**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

Факультет комп’ютерних наук та кібернетики

**Звіт до лабораторної роботи №1**

*Предметна область - Аеропорт*

Виконала студентка ІПС-31:

Карпишин Ольга Богданівна

Київ – 2020

Class Aerocompany – описуємо клас для представлення аерокомпанії

package sample;  
  
import java.util.ArrayList;  
  
public class Aircompany {  
 public int code;  
 public String name;  
 private ArrayList<Flight> flights = new ArrayList<>();  
 Aircompany(int code, String name){  
 this.code = code;  
 this.name = name;  
 }  
 public ArrayList<Flight> getFlights(){  
 return flights;  
 }  
 public void addFlight(Flight flight){  
 flights.add(flight);  
 }  
 public Flight findFlightByCode(int code){  
 for(int i = 0; i < flights.size(); i++){  
 if(flights.get(i).code == code){  
 return flights.get(i);  
 }  
 }  
 return null;  
 }  
  
}

Class Airport – описуємо клас для представлення аеропорту

package sample;  
  
import java.util.ArrayList;  
  
public class Airport {  
 private ArrayList<Aircompany> aircompanies = new ArrayList<>();  
 public void addAircompany(int code, String name){  
 aircompanies.add(new Aircompany(code, name));  
 }  
 public void addAircompany(Aircompany aircompany){ aircompanies.add(aircompany);}  
 public Aircompany getAirCompany(int code){  
 for(int i = 0;i < aircompanies.size(); i++){  
 if(aircompanies.get(i).code == code){  
 return aircompanies.get(i);  
 }  
 }  
 return null;  
 }  
 public ArrayList<Aircompany> getaircompanies(){return aircompanies;}  
 public int countAircompanies(){  
 return aircompanies.size();  
 }  
 public void deleteAircompany(int code) throws Exception{  
 Aircompany aircompanyToDelete = getAirCompany(code);  
 if(aircompanyToDelete == null){  
 throw new Exception("Aircompany doesnt exist");  
 }  
 aircompanies.remove(aircompanyToDelete);  
 }  
 public void addFlight(int code, String from, String to, int aircompanyCode) throws Exception{  
 Aircompany aircompany = getAirCompany(code);  
 if(aircompany == null){  
 throw new Exception("Aircompany doesnt exist");  
 }  
 ArrayList<Flight> flights= aircompany.getFlights();  
 for(int i = 0; i < flights.size(); i++){  
 if(flights.get(i).code == code){  
 throw new Exception("This flight has already exist");  
 }  
 }  
 Flight flight = new Flight(code, from, to);  
 aircompany.addFlight(flight);  
 }  
}

Class Controller

package sample;  
  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.scene.control.\*;  
import org.xml.sax.SAXException;  
  
import javax.xml.parsers.ParserConfigurationException;  
import java.io.IOException;  
import java.util.ArrayList;  
  
public class Controller {  
 @FXML  
 private TreeView treeView;  
 @FXML  
 private Button addCompany;  
 @FXML  
 private Button addFlight;  
 @FXML  
 private Button updateCompany;  
 @FXML  
 private Button updateFlight;  
 @FXML  
 private Button deleteCompany;  
 @FXML  
 private Button deleteFlight;  
 @FXML  
 private Button save;  
 @FXML  
 private TextField codeCompany;  
 @FXML  
 private TextField nameCompany;  
 @FXML  
 private TextField codeFlight;  
 @FXML  
 private TextField fromFlight;  
 @FXML  
 private TextField toFlight;  
 @FXML  
 private Label information;  
  
 XmlWorker xmlWorker;  
 @FXML  
 public void initialize() {  
 try {  
 xmlWorker = new XmlWorker();  
 printToTreeView();  
  
 } catch (ParserConfigurationException e) {  
 e.printStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 } catch (SAXException e) {  
 e.printStackTrace();  
 }  
 }  
 public void printToTreeView(){  
 treeView.setRoot(null);  
 ArrayList<Aircompany> aircompanyes = xmlWorker.getaircompanies();  
 //ArrayList<TreeItem<String>> aircompanyesTree = new ArrayList<>();  
 TreeItem<String> root = new TreeItem<>("Airport");  
  
 for(int i = 0; i < aircompanyes.size(); i++){  
 String info;  
 info = "Code: " + aircompanyes.get(i).code + " Name: " + aircompanyes.get(i).name;  
 ArrayList<Flight> flights = aircompanyes.get(i).getFlights();  
 TreeItem<String> aircompany = new TreeItem<>(info);  
 for(int j = 0; j < flights.size(); j++){  
 String flightInfo = flights.get(j).code + " " + flights.get(j).from + " - " + flights.get(j).to;  
 TreeItem<String> flight = new TreeItem<>(flightInfo);  
 aircompany.getChildren().add(flight);  
 }  
 // aircompanyesTree.add(aircompany);  
 root.getChildren().add(aircompany);  
 }  
 treeView.setRoot(root);  
 }  
 @FXML  
 private void addCompany(ActionEvent event) {  
 String code = codeCompany.getText();  
 String name = nameCompany.getText();  
 Aircompany aircompany = new Aircompany(Integer.valueOf(code), name);  
 xmlWorker.addAircompany(aircompany);  
 printToTreeView();  
 }  
 @FXML  
 private void addFlight(ActionEvent event) {  
 String code = codeFlight.getText();  
 String from = fromFlight.getText();  
 String to = toFlight.getText();  
 TreeItem<String> selectedItem = (TreeItem<String>)treeView.getSelectionModel().getSelectedItem();  
 if(selectedItem == null){  
 information.setText("Not selected aircompany");  
 return;  
 }  
 String codeAircompany = selectedItem.getValue().split(" ")[1];  
 Aircompany aircompany = xmlWorker.findAircompanyByCode(Integer.valueOf(codeAircompany));  
 xmlWorker.addFlight(Integer.valueOf(codeAircompany), new Flight(Integer.valueOf(code), from, to));  
 printToTreeView();  
  
 }  
 @FXML  
 private void updateCompany(ActionEvent event) {  
 String newCode = codeCompany.getText();  
 String newName = nameCompany.getText();  
 TreeItem<String> selectedItem = (TreeItem<String>)treeView.getSelectionModel().getSelectedItem();  
 if(selectedItem == null){  
 information.setText("Not selected aircompany");  
 return;  
 }  
 String codeAircompany = selectedItem.getValue().split(" ")[1];  
 String nameAircompany = selectedItem.getValue().split(" ")[4];  
 //Aircompany aircompany = xmlWorker.findAircompanyByCode(Integer.valueOf(codeAircompany));  
 if(newCode.equals("")){  
 newCode = codeAircompany;  
 }  
 if(newName.equals("")){  
 newName = nameAircompany;  
 }  
 xmlWorker.updateCompany(Integer.valueOf(codeAircompany), Integer.valueOf(newCode), newName);  
 printToTreeView();  
 }  
 @FXML  
 public void updateFlight(ActionEvent event){  
 String newCode = codeFlight.getText();  
 String newFrom = fromFlight.getText();  
 String newTo = toFlight.getText();  
 TreeItem<String> selectedItem =(TreeItem<String>)treeView.getSelectionModel().getSelectedItem();  
 if(selectedItem == null){  
 information.setText("Not selected aircompany");  
 return;  
 }  
 String codeAircompany = selectedItem.getParent().getValue().split(" ")[1];  
 String codeFlight = selectedItem.getValue().split(" ")[0];  
 String fromFlight = selectedItem.getValue().split(" ")[1];  
 String toFlight = selectedItem.getValue().split(" ")[3];  
 if(newCode.equals("")){  
 newCode = codeFlight;  
 }  
 if(newFrom.equals("")){  
 newFrom = fromFlight;  
 }  
 if(newTo.equals("")){  
 newTo = toFlight;  
 }  
 xmlWorker.updateFlight(Integer.valueOf(codeAircompany), Integer.valueOf(codeFlight),  
 Integer.valueOf(newCode), newFrom, newTo);  
 printToTreeView();  
 }  
 @FXML  
 public void deleteCompany(ActionEvent event){  
 TreeItem<String> selectedItem =(TreeItem<String>)treeView.getSelectionModel().getSelectedItem();  
 if(selectedItem == null){  
 information.setText("Not selected aircompany");  
 return;  
 }  
 String codeAircompany = selectedItem.getValue().split(" ")[1];  
 xmlWorker.deleteCompany(Integer.valueOf(codeAircompany));  
 printToTreeView();  
 }  
 @FXML  
 public void deleteFlight(ActionEvent event){  
 TreeItem<String> selectedItem =(TreeItem<String>)treeView.getSelectionModel().getSelectedItem();  
 if(selectedItem == null){  
 information.setText("Not selected aircompany");  
 return;  
 }  
 String codeAircompany = selectedItem.getParent().getValue().split(" ")[1];  
 String codeFlight = selectedItem.getValue().split(" ")[0];  
 xmlWorker.deleteFlight(Integer.valueOf(codeAircompany), Integer.valueOf(codeFlight));  
 printToTreeView();  
 }  
 @FXML  
 public void save(ActionEvent event){  
 xmlWorker.saveToXml();  
 }  
}

Class Flight

package sample;  
  
public class Flight {  
 public int code;  
 public String from;  
 public String to;  
 Flight(int code, String from, String to){  
 this.code = code;  
 this.from = from;  
 this.to = to;  
 }  
}

Class Main

package sample;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
public class Main extends Application {  
  
 @Override  
 public void start(Stage primaryStage) throws Exception{  
 Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));  
 primaryStage.setTitle("Hello World");  
 primaryStage.setScene(new Scene(root));  
 primaryStage.show();  
 }  
  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
}

Class ValidateXmlXsd

package sample;  
  
import org.xml.sax.SAXException;  
  
import javax.xml.XMLConstants;  
import javax.xml.transform.stream.StreamSource;  
import javax.xml.validation.Schema;  
import javax.xml.validation.SchemaFactory;  
import javax.xml.validation.Validator;  
import java.io.File;  
import java.io.IOException;  
  
public class ValidateXmlXsd {  
 public static void main(String[] args) {  
 try {  
 SchemaFactory factory = SchemaFactory.newInstance(XMLConstants.W3C\_XML\_SCHEMA\_NS\_URI);  
 Schema schema = factory.newSchema(new File("sample/airport.xsd"));  
 Validator validator = schema.newValidator();  
 validator.setErrorHandler(new XsdErrorHandler());  
 validator.validate(new StreamSource(new File("sample/airport.xml")));  
 System.out.println("Validation is successful");  
 } catch (SAXException e) {  
 System.out.println("Validation is failed: " + e.getMessage());  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

Class XmlWorker

package sample;  
  
import org.w3c.dom.Document;  
import org.w3c.dom.Element;  
import org.w3c.dom.Node;  
import org.w3c.dom.NodeList;  
import org.xml.sax.SAXException;  
  
import javax.xml.parsers.DocumentBuilder;  
import javax.xml.parsers.DocumentBuilderFactory;  
import javax.xml.parsers.ParserConfigurationException;  
import javax.xml.transform.Transformer;  
import javax.xml.transform.TransformerException;  
import javax.xml.transform.TransformerFactory;  
import javax.xml.transform.dom.DOMSource;  
import javax.xml.transform.stream.StreamResult;  
import java.io.File;  
import java.io.IOException;  
import java.util.ArrayList;  
  
public class XmlWorker {  
 private Airport airport;  
 public ArrayList<Aircompany> getaircompanies(){return airport.getaircompanies();}  
 XmlWorker() throws ParserConfigurationException, IOException, SAXException {  
 airport = new Airport();  
 File inputFile = new File("sample/airport.xml");  
 DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();  
 DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();  
 Document doc = dBuilder.parse(inputFile);  
 doc.getDocumentElement().normalize();  
 System.out.println("Root element :" + doc.getDocumentElement().getNodeName());  
 NodeList nList = doc.getElementsByTagName("aircompany");  
 for (int i = 0; i < nList.getLength(); i++) {  
 Node nNode = nList.item(i);  
  
 if (nNode.getNodeType() == Node.ELEMENT\_NODE) {  
 Element eAirCompany = (Element) nNode;  
 Aircompany aircompany = new Aircompany(Integer.valueOf(eAirCompany.getAttribute("code")),  
 eAirCompany.getAttribute("name"));  
 airport.addAircompany(aircompany);  
 NodeList flights = eAirCompany.getElementsByTagName("flight");  
  
 for(int j = 0; j < flights.getLength(); j++){  
 Node fNode = flights.item(j);  
  
 if (fNode.getNodeType() == Node.ELEMENT\_NODE) {  
 Element eFlight = (Element) fNode;  
 Flight flight = new Flight(Integer.valueOf(eFlight.getAttribute("code")),  
 eFlight.getAttribute("from"), eFlight.getAttribute("to"));  
 aircompany.addFlight(flight);  
 }  
 }  
  
 }  
 }  
 }  
 public void saveToXml(){  
 try {  
  
 DocumentBuilderFactory documentFactory = DocumentBuilderFactory.newInstance();  
 DocumentBuilder documentBuilder = documentFactory.newDocumentBuilder();  
 Document document = documentBuilder.newDocument();  
 Element root = document.createElement("airport");  
 document.appendChild(root);  
 for(int i = 0; i < airport.countAircompanies(); i++){  
 Element aircompanyElement = document.createElement("aircompany");  
 aircompanyElement.setAttribute("code", String.valueOf(airport.getaircompanies().get(i).code));  
 aircompanyElement.setAttribute("name", airport.getaircompanies().get(i).name);  
 for(int j = 0; j < airport.getaircompanies().get(i).getFlights().size(); j++){  
 Flight flight = airport.getaircompanies().get(i).getFlights().get(j);  
 Element flightElement = document.createElement("flight");  
 flightElement.setAttribute("code", String.valueOf(flight.code));  
 flightElement.setAttribute("from", flight.from);  
 flightElement.setAttribute("to", flight.to);  
 aircompanyElement.appendChild(flightElement);  
 }  
 root.appendChild(aircompanyElement);  
 }  
 TransformerFactory transformerFactory = TransformerFactory.newInstance();  
 Transformer transformer = transformerFactory.newTransformer();  
 DOMSource domSource = new DOMSource(document);  
 StreamResult streamResult = new StreamResult(new File("sample/airport.xml"));  
 transformer.transform(domSource, streamResult);  
 }catch (ParserConfigurationException pce) {  
 pce.printStackTrace();  
 } catch (TransformerException tfe) {  
 tfe.printStackTrace();  
 }  
 }  
 public void addAircompany(Aircompany aircompany){  
 airport.addAircompany(aircompany);  
 }  
 public void addFlight(int code, Flight flight){  
 Aircompany aircompany = airport.getAirCompany(code);  
 aircompany.addFlight(flight);  
 }  
 public Aircompany findAircompanyByCode(int code){  
 return airport.getAirCompany(code);  
 }  
 public void updateCompany(int code, int newCode, String newName){  
 Aircompany aircompany = airport.getAirCompany(code);  
 aircompany.name = newName;  
 aircompany.code = newCode;  
 }  
 public void updateFlight(int codeAircompany, int codeFlight, int newCode, String newFrom, String newTo){  
 Aircompany aircompany = airport.getAirCompany(codeAircompany);  
 Flight flight = aircompany.findFlightByCode(codeFlight);  
 flight.code = newCode;  
 flight.from = newFrom;  
 flight.to = newTo;  
 }  
 public void deleteCompany(int code){  
 Aircompany aircompany = airport.getAirCompany(code);  
 getaircompanies().remove(aircompany);  
 }  
 public void deleteFlight(int codeAircompany, int codeFlight){  
 Aircompany aircompany = airport.getAirCompany(codeAircompany);  
 Flight flight = aircompany.findFlightByCode(codeFlight);  
 aircompany.getFlights().remove(flight);  
 }  
}

XsdErrorHandler

package sample;  
  
import org.xml.sax.ErrorHandler;  
import org.xml.sax.SAXException;  
import org.xml.sax.SAXParseException;  
  
public class XsdErrorHandler implements ErrorHandler {  
  
 @Override  
 public void warning(SAXParseException exception) throws SAXException {  
 handleMessage("Warning", exception);  
 }  
  
 @Override  
 public void error(SAXParseException exception) throws SAXException {  
 handleMessage("Error", exception);  
 }  
  
 @Override  
 public void fatalError(SAXParseException exception) throws SAXException {  
 handleMessage("Fatal", exception);  
 }  
  
 private String handleMessage(String level, SAXParseException exception) throws SAXException {  
 int lineNumber = exception.getLineNumber();  
 int columnNumber = exception.getColumnNumber();  
 String message = exception.getMessage();  
 throw new SAXException("[" + level + "] line nr: " + lineNumber + " column nr: " + columnNumber + " message: " + message);  
 }  
}

Airport.xlm

<?xml version="1.0" encoding="UTF-8" standalone="no"?>  
<airport><aircompany code="12" name="airlines">  
 <flight code="123" from="Kiev" to="New York"/>  
 <flight code="124" from="Kiev" to="Paris"/>  
</aircompany><aircompany code="11" name="fedex">  
 <flight code="222" from="Kiev" to="Ternopil"/>  
 <flight code="228" from="Kiev" to="LA"/></aircompany>  
</airport>

Airoport/xsd

<?xml version="1.0" encoding="UTF-8"?>  
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">  
 <xs:element name="airport" type="airportType"/>  
 <xs:complexType name="flightType">  
 <xs:simpleContent>  
 <xs:extension base="xs:string">  
 <xs:attribute name="code" use="required" >  
 <xs:simpleType>  
 <xs:restriction base="xs:integer">  
 <xs:maxInclusive value="120"/>  
 </xs:restriction>  
 </xs:simpleType>  
 </xs:attribute>  
 <xs:attribute type="xs:string" name="from" use="required"/>  
 <xs:attribute type="xs:string" name="to" use="required"/>  
 </xs:extension>  
 </xs:simpleContent>  
 </xs:complexType>  
 <xs:complexType name="aircompanyType">  
 <xs:sequence>  
 <xs:element type="flightType" name="flight" maxOccurs="unbounded" minOccurs="0"/>  
 </xs:sequence>  
 <xs:attribute type="xs:string" name="code" use="required"/>  
 <xs:attribute type="xs:string" name="name" use="required"/>  
 </xs:complexType>  
 <xs:complexType name="airportType">  
 <xs:sequence>  
 <xs:element type="aircompanyType" name="aircompany" maxOccurs="unbounded" minOccurs="0"/>  
 </xs:sequence>  
 </xs:complexType>  
</xs:schema>