Aachen Summer Simulation Seminar 2014

Practice 02

Java Basics + First Model

Peer-Olaf Siebers

pos@cs.nott.ac.uk

Seminar Organisation

The University of Nottingham
UNITED KINGDOM · CHINA · MALAYSIA

Seminar Organisation

- Day 1: Introduction to Modelling and Simulation
  1. Lec01: Introduction to Modelling an Simulation
  2. Lec02: Foundations of Simulation Modelling
  3. Lec03: Conceptual Modelling
  4. Practice01: AnyLogic Introduction (in pairs)
- Day 2: Application of Modelling and Simulation Methods
  5. Practice02: Java Basics + First Model (in pairs)
  6. Lec04: SDS Modelling
  7. GroupActivity01: Conceptual Modelling (in small groups)
  8. Lec05: DES Modelling
  9. Practice03: SDS or DES Tutorials (in pairs)

Seminar Organisation

- Day 3: Application of Modelling and Simulation Methods
  10. GroupActivity02: Focus Group (in small groups)
  11. Lec06: ABS Modelling + Hybrids
  12. Practice04: ABS Tutorial 1: Wind Turbine Maintenance (in pairs)
  13. Practice05: Exploring the AnyLogic Model Library (optional)
- Day 4: Knowledge Gathering
  14. Practice06: ABS Tutorial 2: Blob World (in pairs)
  15. Lec07: Input Modelling / Experimentation / Output Analysis
  16. GroupActivity03: Discussion of your own project ideas (whole group)
  17. Lec08: Model Verification and Validation

  Feedback

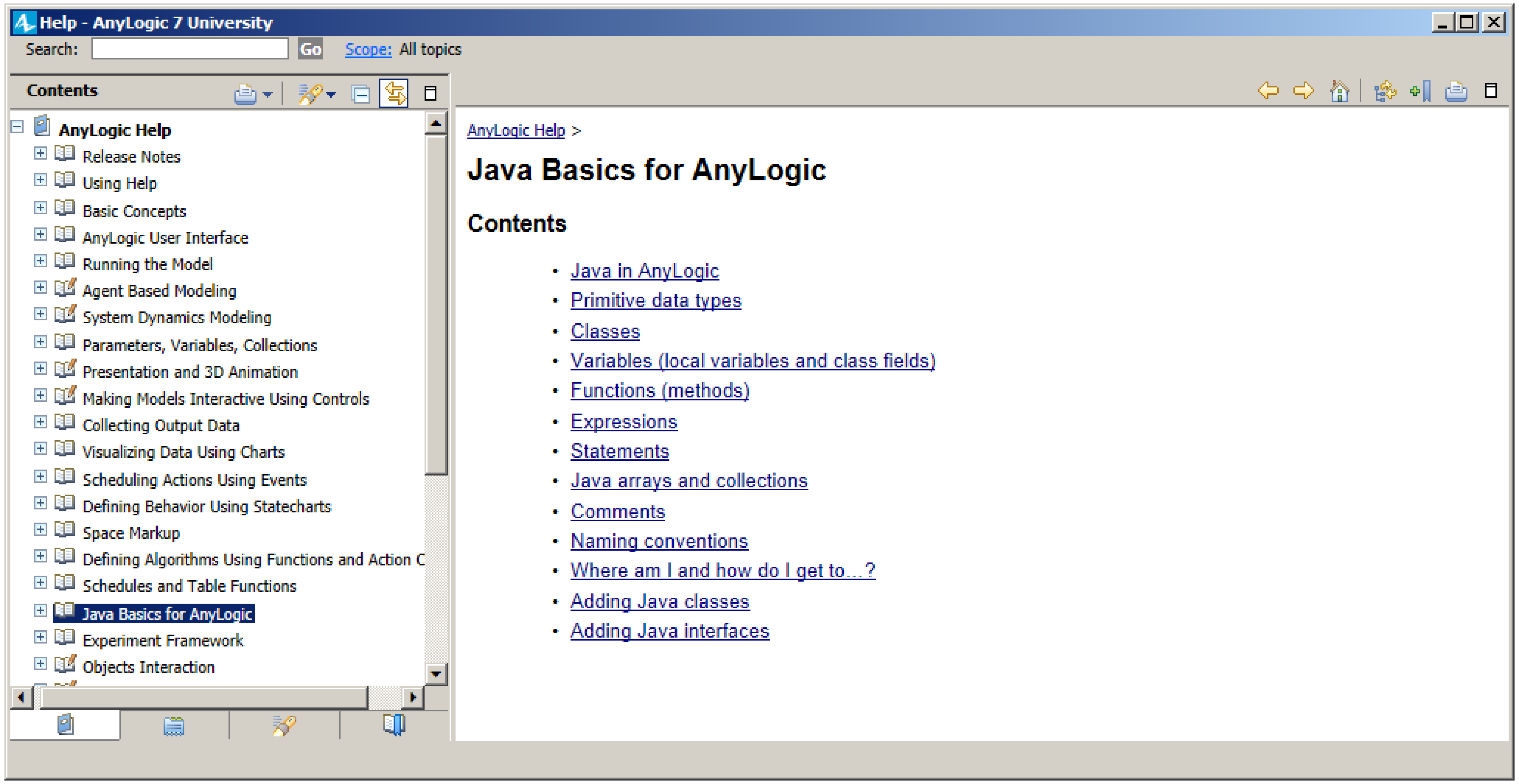
Java Basics + First Model

Motivation

- Provide the minimum Java knowledge required for AnyLogic
- Design and implement a first model in AnyLogic

Java Basics for AnyLogic

- AnyLogic Help



AnyLogic Help
- Release Notes
- Using Help
- Basic Concepts
- AnyLogic User Interface
- Running the Model
- Agent Based Modeling
- System Dynamics Modeling
- Parameters, Variables, Collections
- Presentation and 3D Animation
- Making Models Interactive Using Controls
- Collecting Output Data
- Visualizing Data Using Charts
- Scheduling Actions Using Events
- Defining Behavior Using Statecharts
- Space Markup
- Defining Algorithms Using Functions and Action C
- Schedules and Table Functions
- Java Basics for AnyLogic
- Experiment Framework
- Objects Interaction

AnyLogic Help >

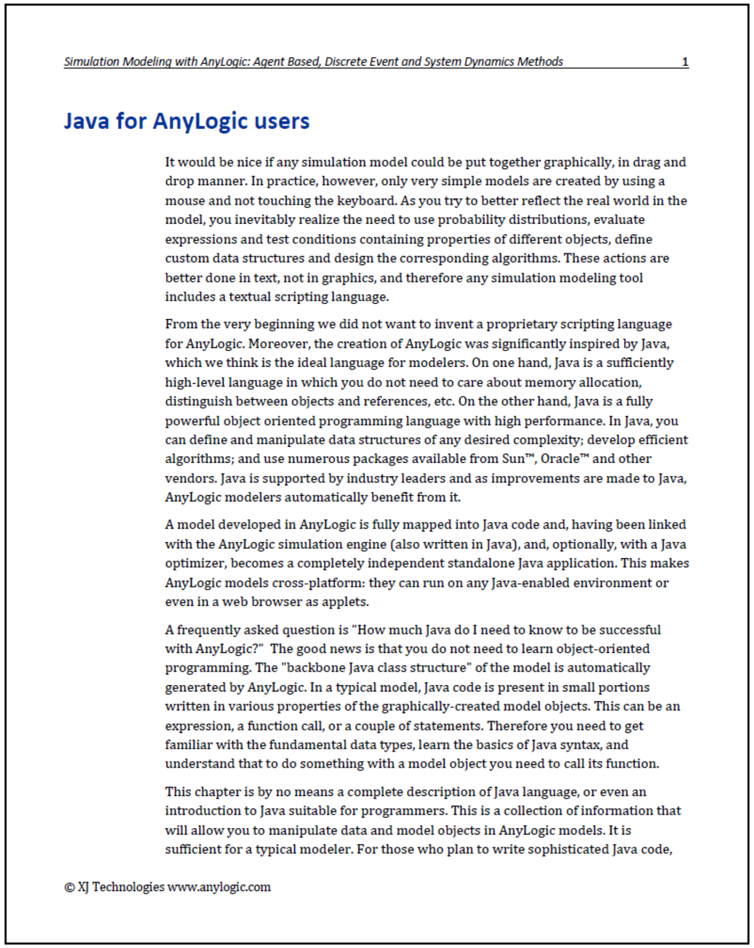
Java Basics for AnyLogic

Contents

- Java in AnyLogic
- Primitive data types
- Classes
- Variables (local variables and class fields)
- Functions (methods)
- Expressions
- Statements
- Java arrays and collections
- Comments
- Naming conventions
- Where am I and how do I get to...?
- Adding Java classes
- Adding Java interfaces

Java Basics for AnyLogic

- Book Chapter: [url]



Java for AnyLogic users

It would be nice if any simulation model could be put together graphically, in drag and drop manner. In practice, however, only very simple models are created by using a mouse and not touching the keyboard. As you try to better reflect the real world in the model, you inevitably realize the need to use probability distributions, evaluate expressions and test conditions containing properties of different objects, define custom data structures and design the corresponding algorithms. These actions are better done in text, not in graphics, and therefore any simulation modeling tool includes a textual scripting language.

From the very beginning we did not want to invent a proprietary scripting language for AnyLogic. Moreover, the creation of AnyLogic was significantly inspired by Java, which we think is the ideal language for modelers. On one hand, Java is a sufficiently high-level language in which you do not need to care about memory allocation, distinguish between objects and references, etc. On the other hand, Java is a fully powerful object oriented programming language with high performance. In Java, you can define and manipulate data structures of any desired complexity; develop efficient algorithms; and use numerous packages available from Sun™, Oracle™ and other vendors. Java is supported by industry leaders and as improvements are made to Java, AnyLogic modelers automatically benefit from it.

A model developed in AnyLogic is fully mapped into Java code and, having been linked with the AnyLogic simulation engine (also written in Java), and, optionally, with a Java optimizer, becomes a completely independent standalone Java application. This makes AnyLogic models cross-platform: they can run on any Java-enabled environment or even in a web browser as applets.

A frequently asked question is "How much Java do I need to know to be successful with AnyLogic?" The good news is that you do not need to learn object-oriented programming. The "backbone Java class structure" of the model is automatically generated by AnyLogic. In a typical model, Java code is present in small portions written in various properties of the graphically-created model objects. This can be an expression, a function call, or a couple of statements. Therefore you need to get familiar with the fundamental data types, learn the basics of Java syntax, and understand that to do something with a model object you need to call its function.

This chapter is by no means a complete description of Java language, or even an introduction to Java suitable for programmers. This is a collection of information that will allow you to manipulate data and model objects in AnyLogic models. It is sufficient for a typical modeler. For those who plan to write sophisticated Java code,

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

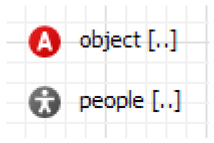
Java Basics for AnyLogic

- 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);

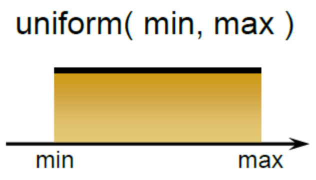


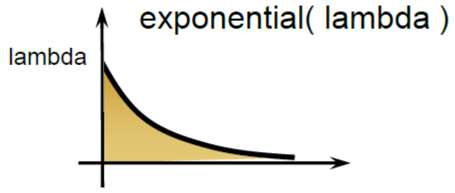
Java Basics for AnyLogic

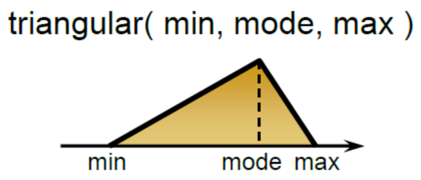
- Built-in Functions
  - System functions
    - time(); getOwner(); pause(); isStateActive(…); etc.
  - Mathematical functions
    - Basic: sqrt; sin; cos; tan; exp; log; round; zidz; xidz; etc.
    - Array: add; sub; mul; sum; avg; min; max; get; etc.
  - Special functions
    - Random numbers: uniform;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

Java Basics for AnyLogic

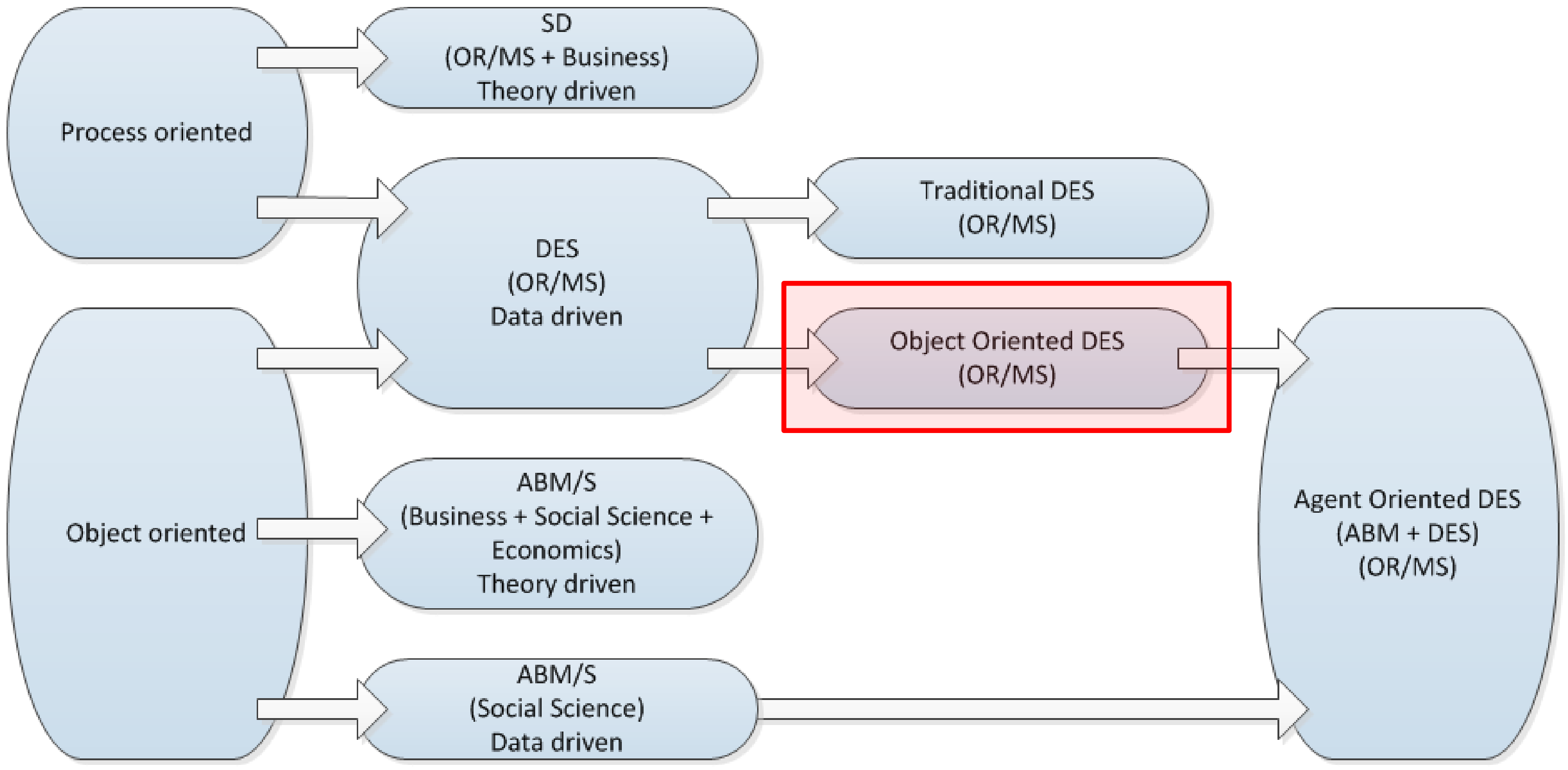
- Probability Distributions
  - **Uniform:** Used to represent a random variable with constant likelihood of being in any small interval between min and max. Its density does not depend on the value of x.
  - **Exponential:** Used to represent the time between random occurrences. The unique property 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.







Simulation Modelling Paradigms


Process oriented
SD (OR/MS + Business) Theory driven
DES (OR/MS) Data driven
Traditional DES (OR/MS)
Object Oriented DES (OR/MS)
Object oriented
ABM/S (Business + Social Science + Economics) Theory driven
ABM/S (Social Science) Data driven
Agent Oriented DES (ABM + DES) (OR/MS)

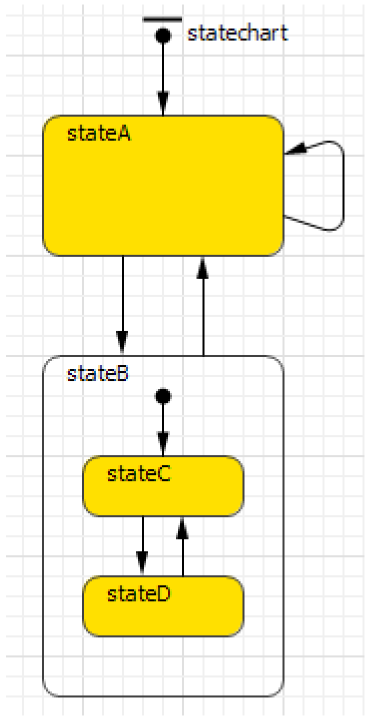
Data driven: Data for model formulation (in Social Sciences can be quantitative and qualitative); data for model validation
Theory driven: Theories for model formulation; data for model validation

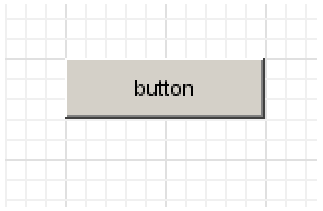
Object Oriented DES

- Laptop model: Considering different power states

Object Oriented DES

- Required modelling elements for the laptop example





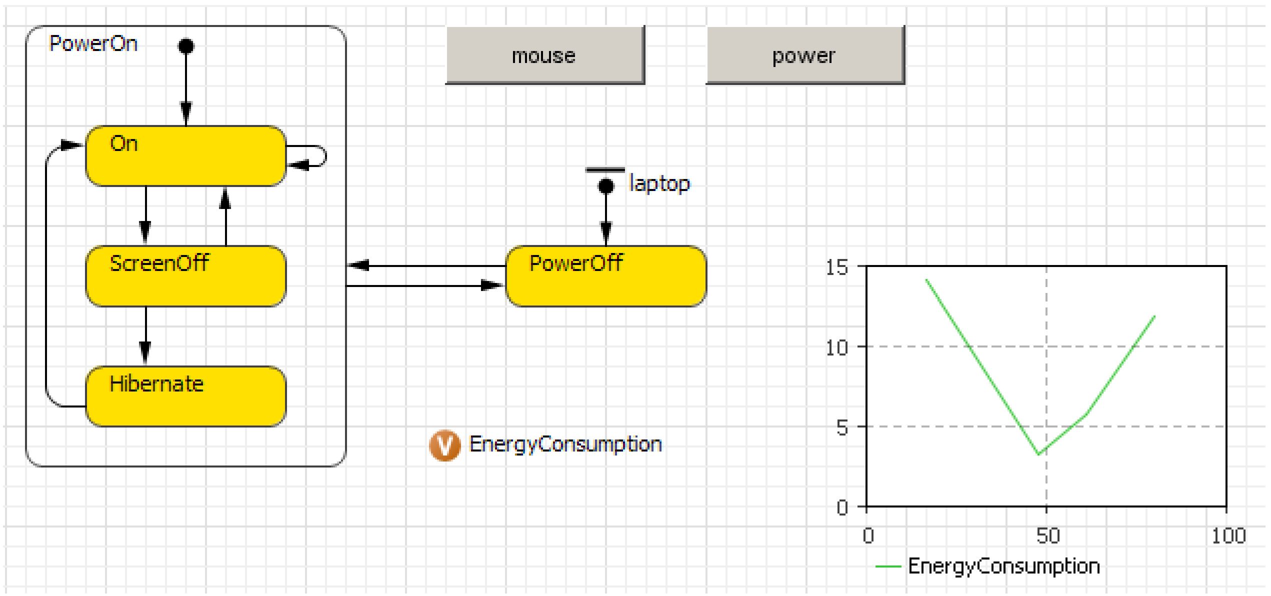
Object Oriented DES

- Modelling task
  - When the laptop is on and the user is working on it then the laptop stays in "On" mode. After 5 minutes of user inactivity the laptop switches to "Screen Off" mode to save energy. If the user remains inactive for another 10 minutes the laptop switches to "Hibernate" mode to further minimise the battery usage. To wake up the laptop from "Hibernate" mode you need to press the "Power" button.
  - In any of the three states "On", "Screen Off" and "Hibernate" the laptop consumes battery power. When the battery level falls down to 10% the laptop is forced to shut down regardless of the state.

Do not look at the next slide!

Object Oriented DES

- Design a conceptual model



Object Oriented DES

- Work in pairs:
  - Follow the tutorial provided

Questions