Aachen Summer Simulation Seminar 2014

Practice 01

Introduction to AnyLogic

Peer-Olaf Siebers

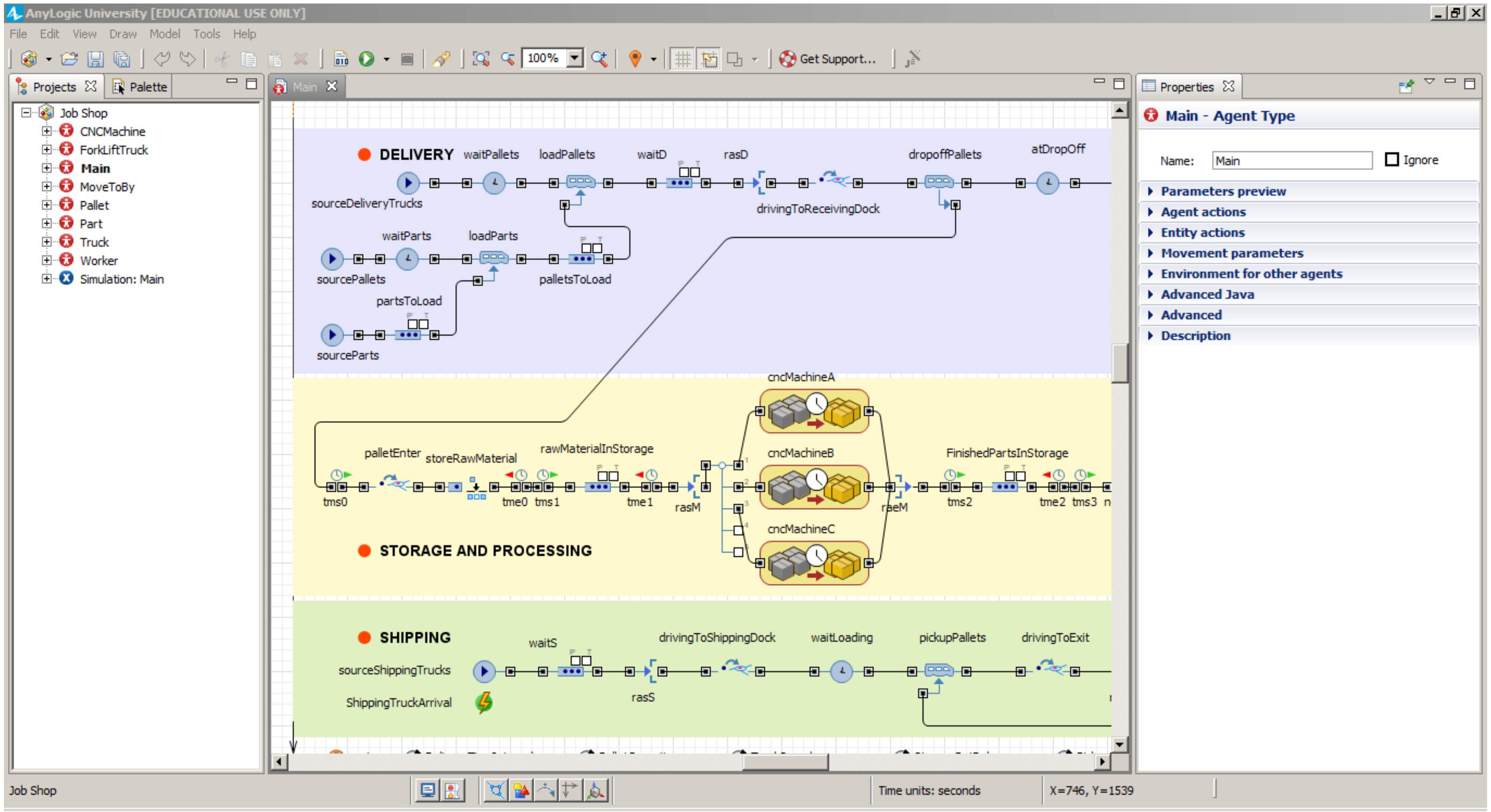
pos@cs.nott.ac.uk

The University of Nottingham
UNITED KINGDOM · CHINA · MALAYSIA

Motivation

- Provide an introduction to AnyLogic IDE
- Introduce some object oriented terminology

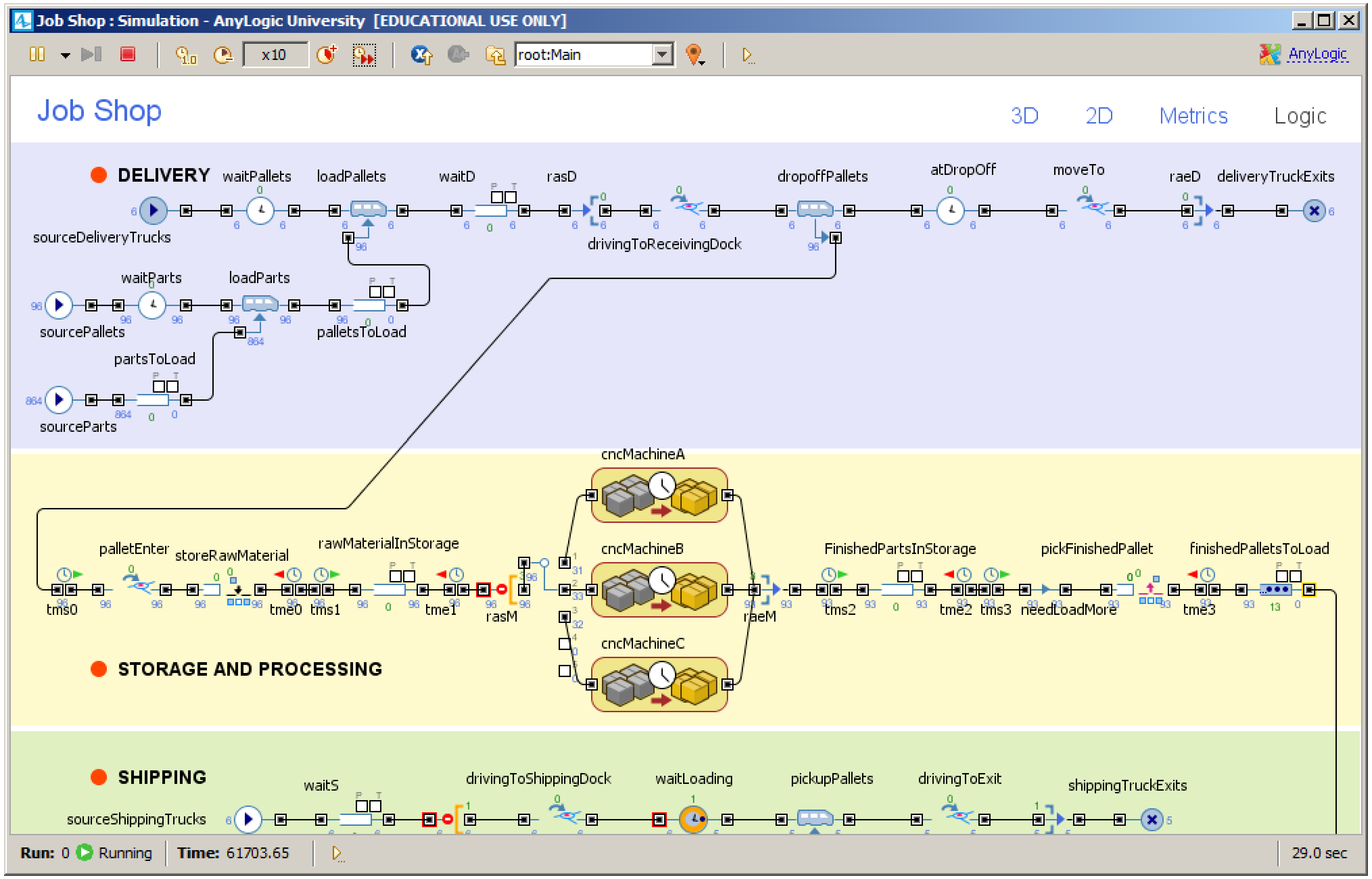
AnyLogic IDE



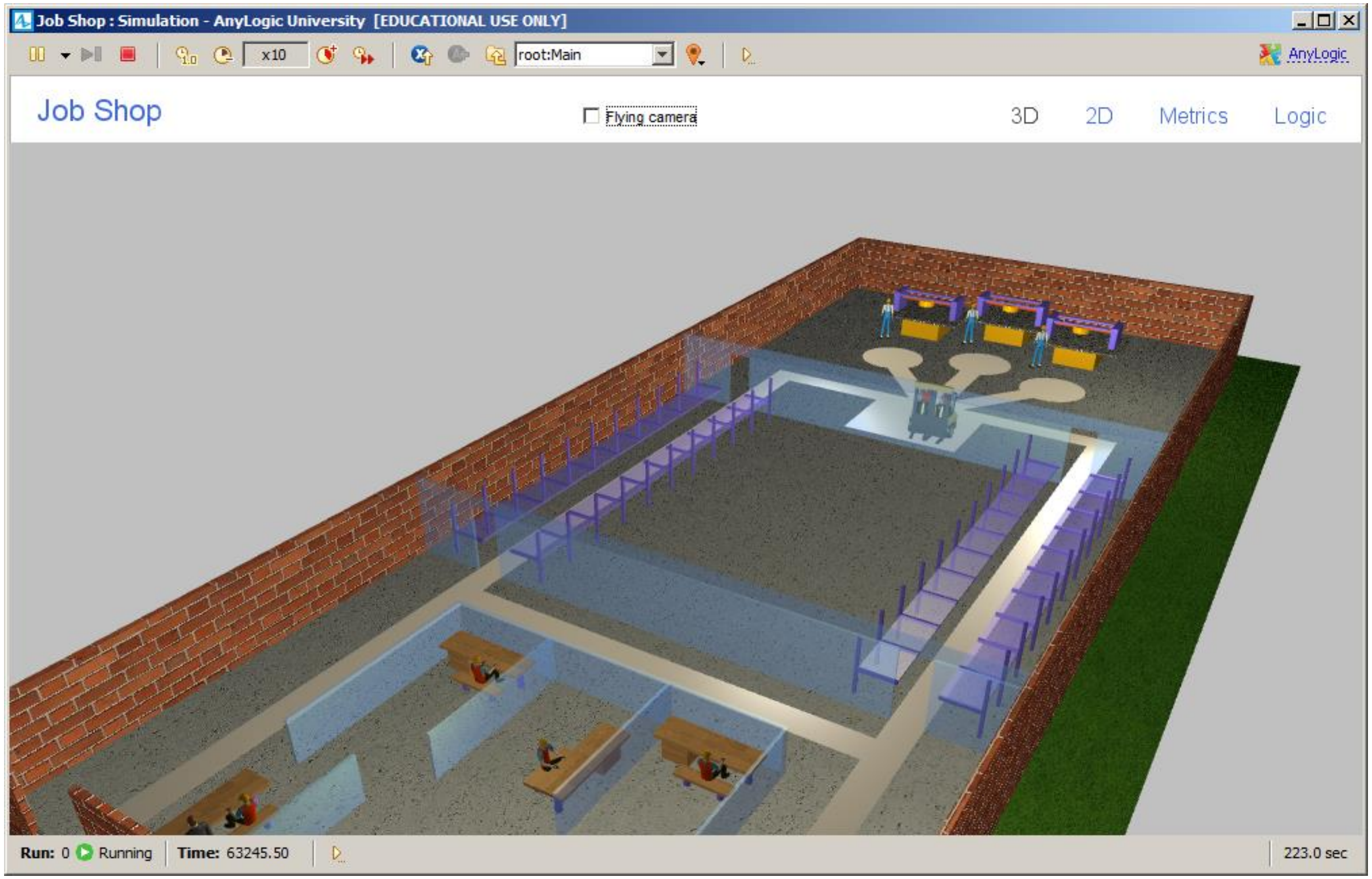
AnyLogic IDE

- Important Keys
  - F1: Help
  - Ctrl-Space: Code completion support
  - Ctrl-Enter: Perform refactoring (replace name occurrences)
  - Run: Select the correct model
  - Simulation: Set up simulation parameters

AnyLogic IDE



AnyLogic IDE



Fundamental Object Oriented Terms

http://www.cs.kent.ac.uk/people/staff/djb/oop/glossary.html

- Class
  - Programming language concept that allows data and methods to be grouped together; defines the implementation of a particular kind of objects; blueprint for objects
- Field
  - Variables defined inside a class but outside the methods; fields are members of a class
- Method
  - The part of a class definition that implements some of the behaviour of objects of the class; a function defined in a class

Fundamental Object Oriented Terms

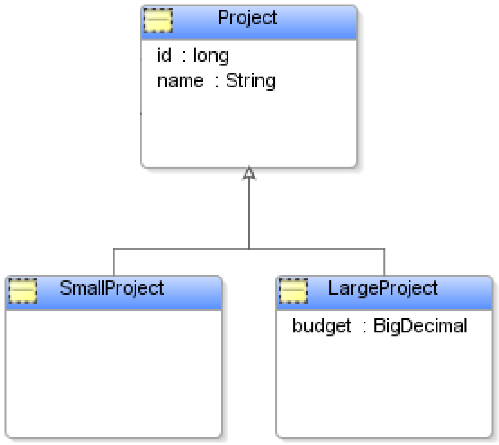
http://www.cs.kent.ac.uk/people/staff/djb/oop/glossary.html

- Object
  - An instance of a particular class. The class to which an object belongs defines the general characteristics of all instances of that class.
- Constructor
  - Creates a new instance of a class
- Instance (or replicated object)
  - An object of a particular class, created using the "new" operator followed by the class name

Fundamental Object Oriented Terms

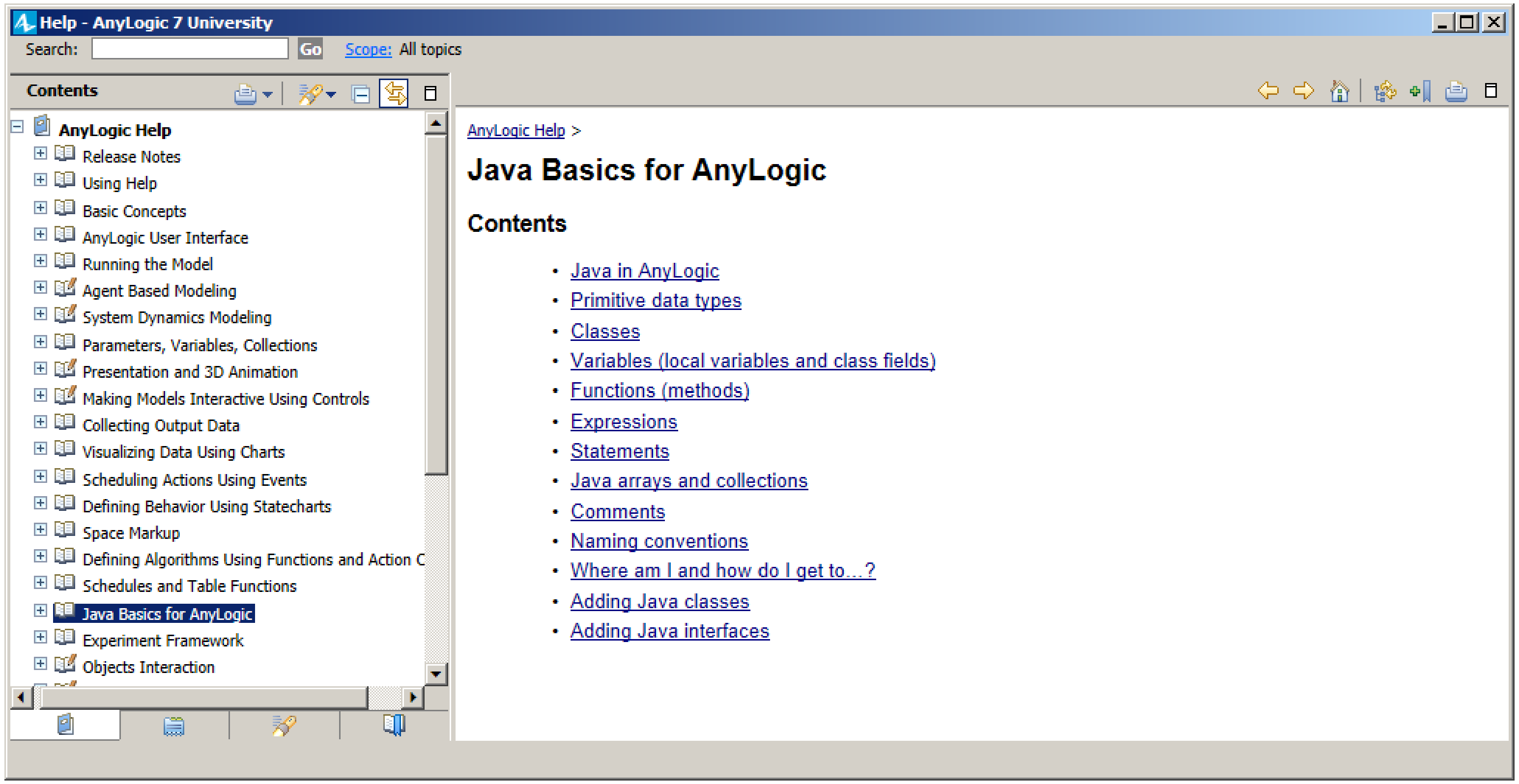
http://www.cs.kent.ac.uk/people/staff/djb/oop/glossary.html

- Inheritance
  - Concept of classes automatically contain fields and methods defined in their superclass



Java Basics for AnyLogic

- AnyLogic Help



Help - AnyLogic 7 University

Search: Go Scope: All topics

Contents

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

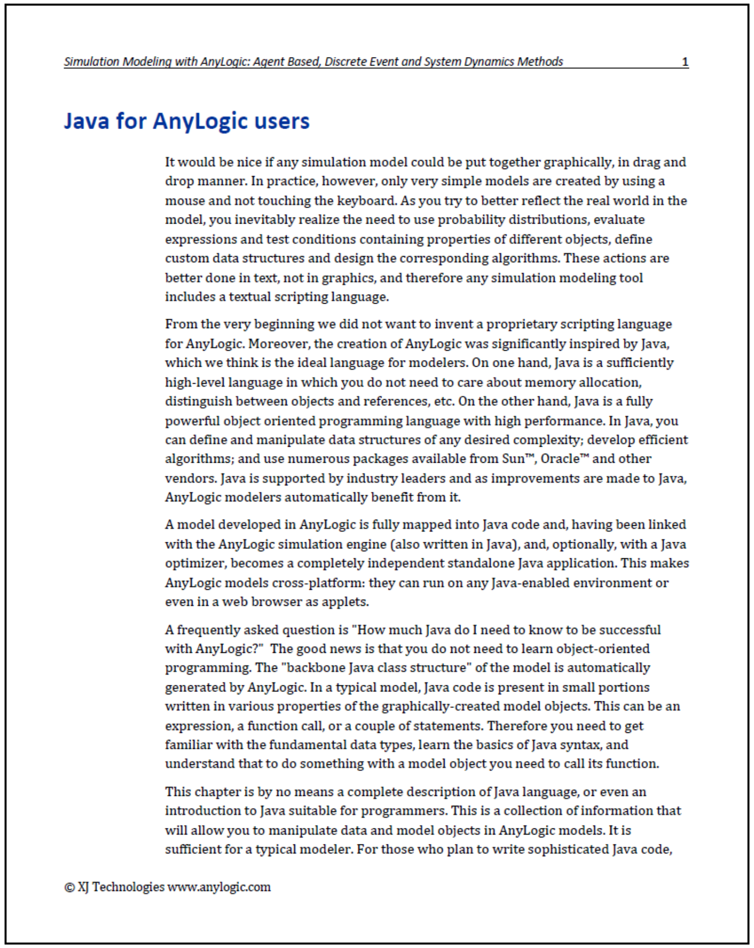
Contents

- 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

The University of Nottingham
UNITED KINGDOM · CHINA · MALAYSIA

Java Basics for AnyLogic

- Book Chapter: [url]



Simulation Modeling with AnyLogic: Agent Based, Discrete Event and System Dynamics Methods 1

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,

Questions