LOGO Instructor README

Table of Contents

1. PROJECT OUTLINE / SPECIFICATION

You must use Java to do the following project.

Main:

Create a LOGO like instruction set language in XML.

Use whatever or specific design patterns you see fit.

Parse it in with some form of ORM and factory to make a singleton LOGO like robot execute the commands by showing it is receiving each command and taking action.

The test script in XML should be designed to output Diagram1.

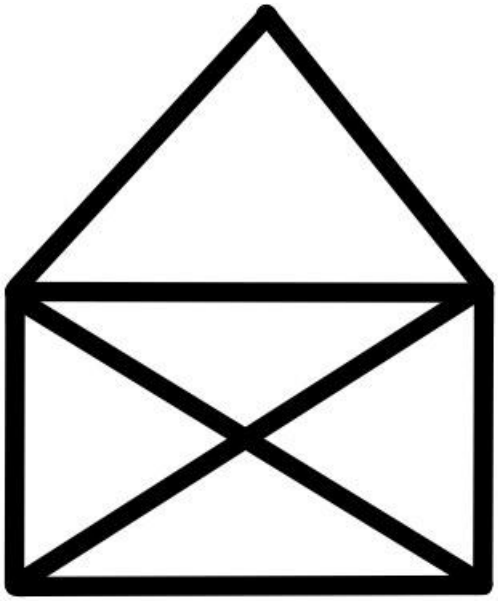
Bonus:

As a bonus use a graphics or paint library to actually draw Diagram1 from the commands received.

To submit your project you will need to create a gitHub account and commit your project.

For us to check it you must supply us with the gitHub url to clone it.

The repository must be fully contained, so when it is cloned all source and instructions to get it to work as well as all data needs to be all inclusive.



2. SYSTEM REQUIREMENTS

A reasonably spec'ed developer machine would be more than sufficient to run this application.

Java Runtime Environment 1.6. (or higher) installed and configured.

More information here:

http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase6-419409.html

3. DEVELOPMENT ENVIRONMENT AND TOOLS

3.1. Environment

- Windows 7
- Java Platform, Standard Edition 6.0

3.2. Tools

- Eclipse Indigo
- Altova XMLSpy

Notes:

The application was developed and only tested in a Windows environment. Although all considerations were made to ensure cross platform portability it must be noted that it was only tested in an Windows environment.

The JAXB 2.0 dependencies are bundled with Java SE 6 and therefore not included in the JAR. More info available here:

http://www.oracle.com/technetwork/articles/marx-jse6-090753.html

4. INSTALL, CONFIGURE AND LAUNCH THE APPLICATION

4.1. Installation

Step 1:

Navigate to the repository on GitHub and download the application:

https://github.com/romeod75/instructionset/zipball/master

Or here and select your preferred download type:

https://github.com/romeod75/instructionset/downloads

Step 2:

Extract to your PC and see README.pdf for further instructions and more info.

4.1.1. Directory Structure

USER_DEFINED_ROOT
| *(Documentation)*
→ **/doc**
| *(Instructions set XML examples)*
→ **/examples**
| *(Java source files)*
→ **/src**
| *(Application JAR)*
→ **LOGOInstructor.jar**
| *(Instruction set schema)*
→ **instructionSet.xsd**
| *(Some default properties)*
→ **config.properties**

4.2. Default configuration

Config.properties

The following values will be used if any of the parameters are omitted when the application is launched. It can be changed as needed.

```
# args[0] Can contain relative (to LOGOInstructor.jar) or absolute path
instructionSetFilename=/examples/multichoice.xml

# args[1] Time in millis to wait between executing each command and action
waitFor=1000

# Can contain relative (to LOGOInstructor.jar) or absolute path
# Used by application and not passed and a parameter to the LOGOInstructor.main()
xsdFilename=/instructionSet.xsd
```

4.3. Launch the application

4.3.1. Usage

From the USER_DEFINED_ROOT directory where the LOGOInstructor.jar is located:

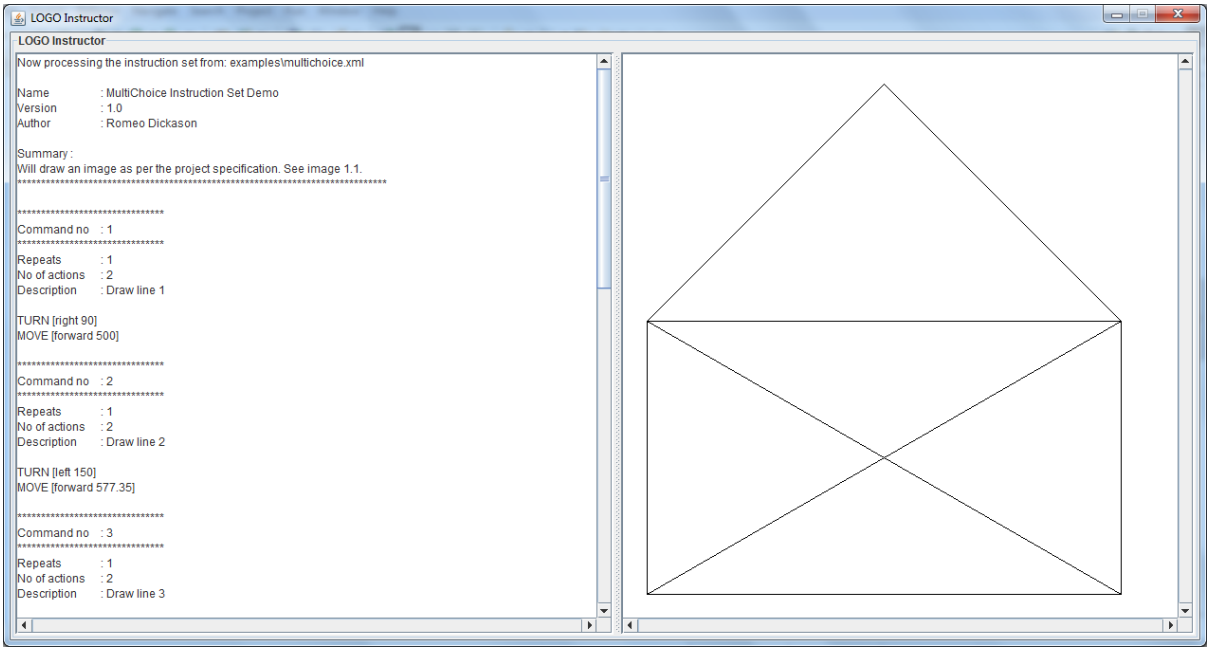
java -jar LOGOInstructor.jar [<path>/**filename**] [**wait**]

4.4. The default example

java -jar LOGOInstructor.jar

Output:

Will execute. USER_DEFINED_ROOT /examples/multichoice.xml (from config.properties) with an waiting period of 1000 milliseconds (from config.properties) between the processing of every command and action.

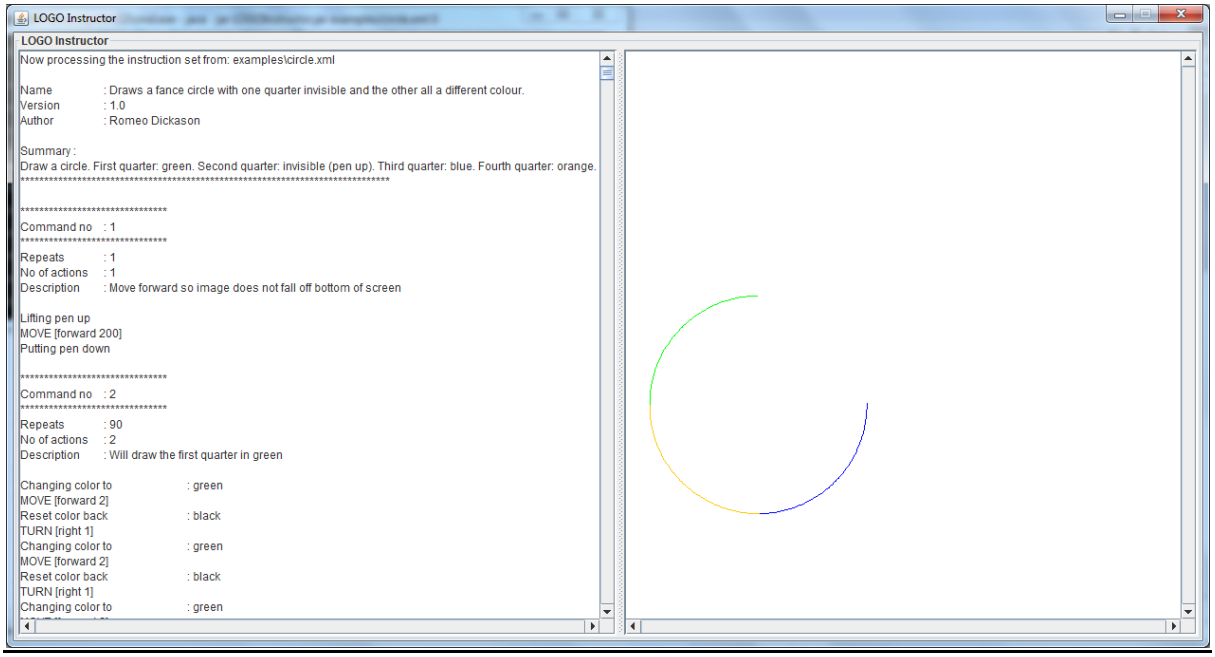


4.5. The circle example

java -jar LOGOInstructor.jar examples/circle.xml 250

Output:

Draws a fance circle with one quarter invisible and the other all a different colour with an interval of 250 milliseconds between executions.

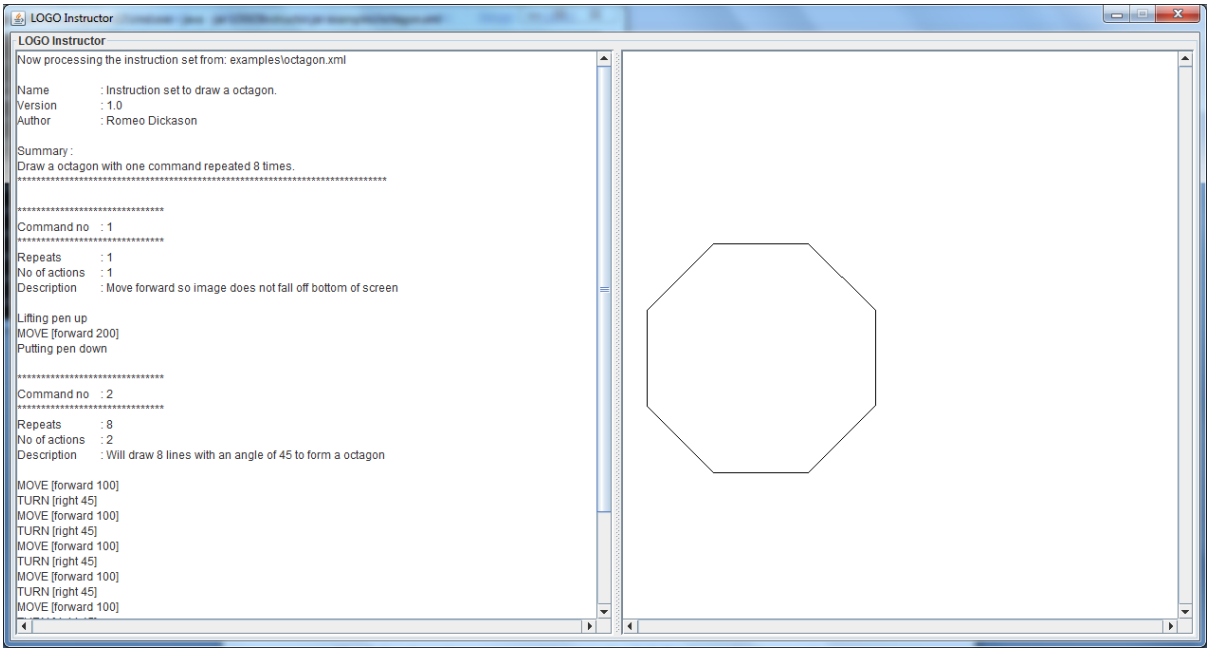


4.6. The octagon example

java -jar LOGOInstructor.jar examples/octagon.xml 0

Output:

Draws an octagon with one command repeated 8 times and no interval between executions.

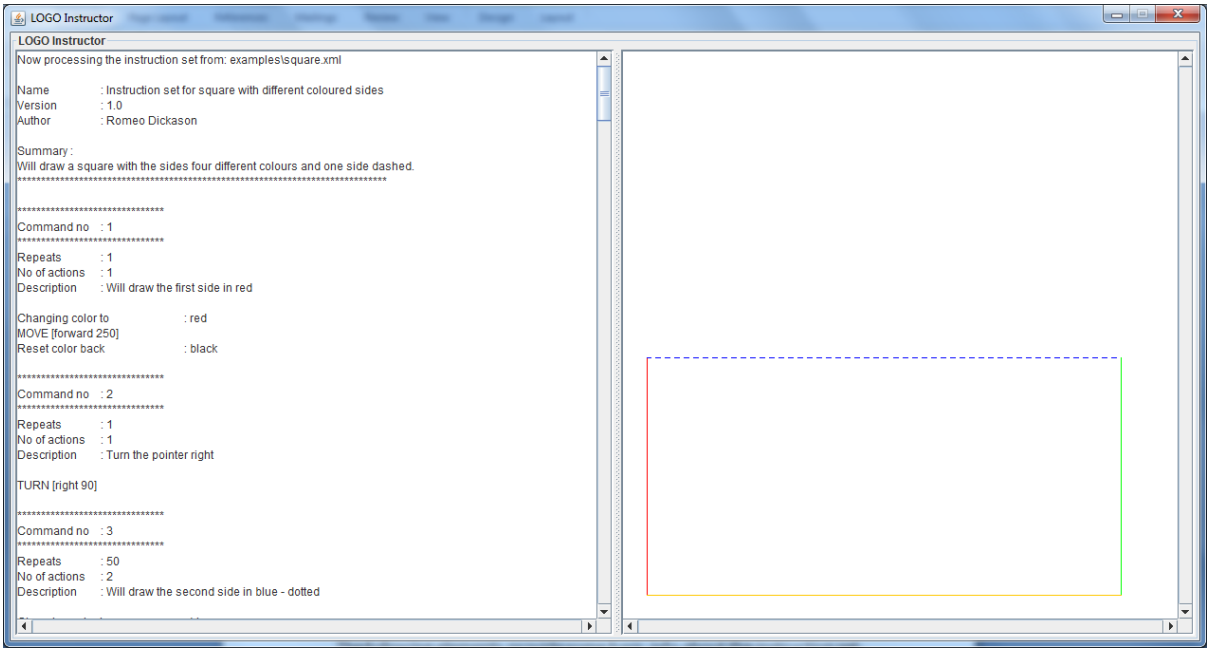


4.7. The square example

java -jar LOGOInstructor.jar examples/square.xml

Output:

Will draw a square with the sides four different colours and one side dashed and the default interval of 1000 milliseconds.



5. THE SCHEMA FOR AN INSTRUCTION SET

Provides an API and describes and controls the syntax elements and parameters for an Instruction set.

5.1. Basic elements

The following elements provide some basic info about the instruction set for reference purposes.

```
<!-- ********************************************************************** -->
<!-- root element instructionSet -->
<!-- ********************************************************************** -->
<xsd:element name="instructionSet">
	<xsd:complexType>
		<xsd:sequence>
			<xsd:element ref="instructionSetName"/>
			<xsd:element ref="version"/>
			<xsd:element ref="author"/>
			<xsd:element ref="instructionSetDescription"/>
```

5.2. The Command element

The following elements contain the command (one or more) with a description of the command(s) and if so how many times to repeat. A specific command must contain one or more actions described in more detail below.

```
			<xsd:element name="commands" type="Commands"/>
		</xsd:sequence>
	</xsd:complexType>
</xsd:element>
<!-- ********************************************************************** -->
<!-- Commands -->
<!-- ********************************************************************** -->
<xsd:complexType name="Commands">
	<xsd:sequence>
		<xsd:element name="command" minOccurs="1" maxOccurs="unbounded">
			<xsd:complexType>
				<xsd:sequence>
					<xsd:element name="commandDescription" .../>
					<!-- [Optional] How many times to repeat ...-->
					<xsd:element name="repeat" minOccurs="0">
						<xsd:simpleType>
							<xsd:restriction ...>
								<xsd:minInclusive value="2"/>
							</xsd:restriction>
						</xsd:simpleType>
					</xsd:element>
```

5.2.1. Command rules and examples

A command could contain a single action to lift the pen and move the cursor to a position on the screen to ensure the image doesn't fall off the screen. (See octagon.xml)

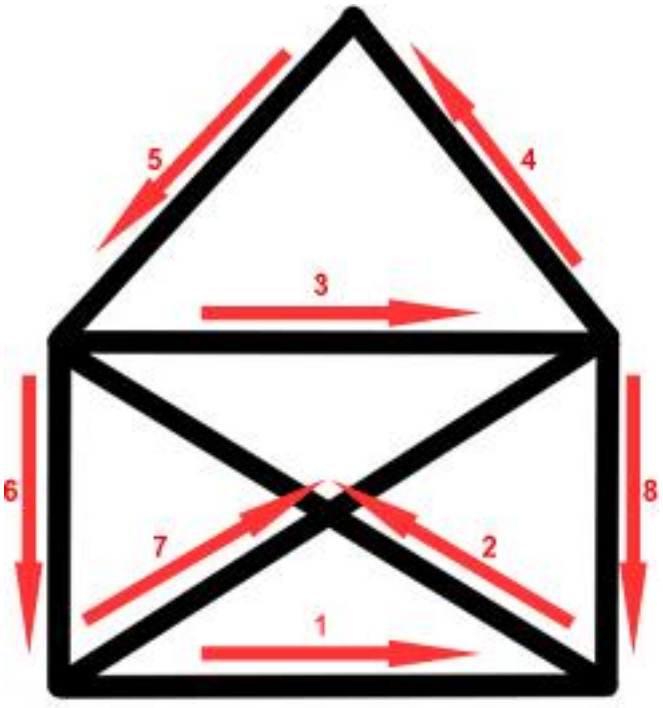
A command could be repeated 8 times with 2 actions of a move forward and a turn at an angle of 45° that will result in an octagon. (See octagon.xml)

A command could contain 4 commands repeated 90 times with 2 actions, one move forward and 1° turn to draw each quarter of a circle in a different colour. Or a single command repeated 360 times with 2 actions, a move forward and a 1° turn to draw a full circle. (See circle.xml)

Again an instruction set could contain 4 commands to draw the 4 sides of a square in different colour or 1

command repeated four times to draw the sides in one colour. (See square.xml)

The example requested by the specification requires 8 commands, with 2 actions each that includes a turn and a move forward to draw the image. (See multichoice.xml)



5.3. The Action element

A command can have 2 but must have at least one action and an action can either be a move or a turn, or a combination of both, in the desired order depending on what's needed and is described by the following elements:

```
                              <xsd:element name="actions"type="Actions"/>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>
      </xsd:sequence>
</xsd:complexType>
<!-- ********************************************************************** -->
<!-- Actions -->
<!-- ********************************************************************** -->
<xsd:complexType name="Actions">
      <xsd:annotation>
            <xsd:documentation>At least one of move or tur ...</xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
            <xsd:element name="action" minOccurs="1" maxOccurs="unbounded">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="move" type="moveType" .../>
                              <xsd:element name="turn" type="turnType" .../>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>
      </xsd:sequence>
</xsd:complexType>
```

5.3.1. Action rules, combinations and examples

A command must have at least one but could have 2 actions to be used as follows:

- A single move or turn
- A combination of move and turn. Turn first then move, or move first then turn.

5.3.1.1. The Move action:

A move action must have at least the direction and the distance but can also change the pen colour or move invisible which could be used to move to a position on the canvas.

> Notes:
>
> Color and invisible will be reset, to black and pendown after each move action.
>
> The starting point on the canvas is the bottom left corner pointing 0°.

```
<!-- ********************************************************************** -->
<!-- Move -->
<!-- ********************************************************************** -->
<xsd:complexType name="moveType">
      <xsd:sequence>
            <!-- [Mandatory] where? -->
            <xsd:element name="direction">
                  <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                              <xsd:enumeration value="forward"/>
                              <xsd:enumeration value="back"/>
                        </xsd:restriction>
                  </xsd:simpleType>
            </xsd:element>
            <!-- [Mandatory] how far? -->
            <xsd:element name="distance">
                  <xsd:simpleType>
                        <xsd:restriction base="xsd:decimal">
                              <xsd:minInclusive value="1"/>
                        </xsd:restriction>
                  </xsd:simpleType>
            </xsd:element>
            <!-- [Optional] Do not output to screen -->
            <xsd:element name="invisible" minOccurs="0">
                  <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                              <xsd:enumeration value="Y"/>
                        </xsd:restriction>
                  </xsd:simpleType>
            </xsd:element>
            <!-- [Optional] Use color other than black -->
```

```
            <xsd:element name="color" minOccurs="0">
                  <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                              <xsd:enumeration value="blue"/>
                              <xsd:enumeration value="green"/>
                              <xsd:enumeration value="orange"/>
                              <xsd:enumeration value="red"/>
                              <xsd:enumeration value="yellow"/>
                        </xsd:restriction>
                  </xsd:simpleType>
            </xsd:element>
      </xsd:sequence>
</xsd:complexType>
```

5.3.1.2. The Turn action:

Simply turn the pointer left or right at the specified angle.

```
<!-- ********************************************************************** -->
<!-- Turn -->
<!-- ********************************************************************** -->
<xsd:complexType name="turnType">
      <xsd:sequence>
            <!-- [Manadtory] Which direction -->
            <xsd:element name="direction">
                  <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                              <xsd:enumeration value="left"/>
                              <xsd:enumeration value="right"/>
                        </xsd:restriction>
                  </xsd:simpleType>
            </xsd:element>
            <!-- [Manadtory] What angel -->
            <xsd:element name="angle">
                  <xsd:simpleType>
                        <xsd:restriction base="xsd:positiveInteger">
                              <xsd:minInclusive value="1"/>
                              <xsd:maxInclusive value="360"/>
                        </xsd:restriction>
                  </xsd:simpleType>
            </xsd:element>
      </xsd:sequence>
</xsd:complexType>
```

6. TROUBLESHOOTING

Please contact Romeo Dickason on 076-824-5325