package com.meyer.chemicalvirtuallab;

/\*

\* Copyright (C) 2018 Ragil Rynaldo Meyer (meyer.ananda@gmail.com)

\*

\* This program is free software; you can redistribute it and/or

\* modify it under the terms of the GNU General Public License

\* as published by the Free Software Foundation; either version 2

\* of the License, or (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

\*/

import com.meyer.controller.MainMenuController;

import java.io.File;

import java.nio.file.Paths;

import javafx.application.Application;

import javafx.event.EventHandler;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.input.KeyCode;

import javafx.scene.input.KeyEvent;

import javafx.stage.Stage;

import javafx.stage.WindowEvent;

import javax.swing.JOptionPane;

/\*\*

\*

\* @author meyer

\*/

public class MainApp extends Application {

boolean dirAssets = false;

boolean dirJSMOL = false;

boolean fileAtoms = false;

boolean filePolyatoms = false;

@Override

public void start(final Stage stage) throws Exception {

File appDir = new File(System.getProperty("user.dir"));

File dataDir = new File(Paths.get(System.getProperty("user.dir"), "data").toString());

for (File f : appDir.listFiles()) {

if ("jsmol".equals(f.getName())) {

dirJSMOL = true;

}

if ("assets".equals(f.getName())) {

dirAssets = true;

}

}

for (File f : dataDir.listFiles()) {

if ("elements.csv".equals(f.getName())) {

fileAtoms = true;

}

if ("polyatomic.csv".equals(f.getName())) {

filePolyatoms = true;

}

}

if (!fileAtoms || !filePolyatoms) {

JOptionPane.showMessageDialog(null, "Berkas elements.csv dan polyatomic.csv tidak ditemukan.\n"

+ "Program akan keluar");

throw new RuntimeException();

}

if (!dirAssets) {

JOptionPane.showMessageDialog(null, "Direktori assets tidak ditemukan.\n"

+ "Gambar pada tampilan akan kosong");

}

if (!dirJSMOL) {

JOptionPane.showMessageDialog(null, "Direktori jsmol tidak ditemukan.\n"

+ "Tidak dapat menampilkan geometri 3D pada Menu Geometri");

}

FXMLLoader loader = new FXMLLoader(getClass().getResource("/fxml/MainMenu.fxml"));

Parent root = loader.load();

final MainMenuController mmc = loader.<MainMenuController>getController();

final Scene scene = new Scene(root);

stage.setTitle("Chemical Virtual Lab");

stage.setResizable(false);

stage.setScene(scene);

stage.show();

stage.addEventHandler(KeyEvent.KEY\_PRESSED, new EventHandler<KeyEvent>() {

@Override

public void handle(KeyEvent event) {

if (event.isControlDown() && event.getCode() == KeyCode.A) {

mmc.toggleAbout();

}

}

});

stage.addEventHandler(WindowEvent.WINDOW\_CLOSE\_REQUEST, new EventHandler<WindowEvent>() {

@Override

public void handle(WindowEvent event) {

if (event.getEventType() == WindowEvent.WINDOW\_CLOSE\_REQUEST) {

mmc.closeAll();

}

}

});

}

/\*\*

\* The main() method is ignored in correctly deployed JavaFX application.

\* main() serves only as fallback in case the application can not be

\* launched through deployment artifacts, e.g., in IDEs with limited FX

\* support. NetBeans ignores main().

\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

launch(args);

}

}

package com.meyer.controller;

/\*

\* Copyright (C) 2018 Ragil Rynaldo Meyer (meyer.ananda@gmail.com)

\*

\* This program is free software; you can redistribute it and/or

\* modify it under the terms of the GNU General Public License

\* as published by the Free Software Foundation; either version 2

\* of the License, or (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

\*/

import java.io.File;

import java.io.IOException;

import java.net.MalformedURLException;

import java.net.URL;

import java.nio.file.Paths;

import java.util.ResourceBundle;

import java.util.logging.Level;

import java.util.logging.Logger;

import javafx.event.EventHandler;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Cursor;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.effect.Glow;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.scene.input.MouseButton;

import javafx.scene.input.MouseEvent;

import javafx.scene.layout.AnchorPane;

import javafx.scene.layout.StackPane;

import javafx.scene.text.Font;

import javafx.scene.text.Text;

import javafx.scene.web.WebView;

import javafx.stage.Stage;

import javafx.stage.WindowEvent;

/\*\*

\* FXML Controller class

\*

\* @author meyer

\*/

public class MainMenuController implements Initializable {

@FXML private ImageView imgTable; @FXML private ImageView imgPredict; @FXML private ImageView img3D;

@FXML private StackPane paneTable; @FXML private Text txtTable; @FXML private StackPane panePredict;

@FXML private Text txtPredict; @FXML private StackPane pane3D; @FXML private Text txt3D;

@FXML private ImageView imgAbout; @FXML private AnchorPane aboutContainer;

private Stage sTable = null;

private Stage sPredict = null;

private Stage s3D = null;

private Stage sAbout = null;

private final Glow glow = new Glow();

private final EventHandler<MouseEvent> evtTable = new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if (event.getEventType() == MouseEvent.MOUSE\_CLICKED && event.getButton() == MouseButton.PRIMARY) {

launchTableScene();

} else if (event.getEventType() == MouseEvent.MOUSE\_ENTERED) {

txtTable.setEffect(glow);

} else if (event.getEventType() == MouseEvent.MOUSE\_EXITED) {

txtTable.setEffect(null);

}

}

};

private final EventHandler<MouseEvent> evtPredict = new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if (event.getEventType() == MouseEvent.MOUSE\_CLICKED && event.getButton() == MouseButton.PRIMARY) {

launchPredictScene();

} else if (event.getEventType() == MouseEvent.MOUSE\_ENTERED) {

txtPredict.setEffect(glow);

} else if (event.getEventType() == MouseEvent.MOUSE\_EXITED) {

txtPredict.setEffect(null);

}

}

};

private final EventHandler<MouseEvent> evt3D = new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if (event.getEventType() == MouseEvent.MOUSE\_CLICKED && event.getButton() == MouseButton.PRIMARY) {

launch3DScene();

} else if (event.getEventType() == MouseEvent.MOUSE\_ENTERED) {

txt3D.setEffect(glow);

} else if (event.getEventType() == MouseEvent.MOUSE\_EXITED) {

txt3D.setEffect(null);

}

}

};

private final EventHandler<MouseEvent> evtAbout = new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if (event.getEventType() == MouseEvent.MOUSE\_CLICKED && event.getButton() == MouseButton.PRIMARY) {

launchAboutScene();

}

}

};

private final Image table = new Image("file:" + Paths.get(System.getProperty("user.dir"), "assets", "png", "047-test-tube.png").toString(), true);

private final Image predict = new Image("file:" + Paths.get(System.getProperty("user.dir"), "assets", "png", "012-mortar.png").toString(), true);

private final Image geo = new Image("file:" + Paths.get(System.getProperty("user.dir"), "assets", "png", "024-science-2.png").toString(), true);

private final Image about = new Image("file:" + Paths.get(System.getProperty("user.dir"), "assets", "png", "man-thinking.png").toString(), true);

private final File font = new File(Paths.get(System.getProperty("user.dir"), "assets", "notosans", "NotoSans-Bold.ttf").toString());

@Override

public void initialize(URL url, ResourceBundle rb) {

try {

txtTable.setFont(Font.loadFont(font.toURI().toURL().toString(), 22));

txtPredict.setFont(Font.loadFont(font.toURI().toURL().toString(), 22));

txt3D.setFont(Font.loadFont(font.toURI().toURL().toString(), 22));

}

catch (MalformedURLException ex) {

Logger.getLogger(MainMenuController.class.getName()).log(Level.SEVERE, null, ex);

}

aboutContainer.setVisible(false);

paneTable.addEventHandler(MouseEvent.ANY, evtTable);

panePredict.addEventHandler(MouseEvent.ANY, evtPredict);

pane3D.addEventHandler(MouseEvent.ANY, evt3D);

aboutContainer.addEventHandler(MouseEvent.MOUSE\_CLICKED, evtAbout);

imgTable.setImage(table);

imgPredict.setImage(predict);

img3D.setImage(geo);

imgAbout.setImage(about);

paneTable.setCursor(Cursor.HAND);

panePredict.setCursor(Cursor.HAND);

pane3D.setCursor(Cursor.HAND);

aboutContainer.setCursor(Cursor.HAND);

}

private void launchTableScene() {

if (sTable == null) {

try {

FXMLLoader loader = new FXMLLoader(getClass().getResource("/fxml/PeriodTables.fxml"));

Parent root = loader.load();

final PeriodTablesController cntl = loader.<PeriodTablesController>getController();

Scene scene = new Scene(root);

sTable = new Stage();

sTable.setTitle("Tabel Periode");

sTable.setScene(scene);

sTable.show();

sTable.requestFocus();

sTable.addEventHandler(WindowEvent.WINDOW\_CLOSE\_REQUEST, new EventHandler<WindowEvent>() {

@Override

public void handle(WindowEvent event) {

sTable = null;

}

});

}

catch (IOException ex) {

Logger.getLogger(MainMenuController.class.getName()).log(Level.SEVERE, null, ex);

}

} else {

sTable.requestFocus();

}

}

private void launchPredictScene() {

if (sPredict == null) {

try {

FXMLLoader loader = new FXMLLoader(getClass().getResource("/fxml/PredictionTab.fxml"));

Parent root = loader.load();

final PredictionTabController cntl = loader.<PredictionTabController>getController();

Scene scene = new Scene(root);

sPredict = new Stage();

sPredict.setTitle("Prediksi");

sPredict.setScene(scene);

sPredict.setResizable(false);

sPredict.show();

sPredict.requestFocus();

sPredict.addEventHandler(WindowEvent.WINDOW\_CLOSE\_REQUEST, new EventHandler<WindowEvent>() {

@Override

public void handle(WindowEvent event) {

cntl.saveDic();

sPredict = null;

}

});

}

catch (IOException ex) {

Logger.getLogger(MainMenuController.class.getName()).log(Level.SEVERE, null, ex);

}

} else {

sPredict.requestFocus();

}

}

private void launch3DScene() {

if (s3D == null) {

try {

FXMLLoader loader = new FXMLLoader(getClass().getResource("/fxml/Atom3DViewer.fxml"));

Parent root = loader.load();

final Atom3DViewerController cntl = loader.<Atom3DViewerController>getController();

Scene scene = new Scene(root);

s3D = new Stage();

s3D.setTitle("Geometri");

s3D.setScene(scene);

s3D.setMinHeight(600);

s3D.show();

s3D.requestFocus();

s3D.addEventHandler(WindowEvent.WINDOW\_CLOSE\_REQUEST, new EventHandler<WindowEvent>() {

@Override

public void handle(WindowEvent event) {

cntl.saveDic();

s3D = null;

}

});

}

catch (IOException ex) {

Logger.getLogger(MainMenuController.class.getName()).log(Level.SEVERE, null, ex);

}

} else {

s3D.requestFocus();

}

}

}

package com.meyer.controller;

/\*

\* Copyright (C) 2018 Ragil Rynaldo Meyer (meyer.ananda@gmail.com)

\*

\* This program is free software; you can redistribute it and/or

\* modify it under the terms of the GNU General Public License

\* as published by the Free Software Foundation; either version 2

\* of the License, or (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

\*/

import com.meyer.object.Atom;

import java.io.File;

import java.io.IOException;

import java.net.MalformedURLException;

import java.net.URL;

import java.nio.charset.Charset;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.List;

import java.util.ResourceBundle;

import java.util.logging.Level;

import java.util.logging.Logger;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.control.Label;

import javafx.scene.control.TextArea;

import javafx.scene.text.Font;

/\*\*

\* FXML Controller class

\*

\* @author meyer

\*/

public class ElementDescriptionController implements Initializable {

@FXML private Label number; @FXML private Label symbol; @FXML private Label name;

@FXML private Label mass; @FXML private Label group; @FXML private Label iupac;

@FXML private Label period; @FXML private Label type; @FXML private Label ionization;

@FXML private Label electro; @FXML private Label valence; @FXML private Label charge;

@FXML private TextArea txtHistory; @FXML private TextArea txtUses;

private final File dir = new File(Paths.get(System.getProperty("user.dir"), "data").toString());

private final File font = new File(Paths.get(System.getProperty("user.dir"), "assets", "notosans", "NotoSans-Regular.ttf").toString());

/\*\*

\* Initializes the controller class.

\*

\* @param url

\* @param rb

\*/

@Override

public void initialize(URL url, ResourceBundle rb) {

try {

txtHistory.setFont(Font.loadFont(font.toURI().toURL().toString(), 14));

txtUses.setFont(Font.loadFont(font.toURI().toURL().toString(), 14));

}

catch (MalformedURLException ex) {

Logger.getLogger(ElementDescriptionController.class.getName()).log(Level.SEVERE, null, ex);

}

}

public void setAtom(Atom atom) {

number.setText(String.valueOf(atom.getNumber()));

symbol.setText(atom.getSymbol());

name.setText(atom.getName());

mass.setText(String.valueOf(atom.getMass()));

group.setText(atom.getGroup());

iupac.setText(atom.getIupac());

period.setText(String.valueOf(atom.getPeriod()));

type.setText(atom.getType());

ionization.setText(String.valueOf(atom.getIonEnergy()));

electro.setText(String.valueOf(atom.getNegativity()));

valence.setText(String.valueOf(atom.getValence()));

charge.setText(atom.getCharge());

searchOtherData(atom.getName());

}

private void searchOtherData(String name) {

if (!name.isEmpty()) {

for (File f : dir.listFiles()) {

if (f.getName().contains(name)) {

try {

List<String> lines = Files.readAllLines(f.toPath(), Charset.forName("Cp1253"));

txtHistory.setText(lines.get(1).replace("History Elements and Periodic Table History ", "HISTORY\n\n"));

String uses = lines.get(2).substring(lines.get(2).indexOf("Appearance"), lines.get(2).length());

uses = uses.replace("Appearance ", "APPEARANCE\n\n")

.replace("Uses ", "\n\nUSES\n\n")

.replace("Biological role ", "\n\nBIOLOGICAL ROLE\n\n")

.replace("Natural abundance ", "\n\nNATURAL ABUNDANCE\n\n");

txtUses.setText(uses);

}

catch (IOException ex) {

Logger.getLogger(ElementDescriptionController.class.getName()).log(Level.SEVERE, null, ex);

}

break;

}

}

}

}

}

package com.meyer.controller;

/\*

\* Copyright (C) 2018 Ragil Rynaldo Meyer (meyer.ananda@gmail.com)

\*

\* This program is free software; you can redistribute it and/or

\* modify it under the terms of the GNU General Public License

\* as published by the Free Software Foundation; either version 2

\* of the License, or (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

\*/

import com.meyer.constants.Bonding;

import com.meyer.constants.Reactions;

import com.meyer.element.ProgressContainer;

import com.meyer.ioutils.ElementReader;

import com.meyer.ioutils.FileDownload;

import com.meyer.ioutils.NameResolver;

import com.meyer.ioutils.PolyatomicReader;

import com.meyer.notation.NotationUtils;

import static com.meyer.notation.NotationUtils.distract;

import static com.meyer.notation.NotationUtils.isCations;

import static com.meyer.notation.NotationUtils.reduce;

import com.meyer.object.Atom;

import com.meyer.object.Compound;

import com.meyer.object.Polyatom;

import com.sun.javafx.util.Utils;

import java.io.ByteArrayInputStream;

import java.io.IOException;

import java.net.URL;

import java.nio.charset.StandardCharsets;

import java.util.ArrayList;

import java.util.LinkedHashMap;

import java.util.List;

import java.util.Map.Entry;

import java.util.ResourceBundle;

import java.util.TreeMap;

import java.util.function.Predicate;

import java.util.logging.Level;

import java.util.logging.Logger;

import javafx.beans.binding.Bindings;

import javafx.beans.value.ChangeListener;

import javafx.beans.value.ObservableValue;

import javafx.concurrent.Task;

import javafx.concurrent.WorkerStateEvent;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Cursor;

import javafx.scene.Node;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.ScrollPane;

import javafx.scene.control.TextField;

import javafx.scene.control.TitledPane;

import javafx.scene.input.KeyEvent;

import javafx.scene.input.MouseButton;

import javafx.scene.input.MouseEvent;

import javafx.scene.layout.StackPane;

import javafx.scene.layout.VBox;

import javafx.scene.text.Text;

import javafx.scene.text.TextFlow;

import javax.swing.JOptionPane;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.DOMException;

import org.w3c.dom.Document;

import org.xml.sax.SAXException;

/\*\*

\* FXML Controller class

\*

\* @author meyer

\*/

public class PredictionTabController implements Initializable {

@FXML private TextField textComp1; @FXML private TextField textComp2; @FXML private TitledPane listCations;

@FXML private TitledPane listAnions; @FXML private Button btnPredict; @FXML private StackPane boxResult;

@FXML private ScrollPane boxInfo;

private final EventHandler<KeyEvent> textTyped = new EventHandler<KeyEvent>() {

@Override

public void handle(KeyEvent event) {

if (!textComp1.getText().isEmpty() && !textComp2.getText().isEmpty()) {

btnPredict.setDisable(false);

} else {

btnPredict.setDisable(true);

}

}

};

private final EventHandler<ActionEvent> clicked = new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

predict(textComp1.getText(), textComp2.getText());

}

};

private final EventHandler<MouseEvent> mouse = new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if (event.getEventType() == MouseEvent.MOUSE\_ENTERED) {

if (event.getSource() instanceof TextFlow) {

TextFlow tf = (TextFlow) event.getSource();

tf.setStyle("-fx-background-color: #FF1493;");

}

} else if (event.getEventType() == MouseEvent.MOUSE\_EXITED) {

if (event.getSource() instanceof TextFlow) {

TextFlow tf = (TextFlow) event.getSource();

tf.setStyle("-fx-background-color: #ffffff;");

}

} else if (event.getEventType() == MouseEvent.MOUSE\_CLICKED

&& event.getButton() == MouseButton.PRIMARY) {

search(String.valueOf(((TextFlow) event.getSource()).getUserData()));

}

}

};

private Label selectedLabelCations = null;

private Label selectedLabelAnions = null;

private final TreeMap<String, String> dictionary = new TreeMap<>();

private Thread thread = null;

/\*\*

\* Initializes the controller class.

\*

\* @param url

\* @param rb

\*/

@Override

public void initialize(URL url, ResourceBundle rb) {

btnPredict.addEventHandler(ActionEvent.ACTION, clicked);

textComp1.addEventHandler(KeyEvent.KEY\_TYPED, textTyped);

textComp2.addEventHandler(KeyEvent.KEY\_TYPED, textTyped);

listCations.prefWidthProperty().bind(Bindings.selectDouble(listCations.parentProperty(), "width"));

listCations.prefHeightProperty().bind(Bindings.selectDouble(listCations.parentProperty(), "height"));

listAnions.prefWidthProperty().bind(Bindings.selectDouble(listCations.parentProperty(), "width"));

listAnions.prefHeightProperty().bind(Bindings.selectDouble(listCations.parentProperty(), "height"));

VBox vCations = new VBox();

VBox vAnions = new VBox();

vCations.setPadding(new Insets(5, 5, 5, 5));

vAnions.setPadding(new Insets(5, 5, 5, 5));

vCations.setSpacing(5);

vAnions.setSpacing(5);

for (Entry<String, Object> e : getCations().entrySet()) {

final Label lbl = new Label(e.getKey());

lbl.setUserData(e.getValue());

lbl.setCursor(Cursor.HAND);

lbl.addEventHandler(MouseEvent.ANY, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if (event.getEventType() == MouseEvent.MOUSE\_CLICKED) {

click("cation", lbl);

}

}

});

vCations.getChildren().add(lbl);

}

for (Entry<String, Object> e : getAnions().entrySet()) {

final Label lbl = new Label(e.getKey());

lbl.setUserData(e.getValue());

lbl.setCursor(Cursor.HAND);

lbl.setCursor(Cursor.HAND);

lbl.addEventHandler(MouseEvent.ANY, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if (event.getEventType() == MouseEvent.MOUSE\_CLICKED) {

click("anion", lbl);

}

}

});

vAnions.getChildren().add(lbl);

}

listCations.setContent(vCations);

listAnions.setContent(vAnions);

}

private void predict(String comp1, String comp2) {

System.out.println(Reactions.getType(comp1 + "+" + comp2));

Reactions react = Reactions.getType(comp1 + "+" + comp2);

switch (react) {

case SINGLE\_REPLACEMENT:

reactSingle(react, comp1, comp2);

break;

case DOUBLE\_REPLACEMENT:

reactDouble(react, comp1, comp2);

break;

case COMBUSTION:

LinkedHashMap<String, Object> result = new LinkedHashMap<>();

List<Node> list = new ArrayList<>();

result.put("reactions", react);

list.add(new Compound().getTextNode("CO2"));

list.add(new Compound().getTextNode("H2O"));

result.put("product1", list);

showResult(result);

break;

case ACID\_BASE:

reactDouble(react, comp1, comp2);

break;

default:

showResult(null);

break;

}

}

private void click(String type, Label label) {

if ("cation".equals(type)) {

if (selectedLabelCations == null) {

selectedLabelCations = label;

label.setStyle("-fx-text-fill: #6bf204;");

} else if (selectedLabelCations != null && selectedLabelCations == label) {

selectedLabelCations.setStyle("-fx-text-fill: #000000;");

selectedLabelCations = null;

} else if (selectedLabelCations != null && selectedLabelCations != label) {

selectedLabelCations.setStyle("-fx-text-fill: #000000;");

selectedLabelCations = label;

label.setStyle("-fx-text-fill: #6bf204;");

}

} else {

if (selectedLabelAnions == null) {

selectedLabelAnions = label;

label.setStyle("-fx-text-fill: #6bf204;");

} else if (selectedLabelAnions != null && selectedLabelAnions == label) {

selectedLabelAnions.setStyle("-fx-text-fill: #000000;");

selectedLabelAnions = null;

} else if (selectedLabelAnions != null && selectedLabelAnions != label) {

selectedLabelAnions.setStyle("-fx-text-fill: #000000;");

selectedLabelAnions = label;

label.setStyle("-fx-text-fill: #6bf204;");

}

}

if (selectedLabelCations != null && selectedLabelAnions != null) {

System.out.println(Bonding.getBond(selectedLabelCations.getUserData(), selectedLabelAnions.getUserData()));

if (Bonding.getBond(selectedLabelCations.getUserData(), selectedLabelAnions.getUserData()) != Bonding.UNKNOWN) {

Compound c = new Compound();

Reactions react = Reactions.SYNTHESIS;

if (c.addCation(selectedLabelCations.getUserData()) && c.addAnion(selectedLabelAnions.getUserData())) {

System.out.println("cation and anion added");

if (c.isValid()) {

System.out.println("compound is valid");

LinkedHashMap<String, Object> result = new LinkedHashMap<>();

result.put("reactions", react);

result.put("product1", c.getProduct());

showResult(result);

} else {

System.out.println("compound is not valid");

}

} else {

System.out.println("something is wrong, please check your input");

}

}

}

}

private void reactSingle(Reactions react, String input1, String input2) {

Compound c = new Compound();

Object ionCation = null;

Object ionAnion = null;

List<Object> comp = null;

List<Object> polyatom = new ArrayList();

if (NotationUtils.isCompound(input1) && !NotationUtils.isCompound(input2)) {

ionCation = reduce(distract(input2)).get(0);

comp = reduce(distract(input1));

} else if (!NotationUtils.isCompound(input1) && NotationUtils.isCompound(input2)) {

ionCation = reduce(distract(input1)).get(0);

comp = reduce(distract(input2));

} else {

System.out.println("Cannot Predict Result of Single Replacement");

}

for (Object o : comp) {

if (o instanceof Polyatom) {

polyatom.add((Polyatom) o);

}

}

comp.removeAll(polyatom);

if (comp.size() > 0 && polyatom.size() > 0) {

if (isCations(reduce(comp).get(0)) && !isCations(reduce(polyatom).get(0))) {

ionAnion = reduce(polyatom).get(0);

} else if (!isCations(reduce(comp).get(0)) && isCations(reduce(polyatom).get(0))) {

ionAnion = reduce(comp).get(0);

}

} else if (comp.size() < 1 && polyatom.size() > 0) {

if (reduce(polyatom).size() > 1) {

if (isCations(reduce(polyatom).get(0)) && !isCations(reduce(polyatom).get(1))) {

ionAnion = reduce(polyatom).get(1);

} else if (!isCations(reduce(polyatom).get(0)) && isCations(reduce(polyatom).get(0))) {

ionAnion = reduce(polyatom).get(0);

}

}

} else if (comp.size() > 0 && polyatom.size() < 1) {

if (reduce(comp).size() > 1) {

if (isCations(reduce(comp).get(0)) && !isCations(reduce(comp).get(1))) {

ionAnion = reduce(comp).get(1);

} else if (!isCations(reduce(comp).get(0)) && isCations(reduce(comp).get(0))) {

ionAnion = reduce(comp).get(0);

}

}

} else {

System.out.println("Failed predict Single Replacement");

}

if (ionAnion != null) {

if (c.addCation(ionCation) && c.addAnion(ionAnion)) {

if (c.isValid()) {

LinkedHashMap<String, Object> result = new LinkedHashMap<>();

result.put("reactions", react);

result.put("product1", c.getProduct());

result.put("product2", (comp.get(0) instanceof Atom ? ((Atom) comp.get(0)).getSymbol() : ((Polyatom) comp.get(0)).getIon()));

showResult(result);

}

}

}

}

private void reactDouble(Reactions react, final String strComp1, final String strComp2) {

final Compound c1 = new Compound();

final Compound c2 = new Compound();

List<Object> comp1 = null;

List<Object> comp2 = null;

if (NotationUtils.isCompound(strComp1) && NotationUtils.isCompound(strComp2)) {

if (c1.addCation(NotationUtils.getCation(strComp1))

&& c2.addCation(NotationUtils.getCation(strComp2))) {

System.out.println("Success to add cation to the comp1 and comp2");

} else {

System.out.println("failed to add cation to the comp1 and comp2");

}

comp1 = reduce(distract(strComp1));

comp2 = reduce(distract(strComp2));

System.out.println("before removing");

console(comp1);

console(comp2);

comp1.removeIf(new Predicate() {

@Override

public boolean test(Object t) {

if (t instanceof Atom && c1.getCation() instanceof Atom) {

Atom a1 = (Atom) t;

Atom a2 = (Atom) c1.getCation();

return a1.getSymbol().equals(a2.getSymbol());

} else if (t instanceof Polyatom && c1.getCation() instanceof Polyatom) {

Polyatom p1 = (Polyatom) t;

Polyatom p2 = (Polyatom) c1.getCation();

return p1.getIon().equals(p2.getIon());

} else {

return false;

}

}

});

comp2.removeIf(new Predicate() {

@Override

public boolean test(Object t) {

if (t instanceof Atom && c2.getCation() instanceof Atom) {

Atom a1 = (Atom) t;

Atom a2 = (Atom) c2.getCation();

return a1.getSymbol().equals(a2.getSymbol());

} else if (t instanceof Polyatom && c2.getCation() instanceof Polyatom) {

Polyatom p1 = (Polyatom) t;

Polyatom p2 = (Polyatom) c2.getCation();

return p1.getIon().equals(p2.getIon());

} else {

return false;

}

}

});

System.out.println("After removed");

console(comp1);

console(comp2);

if (c1.addAnion(comp2.get(0))

&& c2.addAnion(comp1.get(0))) {

if (c1.isValid() && c2.isValid()) {

LinkedHashMap<String, Object> result = new LinkedHashMap<>();

result.put("reactions", react);

result.put("product1", c1.getProduct());

result.put("product2", c2.getProduct());

showResult(result);

}

}

}

}

private void console(List<Object> list) {

for (Object o : list) {

if (o instanceof Polyatom) {

System.out.print("(P)" + ((Polyatom) o).getName() + " ");

} else if (o instanceof Atom) {

System.out.print("(A)" + ((Atom) o).getName() + " ");

}

}

System.out.println();

}

private LinkedHashMap<String, Object> getCations() {

LinkedHashMap<String, Object> cations = new LinkedHashMap();

for (Entry<String, Atom> e : ElementReader.read().entrySet()) {

if (!e.getValue().getCharge().contains("unknown")) {

if (Utils.stripQuotes(e.getValue().getCharge()).split(";").length > 1) {

List<Integer> charges = new ArrayList();

charges.addAll(NotationUtils.getAtomCharge(e.getValue()));

for (int i : charges) {

if (i > 0) {

cations.put(e.getValue().getName(), e.getValue());

}

}

} else {

if (Integer.valueOf(Utils.stripQuotes(e.getValue().getCharge())) > 0) {

cations.put(e.getValue().getName(), e.getValue());

}

}

}

}

for (Entry<String, Polyatom> e : PolyatomicReader.read().entrySet()) {

if (e.getValue().getCharge() > 0) {

cations.put(e.getValue().getName(), e.getValue());

}

}

return cations;

}

private LinkedHashMap<String, Object> getAnions() {

LinkedHashMap<String, Object> anions = new LinkedHashMap();

for (Entry<String, Atom> e : ElementReader.read().entrySet()) {

if (!e.getValue().getCharge().contains("unknown")) {

if (Utils.stripQuotes(e.getValue().getCharge()).split(";").length > 1) {

List<Integer> charges = new ArrayList();

charges.addAll(NotationUtils.getAtomCharge(e.getValue()));

for (int i : charges) {

if (i < 0) {

anions.put(e.getValue().getName(), e.getValue());

}

}

} else {

if (Integer.valueOf(Utils.stripQuotes(e.getValue().getCharge())) < 0) {

anions.put(e.getValue().getName(), e.getValue());

}

}

}

}

for (Entry<String, Polyatom> e : PolyatomicReader.read().entrySet()) {

if (e.getValue().getCharge() < 0) {

anions.put(e.getValue().getName(), e.getValue());

}

}

return anions;

}

private void showResult(LinkedHashMap<String, Object> results) {

if (boxResult.getChildren().size() > 0) {

boxResult.getChildren().clear();

}

if (results != null) {

VBox v = new VBox();

v.setSpacing(5);

v.setPadding(new Insets(5, 5, 5, 5));

v.setAlignment(Pos.CENTER);

v.getChildren().add(new Text("Hasil Prediksi:"));

v.getChildren().add(new Text(((Reactions) results.get("reactions")).toString()));

v.getChildren().add(new Text("Produk Hasil Reaksi:"));

System.out.println(results.get("product1") instanceof List);

List<Node> l = (List<Node>) results.get("product1");

for (Node n : l) {

n.setCursor(Cursor.HAND);

n.addEventHandler(MouseEvent.ANY, mouse);

v.getChildren().add(n);

}

if (results.get("product2") != null) {

if (results.get("product2") instanceof List) {

List<Node> ll = (List<Node>) results.get("product2");

for (Node n : ll) {

n.setCursor(Cursor.HAND);

n.addEventHandler(MouseEvent.ANY, mouse);

v.getChildren().add(n);

}

} else {

Text t = new Text((String) results.get("product2"));

t.setStyle("-fx-font-size: 18;");

v.getChildren().add(t);

}

}

boxResult.getChildren().add(v);

} else {

VBox v = new VBox();

v.setSpacing(5);

v.setPadding(new Insets(5, 5, 5, 5));

v.setAlignment(Pos.CENTER);

v.getChildren().add(new Text("Hasil Prediksi:"));

v.getChildren().add(new Text("Reaksi Tidak Diketahui"));

v.getChildren().add(new Text("Produk Hasil Reaksi:"));

v.getChildren().add(new Text("Produk Tidak Diketahui"));

boxResult.getChildren().add(v);

}

}

private void search(final String compName) {

if (thread == null) {

final ProgressContainer pc = new ProgressContainer();

pc.showProgress(boxInfo);

final Task<LinkedHashMap<String, String>> task = new Task<LinkedHashMap<String, String>>() {

@Override

protected LinkedHashMap<String, String> call() throws Exception {

LinkedHashMap<String, String> result = new LinkedHashMap<>();

String iupacName = "";

String casNumber = "";

String name = getName(compName);

updateMessage("Mencari Nama Produk");

if (!name.isEmpty()) {

updateMessage("Mendapatkan Nama");

result.put("name", name);

updateMessage("Mencari Nama IUPAC");

iupacName = FileDownload.getInfo(name, FileDownload.Source.NCI, FileDownload.ContentType.IUPAC);

if (!iupacName.isEmpty()) {

result.put("iupac", iupacName);

updateMessage("Nama IUPAC Ditemukan (NCI)");

} else {

updateMessage("Nama IUPAC Tidak Ditemukan (NCI)");

String xml = FileDownload.getInfo(name, FileDownload.Source.NIH, FileDownload.ContentType.IUPAC);

if (!xml.isEmpty()) {

try {

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(new ByteArrayInputStream(xml.getBytes(StandardCharsets.UTF\_8)));

if (doc.getElementsByTagName("CID").getLength() > -1) {

updateMessage("Nama IUPAC Ditemukan (NIH)");

String cid = doc.getElementsByTagName("CID").item(0).getTextContent();

String iupac = doc.getElementsByTagName("IUPACName").item(0).getTextContent();

result.put("iupac", iupac);

updateMessage("Mencari Nomor Registri CAS (NIH)");

delay(3000);

String xcas = FileDownload.getInfo(cid, FileDownload.Source.NIH, FileDownload.ContentType.CAS);

doc = dBuilder.parse(new ByteArrayInputStream(xcas.getBytes(StandardCharsets.UTF\_8)));

if (doc.getElementsByTagName("RecordNumber").getLength() > 0) {

updateMessage("Nomor Registri CAS Ditemukan (NIH)");

result.put("cas", doc.getElementsByTagName("StringValue").item(0).getTextContent());

}

}

}

catch (IOException | ParserConfigurationException | DOMException | SAXException ex) {

Logger.getLogger(PredictionTabController.class.getName()).log(Level.SEVERE, null, ex);

updateMessage("Terjadi Kesalahan Ketika Memproses Nama IUPAC");

}

}

}

if (result.get("cas") == null) {

delay(3000);

casNumber = FileDownload.getInfo(name, FileDownload.Source.NCI, FileDownload.ContentType.CAS);

if (casNumber.length() > 0) {

updateMessage("Nomor Registry CAS Ditemukan (NCI)");

result.put("cas", casNumber);

}

}

updateMessage("Memproses Hasil");

}

return result;

}

};

task.addEventHandler(WorkerStateEvent.ANY, new EventHandler<WorkerStateEvent>() {

@Override

public void handle(WorkerStateEvent event) {

ChangeListener<String> messageListener = new ChangeListener<String>() {

@Override

public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

pc.updateText(task.getMessage());

}

};

if (event.getEventType() == WorkerStateEvent.WORKER\_STATE\_RUNNING) {

task.messageProperty().addListener(messageListener);

} else if (event.getEventType() == WorkerStateEvent.WORKER\_STATE\_SUCCEEDED) {

task.messageProperty().removeListener(messageListener);

thread = null;

if (boxInfo.getContent() != null) {

boxInfo.setContent(null);

}

showSearch(task.getValue());

}

}

});

thread = new Thread(task);

thread.start();

} else {

JOptionPane.showMessageDialog(null, "Sedang Melakukan Proses Pencarian\n"

+ "Mohon Menunggu", "Peringatan", JOptionPane.WARNING\_MESSAGE);

}

}

private void delay(long milis) {

try {

Thread.sleep(milis);

}

catch (InterruptedException ex) {

Logger.getLogger(PredictionTabController.class.getName()).log(Level.SEVERE, null, ex);

}

}

private void showSearch(LinkedHashMap<String, String> result) {

VBox vv = new VBox();

vv.setPadding(new Insets(5, 5, 5, 5));

vv.setSpacing(5);

vv.setAlignment(Pos.CENTER);

vv.setFillWidth(true);

if (!result.isEmpty()) {

String name = (result.get("name") != null ? (String) result.get("name").replace("\n", "").trim() : "Nama Tidak Ditemukan");

String iupacName = (result.get("iupac") != null ? (String) result.get("iupac") : "Nama Tidak Ditemukan");

String casNumber = (result.get("cas") != null ? (String) result.get("cas") : "Nomor Tidak Ditemukan");

vv.setId("vbInfo");

vv.getChildren().add(new Text("Nama Produk:"));

vv.getChildren().add(new Text(name + "\n"));

vv.getChildren().add(new Text("Nama Berdasarkan IUPAC:"));

vv.getChildren().add(new Text(iupacName));

vv.getChildren().add(new Text("Nomor Registri CAS:"));

vv.getChildren().add(new Text(casNumber));

} else {

vv.getChildren().add(new Text("Nama Produk:"));

vv.getChildren().add(new Text("Nama Tidak Ditemukan\n"));

vv.getChildren().add(new Text("Nama Berdasarkan IUPAC:"));

vv.getChildren().add(new Text("Nama Tidak Ditemukan\n"));

vv.getChildren().add(new Text("Nomor Registri CAS:"));

vv.getChildren().add(new Text("Nomor Tidak Ditemukan\n"));

}

boxInfo.setContent(vv);

vv.prefWidthProperty().bind(Bindings.selectDouble(vv.parentProperty(), "width"));

vv.prefHeightProperty().bind(Bindings.selectDouble(vv.parentProperty(), "height"));

}

private String getName(String comp) {

System.out.println("Finding nomenclature");

String nomen = "";

if (dictionary.containsKey(comp)) {

nomen = dictionary.get(comp);

} else {

for (Entry<String, String> e : dictionary.entrySet()) {

if (e.getValue().trim().contains(comp)) {

nomen = e.getValue();

}

}

}

if (nomen.isEmpty()) {

nomen = FileDownload.getName(comp);

dictionary.put(comp.trim(), nomen);

}

return nomen;

}

public void saveDic() {

NameResolver.saveDictionary(dictionary);

}

}

package com.meyer.controller;

/\*

\* Copyright (C) 2018 Ragil Rynaldo Meyer (meyer.ananda@gmail.com)

\*

\* This program is free software; you can redistribute it and/or

\* modify it under the terms of the GNU General Public License

\* as published by the Free Software Foundation; either version 2

\* of the License, or (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

\*/

import com.meyer.element.ProgressContainer;

import com.meyer.ioutils.FileDownload;

import com.meyer.ioutils.NameResolver;

import java.io.BufferedReader;

import java.io.ByteArrayInputStream;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.net.URL;

import java.nio.charset.StandardCharsets;

import java.nio.file.Files;

import java.nio.file.LinkOption;

import java.nio.file.Paths;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.LinkedHashMap;

import java.util.List;

import java.util.Map;

import java.util.Map.Entry;

import java.util.ResourceBundle;

import java.util.TreeMap;

import java.util.logging.Level;

import java.util.logging.Logger;

import javafx.beans.binding.Bindings;

import javafx.concurrent.Task;

import javafx.concurrent.WorkerStateEvent;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.Node;

import javafx.scene.control.Button;

import javafx.scene.control.ScrollPane;

import javafx.scene.control.Tab;

import javafx.scene.control.TabPane;

import javafx.scene.control.TextField;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.scene.input.KeyEvent;

import javafx.scene.layout.StackPane;

import javafx.scene.text.Text;

import javafx.scene.web.HTMLEditor;

import javafx.scene.web.WebEngine;

import javafx.scene.web.WebView;

import javax.swing.JOptionPane;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.DOMException;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

/\*\*

\* FXML Controller class

\*

\* @author meyer

\*/

public class Atom3DViewerController implements Initializable {

@FXML private TextField boxSearch; @FXML private Button btnNIH; @FXML private Button btnNCI;

@FXML private StackPane containerWeb; @FXML private StackPane containerImage; @FXML private StackPane containerInfo;

/\*\*

\* Initializes the controller class.

\*/

private final String home = Paths.get(System.getProperty("user.dir"), "jsmol", "home.html").toString();

private final String modeldir = Paths.get(System.getProperty("user.dir"), "caches", "models").toString();

private final String datadir = Paths.get(System.getProperty("user.dir"), "caches", "datas").toString();

private final String imagedir = Paths.get(System.getProperty("user.dir"), "caches", "images").toString();

private final TreeMap<String, String> dictionary = new TreeMap<>();

Thread thread = null;

private final EventHandler clickSearch = new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

search(event);

}

};

private final EventHandler searchField = new EventHandler<KeyEvent>() {

@Override

public void handle(KeyEvent event) {

if (event.getEventType() == KeyEvent.KEY\_TYPED) {

if (boxSearch.getText().length() > 1) {

if (btnNIH.isDisable()) {

btnNIH.setDisable(false);

}

} else {

btnNIH.setDisable(true);

}

if (boxSearch.getText().length() > 1) {

if (btnNCI.isDisable()) {

btnNCI.setDisable(false);

}

} else {

btnNCI.setDisable(true);

}

}

}

};

@Override

public void initialize(URL url, ResourceBundle rb) {

dictionary.putAll(NameResolver.getDictionary());

try {

if (!Files.exists(Paths.get(modeldir), LinkOption.NOFOLLOW\_LINKS)) {

Files.createDirectories(Paths.get(modeldir));

}

if (!Files.exists(Paths.get(imagedir), LinkOption.NOFOLLOW\_LINKS)) {

Files.createDirectories(Paths.get(imagedir));

}

if (!Files.exists(Paths.get(datadir), LinkOption.NOFOLLOW\_LINKS)) {

Files.createDirectories(Paths.get(datadir));

}

} catch (IOException ex) {

Logger.getLogger(Atom3DViewerController.class.getName()).log(Level.SEVERE, null, ex);

}

btnNIH.setDisable(true);

btnNIH.addEventHandler(ActionEvent.ACTION, clickSearch);

btnNCI.setDisable(true);

btnNCI.addEventHandler(ActionEvent.ACTION, clickSearch);

boxSearch.addEventHandler(KeyEvent.KEY\_TYPED, searchField);

}

private void search(ActionEvent event) {

if (((Button) event.getSource()).getId().equals("btnNIH")) {

searching(boxSearch.getText().trim(), FileDownload.Source.NIH);

} else if (((Button) event.getSource()).getId().equals("btnNCI")) {

searching(boxSearch.getText().trim(), FileDownload.Source.NCI);

}

}

private void searching(final String comp, final FileDownload.Source source) {

if (thread == null) {

ProgressContainer pcWeb = new ProgressContainer();

ProgressContainer pcPNG = new ProgressContainer();

ProgressContainer pcInfo = new ProgressContainer();

if (containerWeb.getChildren().size() > 0) {

containerWeb.getChildren().clear();

}

if (containerImage.getChildren().size() > 0) {

containerImage.getChildren().clear();

}

if (containerInfo.getChildren().size() > 0) {

containerInfo.getChildren().clear();

}

pcWeb.showProgress(containerWeb);

pcPNG.showProgress(containerImage);

pcInfo.showProgress(containerInfo);

final Task task = new Task<Map<String, Object>>() {

@Override

protected Map<String, Object> call() throws Exception {

Map<String, Object> result = new HashMap<>();

String nomen = getName(comp);

if (!nomen.isEmpty()) {

File sdf = searchFile(source.toString() + "\_" + nomen, modeldir);

if (sdf != null) {

result.put("sdf", sdf);

} else {

result.put("sdf", FileDownload.download(nomen, source, FileDownload.ContentType.SDF));

}

File img = searchFile(source.toString() + "\_" + nomen, imagedir);

if (img != null) {

result.put("img", img);

} else {

delay(3000);

result.put("img", FileDownload.download(nomen,

source,

FileDownload.ContentType.IMAGE));

}

delay(3000);

String xmlNIH = FileDownload.getInfo(nomen,

FileDownload.Source.NIH,

FileDownload.ContentType.INCHIKEY);

List<String> parsedXML = CIDParser(xmlNIH);

if (!parsedXML.isEmpty()) {

File data = searchFile(parsedXML.get(0), datadir);

if (data != null) {

result.put("data", data);

} else {

delay(3000);

result.put("data", FileDownload.download(parsedXML.get(0),

FileDownload.Source.NIH,

FileDownload.ContentType.DATA));

}

}

System.out.println("End Of Searching");

}

return result;

}

};

task.addEventHandler(WorkerStateEvent.ANY, new EventHandler<WorkerStateEvent>() {

@Override

public void handle(WorkerStateEvent event) {

if (event.getEventType() == WorkerStateEvent.WORKER\_STATE\_SUCCEEDED) {

Map<String, Object> result = new HashMap<>();

result.putAll((Map<String, Object>) task.getValue());

showResultWeb((File) result.get("sdf"));

showResultImage((File) result.get("img"));

showResultData((File) result.get("data"));

thread = null;

}

}

});

thread = new Thread(task);

thread.start();

} else {

JOptionPane.showMessageDialog(null, "Sedang Melakukan Proses Pencarian\n"

+ "Mohon Menunggu", "Peringatan", JOptionPane.WARNING\_MESSAGE);

}

}

private boolean parseHome(File file) {

boolean success = false;

File newHome = new File(home);

StringBuilder homeBuilder = readFile(newHome);

StringBuilder dataBuilder = readFile(file);

homeBuilder.replace(homeBuilder.indexOf("<!-- MOL DATA"), homeBuilder.lastIndexOf("$$$$") + 4, "<!-- MOL DATA\n" + dataBuilder);

try (FileWriter writer = new FileWriter(new File(home))) {

writer.write(homeBuilder.toString());

writer.close();

success = true;

} catch (IOException ex) {

Logger.getLogger(Atom3DViewerController.class.getName()).log(Level.SEVERE, null, ex);

}

return success;

}

private StringBuilder readFile(File file) {

StringBuilder sb = new StringBuilder();

try {

BufferedReader br;

try (FileReader reader = new FileReader(file)) {

br = new BufferedReader(reader);

String line;

while ((line = br.readLine()) != null) {

sb.append(line).append("\n");

}

}

br.close();

} catch (FileNotFoundException ex) {

Logger.getLogger(Atom3DViewerController.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

Logger.getLogger(Atom3DViewerController.class.getName()).log(Level.SEVERE, null, ex);

}

return sb;

}

private void delay(long milis) {

try {

Thread.sleep(milis);

} catch (InterruptedException ex) {

Logger.getLogger(Atom3DViewerController.class.getName()).log(Level.SEVERE, null, ex);

}

}

private List<String> CIDParser(String xml) {

List<String> list = new ArrayList<>();

try {

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(new ByteArrayInputStream(xml.getBytes(StandardCharsets.UTF\_8)));

if (doc.getElementsByTagName("CID").getLength() > -1) {

String cid = doc.getElementsByTagName("CID").item(0).getTextContent();

String inchikey = doc.getElementsByTagName("InChIKey").item(0).getTextContent();

list.add(cid);

list.add(inchikey);

}

} catch (IOException | ParserConfigurationException | DOMException | SAXException ex) {

Logger.getLogger(Atom3DViewerController.class.getName()).log(Level.SEVERE, null, ex);

}

return list;

}

private void showResultWeb(File file) {

if (containerWeb.getChildren().size() > 0) {

containerWeb.getChildren().clear();

}

if (file != null) {

if (parseHome(file)) {

WebView view = new WebView();

WebEngine we = view.getEngine();

we.load("file:" + Paths.get(home).toString());

containerWeb.getChildren().add(view);

view.prefHeightProperty().bind(Bindings.selectDouble(view.parentProperty(), "height"));

view.prefWidthProperty().bind(Bindings.selectDouble(view.parentProperty(), "width"));

}

} else {

Text text = new Text("Geometri 3D tidak ditemukan");

StackPane pane = new StackPane();

pane.getChildren().add(text);

containerWeb.getChildren().add(pane);

}

}

private void showResultImage(File file) {

if (containerImage.getChildren().size() > 0) {

containerImage.getChildren().clear();

}

if (file != null) {

ImageView view = new ImageView();

view.setImage(new Image("file:" + file.getPath()));

containerImage.getChildren().add(view);

} else {

Text text = new Text("Geometri 2D tidak ditemukan");

StackPane pane = new StackPane();

pane.getChildren().add(text);

containerImage.getChildren().add(pane);

}

}

private void showResultData(File file) {

if (containerInfo.getChildren().size() > 0) {

containerInfo.getChildren().clear();

}

if (file != null) {

containerInfo.getChildren().add(parseData(readFile(file).toString()));

} else {

Text text = new Text("Deskripsi fisik tidak ditemukan");

StackPane pane = new StackPane();

pane.getChildren().add(text);

containerInfo.getChildren().add(pane);

}

}

private String getName(String comp) {

System.out.println("Finding nomenclature");

String nomen = "";

if (dictionary.containsKey(comp)) {

nomen = dictionary.get(comp);

} else {

for (Entry<String, String> e : dictionary.entrySet()) {

if (e.getValue().trim().contains(comp)) {

nomen = e.getValue();

}

}

}

if (nomen.isEmpty()) {

nomen = FileDownload.getName(comp);

dictionary.put(comp.trim(), nomen);

}

return nomen;

}

private File searchFile(String name, String dir) {

File result = null;

File directory = new File(dir);

for (File f : directory.listFiles()) {

if (f.getName().contains(name)) {

result = f;

break;

}

}

return result;

}

public void saveDic() {

NameResolver.saveDictionary(dictionary);

}

private Node parseData(String xmlData) {

HTMLEditor editor = new HTMLEditor();

editor.lookupAll(xmlData);

LinkedHashMap<Integer, String> information = new LinkedHashMap<>();

LinkedHashMap<Integer, String> reference = new LinkedHashMap<>();

try {

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(new ByteArrayInputStream(xmlData.getBytes(StandardCharsets.UTF\_8)));

NodeList info = doc.getElementsByTagName("Information");

NodeList ref = doc.getElementsByTagName("Reference");

for (int x = 0; x < info.getLength(); x++) {

Element el = (Element) info.item(x);

int num = Integer.valueOf(el.getElementsByTagName("ReferenceNumber").item(0).getTextContent());

String str = "";

NodeList nn;

if (el.getElementsByTagName("StringValue").getLength() > 0) {

nn = el.getElementsByTagName("StringValue");

} else {

nn = el.getElementsByTagName("StringValueList");

}

for (int aa = 0; aa < nn.getLength(); aa++) {

str += nn.item(aa).getTextContent() + "\n";

}

information.put(num, "<h4>Description: </h4>" + str);

}

for (int x = 0; x < ref.getLength(); x++) {

Element el = (Element) ref.item(x);

System.out.println(info.item(x).getFirstChild().getTextContent());

System.out.println(info.item(x).getLastChild().getTextContent());

int num = Integer.valueOf(el.getElementsByTagName("ReferenceNumber").item(0).getTextContent());

String desc = "";

if (el.getElementsByTagName("Description").getLength() > 0) {

desc = "<h4>Description: </h4><p>" + el.getElementsByTagName("Description").item(0).getTextContent() + "</p>";

}

String str

= "<h4>Source Name: </h4><p>" + el.getElementsByTagName("SourceName").item(0).getTextContent() + "</p>"

+ "<h4>Source ID: </h4><p>" + el.getElementsByTagName("SourceID").item(0).getTextContent() + "</p>"

+ "<h4>Name: </h4><p>" + el.getElementsByTagName("Name").item(0).getTextContent() + "</p>"

+ desc

+ "<h4>URL: </h4><p>" + el.getElementsByTagName("URL").item(0).getTextContent() + "</p>";

reference.put(num, str);

}

} catch (IOException | ParserConfigurationException | DOMException | SAXException ex) {

Logger.getLogger(Atom3DViewerController.class.getName()).log(Level.SEVERE, null, ex);

}

TabPane tp = new TabPane();

for (Entry<Integer, String> e : information.entrySet()) {

Tab tab = new Tab("Description " + e.getKey());

ScrollPane pane = new ScrollPane();

WebView wv = new WebView();

pane.setContent(wv);

tab.setContent(pane);

wv.getEngine().loadContent("<html>" + e.getValue() + "<br/>" + reference.get(e.getKey()) + "</html>");

tp.getTabs().add(tab);

}

return tp;

}

}

package com.meyer.constants;

/\*

\* Copyright (C) 2018 Ragil Rynaldo Meyer (meyer.ananda@gmail.com)

\*

\* This program is free software; you can redistribute it and/or

\* modify it under the terms of the GNU General Public License

\* as published by the Free Software Foundation; either version 2

\* of the License, or (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

\*/

import com.meyer.ioutils.PolyatomicReader;

import com.meyer.notation.NotationToCode;

import com.meyer.object.Polyatom;

import java.util.ArrayList;

import java.util.LinkedHashMap;

import java.util.List;

import java.util.Map.Entry;

/\*\*

\*

\* @author meyer

\*/

public enum Reactions {

SYNTHESIS, DECOMPOSITION, COMBUSTION, SINGLE\_REPLACEMENT, DOUBLE\_REPLACEMENT, ACID\_BASE, UNKNOWN;

public static Reactions getType(String notation) {

LinkedHashMap<String, Polyatom> polyatoms = PolyatomicReader.read();

if (!notation.contains("\\+")) {

List<Polyatom> polycomp1 = new ArrayList();

List<Polyatom> polycomp2 = new ArrayList();

String comp1 = notation.split("\\+")[0].trim();

String comp2 = notation.split("\\+")[1].trim();

for (Entry<String, Polyatom> e : polyatoms.entrySet()) {

if (comp1.matches(e.getKey())) {

polycomp1.add(e.getValue());

comp1 = comp1.replace(e.getKey(), "");

}

if (comp2.matches(e.getKey())) {

polycomp2.add(e.getValue());

comp2 = comp2.replace(e.getKey(), "");

}

}

for (Entry<String, Polyatom> e : polyatoms.entrySet()) {

if (comp1.contains(e.getKey())) {

polycomp1.add(e.getValue());

comp1 = comp1.replace(e.getKey(), "");

}

if (comp2.contains(e.getKey())) {

polycomp2.add(e.getValue());

comp2 = comp2.replace(e.getKey(), "");

}

}

if (polycomp1.isEmpty() && polycomp2.isEmpty()) {

if ((reduce(NotationToCode.parse(comp1)).size() == 1) && (reduce(NotationToCode.parse(comp2)).size() == 1)

|| (polycomp2.size() == 1 && polycomp1.size() == 1 && comp1.length() == 0 && comp2.length() == 0)) {

return SYNTHESIS;

} else if ((reduce(NotationToCode.parse(comp1)).size() == 1

&& reduce(NotationToCode.parse(comp2)).size() == 2)

|| (reduce(NotationToCode.parse(comp2)).size() == 1

&& reduce(NotationToCode.parse(comp1)).size() == 2)) {

if (reduce(NotationToCode.parse(comp1)).contains("C")

&& reduce(NotationToCode.parse(comp1)).contains("H")

&& reduce(NotationToCode.parse(comp2)).contains("O")) {

return COMBUSTION;

} else if (reduce(NotationToCode.parse(comp2)).contains("C")

&& reduce(NotationToCode.parse(comp2)).contains("H")

&& reduce(NotationToCode.parse(comp1)).contains("O")) {

return COMBUSTION;

} else if (reduce(NotationToCode.parse(comp2)).contains("C")

&& reduce(NotationToCode.parse(comp2)).contains("H")

&& reduce(NotationToCode.parse(comp2)).contains("O")

&& reduce(NotationToCode.parse(comp1)).contains("O")) {

return COMBUSTION;

} else if (reduce(NotationToCode.parse(comp1)).contains("C")

&& reduce(NotationToCode.parse(comp1)).contains("H")

&& reduce(NotationToCode.parse(comp1)).contains("O")

&& reduce(NotationToCode.parse(comp2)).contains("O")) {

return COMBUSTION;

} else {

return SINGLE\_REPLACEMENT;

}

} else if (!comp1.isEmpty() && !comp2.isEmpty()) {

if (reduce(NotationToCode.parse(comp1)).size() > 1 && reduce(NotationToCode.parse(comp2)).size() > 1) {

return DOUBLE\_REPLACEMENT;

} else {

return UNKNOWN;

}

} else {

return UNKNOWN;

}

} else if (polycomp1.isEmpty() && polycomp2.size() == 1) {

if (reduce(NotationToCode.parse(comp1)).size() == 1 && comp2.isEmpty()) {

return SYNTHESIS;

} else if (reduce(NotationToCode.parse(comp1)).size() == 1 && !comp2.isEmpty()) {

if (reduce(NotationToCode.parse(comp2)).size() == 1) {

return SINGLE\_REPLACEMENT;

} else {

return UNKNOWN;

}

} else if (reduce(NotationToCode.parse(comp1)).size() > 1 && !comp2.isEmpty()) {

if (reduce(NotationToCode.parse(comp2)).size() == 1) {

return DOUBLE\_REPLACEMENT;

} else {

return UNKNOWN;

}

} else {

return UNKNOWN;

}

} else if (polycomp1.size() == 1 && polycomp2.isEmpty()) {

if (reduce(NotationToCode.parse(comp2)).size() == 1 && comp1.isEmpty()) {

return SYNTHESIS;

} else if (reduce(NotationToCode.parse(comp2)).size() == 1 && !comp1.isEmpty()) {

if (reduce(NotationToCode.parse(comp1)).size() == 1) {

return SINGLE\_REPLACEMENT;

} else {

return UNKNOWN;

}

} else if (reduce(NotationToCode.parse(comp2)).size() > 1 && !comp1.isEmpty()) {

if (reduce(NotationToCode.parse(comp1)).size() == 1) {

return DOUBLE\_REPLACEMENT;

} else {

return UNKNOWN;

}

} else {

return UNKNOWN;

}

} else if (polycomp1.size() == 1 && polycomp2.size() == 1) {

if (!comp1.isEmpty() && comp2.isEmpty()) {

if (!reduce(NotationToCode.parse(comp1)).isEmpty()

&& reduce(NotationToCode.parse(comp2)).isEmpty()) {

return SINGLE\_REPLACEMENT;

} else {

return UNKNOWN;

}

} else if (!comp1.isEmpty() && comp2.isEmpty()) {

if (reduce(NotationToCode.parse(comp1)).isEmpty()

&& !reduce(NotationToCode.parse(comp2)).isEmpty()) {

return SINGLE\_REPLACEMENT;

} else {

return UNKNOWN;

}

} else if (!comp1.isEmpty() && !comp2.isEmpty()) {

if ((reduce(NotationToCode.parse(comp1)).contains("H") && polycomp2.get(0).getIon().equals("OH"))

|| reduce(NotationToCode.parse(comp2)).contains("H") && polycomp1.get(0).getIon().equals("OH")) {

return ACID\_BASE;

} else {

return DOUBLE\_REPLACEMENT;

}

} else {

return UNKNOWN;

}

} else {

return UNKNOWN;

}

} else {

return DECOMPOSITION;

}

}

private static ArrayList<String> reduce(List<String> list) {

ArrayList<String> result = new ArrayList();

for (String s : list) {

if (result.isEmpty()) {

result.add(s);

} else {

if (!result.contains(s)) {

result.add(s);

}

}

}

return result;

}

}

package com.meyer.constants;

/\*

\* Copyright (C) 2018 Ragil Rynaldo Meyer (meyer.ananda@gmail.com)

\*

\* This program is free software; you can redistribute it and/or

\* modify it under the terms of the GNU General Public License

\* as published by the Free Software Foundation; either version 2

\* of the License, or (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

\*/

import com.meyer.object.Atom;

import com.meyer.object.Polyatom;

import java.util.ArrayList;

/\*\*

\*

\* @author meyer

\*/

public enum Bonding {

IONIC\_BOND(1),

COVALENT\_BOND(2),

UNKNOWN(3);

private int type;

private Bonding(int type) {

this.type = type;

}

public int getBond() {

return type;

}

public static Bonding getBonds(ArrayList<Object> list) {

Bonding types = Bonding.UNKNOWN;

if (list.size() < 2 && list.size() > 0) {

types = Bonding.COVALENT\_BOND;

} else if (list.size() > 2) {

types = Bonding.UNKNOWN;

} else {

types = getBond(list.get(0), list.get(1));

}

return types;

}

public static Bonding getBond(Object o1, Object o2) {

if (o1 instanceof Atom && o2 instanceof Atom) {

double atom1 = ((Atom) o1).getNegativity();

double atom2 = ((Atom) o2).getNegativity();

if ((atom1 - atom2) > 2.0 || (atom2 - atom1) > 2.0) {

return IONIC\_BOND;

} else if (((atom1 - atom2) < 1.7 && (atom1 - atom2) > -1)

|| ((atom2 - atom1) < 1.7 && (atom2 - atom1) > -1)) {

return COVALENT\_BOND;

} else {

return UNKNOWN;

}

} else if (o1 instanceof Polyatom && o2 instanceof Atom) {

if (Types.getType(((Atom) o2).getType()) != Types.NON\_METAL) {

return IONIC\_BOND;

} else if (((Atom) o2).getSymbol().equals("H")) {

return COVALENT\_BOND;

} else {

return UNKNOWN;

}

} else if (o1 instanceof Atom && o2 instanceof Polyatom) {

if (Types.getType(((Atom) o1).getType()) != Types.NON\_METAL) {

return IONIC\_BOND;

} else if (((Atom) o1).getSymbol().equals("H")) {

return COVALENT\_BOND;

} else {

return UNKNOWN;

}

} else {

return UNKNOWN;

}

}

}

package com.meyer.constants;

/\*

\* Copyright (C) 2018 Ragil Rynaldo Meyer (meyer.ananda@gmail.com)

\*

\* This program is free software; you can redistribute it and/or

\* modify it under the terms of the GNU General Public License

\* as published by the Free Software Foundation; either version 2

\* of the License, or (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

\*/

import com.meyer.object.Atom;

import com.meyer.object.Polyatom;

/\*\*

\*

\* @author meyer

\*/

public enum Types {

METAL, TRANSITION\_METAL, NON\_METAL, POLYATOMIC, UNKNOWN;

public static Types getType(Object object) {

if (object instanceof Polyatom) {

return POLYATOMIC;

} else if (object instanceof Atom) {

String type = ((Atom) object).getType();

switch (type) {

case "Alkali Metal":

return Types.METAL;

case "Alkaline Earth":

return Types.METAL;

case "Transition Metal":

return Types.TRANSITION\_METAL;

case "Basic Metal":

return Types.METAL;

case "Semimetal":

return Types.METAL;

case "Lanthanide":

return Types.METAL;

case "Actinide":

return Types.METAL;

default:

return Types.NON\_METAL;

}

} else {

return Types.UNKNOWN;

}

}

public static int getInt(Types t) {

switch (t) {

case METAL:

return 1;

case TRANSITION\_METAL:

return 2;

case NON\_METAL:

return 3;

case POLYATOMIC:

return 4;

default:

return 0;

}

}

}

package com.meyer.element;

/\*

\* Copyright (C) 2018 Ragil Rynaldo Meyer (meyer.ananda@gmail.com)

\*

\* This program is free software; you can redistribute it and/or

\* modify it under the terms of the GNU General Public License

\* as published by the Free Software Foundation; either version 2

\* of the License, or (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

\*/

import javafx.beans.binding.Bindings;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Node;

import javafx.scene.control.ProgressIndicator;

import javafx.scene.control.ScrollPane;

import javafx.scene.layout.StackPane;

import javafx.scene.layout.VBox;

import javafx.scene.text.Text;

import javafx.scene.text.TextAlignment;

/\*\*

\*

\* @author meyer

\*/

public class ProgressContainer {

private Text text = new Text();

private ProgressIndicator pi = new ProgressIndicator();

private VBox vb = new VBox();

public void showProgress(Node parent) {

vb.setPadding(new Insets(5, 5, 5, 5));

vb.setSpacing(5);

vb.setAlignment(Pos.CENTER);

text.setTextAlignment(TextAlignment.CENTER);

text.setLineSpacing(1.5);

text.setText("Harap Menunggu");

pi.setProgress(-1);

pi.setMaxSize(50, 50);

pi.setPrefSize(50, 50);

vb.getChildren().add(pi);

vb.getChildren().add(text);

if (parent instanceof ScrollPane) {

((ScrollPane) parent).setContent(vb);

} else if (parent instanceof StackPane) {

((StackPane) parent).getChildren().add(vb);

}

vb.prefHeightProperty().bind(Bindings.selectDouble(vb.parentProperty(), "height"));

vb.prefWidthProperty().bind(Bindings.selectDouble(vb.parentProperty(), "width"));

}

public void updateText(String str) {

text.setText(str);

}

public void updateProgress(double progress) {

pi.setProgress(progress);

}

}

package com.meyer.ioutils;

/\*

\* Copyright (C) 2018 Ragil Rynaldo Meyer (meyer.ananda@gmail.com)

\*

\* This program is free software; you can redistribute it and/or

\* modify it under the terms of the GNU General Public License

\* as published by the Free Software Foundation; either version 2

\* of the License, or (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

\*/

import com.meyer.object.Atom;

import com.sun.javafx.util.Utils;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.nio.file.Paths;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.LinkedHashMap;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\*

\* @author meyer

\*/

public class ElementReader {

public static LinkedHashMap<String, Atom> read() {

File file = new File(Paths.get(System.getProperty("user.dir"), "data", "elements.csv").toString());

java.io.FileReader reader = null;

try {

reader = new java.io.FileReader(file);

}

catch (FileNotFoundException ex) {

Logger.getLogger(ElementReader.class.getName()).log(Level.SEVERE, null, ex);

}

BufferedReader br = new BufferedReader(reader);

LinkedHashMap<String, Atom> atoms = new LinkedHashMap();

String line = null;

int il = 0;

ArrayList<String> title = new ArrayList();

try {

while ((line = br.readLine()) != null) {

if (il == 0) {

title.addAll(Arrays.asList(line.split(line)));

} else {

String[] some = line.split(",");

Atom atom = new Atom();

atom.setNumber(Integer.valueOf(some[0]));

atom.setSymbol(Utils.stripQuotes(some[1]));

atom.setName(Utils.stripQuotes(some[2]));

atom.setMass(Double.valueOf(some[3]));

atom.setGroup(Utils.stripQuotes(some[4]));

atom.setIupac(Utils.stripQuotes(some[5]));

atom.setPeriod(Integer.valueOf(some[6]));

atom.setType(Utils.stripQuotes(some[7]));

atom.setIonEnergy(Double.valueOf(some[8]));

atom.setNegativity(Double.valueOf(some[9]));

atom.setInventor(some[10]);

atom.setYear(some[11]);

atom.setValence(Integer.valueOf(some[12]));

atom.setCharge(some[13]);

atoms.put(atom.getSymbol(), atom);

}

++il;

}

}

catch (IOException ex) {

Logger.getLogger(ElementReader.class.getName()).log(Level.SEVERE, null, ex);

}

return atoms;

}

}

package com.meyer.ioutils;

/\*

\* Copyright (C) 2018 Ragil Rynaldo Meyer (meyer.ananda@gmail.com)

\*

\* This program is free software; you can redistribute it and/or

\* modify it under the terms of the GNU General Public License

\* as published by the Free Software Foundation; either version 2

\* of the License, or (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

\*/

import com.meyer.controller.Atom3DViewerController;

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.HttpURLConnection;

import java.net.MalformedURLException;

import java.net.ProtocolException;

import java.net.URL;

import java.nio.file.Paths;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.net.ssl.HttpsURLConnection;

/\*\*

\*

\* @author meyer

\*/

public class FileDownload {

private final static String HOMEDIR = System.getProperty("user.dir");

public enum Source {

NIH, NCI, OTHERS;

public static String getSource(Source source) {

switch (source) {

case NIH:

return "https://pubchem.ncbi.nlm.nih.gov";

case NCI:

return "https://cactus.nci.nih.gov";

default:

return null;

}

}

}

public enum ContentType {

INCHIKEY, SDF, IMAGE, DATA, CID, CAS, IUPAC, NAME;

public static String PathStore(ContentType type) {

switch (type) {

case SDF:

return Paths.get(HOMEDIR, "caches", "models").toString();

case IMAGE:

return Paths.get(HOMEDIR, "caches", "images").toString();

case DATA:

return Paths.get(HOMEDIR, "caches", "datas").toString();

default:

return null;

}

}

public static String fileType(ContentType type) {

switch (type) {

case SDF:

return ".SDF";

case IMAGE:

return ".PNG";

case DATA:

return ".XML";

default:

return null;

}

}

}

private static String getCallableURL(String name, Source source, ContentType content) {

String trimmed = name.trim().replace(" ", "%20");

if (source == null) {

return "";

} else {

switch (source) {

case NCI:

switch (content) {

case INCHIKEY:

return Source.getSource(Source.NCI) + "/chemical/structure/"

+ trimmed

+ "/stdinchikey";

case SDF:

return Source.getSource(Source.NCI) + "/chemical/structure/"

+ trimmed

+ "/file?format=sdf&get3d=True";

case IMAGE:

return Source.getSource(Source.NCI) + "/chemical/structure/"

+ trimmed

+ "/image";

case CAS:

return Source.getSource(Source.NCI) + "/chemical/structure/"

+ trimmed

+ "/cas";

case IUPAC:

return Source.getSource(Source.NCI) + "/chemical/structure/"

+ trimmed

+ "/iupac\_name";

default:

return "";

}

case NIH:

switch (content) {

case SDF:

return Source.getSource(Source.NIH) + "/rest/pug/compound/name/"

+ trimmed

+ "/SDF?record\_type=3d";

case IMAGE:

return Source.getSource(Source.NIH) + "/rest/pug/compound/name/"

+ trimmed

+ "/PNG";

case DATA:

return Source.getSource(Source.NIH) + "/rest/pug\_view/data/compound/"

+ trimmed

+ "/XML?heading=Physical+Description";

case INCHIKEY:

return Source.getSource(Source.NIH) + "/rest/pug/compound/name/"

+ trimmed

+ "/property/InChIKey/XML";

case CAS:

return Source.getSource(Source.NIH) + "/rest/pug\_view/data/compound/"

+ trimmed

+ "/XML?heading=CAS";

case IUPAC:

return Source.getSource(Source.NIH) + "/rest/pug/compound/name/"

+ trimmed

+ "/property/IUPACName/XML";

default:

return "";

}

default:

return "";

}

}

}

public static String getInfo(String name, Source source, ContentType type) {

String result = "";

String call = "";

if (!name.isEmpty()) {

call = getCallableURL(name, source, type);

}

if (!call.isEmpty()) {

System.out.println(call);

try {

java.net.URL website = new java.net.URL(call);

HttpsURLConnection con = (HttpsURLConnection) website.openConnection();

con.setRequestMethod("POST");

con.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");

con.setRequestProperty("Content-Language", "en-US");

con.setUseCaches(false);

con.setDoInput(true);

con.setDoOutput(true);

if (con.getResponseCode() == 200) {

try (BufferedReader br = new BufferedReader(new InputStreamReader(con.getInputStream()))) {

StringBuilder sb = new StringBuilder();

String line;

while ((line = br.readLine()) != null) {

sb.append(line).append("\n");

}

result = sb.toString();

System.out.println(sb.toString());

}

con.disconnect();

} else {

System.out.println(con.getResponseCode() + " " + con.getResponseMessage());

}

con.disconnect();

}

catch (IOException ex) {

Logger.getLogger(Atom3DViewerController.class.getName()).log(Level.SEVERE, null, ex);

}

}

return result;

}

public static File download(String name, Source source, ContentType type) {

File file = null;

String call = "";

if (!name.isEmpty()) {

call = getCallableURL(name, source, type);

}

if (!call.isEmpty()) {

try {

java.net.URL website = new java.net.URL(call);

HttpsURLConnection con = (HttpsURLConnection) website.openConnection();

con.setRequestMethod("POST");

con.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");

con.setRequestProperty("Content-Language", "en-US");

con.setUseCaches(false);

con.setDoInput(true);

con.setDoOutput(true);

System.out.println(call);

if (con.getResponseCode() == 200) {

if (type == ContentType.DATA) {

file = new File(

Paths.get(ContentType.PathStore(type),

name.trim() + ContentType.fileType(type)).toString());

} else {

file = new File(

Paths.get(ContentType.PathStore(type),

source.toString() + "\_" + name.trim() + ContentType.fileType(type)).toString());

}

BufferedOutputStream bos;

try (BufferedInputStream bis = new BufferedInputStream(con.getInputStream())) {

bos = new BufferedOutputStream(new FileOutputStream(file));

int inbyte;

while ((inbyte = bis.read()) != -1) {

bos.write(inbyte);

}

}

bos.close();

} else {

System.out.println(con.getResponseCode() + " " + con.getResponseMessage());

}

con.disconnect();

}

catch (IOException ex) {

Logger.getLogger(Atom3DViewerController.class.getName()).log(Level.SEVERE, null, ex);

}

}

return file;

}

public static String getName(String name) {

System.out.println(name.trim().replace(" ", "%20"));

name = name.trim().replace(" ", "%20");

StringBuilder sb = new StringBuilder();

String str = "http://www.endmemo.com/chem/ajax/chemsearch\_ajax.php?q=" + name + "&s=n";

try {

URL url = new URL(str);

HttpURLConnection con = (HttpURLConnection) url.openConnection();

con.setRequestMethod("GET");

con.setRequestProperty("User-Agent", "Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.170 Safari/537.36 OPR/53.0.2907.57");

con.setRequestProperty("Accept", "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8");

con.setRequestProperty("Accept-Language", "en-US,en;q=0.9");

con.setRequestProperty("Host", "www.endmemo.com");

con.setDoInput(true);

con.setDoOutput(true);

try (BufferedReader bf = new BufferedReader(new InputStreamReader(con.getInputStream()))) {

if (bf.ready()) {

String line;

while ((line = bf.readLine()) != null) {

sb.append(line).append("\n");

System.out.println(line);

}

} else {

System.out.println("something wrong?");

}

}

con.disconnect();

}

catch (MalformedURLException ex) {

Logger.getLogger(FileDownload.class.getName()).log(Level.SEVERE, null, ex);

}

catch (ProtocolException ex) {

Logger.getLogger(FileDownload.class.getName()).log(Level.SEVERE, null, ex);

}

catch (IOException ex) {

Logger.getLogger(FileDownload.class.getName()).log(Level.SEVERE, null, ex);

}

if (sb.toString().split("\n").length > 1) {

return sb.toString().split("\n")[0].split("\\|\\|\\|")[0].replace("\n", "").trim();

} else if (name.split("\n").length < 2) {

return sb.toString().split("\\|\\|\\|")[0].replace("\n", "").trim();

} else {

return "";

}

}

}

package com.meyer.ioutils;

/\*

\* Copyright (C) 2018 Ragil Rynaldo Meyer (meyer.ananda@gmail.com)

\*

\* This program is free software; you can redistribute it and/or

\* modify it under the terms of the GNU General Public License

\* as published by the Free Software Foundation; either version 2

\* of the License, or (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

\*/

import com.meyer.object.Polyatom;

import com.sun.javafx.util.Utils;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.nio.file.Paths;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.LinkedHashMap;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\*

\* @author meyer

\*/

public class PolyatomicReader extends Object {

public static LinkedHashMap<String, Polyatom> read() {

File file = new File(Paths.get(System.getProperty("user.dir"), "data", "polyatomic.csv").toString());

java.io.FileReader reader = null;

try {

reader = new java.io.FileReader(file);

}

catch (FileNotFoundException ex) {

Logger.getLogger(PolyatomicReader.class.getName()).log(Level.SEVERE, null, ex);

}

BufferedReader br = new BufferedReader(reader);

LinkedHashMap<String, Polyatom> polyatoms = new LinkedHashMap();

String line = null;

int il = 0;

ArrayList<String> title = new ArrayList();

try {

while ((line = br.readLine()) != null) {

if (il == 0) {

title.addAll(Arrays.asList(line.split(line)));

} else {

if (!line.contains("#")) {

String[] some = line.split(",");

Polyatom polyatom = new Polyatom();

polyatom.setIon(Utils.stripQuotes(some[0]));

polyatom.setName(Utils.stripQuotes(some[1]));

polyatom.setCharge(Integer.valueOf(some[2]));

polyatom.setComposition(Utils.stripQuotes(some[3]));

polyatoms.put(polyatom.getIon(), polyatom);

}

}

++il;

}

}

catch (IOException ex) {

Logger.getLogger(PolyatomicReader.class.getName()).log(Level.SEVERE, null, ex);

}

return polyatoms;

}

}

package com.meyer.notation;

/\*

\* Copyright (C) 2018 Ragil Rynaldo Meyer (meyer.ananda@gmail.com)

\*

\* This program is free software; you can redistribute it and/or

\* modify it under the terms of the GNU General Public License

\* as published by the Free Software Foundation; either version 2

\* of the License, or (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

\*/

import java.util.ArrayList;

/\*\*

\*

\* @author meyer

\*/

public class NotationToCode {

public static String norm(String str) {

String mult = "0";

String f = "";

for (int x = str.indexOf("("); x < str.length(); x++) {

if (x <= str.indexOf(")")) {

f += Character.toString(str.charAt(x));

}

}

for (int x = str.indexOf(")") + 1; x < str.length(); x++) {

if (Character.isDigit(str.charAt(x))) {

mult += Character.toString(str.charAt(x));

System.out.println("is digit " + str.charAt(x));

} else {

System.out.println("is not digit " + str.charAt(x));

break;

}

}

System.out.println("Mole : " + f);

System.out.println("mult : " + Integer.valueOf(mult));

String t = f;

f = f.replace("(", "");

f = f.replace(")", "");

String tt = f;

f = "";

if (Integer.valueOf(mult) > 0) {

for (int x = 0; x < Integer.valueOf(mult); x++) {

f += tt;

}

str = str.replace(t + Integer.valueOf(mult), f);

} else {

str = str.replace(t, tt);

}

System.out.println("f : " + f);

System.out.println("t : " + tt);

return str;

}

public static ArrayList<String> parse(String str) {

if (Character.isDigit(str.charAt(0))) {

str = str.substring(1, str.length() - 1);

}

if (str.contains("+")) {

str = str.substring(0, str.indexOf("+") + 1);

}

if (str.contains("-")) {

str = str.substring(0, str.indexOf("+") + 1);

}

if (str.contains("(") && str.contains(")")) {

str = norm(str);

}

ArrayList<String> arr = new ArrayList();

String f = "";

for (int x = 0; x < str.length(); x++) {

if (Character.isAlphabetic(str.charAt(x)) && Character.isUpperCase(str.charAt(x))) {

if (f.length() > 0) {

arr.add(f);

}

f = Character.toString(str.charAt(x));

} else if (Character.isAlphabetic(str.charAt(x)) && Character.isLowerCase(str.charAt(x))) {

f += Character.toString(str.charAt(x));

} else if (Character.isDigit(str.charAt(x)) && x > 0) {

for (int y = 0; y < Integer.valueOf(Character.toString(str.charAt(x))) - 1; y++) {

arr.add(f);

}

}

if (x + 1 == str.length() && f.length() > 0) {

arr.add(f);

}

}

return arr;

}

public static int globMult(String str) {

String mult = "0";

for (int x = 0; x < str.length(); x++) {

if (Character.isDigit(str.charAt(x))) {

mult += Character.toString(str.charAt(x));

} else {

break;

}

}

return Integer.valueOf(mult);

}

public static int numCharge(String str) {

int charge = 0;

if (str.contains("-")) {

Integer.valueOf(str.substring(str.indexOf("-")), str.length());

} else if (str.contains("+")) {

Integer.valueOf(str.substring(str.indexOf("+")), str.length());

}

return charge;}

package com.meyer.notation;

/\*

\* Copyright (C) 2018 Ragil Rynaldo Meyer (meyer.ananda@gmail.com)

\*

\* This program is free software; you can redistribute it and/or

\* modify it under the terms of the GNU General Public License

\* as published by the Free Software Foundation; either version 2

\* of the License, or (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

\*/

import com.meyer.ioutils.ElementReader;

import com.meyer.ioutils.PolyatomicReader;

import com.meyer.object.Atom;

import com.meyer.object.Polyatom;

import com.sun.javafx.util.Utils;

import java.util.ArrayList;

import java.util.LinkedHashMap;

import java.util.List;

import java.util.Map;

/\*\*

\*

\* @author meyer

\*/

public class NotationUtils {

public final static List<Object> distract(String comp) {

List<Object> list = new ArrayList();

LinkedHashMap<String, Polyatom> polyatoms = PolyatomicReader.read();

LinkedHashMap<String, Atom> atoms = ElementReader.read();

for (Map.Entry<String, Polyatom> e : polyatoms.entrySet()) {

if (comp.matches(e.getKey())) {

System.out.println(e.getKey());

list.add(e.getValue());

comp = comp.replace(e.getKey(), "");

}

}

for (Map.Entry<String, Polyatom> e : polyatoms.entrySet()) {

if (comp.contains(e.getKey())) {

System.out.println(e.getKey());

list.add(e.getValue());

comp = comp.replace(e.getKey(), "");

}

}

if (comp.length() > 0) {

for (String s : NotationToCode.parse(comp)) {

for (Map.Entry<String, Atom> e : atoms.entrySet()) {

if (s.equals(e.getKey())) {

list.add(e.getValue());

}

}

}

}

return list;

}

public final static List<Object> reduce(List<Object> list) {

LinkedHashMap<String, Object> map = new LinkedHashMap();

for (Object a : list) {

if (a instanceof Atom) {

map.put(((Atom) a).getSymbol(), a);

} else if (a instanceof Polyatom) {

map.put(((Polyatom) a).getIon(), a);

}

}

List<Object> result = new ArrayList();

for (Map.Entry<String, Object> e : map.entrySet()) {

result.add(e.getValue());

}

return result;

}

public final static List<Integer> getAtomCharge(Atom atom) {

List<Integer> charges = new ArrayList();

if (Utils.stripQuotes(atom.getCharge()).split(";").length > 1) {

for (String s : Utils.stripQuotes(atom.getCharge()).split(";")) {

charges.add(Integer.valueOf(s));

}

}

return charges;

}

public final static void removeAll(List<Object> list, Object obj) {

for (int x = 0; x < list.size(); x++) {

if (list.get(x) instanceof Atom && obj instanceof Atom) {

if (((Atom) list.get(x)).getSymbol().equals(((Atom) obj).getSymbol())) {

list.remove(x);

break;

}

}

}

}

public final static boolean isCations(Object obj) {

boolean yes = false;

if (obj instanceof Atom) {

Atom a = (Atom) obj;

if (a.getCharge().contains(";")) {

List<Integer> charges = getAtomCharge(a);

for (int i : charges) {

if (i > 0) {

yes = true;

break;

}

}

} else {

if (Integer.valueOf(a.getCharge()) > 0) {

yes = true;

}

}

} else {

Polyatom p = (Polyatom) obj;

if (p.getCharge() > 0) {

yes = true;

}

}

return yes;

}

public final static boolean isAnions(Object obj) {

boolean yes = false;

if (obj instanceof Atom) {

Atom a = (Atom) obj;

if (a.getCharge().contains(";")) {

List<Integer> charges = getAtomCharge(a);

for (int i : charges) {

if (i < 0) {

yes = true;

break;

}

}

} else {

if (Integer.valueOf(a.getCharge()) < 0) {

yes = true;

}

}

} else {

Polyatom p = (Polyatom) obj;

if (p.getCharge() < 0) {

yes = true;

}

}

return yes;

}

public final static int[] getBalance(int cCations, int cAnions) {

int[] factor = new int[2];

System.out.println("getBalance: ");

System.out.println(cCations);

System.out.println(cAnions);

if (cCations < 0) {

cCations = Math.negateExact(cCations);

}

if (cAnions < 0) {

cAnions = Math.negateExact(cAnions);

}

System.out.println(cCations);

System.out.println(cAnions);

if (Integer.compare(cCations, cAnions) > 0) {

if (cAnions == 1) {

cAnions = cCations;

cCations = 1;

} else if (cCations % cAnions == 0 && cCations == 2) {

cAnions \*= 2;

cCations = 1;

} else if (cCations % cAnions == 0 && cCations != 2) {

int a = (int) Math.sqrt(cAnions);

cAnions \*= a;

cCations = 1;

} else if (cCations % cAnions != 0 && cCations != 2) {

int a = cAnions;

cAnions = cCations;

cCations = a;

}

} else if (Integer.compare(cCations, cAnions) < 0) {

if (cCations == 1) {

cCations = cAnions;

cAnions = 1;

} else if (cAnions % cCations == 0 && cAnions == 2) {

cCations \*= 2;

cAnions = 1;

} else if (cAnions % cCations == 0 && cAnions != 2) {

int a = (int) Math.sqrt(cCations);

cCations \*= a;

cAnions = 1;

} else if (cAnions % cCations != 0 && cAnions != 2) {

int a = cCations;

cCations = cAnions;

cAnions = a;

}

} else {

cCations = 1;

cAnions = 1;

}

factor[0] = cCations;

factor[1] = cAnions;

return factor;

}

public final static boolean isCompound(String notation) {

List<Object> comp = reduce(distract(notation));

List<Object> polyatom = new ArrayList();

for (Object o : comp) {

if (o instanceof Polyatom) {

polyatom.add((Polyatom) o);

}

}

comp.removeAll(polyatom);

if (comp.size() > 0 && polyatom.size() > 0) {

return (isCations(reduce(comp).get(0)) && !isCations(reduce(polyatom).get(0)))

|| (!isCations(reduce(comp).get(0)) && isCations(reduce(polyatom).get(0)));

} else if (comp.size() < 1 && polyatom.size() > 0) {

if (reduce(polyatom).size() > 1) {

return (isCations(reduce(polyatom).get(0)) && !isCations(reduce(polyatom).get(1)))

|| (!isCations(reduce(polyatom).get(0)) && isCations(reduce(polyatom).get(0)));

} else {

return false;

}

} else if (comp.isEmpty() && polyatom.isEmpty()) {

System.out.println("Something wrong when predict compound (rare)");

return false;

} else if (comp.size() > 0 && polyatom.size() < 1) {

if (reduce(comp).size() > 1) {

return (isCations(reduce(comp).get(0)) && !isCations(reduce(comp).get(1)))

|| (!isCations(reduce(comp).get(0)) && isCations(reduce(comp).get(0)));

} else {

return false;

}

} else {

return false;

}

}

public final static Object getCation(String notation) {

List<Object> comp = reduce(distract(notation));

List<Object> polyatom = new ArrayList();

for (Object o : comp) {

if (o instanceof Polyatom) {

polyatom.add((Polyatom) o);

}

}

comp.removeAll(polyatom);

if (comp.size() > 0 && polyatom.size() > 0) {

if (isCations(reduce(comp).get(0)) && !isCations(reduce(polyatom).get(0))) {

return reduce(comp).get(0);

} else if (!isCations(reduce(comp).get(0)) && isCations(reduce(polyatom).get(0))) {

return reduce(polyatom).get(0);

} else {

return null;

}

} else if (comp.size() < 1 && polyatom.size() > 0) {

if (reduce(polyatom).size() > 1) {

if (isCations(reduce(polyatom).get(0)) && !isCations(reduce(polyatom).get(1))) {

return reduce(polyatom).get(0);

} else if (!isCations(reduce(polyatom).get(0)) && isCations(reduce(polyatom).get(0))) {

return reduce(polyatom).get(0);

} else {

return null;

}

} else {

return null;

}

} else if (comp.size() > 0 && polyatom.size() < 1) {

if (reduce(comp).size() > 1) {

if (isCations(reduce(comp).get(0)) && !isCations(reduce(comp).get(1))) {

return reduce(comp).get(0);

} else if (!isCations(reduce(comp).get(0)) && isCations(reduce(comp).get(0))) {

return reduce(comp).get(0);

} else {

return null;

}

} else {

return null;

}

} else {

return null;

}

}

}

package com.meyer.object;

/\*

\* Copyright (C) 2018 Ragil Rynaldo Meyer (meyer.ananda@gmail.com)

\*

\* This program is free software; you can redistribute it and/or

\* modify it under the terms of the GNU General Public License

\* as published by the Free Software Foundation; either version 2

\* of the License, or (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

\*/

import com.meyer.notation.NotationUtils;

import com.sun.javafx.util.Utils;

import java.util.ArrayList;

import java.util.List;

import javafx.scene.Node;

import javafx.scene.layout.Region;

import javafx.scene.text.Text;

import javafx.scene.text.TextAlignment;

import javafx.scene.text.TextFlow;

/\*\*

\*

\* @author meyer

\*/

public class Compound {

private Object cation = null;

private Object anion = null;

public boolean addCation(Object cation) {

if (NotationUtils.isCations(cation)) {

this.cation = cation;

return true;

} else {

System.out.println("not cation object");

return false;

}

}

public boolean addAnion(Object anion) {

if (!NotationUtils.isCations(anion)) {

this.anion = anion;

return true;

} else {

System.out.println("not anion object");

return false;

}

}

public Object getCation() {

return this.cation;

}

public Object getAnion() {

return this.anion;

}

public void printIon(String ion) {

switch (ion) {

case "anion":

if (this.anion instanceof Atom) {

System.out.println(((Atom) this.anion).getSymbol());

} else if (this.anion instanceof Polyatom) {

System.out.println(((Polyatom) this.anion).getIon());

} else {

System.out.println("null");

}

break;

case "cation":

if (this.cation instanceof Atom) {

System.out.println(((Atom) this.cation).getSymbol());

} else if (this.cation instanceof Polyatom) {

System.out.println(((Polyatom) this.cation).getIon());

} else {

System.out.println("null");

}

break;

}

}

public boolean isValid() {

return cation != null && anion != null;

}

private final List<Integer> getAtomCharge(Atom atom) {

List<Integer> charges = new ArrayList();

if (Utils.stripQuotes(atom.getCharge()).split(";").length > 1) {

for (String s : Utils.stripQuotes(atom.getCharge()).split(";")) {

charges.add(Integer.valueOf(s));

}

} else {

charges.add(Integer.valueOf(atom.getCharge()));

}

return charges;

}

private String ionCombine(Object cat, Object ani, int cCat, int cAni) {

int x = NotationUtils.getBalance(cCat, cAni)[0];

int y = NotationUtils.getBalance(cCat, cAni)[1];

if (cat instanceof Atom && ani instanceof Atom) {

String a = ((Atom) cat).getSymbol();

String b = ((Atom) ani).getSymbol();

return (x == 1 ? a : a + x) + (y == 1 ? b : b + y);

} else if (cat instanceof Polyatom && ani instanceof Atom) {

String a = ((Polyatom) cat).getIon();

String b = ((Atom) ani).getSymbol();

return (x == 1 ? a : "(" + a + ")" + x) + (y == 1 ? b : b + y);

} else if (cat instanceof Atom && ani instanceof Polyatom) {

String a = ((Atom) cat).getSymbol();

String b = ((Polyatom) ani).getIon();

return (x == 1 ? a : a + x) + (y == 1 ? b : "(" + b + ")" + y);

} else if (cat instanceof Polyatom && ani instanceof Polyatom) {

String a = ((Polyatom) cat).getIon();

String b = ((Polyatom) ani).getIon();

return (x == 1 ? a : "(" + a + ")" + x) + (y == 1 ? b : "(" + b + ")" + y);

} else {

return null;

}

}

public Node getTextNode(String compound) {

TextFlow tf = new TextFlow();

tf.setStyle("-fx-background-color: #ffffff;");

tf.setUserData(compound);

tf.setTextAlignment(TextAlignment.CENTER);

tf.setPrefSize(Region.USE\_COMPUTED\_SIZE, Region.USE\_COMPUTED\_SIZE);

StringBuilder sb = new StringBuilder();

char[] c = compound.toCharArray();

for (int x = 0; x < c.length; x++) {

if (Character.isDigit(c[x])) {

if (sb.length() > 0) {

Text t2 = new Text(sb.toString());

t2.setStyle("-fx-font-size: 18;");

tf.getChildren().add(t2);

sb.setLength(0);

}

Text t1 = new Text(String.valueOf(c[x]));

t1.setStyle("-fx-font-size: 14;");

t1.setTranslateY(4.0);

tf.getChildren().add(t1);

} else {

sb.append(c[x]);

}

if (x + 1 == c.length) {

if (sb.length() > 0) {

Text t2 = new Text(sb.toString());

t2.setStyle("-fx-font-size: 18;");

tf.getChildren().add(t2);

}

}

}

return tf;

}

public List<Node> getProduct() {

if (this.isValid()) {

List<Node> list = new ArrayList<>();

List<Integer> lcCat = new ArrayList<>();

List<Integer> lcAni = new ArrayList<>();

if (this.cation instanceof Atom) {

lcCat.addAll(getAtomCharge((Atom) this.cation));

} else {

lcCat.add(((Polyatom) cation).getCharge());

}

if (this.anion instanceof Atom) {

lcAni.addAll(getAtomCharge((Atom) this.anion));

} else {

lcAni.add(((Polyatom) anion).getCharge());

}

List<String> resultedIons = new ArrayList<>();

for (int i : lcCat) {

for (int ii : lcAni) {

String s = ionCombine(this.cation, this.anion, i, ii);

if (s != null) {

if (!resultedIons.contains(s)) {

if (s.contains("HOH")) {

s = "H2O";

}

resultedIons.add(s);

}

}

}

}

for (String s : resultedIons) {

list.add(getTextNode(s));

}

return list;

} else {

return null;

}

}

}

package com.meyer.object;

/\*

\* Copyright (C) 2018 Ragil Rynaldo Meyer (meyer.ananda@gmail.com)

\*

\* This program is free software; you can redistribute it and/or

\* modify it under the terms of the GNU General Public License

\* as published by the Free Software Foundation; either version 2

\* of the License, or (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

\*/

/\*\*

\*

\* @author meyer

\*/

public class Atom extends Object {

private int number = 0;

private String name = "unknown";

private String symbol = "unknown";

private double mass = 0;

private String group = "unknown";

private String iupac = "unknown";

private int period = 0;

private String type = "unknown";

private double ionEnergy = 0;

private double negativity = 0;

private String inventor = "unknown";

private String year = "unknown";

private String charge = "unknown";

private int valence = 0;

public int getNumber() {

return number;

}

public void setNumber(int number) {

this.number = number;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getSymbol() {

return symbol;

}

public void setSymbol(String symbol) {

this.symbol = symbol;

}

public double getMass() {

return mass;

}

public void setMass(double mass) {

this.mass = mass;

}

public String getGroup() {

return group;

}

public void setGroup(String group) {

this.group = group;

}

public String getIupac() {

return iupac;

}

public void setIupac(String iupac) {

this.iupac = iupac;

}

public int getPeriod() {

return period;

}

public void setPeriod(int period) {

this.period = period;

}

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

public double getIonEnergy() {

return ionEnergy;

}

public void setIonEnergy(double ionEnergy) {

this.ionEnergy = ionEnergy;

}

public double getNegativity() {

return negativity;

}

public void setNegativity(double negativity) {

this.negativity = negativity;

}

public String getInventor() {

return inventor;

}

public void setInventor(String inventor) {

this.inventor = inventor;

}

public String getYear() {

return year;

}

public void setYear(String year) {

this.year = year;

}

public String getCharge() {

return charge;

}

public void setCharge(String charge) {this.charge = charge;

package com.meyer.object;

/\*

\* Copyright (C) 2018 Ragil Rynaldo Meyer (meyer.ananda@gmail.com)

\*

\* This program is free software; you can redistribute it and/or

\* modify it under the terms of the GNU General Public License

\* as published by the Free Software Foundation; either version 2

\* of the License, or (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

\*/

/\*\*

\*

\* @author meyer

\*/

public class Polyatom extends Object {

private int charge = 0;

private String name = "unknown";

private String composition = "unknown";

private String ion = "unknown";

public int getCharge() {

return charge;

}

public void setCharge(int charge) {

this.charge = charge;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getComposition() {

return composition;

}

public void setComposition(String composition) {

this.composition = composition;

}

public String getIon() {

return ion;

}

public void setIon(String notation) {

this.ion = notation;

}

}

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>

<?import javafx.scene.image.ImageView?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.ColumnConstraints?>

<?import javafx.scene.layout.GridPane?>

<?import javafx.scene.layout.RowConstraints?>

<?import javafx.scene.layout.StackPane?>

<?import javafx.scene.text.Font?>

<?import javafx.scene.text.Text?>

<GridPane alignment="CENTER" hgap="10.0" prefHeight="441.0" prefWidth="640.0" vgap="10.0" xmlns="http://javafx.com/javafx/8.0.141" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.meyer.controller.MainMenuController">

<columnConstraints>

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" maxWidth="335.0" minWidth="10.0" prefWidth="321.0" />

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" maxWidth="469.0" minWidth="10.0" prefWidth="319.0" />

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" maxWidth="469.0" minWidth="10.0" prefWidth="319.0" />

</columnConstraints>

<rowConstraints>

<RowConstraints maxHeight="225.0" minHeight="10.0" prefHeight="207.0" valignment="CENTER" vgrow="SOMETIMES" />

<RowConstraints maxHeight="217.0" minHeight="10.0" prefHeight="204.0" valignment="CENTER" vgrow="SOMETIMES" />

</rowConstraints>

<children>

<ImageView fx:id="imgTable" fitHeight="232.0" fitWidth="409.0" pickOnBounds="true" preserveRatio="true" GridPane.columnSpan="2" GridPane.halignment="CENTER" GridPane.hgrow="ALWAYS" GridPane.valignment="CENTER" GridPane.vgrow="NEVER" />

<ImageView fx:id="imgPredict" fitHeight="421.0" fitWidth="203.0" pickOnBounds="true" preserveRatio="true" GridPane.columnIndex="2" GridPane.halignment="CENTER" GridPane.hgrow="NEVER" GridPane.rowSpan="2" GridPane.valignment="CENTER" GridPane.vgrow="ALWAYS" />

<ImageView fx:id="img3D" fitHeight="203.0" fitWidth="414.0" pickOnBounds="true" preserveRatio="true" GridPane.columnSpan="2" GridPane.halignment="CENTER" GridPane.hgrow="ALWAYS" GridPane.rowIndex="1" GridPane.valignment="CENTER" GridPane.vgrow="NEVER" />

<StackPane fx:id="paneTable" prefHeight="65.0" prefWidth="410.0" style="-fx-background-color: transparent;" GridPane.columnSpan="2" GridPane.halignment="CENTER" GridPane.hgrow="NEVER" GridPane.valignment="CENTER" GridPane.vgrow="NEVER">

<children>

<Text fx:id="txtTable" strokeType="OUTSIDE" strokeWidth="0.0" text="Tabel Periode">

<font>

<Font size="30.0" />

</font>

</Text>

</children>

</StackPane>

<StackPane fx:id="panePredict" prefHeight="150.0" prefWidth="200.0" style="-fx-background-color: transparent;" GridPane.columnIndex="2" GridPane.halignment="CENTER" GridPane.hgrow="NEVER" GridPane.rowSpan="2" GridPane.valignment="CENTER" GridPane.vgrow="NEVER">

<children>

<Text fx:id="txtPredict" strokeType="OUTSIDE" strokeWidth="0.0" text="Prediksi">

<font>

<Font size="30.0" />

</font>

</Text>

</children>

</StackPane>

<StackPane fx:id="pane3D" prefHeight="150.0" prefWidth="200.0" style="-fx-background-color: transparent;" GridPane.columnSpan="2" GridPane.halignment="CENTER" GridPane.hgrow="NEVER" GridPane.rowIndex="1" GridPane.valignment="CENTER" GridPane.vgrow="NEVER">

<children>

<Text fx:id="txt3D" strokeType="OUTSIDE" strokeWidth="0.0" text="Geometri">

<font>

<Font size="30.0" />

</font>

</Text>

</children>

</StackPane>

<AnchorPane fx:id="aboutContainer" maxHeight="50.0" maxWidth="50.0" minHeight="50.0" minWidth="50.0" prefHeight="50.0" prefWidth="50.0" translateX="495.0" translateY="292.0">

<children>

<ImageView fx:id="imgAbout" fitHeight="50.0" fitWidth="50.0" pickOnBounds="true" preserveRatio="true" />

</children>

</AnchorPane>

</children>

<padding>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />

</padding>

</GridPane>

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.ScrollPane?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.control.TitledPane?>

<?import javafx.scene.layout.HBox?>

<?import javafx.scene.layout.StackPane?>

<?import javafx.scene.layout.VBox?>

<HBox maxHeight="542.0" maxWidth="846.0" minHeight="542.0" minWidth="846.0" prefHeight="542.0" prefWidth="846.0" xmlns="http://javafx.com/javafx/8.0.141" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.meyer.controller.PredictionTabController">

<children>

<VBox prefHeight="542.0" prefWidth="282.0" spacing="10.0" HBox.hgrow="NEVER">

<children>

<StackPane prefHeight="315.0" prefWidth="200.0" VBox.vgrow="ALWAYS">

<children>

<ScrollPane prefHeight="200.0" prefWidth="200.0">

<content>

<TitledPane fx:id="listCations" animated="false" text="Kation" />

</content>

</ScrollPane>

</children>

</StackPane>

<VBox prefHeight="150.0" prefWidth="200.0" VBox.vgrow="NEVER">

<children>

<StackPane prefHeight="150.0" prefWidth="200.0" VBox.vgrow="NEVER">

<children>

<TextField fx:id="textComp1" promptText="Kompon / Ion" />

</children>

<padding>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />

</padding>

</StackPane>

<StackPane prefHeight="73.0" prefWidth="182.0" VBox.vgrow="NEVER">

<children>

<TextField fx:id="textComp2" promptText="Kompon / Ion" />

</children>

<padding>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />

</padding>

</StackPane>

</children>

</VBox>

</children>

<padding>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />

</padding>

</VBox>

<VBox prefHeight="542.0" prefWidth="275.0" spacing="10.0" HBox.hgrow="NEVER">

<children>

<StackPane prefHeight="315.0" prefWidth="200.0" VBox.vgrow="ALWAYS">

<children>

<ScrollPane prefHeight="290.0" prefWidth="202.0">

<content>

<TitledPane fx:id="listAnions" animated="false" text="Anion" />

</content>

</ScrollPane>

</children>

</StackPane>

<StackPane prefHeight="150.0" prefWidth="200.0" VBox.vgrow="NEVER">

<children>

<Button fx:id="btnPredict" mnemonicParsing="false" text="Prediksi" />

</children>

</StackPane>

</children>

<padding>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />

</padding>

</VBox>

<VBox alignment="CENTER" prefHeight="542.0" prefWidth="289.0" spacing="10.0">

<children>

<StackPane fx:id="boxResult" prefHeight="542.0" prefWidth="287.0" VBox.vgrow="ALWAYS">

<padding>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />

</padding>

</StackPane>

<StackPane prefHeight="625.0" prefWidth="289.0" VBox.vgrow="ALWAYS">

<padding>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />

</padding>

<children>

<ScrollPane fx:id="boxInfo" prefHeight="200.0" prefWidth="200.0" StackPane.alignment="CENTER" />

</children>

</StackPane>

</children>

<padding>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />

</padding>

</VBox>

</children>

</HBox>

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>

<?import javafx.scene.layout.ColumnConstraints?>

<?import javafx.scene.layout.GridPane?>

<?import javafx.scene.layout.RowConstraints?>

<?import javafx.scene.layout.StackPane?>

<?import javafx.scene.text.Text?>

<GridPane fx:id="container" alignment="CENTER" hgap="2.0" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="571.0" prefWidth="981.0" vgap="2.0" xmlns="http://javafx.com/javafx/8.0.141" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.meyer.controller.PeriodTablesController">

<columnConstraints>

<ColumnConstraints halignment="CENTER" hgrow="NEVER" maxWidth="30.0" minWidth="30.0" prefWidth="30.0" />

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

</columnConstraints>

<rowConstraints>

<RowConstraints maxHeight="20.0" minHeight="20.0" prefHeight="20.0" valignment="CENTER" vgrow="NEVER" />

<RowConstraints minHeight="10.0" prefHeight="30.0" valignment="CENTER" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" valignment="CENTER" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" valignment="CENTER" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" valignment="CENTER" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" valignment="CENTER" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" valignment="CENTER" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" valignment="CENTER" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" valignment="CENTER" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" valignment="CENTER" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" valignment="CENTER" vgrow="SOMETIMES" />

</rowConstraints>

<children>

<StackPane fx:id="a1" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="1" GridPane.rowIndex="1" />

<StackPane fx:id="a3" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="1" GridPane.rowIndex="2" />

<StackPane fx:id="a11" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="1" GridPane.rowIndex="3" />

<StackPane fx:id="a19" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="1" GridPane.rowIndex="4" />

<StackPane fx:id="a37" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="1" GridPane.rowIndex="5" />

<StackPane fx:id="a55" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="1" GridPane.rowIndex="6" />

<StackPane fx:id="a87" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="1" GridPane.rowIndex="7" />

<StackPane fx:id="a4" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="2" GridPane.rowIndex="2" />

<StackPane fx:id="a12" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="2" GridPane.rowIndex="3" />

<StackPane fx:id="a20" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="2" GridPane.rowIndex="4" />

<StackPane fx:id="a38" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="2" GridPane.rowIndex="5" />

<StackPane fx:id="a56" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="2" GridPane.rowIndex="6" />

<StackPane fx:id="a88" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="2" GridPane.rowIndex="7" />

<StackPane fx:id="a39" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="3" GridPane.rowIndex="5" />

<StackPane fx:id="a40" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="4" GridPane.rowIndex="5" />

<StackPane fx:id="a41" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="5" GridPane.rowIndex="5" />

<StackPane fx:id="a42" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="6" GridPane.rowIndex="5" />

<StackPane fx:id="a43" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="7" GridPane.rowIndex="5" />

<StackPane fx:id="a44" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="8" GridPane.rowIndex="5" />

<StackPane fx:id="a45" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="9" GridPane.rowIndex="5" />

<StackPane fx:id="a46" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="10" GridPane.rowIndex="5" />

<StackPane fx:id="a47" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="11" GridPane.rowIndex="5" />

<StackPane fx:id="a48" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="12" GridPane.rowIndex="5" />

<StackPane fx:id="a49" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="13" GridPane.rowIndex="5" />

<StackPane fx:id="a50" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="14" GridPane.rowIndex="5" />

<StackPane fx:id="a51" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="15" GridPane.rowIndex="5" />

<StackPane fx:id="a52" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="16" GridPane.rowIndex="5" />

<StackPane fx:id="a53" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="17" GridPane.rowIndex="5" />

<StackPane fx:id="a54" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="18" GridPane.rowIndex="5" />

<StackPane fx:id="a2" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="18" GridPane.rowIndex="1" />

<StackPane fx:id="a10" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="18" GridPane.rowIndex="2" />

<StackPane fx:id="a18" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="18" GridPane.rowIndex="3" />

<StackPane fx:id="a36" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="18" GridPane.rowIndex="4" />

<StackPane fx:id="actinides" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="3" GridPane.rowIndex="7" />

<StackPane fx:id="a104" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="4" GridPane.rowIndex="7" />

<StackPane fx:id="a105" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="5" GridPane.rowIndex="7" />

<StackPane fx:id="a106" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="6" GridPane.rowIndex="7" />

<StackPane fx:id="a107" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="7" GridPane.rowIndex="7" />

<StackPane fx:id="a108" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="8" GridPane.rowIndex="7" />

<StackPane fx:id="a109" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="9" GridPane.rowIndex="7" />

<StackPane fx:id="a110" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="10" GridPane.rowIndex="7" />

<StackPane fx:id="a111" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="11" GridPane.rowIndex="7" />

<StackPane fx:id="a112" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="12" GridPane.rowIndex="7" />

<StackPane fx:id="a113" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="13" GridPane.rowIndex="7" />

<StackPane fx:id="a114" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="14" GridPane.rowIndex="7" />

<StackPane fx:id="a115" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="15" GridPane.rowIndex="7" />

<StackPane fx:id="a116" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="16" GridPane.rowIndex="7" />

<StackPane fx:id="a117" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="17" GridPane.rowIndex="7" />

<StackPane fx:id="a118" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="18" GridPane.rowIndex="7" />

<StackPane fx:id="a86" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="18" GridPane.rowIndex="6" />

<StackPane fx:id="lanthanides" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="3" GridPane.rowIndex="6" />

<StackPane fx:id="a72" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="4" GridPane.rowIndex="6" />

<StackPane fx:id="a73" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="5" GridPane.rowIndex="6" />

<StackPane fx:id="a74" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="6" GridPane.rowIndex="6" />

<StackPane fx:id="a75" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="7" GridPane.rowIndex="6" />

<StackPane fx:id="a76" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="8" GridPane.rowIndex="6" />

<StackPane fx:id="a77" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="9" GridPane.rowIndex="6" />

<StackPane fx:id="a78" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="10" GridPane.rowIndex="6" />

<StackPane fx:id="a79" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="11" GridPane.rowIndex="6" />

<StackPane fx:id="a80" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="12" GridPane.rowIndex="6" />

<StackPane fx:id="a81" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="13" GridPane.rowIndex="6" />

<StackPane fx:id="a82" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="14" GridPane.rowIndex="6" />

<StackPane fx:id="a83" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="15" GridPane.rowIndex="6" />

<StackPane fx:id="a84" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="16" GridPane.rowIndex="6" />

<StackPane fx:id="a85" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="17" GridPane.rowIndex="6" />

<StackPane fx:id="a9" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="17" GridPane.rowIndex="2" />

<StackPane fx:id="a8" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="16" GridPane.rowIndex="2" />

<StackPane fx:id="a7" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="15" GridPane.rowIndex="2" />

<StackPane fx:id="a6" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="14" GridPane.rowIndex="2" />

<StackPane fx:id="a5" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="13" GridPane.rowIndex="2" />

<StackPane fx:id="a17" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="17" GridPane.rowIndex="3" />

<StackPane fx:id="a16" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="16" GridPane.rowIndex="3" />

<StackPane fx:id="a15" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="15" GridPane.rowIndex="3" />

<StackPane fx:id="a14" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="14" GridPane.rowIndex="3" />

<StackPane fx:id="a13" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="13" GridPane.rowIndex="3" />

<StackPane fx:id="a35" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="17" GridPane.rowIndex="4" />

<StackPane fx:id="a34" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="16" GridPane.rowIndex="4" />

<StackPane fx:id="a33" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="15" GridPane.rowIndex="4" />

<StackPane fx:id="a32" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="14" GridPane.rowIndex="4" />

<StackPane fx:id="a31" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="13" GridPane.rowIndex="4" />

<StackPane fx:id="a21" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="3" GridPane.rowIndex="4" />

<StackPane fx:id="a22" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="4" GridPane.rowIndex="4" />

<StackPane fx:id="a23" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="5" GridPane.rowIndex="4" />

<StackPane fx:id="a24" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="6" GridPane.rowIndex="4" />

<StackPane fx:id="a25" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="7" GridPane.rowIndex="4" />

<StackPane fx:id="a26" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="8" GridPane.rowIndex="4" />

<StackPane fx:id="a27" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="9" GridPane.rowIndex="4" />

<StackPane fx:id="a28" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="10" GridPane.rowIndex="4" />

<StackPane fx:id="a29" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="11" GridPane.rowIndex="4" />

<StackPane fx:id="a30" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="12" GridPane.rowIndex="4" />

<StackPane fx:id="a57" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="3" GridPane.rowIndex="9" />

<StackPane fx:id="a58" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="4" GridPane.rowIndex="9" />

<StackPane fx:id="a59" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="5" GridPane.rowIndex="9" />

<StackPane fx:id="a60" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="6" GridPane.rowIndex="9" />

<StackPane fx:id="a61" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="7" GridPane.rowIndex="9" />

<StackPane fx:id="a62" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="8" GridPane.rowIndex="9" />

<StackPane fx:id="a63" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="9" GridPane.rowIndex="9" />

<StackPane fx:id="a64" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="10" GridPane.rowIndex="9" />

<StackPane fx:id="a65" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="11" GridPane.rowIndex="9" />

<StackPane fx:id="a66" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="12" GridPane.rowIndex="9" />

<StackPane fx:id="a67" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="13" GridPane.rowIndex="9" />

<StackPane fx:id="a68" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="14" GridPane.rowIndex="9" />

<StackPane fx:id="a69" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="15" GridPane.rowIndex="9" />

<StackPane fx:id="a70" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="16" GridPane.rowIndex="9" />

<StackPane fx:id="a71" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="17" GridPane.rowIndex="9" />

<StackPane fx:id="a103" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="17" GridPane.rowIndex="10" />

<StackPane fx:id="a102" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="16" GridPane.rowIndex="10" />

<StackPane fx:id="a101" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="15" GridPane.rowIndex="10" />

<StackPane fx:id="a100" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="14" GridPane.rowIndex="10" />

<StackPane fx:id="a99" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="13" GridPane.rowIndex="10" />

<StackPane fx:id="a98" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="12" GridPane.rowIndex="10" />

<StackPane fx:id="a97" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="11" GridPane.rowIndex="10" />

<StackPane fx:id="a96" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="10" GridPane.rowIndex="10" />

<StackPane fx:id="a95" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="9" GridPane.rowIndex="10" />

<StackPane fx:id="a94" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="8" GridPane.rowIndex="10" />

<StackPane fx:id="a93" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="7" GridPane.rowIndex="10" />

<StackPane fx:id="a92" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="6" GridPane.rowIndex="10" />

<StackPane fx:id="a91" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="5" GridPane.rowIndex="10" />

<StackPane fx:id="a90" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="4" GridPane.rowIndex="10" />

<StackPane fx:id="a89" prefHeight="150.0" prefWidth="200.0" GridPane.columnIndex="3" GridPane.rowIndex="10" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="1" GridPane.columnIndex="1" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="2" GridPane.columnIndex="2" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="3" GridPane.columnIndex="3" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="4" GridPane.columnIndex="4" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="5" GridPane.columnIndex="5" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="6" GridPane.columnIndex="6" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="7" GridPane.columnIndex="7" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="8" GridPane.columnIndex="8" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="9" GridPane.columnIndex="9" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="10" GridPane.columnIndex="10" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="11" GridPane.columnIndex="11" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="12" GridPane.columnIndex="12" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="13" GridPane.columnIndex="13" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="14" GridPane.columnIndex="14" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="18" GridPane.columnIndex="18" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="17" GridPane.columnIndex="17" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="16" GridPane.columnIndex="16" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="15" GridPane.columnIndex="15" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="1" GridPane.rowIndex="1" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="2" GridPane.rowIndex="2" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="3" GridPane.rowIndex="3" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="4" GridPane.rowIndex="4" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="5" GridPane.rowIndex="5" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="6" GridPane.rowIndex="6" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="7" GridPane.rowIndex="7" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="6" GridPane.rowIndex="9" />

<Text strokeType="OUTSIDE" strokeWidth="0.0" text="7" GridPane.rowIndex="10" />

</children>

<padding>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />

</padding>

</GridPane>

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.TextArea?>

<?import javafx.scene.layout.ColumnConstraints?>

<?import javafx.scene.layout.GridPane?>

<?import javafx.scene.layout.HBox?>

<?import javafx.scene.layout.RowConstraints?>

<?import javafx.scene.text.Font?>

<HBox alignment="CENTER\_LEFT" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" spacing="10.0" xmlns="http://javafx.com/javafx/8.0.141" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.meyer.controller.ElementDescriptionController">

<children>

<GridPane alignment="CENTER" prefHeight="380.0" prefWidth="282.0" HBox.hgrow="ALWAYS">

<columnConstraints>

<ColumnConstraints hgrow="NEVER" maxWidth="184.0" minWidth="10.0" prefWidth="147.0" />

<ColumnConstraints hgrow="ALWAYS" maxWidth="249.0" minWidth="10.0" prefWidth="62.0" />

</columnConstraints>

<rowConstraints>

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

</rowConstraints>

<children>

<Label prefHeight="19.0" prefWidth="156.0" text="Nomor Atom">

<font>

<Font size="15.0" />

</font>

</Label>

<Label prefHeight="19.0" prefWidth="156.0" text="Simbol Atom" GridPane.rowIndex="1">

<font>

<Font size="15.0" />

</font>

</Label>

<Label prefWidth="156.0" text="Nama Atom" GridPane.rowIndex="2">

<font>

<Font size="15.0" />

</font>

</Label>

<Label prefWidth="156.0" text="Massa Atom" GridPane.rowIndex="3">

<font>

<Font size="15.0" />

</font>

</Label>

<Label prefWidth="156.0" text="Grup Atom" GridPane.rowIndex="4">

<font>

<Font size="15.0" />

</font>

</Label>

<Label prefWidth="156.0" text="Nomor IUPAC" GridPane.rowIndex="5">

<font>

<Font size="15.0" />

</font>

</Label>

<Label prefWidth="156.0" text="Periode Atom" GridPane.rowIndex="6">

<font>

<Font size="15.0" />

</font>

</Label>

<Label prefWidth="156.0" text="Jenis Atom" GridPane.rowIndex="7">

<font>

<Font size="15.0" />

</font>

</Label>

<Label prefWidth="156.0" text="Energi Ionisasi" GridPane.rowIndex="8">

<font>

<Font size="15.0" />

</font>

</Label>

<Label prefWidth="156.0" text="Elektronegativitas" GridPane.rowIndex="9">

<font>

<Font size="15.0" />

</font>

</Label>

<Label prefWidth="156.0" text="Elektron Valensi" GridPane.rowIndex="10">

<font>

<Font size="15.0" />

</font>

</Label>

<Label prefWidth="156.0" text="Charge" GridPane.rowIndex="11">

<font>

<Font size="15.0" />

</font>

</Label>

<Label fx:id="number" text="0" GridPane.columnIndex="1">

<font>

<Font size="15.0" />

</font>

</Label>

<Label fx:id="symbol" prefHeight="19.0" text="Unknown" GridPane.columnIndex="1" GridPane.rowIndex="1">

<font>

<Font size="15.0" />

</font>

</Label>

<Label fx:id="name" text="Unknown" GridPane.columnIndex="1" GridPane.rowIndex="2">

<font>

<Font size="15.0" />

</font>

</Label>

<Label fx:id="mass" text="0.0" GridPane.columnIndex="1" GridPane.rowIndex="3">

<font>

<Font size="15.0" />

</font>

</Label>

<Label fx:id="group" text="Unknown" GridPane.columnIndex="1" GridPane.rowIndex="4">

<font>

<Font size="15.0" />

</font>

</Label>

<Label fx:id="iupac" text="0" GridPane.columnIndex="1" GridPane.rowIndex="5">

<font>

<Font size="15.0" />

</font>

</Label>

<Label fx:id="period" text="0" GridPane.columnIndex="1" GridPane.rowIndex="6">

<font>

<Font size="15.0" />

</font>

</Label>

<Label fx:id="type" text="Unknown" GridPane.columnIndex="1" GridPane.rowIndex="7">

<font>

<Font size="15.0" />

</font>

</Label>

<Label fx:id="ionization" text="0.0" GridPane.columnIndex="1" GridPane.rowIndex="8">

<font>

<Font size="15.0" />

</font>

</Label>

<Label fx:id="electro" text="0.0" GridPane.columnIndex="1" GridPane.rowIndex="9">

<font>

<Font size="15.0" />

</font>

</Label>

<Label fx:id="valence" text="0" GridPane.columnIndex="1" GridPane.rowIndex="10">

<font>

<Font size="15.0" />

</font>

</Label>

<Label fx:id="charge" text="Unknown" GridPane.columnIndex="1" GridPane.rowIndex="11">

<font>

<Font size="15.0" />

</font>

</Label>

</children>

</GridPane>

<TextArea fx:id="txtHistory" editable="false" prefHeight="200.0" prefWidth="200.0" promptText="Tidak ada informasi" wrapText="true" HBox.hgrow="ALWAYS" />

<TextArea fx:id="txtUses" editable="false" prefHeight="200.0" prefWidth="200.0" promptText="Tidak ada informasi" wrapText="true" HBox.hgrow="ALWAYS" />

</children>

<padding>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />

</padding>

</HBox>

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.HBox?>

<?import javafx.scene.layout.StackPane?>

<?import javafx.scene.layout.VBox?>

<HBox maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="539.0" prefWidth="882.0" spacing="10.0" xmlns="http://javafx.com/javafx/8.0.141" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.meyer.controller.Atom3DViewerController">

<children>

<VBox alignment="CENTER" prefHeight="539.0" prefWidth="593.0" spacing="10.0" HBox.hgrow="NEVER">

<children>

<StackPane fx:id="containerWeb" prefHeight="502.0" prefWidth="593.0" VBox.vgrow="ALWAYS" />

<TextField fx:id="boxSearch" maxHeight="26.0" maxWidth="315.0" prefHeight="26.0" prefWidth="315.0" promptText="Input Name Or Molecular Notation" VBox.vgrow="NEVER" />

<HBox alignment="CENTER" prefHeight="100.0" prefWidth="200.0" spacing="10.0" VBox.vgrow="NEVER">

<children>

<Button fx:id="btnNIH" mnemonicParsing="false" prefHeight="26.0" prefWidth="154.0" text="NIH" HBox.hgrow="NEVER" />

<Button fx:id="btnNCI" mnemonicParsing="false" prefHeight="26.0" prefWidth="144.0" text="NCI" HBox.hgrow="NEVER" />

</children>

</HBox>

</children>

<padding>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />

</padding>

</VBox>

<VBox prefHeight="400.0" prefWidth="240.0" spacing="10.0" HBox.hgrow="ALWAYS">

<children>

<StackPane fx:id="containerImage" prefHeight="150.0" prefWidth="200.0" VBox.vgrow="NEVER" />

<StackPane fx:id="containerInfo" prefHeight="205.0" prefWidth="320.0" VBox.vgrow="ALWAYS" />

</children>

<padding>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />

</padding>

</VBox>

</children>

<padding>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />

</padding>

</HBox>