

# G54SIM (Spring 2016)

Lab 02

Introduction to AnyLogic

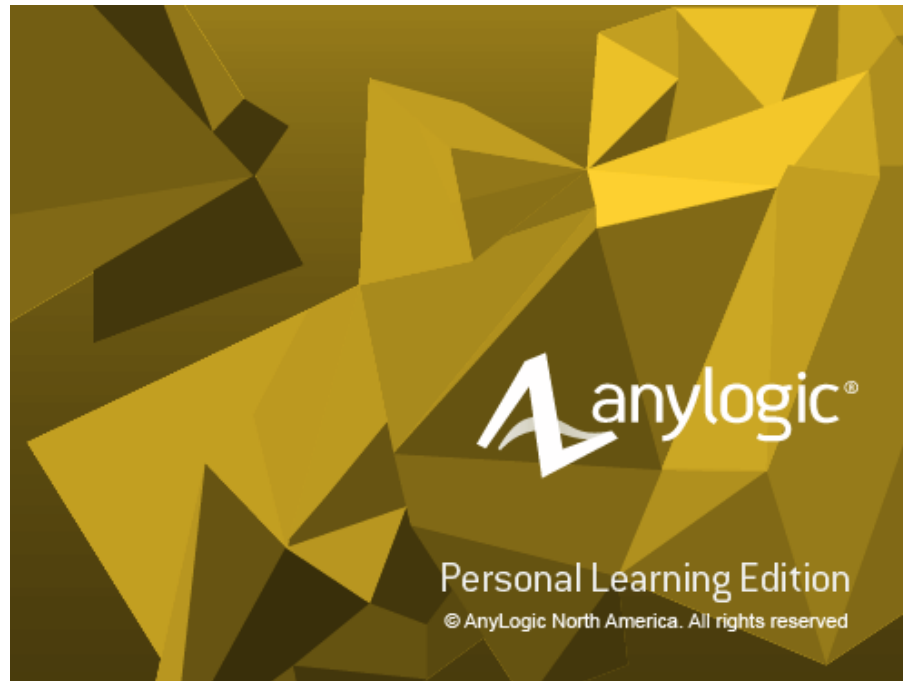
Peer-Olaf Siebers

[pos@cs.nott.ac.uk](mailto:pos@cs.nott.ac.uk)



# AnyLogic

- We use AnyLogic 7.2.0 PLE
  - You will have to apply for a new license key whenever you change your desktop machine (not sure about the virtual desktop)



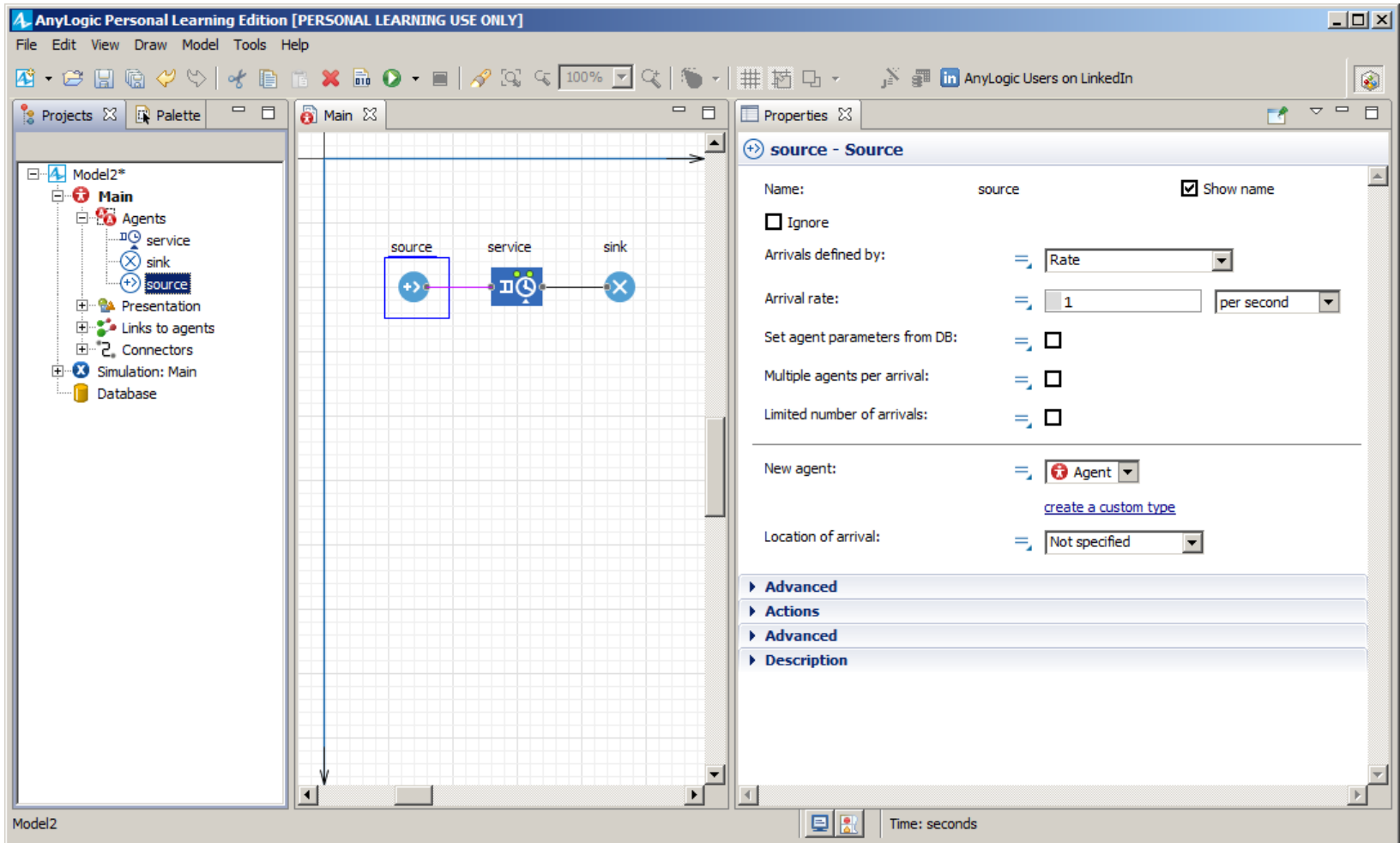
# AnyLogic IDE

The screenshot displays the AnyLogic IDE interface. The main workspace shows a process model with three nodes: 'source', 'service', and 'sink'. The 'service' node is highlighted with a blue box. The 'Properties' panel on the right is titled 'service - Service' and contains the following settings:

- Name: service  Show name  Ignore
- Seize:  (alternative) resource sets  units of the same pool
- Resource sets (alternatives):
- Queue capacity:
- Maximum queue capacity:
- Delay time:  seconds
- Send seized resources:
- Agent location (queue):
- Agent location (delay):

Below the main settings, there are expandable sections: 'Priorities / preemption', 'Advanced', 'Actions', and 'Advanced'.

# AnyLogic IDE



# AnyLogic IDE

- Important things
  - F1: Help
  - Ctrl-Space: Code completion support
  - Ctrl-Enter: Perform refactoring (replace name occurrences)
  - Make sure you select the correct model when pressing "Run"
  - Make sure you set up model time units correctly in the "Model"
  - Use the "magic lightbulb" ...
- Since AnyLogic 7 ...
  - Everything is called "Agent" (entities, resources, agents, ...)
  - PLE version limits number of entities per simulation run to 50,000

# AnyLogic IDE

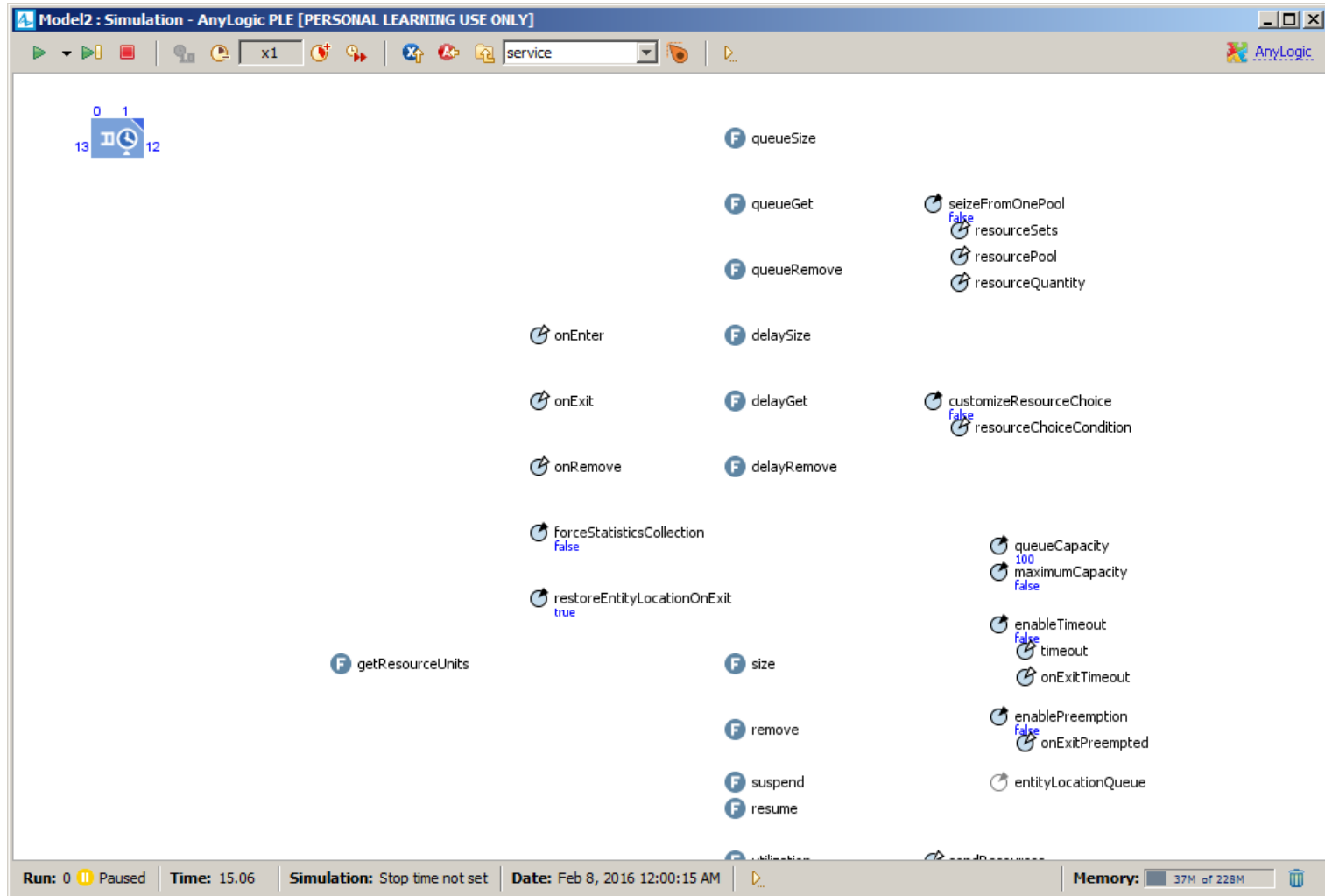
The screenshot displays the AnyLogic IDE interface for a simulation model. The main workspace shows a flow diagram with three components: a source (blue circle with a plus sign), a service (blue square with a clock icon), and a sink (blue circle with an X). The source is labeled 'source' and has an output of 13. The service is labeled 'service' and has an input of 13 and an output of 12. The sink is labeled 'sink' and has an input of 12. A yellow tooltip window is open over the service component, displaying the following details:

```
service
root.service: Service
Queue capacity: 100
Timeout: disabled
Preemption: disabled
in: 13
out: 12
Queue contains: 0
Delay contains: 1
13
```

The status bar at the bottom of the IDE shows the following information:

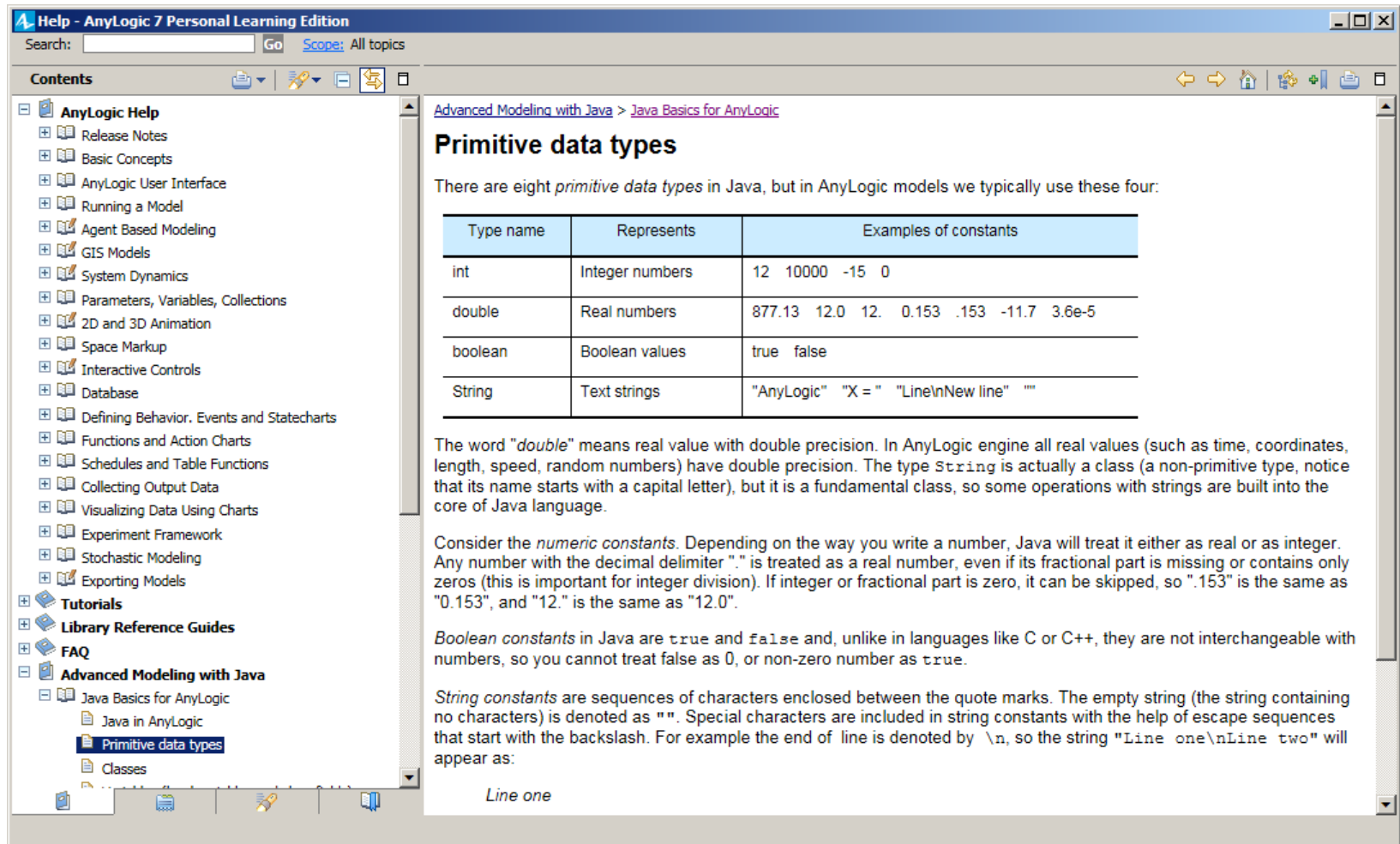
- Run: 0 (Paused)
- Time: 15.06
- Simulation: Stop time not set
- Date: Feb 8, 2016 12:00:15 AM
- Memory: 32M of 228M

# AnyLogic IDE





# Java Basics for AnyLogic



The screenshot shows the 'Help - AnyLogic 7 Personal Learning Edition' window. The left sidebar contains a 'Contents' tree with the following items: AnyLogic Help, Release Notes, Basic Concepts, AnyLogic User Interface, Running a Model, Agent Based Modeling, GIS Models, System Dynamics, Parameters, Variables, Collections, 2D and 3D Animation, Space Markup, Interactive Controls, Database, Defining Behavior, Events and Statecharts, Functions and Action Charts, Schedules and Table Functions, Collecting Output Data, Visualizing Data Using Charts, Experiment Framework, Stochastic Modeling, Exporting Models, Tutorials, Library Reference Guides, FAQ, and Advanced Modeling with Java. Under 'Advanced Modeling with Java', the following items are listed: Java Basics for AnyLogic, Java in AnyLogic, Primitive data types (highlighted), and Classes. The main content area shows the breadcrumb 'Advanced Modeling with Java > Java Basics for AnyLogic' and the title 'Primitive data types'. The text explains that there are eight primitive data types in Java, but only four are typically used in AnyLogic models. A table lists these four types: int, double, boolean, and String. Below the table, the text explains the meaning of 'double', 'boolean constants', and 'string constants'. An example of a string constant is shown as 'Line one'.

Search:  Go Scope: All topics

Contents

- AnyLogic Help
  - Release Notes
  - Basic Concepts
  - AnyLogic User Interface
  - Running a Model
  - Agent Based Modeling
  - GIS Models
  - System Dynamics
  - Parameters, Variables, Collections
  - 2D and 3D Animation
  - Space Markup
  - Interactive Controls
  - Database
  - Defining Behavior, Events and Statecharts
  - Functions and Action Charts
  - Schedules and Table Functions
  - Collecting Output Data
  - Visualizing Data Using Charts
  - Experiment Framework
  - Stochastic Modeling
  - Exporting Models
- Tutorials
- Library Reference Guides
- FAQ
- Advanced Modeling with Java
  - Java Basics for AnyLogic
    - Java in AnyLogic
    - Primitive data types
    - Classes

Advanced Modeling with Java > Java Basics for AnyLogic

## Primitive data types

There are eight *primitive data types* in Java, but in AnyLogic models we typically use these four:

Type name	Represents	Examples of constants
int	Integer numbers	12 10000 -15 0
double	Real numbers	877.13 12.0 12. 0.153 .153 -11.7 3.6e-5
boolean	Boolean values	true false
String	Text strings	"AnyLogic" "X = " "Line\nNew line" ""

The word "*double*" means real value with double precision. In AnyLogic engine all real values (such as time, coordinates, length, speed, random numbers) have double precision. The type `String` is actually a class (a non-primitive type, notice that its name starts with a capital letter), but it is a fundamental class, so some operations with strings are built into the core of Java language.

Consider the *numeric constants*. Depending on the way you write a number, Java will treat it either as real or as integer. Any number with the decimal delimiter "." is treated as a real number, even if its fractional part is missing or contains only zeros (this is important for integer division). If integer or fractional part is zero, it can be skipped, so ".153" is the same as "0.153", and "12." is the same as "12.0".

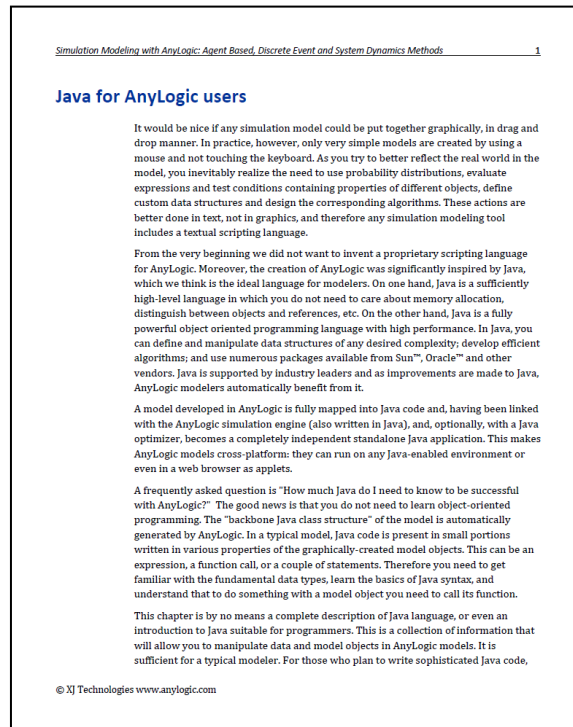
*Boolean constants* in Java are `true` and `false` and, unlike in languages like C or C++, they are not interchangeable with numbers, so you cannot treat `false` as 0, or non-zero number as `true`.

*String constants* are sequences of characters enclosed between the quote marks. The empty string (the string containing no characters) is denoted as `""`. Special characters are included in string constants with the help of escape sequences that start with the backslash. For example the end of line is denoted by `\n`, so the string "Line one\nLine two" will appear as:

Line one

# Java Basics for AnyLogic

- Book Chapter:
  - [http://www.xjitek.com/files/book/Java for AnyLogic users.pdf](http://www.xjitek.com/files/book/Java%20for%20AnyLogic%20users.pdf)



# Java Basics for AnyLogic

- General remarks
  - You do not have to learn full OO programming
    - You need to understand Java data types, expression, and statement syntax
  - Please note:
    - Java is case-sensitive: MyVar is different to myVar!
    - Spaces are not allowed in names: "My Var" is an illegal name!
    - Each statement has to be finished with ";": MyVar=150;
    - Each function has to have parenthesis: time(), add(a)
    - Mind integer division:  $3/2=1$ , not 1.5
    - Boolean values are only true and false, you cannot use 1 and 0
    - Dot "." brings you "inside" the object: agent.event.restart()
    - Array elements have indexes from 0 to n-1

# Java Basics for AnyLogic

- Primitive Types
  - double: Represents real numbers: 1.43, 3.6E18, -14.0
  - int: Represents integer numbers: 12, 16384, -5000
  - boolean: Represents Boolean (true/false) values
- Compound Types –Classes
  - String: Represents textual strings, e.g. "MSFT", "Hi there!", etc.
  - ArrayList; LinkedList: Represents collections of objects
  - HyperArray: Represents multi-dimensional array
  - ...many others. See AnyLogic and Java Class References

# Java Basics for AnyLogic

- Arithmetic operations
  - Notation: +; −; \*; /; % (remainder)
  - In integer divisions, the fraction part is lost, e.g.  $3/2=1$ , and  $2/3=0$
  - Multiplication operators have priority over addition operators
  - The "+" operator allows operands of type String
- Comparison operations
  - Notation: >; >=; <; <=; ==; !=
- Boolean operations
  - Notation: && (AND); || (OR); ! (NOT)

# Java Basics for AnyLogic

- Conditional operator
  - Notation: condition ? value-if-true : value-if-false
- Assignments and shortcuts
  - Notation: =; +=; -=; \*=; /=; %=; ++; -- (a+=b is the same as a=a+b)
- Please note:
  - Within most of operators, left-to-right precedence holds
  - Parentheses may be used to alter the precedence of operations

# Java Basics for AnyLogic

- Method call
  - To call a method, type its name followed by parenthesis; if necessary, put parameters separated by commas within the parenthesis
    - Examples:
      - `x=time(); moveTo(getX(),getY()+100); traceln("Population is increasing");`
- Accessing object fields and methods
  - To access a field or method of a model element (statechart, timer, animation), use the model element name followed by dot "." followed by the field/method name
    - Examples:
      - `statechart.fireEvent("go"); sum=sum+agents.get(i).x;`

# Java Basics for AnyLogic

- Replicated objects are stored in a collection
  - Items are indexed from 0 to n-1
  - Getting the current size of the collection:
    - `people.size()`
  - Obtaining i-th item of the collection:
    - `people.get(i)`
  - Adding a new object to the collection:
    - `add_people();`
  - Removing an object from the collection:
    - `remove_people(person);`



# Java Basics for AnyLogic

- Built-in Functions
  - System functions
    - `time()`; `getOwner()`; `pause()`; `isStateActive(...)`; etc.
  - Mathematical functions
    - Basic: `sqrt`; `sin`; `cos`; `tan`; `exp`; `log`; `round`; `zidz`; `xidz`; etc.
    - Array: `add`; `sub`; `mul`; `sum`; `avg`; `min`; `max`; `get`; etc.
  - Special functions
    - Random numbers: `uniform`; `exponential`; `bernoulli`; `beta`; etc.
    - Time related: `delay`; etc.
  - And more...
    - See AnyLogic Class Reference

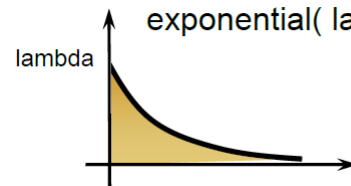
# Java Basics for AnyLogic

- Probability Distributions
  - **Uniform:** Used to represent a random variable with constant likelihood of being in any small interval between min and max. Its density does not depend on the value of  $x$ .
  - **Exponential:** Used to represent the time between random occurrences. The unique property is history independence, i.e. it has the same set of probabilities when shifted in time.
  - **Triangular:** Used when no or little data is available to represent e.g. a process duration.

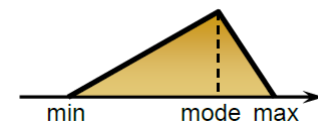
uniform( min, max )



exponential( lambda )



triangular( min, mode, max )



# Tutorial: Object Oriented DES

- Laptop model: Considering different power states



# Questions

