Федеральное государственное бюджетное образовательное учреждение   
высшего образования

|  |  |
| --- | --- |
| Gerb-BMSTU_01 | «Московский государственный технический университет  им. Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана) |

ФАКУЛЬТЕТ – Информатика и управления

КАФЕДРА – Информационные системы и телекоммуникации

РАСЧЁТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе на тему

«Разработка примера переноса графических образов из библиотеки jgraphx в GEF4»

Студент группы ИУ3-72 (подпись) 13.02.2017 Герасимов С.А.

Руководитель курсовой работы (подпись) 13.02.2017 А. М. Иванов

Москва, 2017

Код разработанной программы XMLReader.java:

**package com.bmstu.coursework.gef4;**

**import java.io.FileReader;**

**import javax.xml.namespace.QName;**

**import javax.xml.stream.XMLEventReader;**

**import javax.xml.stream.XMLInputFactory;**

**import javax.xml.stream.XMLStreamConstants;**

**import javax.xml.stream.events.EndElement;**

**import javax.xml.stream.events.StartElement;**

**import javax.xml.stream.events.XMLEvent;**

**import org.eclipse.gef4.fx.nodes.GeometryNode;**

**import org.eclipse.gef4.fx.nodes.InfiniteCanvas;**

**import org.eclipse.gef4.geometry.euclidean.Angle;**

**import org.eclipse.gef4.geometry.planar.Arc;**

**import org.eclipse.gef4.geometry.planar.Ellipse;**

**import org.eclipse.gef4.geometry.planar.Line;**

**import org.eclipse.gef4.geometry.planar.Rectangle;**

**import org.eclipse.gef4.geometry.planar.RoundedRectangle;**

**import javafx.application.Application;**

**import javafx.scene.Node;**

**import javafx.scene.Scene;**

**import javafx.scene.control.Label;**

**import javafx.scene.paint.Color;**

**import javafx.stage.Stage;**

**/\*\***

**\***

**\***

**\* Instance of this class reads data and draws shapes**

**\***

**\*/**

**public class XmlReader**

**extends Application {**

**private static final String HEIGHT = "h";**

**private static final String WIDTH = "w";**

**private static final String RADIUS\_Y = "ry";**

**private static final String RADIUS\_X = "rx";**

**private static final String NAME = "name";**

**private static final String Y\_COORDINATE = "y";**

**private static final String X\_COORDINATE = "x";**

**private static final int ARC\_SIZE = 10;**

**private static final int SHAPE\_OFFSET = 120;**

**private static final int FIGURES\_OFFSET\_Y = 20;**

**private static final String FILE\_NAME = "shapesMin.txt";**

**private static final String TITLE = "Coursework";**

**private InfiniteCanvas canvas;**

**private int currentPositionX;**

**private int currentPositionY;**

**private int xOffset;**

**public static void main(String[] args) {**

**Application.launch(args);**

**}**

**@Override**

**public void start(Stage primaryStage) throws Exception {**

**primaryStage.setScene(createScene());**

**primaryStage.sizeToScene();**

**primaryStage.setTitle(TITLE);**

**primaryStage.show();**

**}**

**private Scene createScene() {**

**canvas = new InfiniteCanvas();**

**Scene scene = new Scene(canvas, 600, 200);**

**readXMLFile();**

**canvas.setVerticalScrollOffset(FIGURES\_OFFSET\_Y);**

**return scene;**

**}**

**private void readXMLFile() {**

**XMLInputFactory factory = XMLInputFactory.newInstance();**

**try**

**{**

**XMLEventReader eventReader = factory.createXMLEventReader(new FileReader(FILE\_NAME));**

**while (eventReader.hasNext())**

**{**

**XMLEvent event = eventReader.nextEvent();**

**switch (event.getEventType()) {**

**case XMLStreamConstants.START\_ELEMENT:**

**processStartElement(event.asStartElement());**

**break;**

**case XMLStreamConstants.END\_ELEMENT:**

**processEndElement(event.asEndElement());**

**break;**

**default:**

**break;**

**}**

**}**

**}**

**catch (Exception e)**

**{**

**// Do nothing**

**}**

**}**

**private void processStartElement(StartElement startElement) {**

**String startElementName = startElement.getName().getLocalPart();**

**if ("move".equals(startElementName))**

**{**

**currentPositionX = Integer.valueOf(startElement.getAttributeByName(new QName(X\_COORDINATE)).getValue());**

**currentPositionY = Integer.valueOf(startElement.getAttributeByName(new QName(Y\_COORDINATE)).getValue());**

**return;**

**}**

**else**

**{**

**Node shape = null;**

**switch (startElementName) {**

**case "shape":**

**processShapeStart(startElement);**

**break;**

**case "arc":**

**shape = createArc(startElement);**

**break;**

**case "line":**

**shape = createLine(startElement);**

**break;**

**case "ellipse":**

**shape = createEllipse(startElement);**

**break;**

**case "rect":**

**shape = createRectangle(startElement);**

**break;**

**case "roundrect":**

**shape = createRoundRectangle(startElement);**

**break;**

**default:**

**break;**

**}**

**if (shape != null)**

**{**

**canvas.getContentGroup().getChildren().add(shape);**

**}**

**}**

**}**

**private void processEndElement(EndElement endElement) {**

**String endElementName = endElement.getName().getLocalPart();**

**if ("shape".equals(endElementName))**

**{**

**xOffset += SHAPE\_OFFSET;**

**}**

**}**

**private void processShapeStart(StartElement startElement) {**

**String name = startElement.getAttributeByName(new QName(NAME)).getValue();**

**Label nameLabel = new Label(name);**

**nameLabel.setLayoutX(xOffset);**

**nameLabel.setLayoutY(-FIGURES\_OFFSET\_Y);**

**canvas.getContentGroup().getChildren().add(nameLabel);**

**}**

**private Node createArc(StartElement startElement) {**

**int rx = Integer.valueOf(startElement.getAttributeByName(new QName(RADIUS\_X)).getValue());**

**int ry = Integer.valueOf(startElement.getAttributeByName(new QName(RADIUS\_Y)).getValue());**

**GeometryNode<Arc> arc = new GeometryNode<>(new Arc(currentPositionX + xOffset - rx, currentPositionY - ry,**

**2 \* rx, 2 \* ry, Angle.fromDeg(0), Angle.fromDeg(180)));**

**return arc;**

**}**

**private Node createLine(StartElement startElement) {**

**int x = Integer.valueOf(startElement.getAttributeByName(new QName(X\_COORDINATE)).getValue());**

**int y = Integer.valueOf(startElement.getAttributeByName(new QName(Y\_COORDINATE)).getValue());**

**GeometryNode<Line> line =**

**new GeometryNode<>(new Line(currentPositionX + xOffset, currentPositionY, x + xOffset, y));**

**currentPositionX = x;**

**currentPositionY = y;**

**return line;**

**}**

**private Node createEllipse(StartElement startElement) {**

**int x = Integer.valueOf(startElement.getAttributeByName(new QName(X\_COORDINATE)).getValue());**

**int y = Integer.valueOf(startElement.getAttributeByName(new QName(Y\_COORDINATE)).getValue());**

**int width = Integer.valueOf(startElement.getAttributeByName(new QName(WIDTH)).getValue());**

**int height = Integer.valueOf(startElement.getAttributeByName(new QName(HEIGHT)).getValue());**

**GeometryNode<Ellipse> ellipse = new GeometryNode<>(new Ellipse(xOffset + x, y, width, height));**

**ellipse.setFill(Color.WHITE);**

**ellipse.setStroke(Color.BLACK);**

**return ellipse;**

**}**

**private Node createRectangle(StartElement startElement) {**

**int x = Integer.valueOf(startElement.getAttributeByName(new QName(X\_COORDINATE)).getValue());**

**int y = Integer.valueOf(startElement.getAttributeByName(new QName(Y\_COORDINATE)).getValue());**

**int width = Integer.valueOf(startElement.getAttributeByName(new QName(WIDTH)).getValue());**

**int height = Integer.valueOf(startElement.getAttributeByName(new QName(HEIGHT)).getValue());**

**GeometryNode<Rectangle> rectangle = new GeometryNode<>(new Rectangle(xOffset + x, y, width, height));**

**rectangle.setFill(Color.WHITE);**

**rectangle.setStroke(Color.BLACK);**

**return rectangle;**

**}**

**private Node createRoundRectangle(StartElement startElement) {**

**int x = Integer.valueOf(startElement.getAttributeByName(new QName(X\_COORDINATE)).getValue());**

**int y = Integer.valueOf(startElement.getAttributeByName(new QName(Y\_COORDINATE)).getValue());**

**int width = Integer.valueOf(startElement.getAttributeByName(new QName(WIDTH)).getValue());**

**int height = Integer.valueOf(startElement.getAttributeByName(new QName(HEIGHT)).getValue());**

**GeometryNode<RoundedRectangle> roundedRectangle =**

**new GeometryNode<>(new RoundedRectangle(xOffset + x, y, width, height, ARC\_SIZE, ARC\_SIZE));**

**roundedRectangle.setFill(Color.WHITE);**

**roundedRectangle.setStroke(Color.BLACK);**

**return roundedRectangle;**

**}**

**}**

shapesMin.txt

<shapes>

<!-- Timer Intermediate -->

<shape name="timer" aspect="fixed">

<foreground>

<ellipse x="10" y="10" w="80" h="80"/>

<stroke/>

<path>

<move x="10" y="50"/>

<line x="16" y="50"/>

<move x="50" y="10"/>

<line x="50" y="16"/>

<move x="29" y="16"/>

<line x="32" y="21"/>

<move x="71" y="16"/>

<line x="68" y="21"/>

<move x="16" y="29"/>

<line x="21" y="32"/>

<move x="16" y="71"/>

<line x="21" y="68"/>

<move x="90" y="50"/>

<line x="84" y="50"/>

<move x="50" y="90"/>

<line x="50" y="84"/>

<move x="71" y="84"/>

<line x="68" y="79"/>

<move x="29" y="84"/>

<line x="32" y="79"/>

<move x="84" y="71"/>

<line x="79" y="68"/>

<move x="84" y="29"/>

<line x="79" y="32"/>

<move x="50" y="50"/>

<line x="52" y="17"/>

<move x="50" y="50"/>

<line x="70" y="50"/>

</path>

<stroke/>

</foreground>

</shape>

<shape name="cross" aspect="fixed">

<foreground>

<strokecolor color="#000000"/>

<strokewidth width="6"/>

<linecap cap="round"/>

<path crisp="0">

<move x="25" y="25"/>

<line x="75" y="75"/>

</path>

<stroke/>

<path>

<move x="75" y="25"/>

<line x="25" y="75"/>

</path>

<stroke/>

</foreground>

</shape>

<!-- Electronics - Transitors - MOSFET\_P\_CH\_DEP -->

<shape name="MOSFET\_P\_CH\_DEP" aspect="fixed">

<connections>

<constraint x="0.8" y="0" perimeter="0"/>

<constraint x="0.8" y="1" perimeter="0"/>

<constraint x="0.1" y="0.5" perimeter="0"/>

</connections>

<foreground>

<strokewidth width = "4"/>

<path crisp="1">

<move x="10" y="50"/>

<line x="40" y="50"/>

<move x="40" y="20"/>

<line x="40" y="80"/>

<move x="50" y="10"/>

<line x="50" y="90"/>

<move x="50" y="20"/>

<line x="80" y="20"/>

<line x="80" y="0"/>

<move x="50" y="50"/>

<line x="80" y="50"/>

<line x="80" y="100"/>

<move x="50" y="80"/>

<line x="80" y="80"/>

</path>

<stroke/>

<path>

<move x="80" y="50"/>

<line x="72" y="46"/>

<line x="72" y="54"/>

<line x="80" y="50"/>

</path>

<fillstroke/>

<ellipse x="78" y="78" w="4" h="4"/>

<fillstroke/>

</foreground>

</shape>

<shape name="gate" aspect="fixed">

<connections>

<constraint x="0.1" y="0.1" perimeter="0"/>

<constraint x="0.1" y="0.9" perimeter="0"/>

<constraint x="0.9" y="0.5" perimeter="0"/>

<constraint x="0.9" y="0.7" perimeter="0"/>

</connections>

<foreground>

<rect x="0" y="0" w="20" h="20" crisp="1"/>

<fillstroke/>

<rect x="0" y="80" w="20" h="20" crisp="1"/>

<fillstroke/>

<fillcolor color="#ff0000"/>

<rect x="80" y="40" w="20" h="20" crisp="1"/>

<fillstroke/>

<fillcolor color="#00ff00"/>

<roundrect x="80" y="60" w="20" h="20" crisp="1"/>

<fillstroke/>

<path crisp="1">

<move x="20" y="10"/>

<line x="90" y="10"/>

<line x="90" y="40"/>

<move x="20" y="90"/>

<line x="50" y="90"/>

<line x="50" y="70"/>

<line x="80" y="70"/>

</path>

<stroke/>

</foreground>

</shape>

</shapes>

Результат работы программы:

