

# Java Basics for AnyLogic

This Presentation is a part  
of AnyLogic Standard  
Training Program



# 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 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`



# Types

---

- **Primitive Types**

- `double` – represent 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` – collections of objects
- `HyperArray` – represents multi-dimensional array in System Dynamics models
- ... many others. See AnyLogic and Java Class References



# Expressions

---

- **Arithmetic operations**

- Notation: **+**, **-**, **\***, **/**, **%** (remainder)
- In integer divisions, the fraction part is lost, e.g. **3 / 2** equals 1, and **2 / 3** equals 0
- Multiplication operators have priority over addition operators
- The '**+**' operator allows operands of type **String**

- **Comparison operations**

- Notation: **>**, **>=**, **<**, **<=**, **==**, **!=**

- **Boolean operations**

- Notation: **&&** (AND), **||** (OR), **!** (NOT)

- **Conditional operator**

- Notation: condition **?** value-if-true **:** value-if-false

- **Assignments and shortcuts**

- Notation: **=**, **+=**, **-=**, **\*=**, **/=**, **%=**, **++**, **--**
- Example: **a+=b** is equivalent to **a=a+b**



Within most of operators, left-to-right precedence holds

Parentheses may be used to alter the precedence of operations



# Some examples

- $5 \% 2 \equiv ?$

- $5 / 2 \equiv ?$

- $5. / 2 \equiv 5 / 2. \equiv ?$

- $(\text{double})5 / 2 \equiv ?$

- $a += b; \equiv ?$

- $a++; \equiv ?$

- "Any" + "Logic"  $\equiv ?$

- Let  $x = 14.3$ , then:

- "x = " + x  $\equiv ?$

- ""  $\equiv ?$

- "" + x  $\equiv ?$

- $y = x > 0 ? x : 0$

- $\equiv ?$

- $x == 5 \equiv ?$

- $x = 5 \equiv ?$

# Calling Methods and Accessing Fields

---

- **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 );
```

```
println( "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;
```



# Writing comments in Java code

---

- **There are two kinds of comments:**

`/* text */`

A traditional comment: all the text from the ASCII characters `/*` to the ASCII characters `*/` is ignored (as in C and C++)

```
/**
 * The class represents AnyLogic 3D animation. It contains the canvas object.
 *
 * @author Daniil Chunosov
 * @version 5.0
 */
public class Animation3DPanel extends javax.swing.JPanel ...
```

`// text`


A end-of-line comment: all the text from the ASCII characters `//` to the end of the line is ignored (as in C++).


```
// Prepare Engine for simulation:
engine.start( root );
engine.runFast(); // fast mode – no animation
```



# Replicated Objects

---

 objects [..]

 people [..]

- 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 );`



# 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 `Utilities`, `Presentable`, `ActiveObject` and `Agent` classes in AnyLogic Class Reference

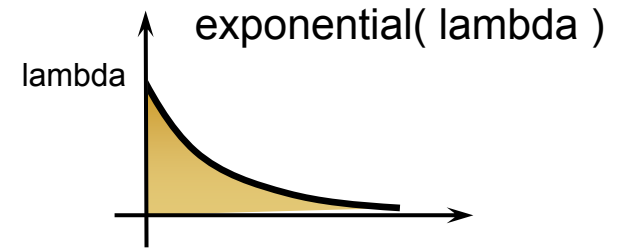
# Probability Distributions

uniform( min, max )

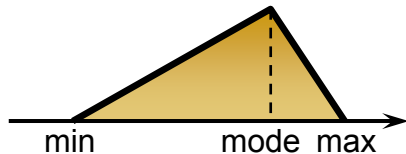


Used to represent a random variable with constant likelihood of being in any small interval between min and max. Its density does not depend on the value of  $x$ .

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.

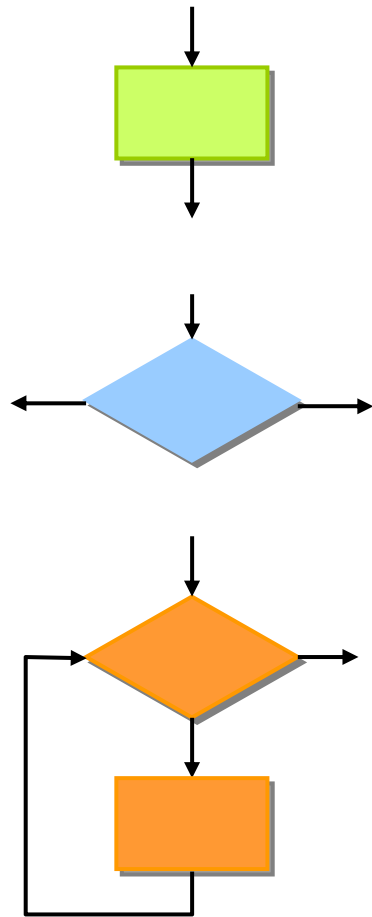


triangular( min, mode, max )



Used when no or little data is available to represent e.g. a process duration.

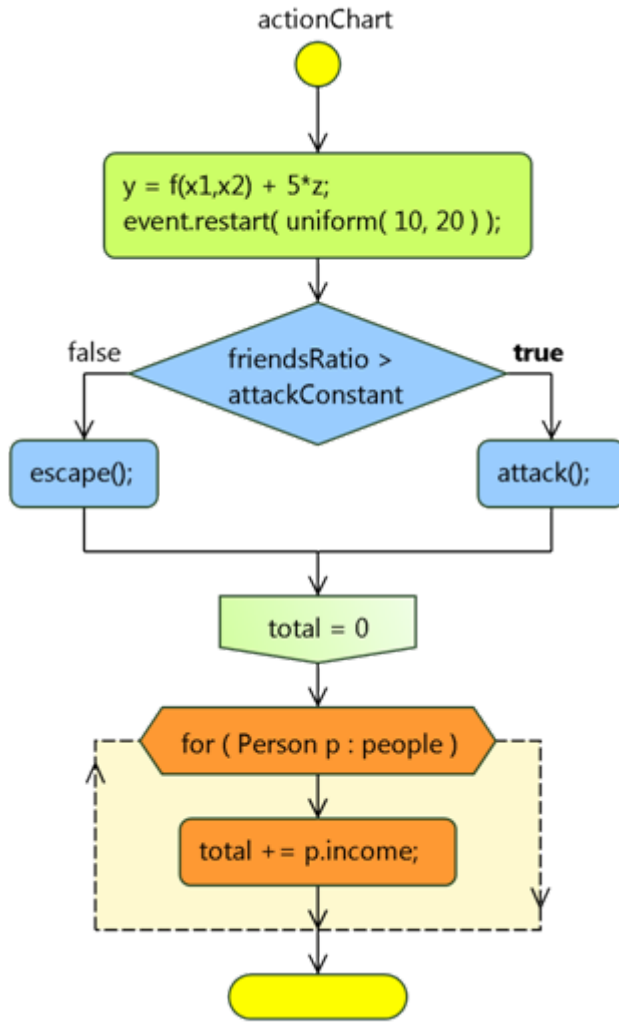
# Main Language Constructs



- **Assignment / action statement:**  
`y = f(x1,x2) + 5*z;`  
`event.restart( uniform( 10, 20 ) );`
- **Decision statement:**  
`if ( friendsRatio > attackConstant )`  
`attack();`  
`else {`  
`escape();`  
`}`
- **Loop statement:**  
`double total = 0;`  
`for ( Person p : people )`  
`total += p.income;`  
`for( int i=0; i<100; i++ )`  
`send( msg, RANDOM );`



# Main Language Constructs



- **Assignment / action statement:**  
`y = f(x1,x2) + 5*z;`  
`event.restart( uniform( 10, 20 ) );`
- **Decision statement:**  
`if ( friendsRatio > attackConstant )`  
`attack();`  
`else {`  
`escape();`  
`}`
- **Loop statement:**  
`double total = 0;`  
`for ( Person p : people )`  
`total += p.income;`  
`for( int i=0; i<100; i++ )`  
`send( msg, RANDOM );`



# Collections

- Collection framework is a set of classes representing basic data structures
- These classes have different timing characteristics of different operations

	ArrayList Vector	LinkedList	HashSet HashMap	SortedSet SortedMap
Size	<b>Const</b>	<b>Const</b>	<b>Const</b>	<b>Const</b>
Add an item	<b>Const</b>	<b>Const</b>	<b>Const</b>	<b>Log</b>
Remove found item	<b>Linear</b>	<b>Linear</b>	<b>Const</b>	<b>Log</b>
Remove by index	<b>Linear</b>	<b>Linear</b>	-	-
Get by index/random	<b>Const</b>	<b>Linear</b>	-	-
Find an item	<b>Linear</b>	<b>Linear</b>	<b>Const</b>	<b>Log</b>
Get smallest/largest	<b>Linear</b>	<b>Linear</b>	<b>Linear</b>	<b>Const</b>



# Reference Information Sources

---

- **AnyLogic Class Reference**
  - [API Reference](#) section of [AnyLogic Help](#)
- **Java API Specification**
  - Open <http://java.sun.com/>
- **Best book on Java:**
  - Bruce Eckel. Thinking in Java.  
Available online at: <http://www.mindview.net/Books/TIJ>

