapeGroup() { ...}` class:

- `CurrentAge() : double`
- `drawModelElements(Panel, Graphics2D)`
- `enterState(short, boolean) : void`
- `EstablishOffspringConnectionsBasedOnM...`
- `EstablishOffspringLocationBasedOnMotl...`
- `evaluateRateOf(TransitionRate) : double`
- `evaluateTimeoutOf(TransitionTimeout) :`
- `executeActionOf(Statechart) : void`
- `executeActionOf(TransitionMessage, Ob...`
- `executeActionOf(TransitionRate) : void`
- `executeActionOf(TransitionTimeout) : vc...`
- `exitState(short, Transition, boolean, Sta...`
- `FertilityRateAgeSexEthnicity(double, Sex,`
- `FinalizeDeath() : void`
- `get_Main() : Main`
- `getNameOf(Statechart) : String`
- `getNameOf(TransitionMessage) : String`
- `getNameOf(TransitionRate) : String`
- `getNameOf(TransitionTimeout) : String`
- `getNameOfState(short) : String`
- `getPersistentShape(int) : Object`
- `getStatechartOf(TransitionMessage) : Sta...`
- `getStatechartOf(TransitionRate) : Statech...`
- `getStatechartOf(TransitionTimeout) : Sta...`
- `IsInfected() : boolean`
- `isInReproductiveYears(double) : boolean`
- `onChange() : void`
- `onChange_ethnicity() : void`
- `onChange_InitialAge() : void`
- `onChange_isInitiallyInfected() : void`
- `onChange_sex() : void`
- `onClickModelAt(Panel, double, double, i...`
- `onDestroy() : void`
- `onReceive(Object, Agent) : void`
- `PerformBirth() : void`

The **Variables** view is currently empty. The **Breakpoints** and **Expressions** views are also empty.

The **Editor** view shows the source code for `Person.java` with the following visible code:

```
void PerformBirth(..) {
    Person mother = this;
    Person offspring = get_Main().add_Population((double) 0, ethnicity, RandomSex(), this.IsInfected());
    println("A baby has been born! - Baby's id is " + offspring + " while the mother is " + this);
    // establish connections of infant
    EstablishOffspringConnectionsBasedOnMothersConnections(offspring, mother);
    // now position the baby to be close to the mother (otherwise leads to stretching of mother's connections ac
    EstablishOffspringLocationBasedOnMothersLocation(offspring, mother);
}

void EstablishOffspringConnectionsBasedOnMothersConnections( Person offspring, Person mother ) {
```

The bottom status bar shows the zoom level as `1:1` and the current project as `Build Project`.

Remembering Breakpoints

- Note Eclipse *does* remember breakpoints from session to session
- So breakpoints that set earlier in an anylogic session will work again even after close eclipse and restart it again
- Suggestions
 - Consider creating a common breakpoints (e.g. at Main.start)
 - Disable and enable breakpoints rather than deleting them

Example of Debugging Session

Debug - U:\Classes\371_Spring2009\Project\Deliverable 4\Milestone4\newDev\newDev\src\oneoak\digdog\gui\MainWindow.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Debug

Anylogic Application [Remote Java Application]

- Java HotSpot(TM) Client VM[localhost:8321] (Suspended)
 - Daemon Thread [AnyLogic model execution thread] (Suspended)
 - FileOutputStream.writeBytes(byte[], int, int) line: not available [native method]
 - FileOutputStream.write(byte[], int, int) line: not available
 - BufferedOutputStream.flushBuffer() line: not available
 - BufferedOutputStream.flush() line: not available
 - PrintStream.write(byte[], int, int) line: not available
 - StreamEncoder.writeBytes() line: not available
 - StreamEncoder.implFlushBuffer() line: not available
 - StreamEncoder.flushBuffer() line: not available
 - OutputStreamWriter.flushBuffer() line: not available
 - PrintStream.write(String) line: not available
 - PrintStream.print(String) line: not available
 - PrintStream.println(Object) line: not available
 - Utilities.traceln(Object) line: not available
 - Person.PerformBirth() line: 520
 - Person.executeActionOf(TransitionTimeout) line: 333
 - TransitionTimeout.execute() line: not available
 - Engine.h() line: not available
 - Engine.a(Engine) line: not available
 - Engine\$.run() line: not available
 - Daemon System Thread [TimerQueue] (Suspended)
 - Thread [DestroyJavaVM] (Suspended)
 - Daemon Thread [AnyLogic presentation frame manager] (Suspended)
 - Thread [AnyLogic simulation performance monitor] (Suspended)
 - Thread [AWT-EventQueue-0] (Suspended)
 - Thread [AWT-Shutdown] (Suspended)
 - Daemon Thread [AWT-Windows] (Suspended)
 - Daemon System Thread [Java2D Disposer] (Suspended)
 - Daemon System Thread [Attach Listener] (Suspended)
 - Daemon System Thread [Signal Dispatcher] (Suspended)
 - Daemon System Thread [Finalizer] (Suspended)
 - Daemon System Thread [Reference Handler] (Suspended)
 - Daemon Thread [Image Fetcher 0] (Suspended)