Unit Converter

For Object, Oriented Programming using Java

This Project is built Using Java 8, Javafxml 2.0 and Jfoenix material design theme On the Netbeans IDE 2.x and Atom 1.x. Also this project requires a minimum of JRE 8u60.

It follows the mvc (Model view control framework) for handling the gui and backend logic. it follows a systematic approach to convert from one unit to another using a sophesticated data structures as opposed to hard coded values and functions in an optimum manner.

By

Joy Jain (159102060)

Jayesh Bansal (159102059)

Contents

[2 Source Code 3](#_Toc469072619)

[2.1 Area.java 3](#_Toc469072620)

[2.2 DataTransferRate.java 4](#_Toc469072621)

[2.3 DigitalStorage.java 6](#_Toc469072622)

[2.4 Frequency.java 8](#_Toc469072623)

[2.5 Length.java 9](#_Toc469072624)

[2.6 Temperature.java 10](#_Toc469072625)

[2.7 Volume.java 11](#_Toc469072626)

[2.8 CustomValidator.java 13](#_Toc469072627)

[2.9 MainController.java 14](#_Toc469072628)

[2.10 Converter.java 18](#_Toc469072629)

[2.11 UnitTypes.java 19](#_Toc469072630)

[2.12 Main.fxml 20](#_Toc469072631)

[3 Output Samples 23](#_Toc469072632)

# Source Code

## Area.java

package converter.converters;

/\*\*

\*

\* Area a = new Area(2, 4, 0, 1, 1); a.convert();

\*/

public class Area {

/\*\*

\* "Square metre", "Hectare", "Square kilometre", "Square inch", "Square

\* foot", "Square yard", "Acre", "Square mile"

\*/

private int from, to, type1, type2;

private final double units[][] = {

{0, 4, 6},

{1, 144, 1296, 6272640, Math.pow(63360, 2)}

};

private double input, output;

public Area(int from, int to, int type1, int type2, double input) {

this.from = from;

this.to = to;

this.type1 = type1;

this.type2 = type2;

this.input = input;

}

public double convert() {

if (type1 == type2) {

if (type1 == 0) {

output = convertImperial(from, to);

} else {

output = convertUS(from, to);

}

} else if (type1 == 0 && type2 == 1) {

input = imperialToUS(convertImperial(from, 0));

output = convertUS(0, to, input);

} else {

input = USToImperial(convertUS(from, 0));

output = convertImperial(0, to, input);

}

return output;

}

public double convertImperial(int from, int to) {

return (Math.pow(10, units[0][from]) / Math.pow(10, units[0][to])) \* input;

}

public double convertUS(int from, int to) {

return (units[1][from] / units[1][to]) \* input;

}

public double convertImperial(int from, int to, double input) {

double output;

if (from < to) {

output = (Math.pow(10, units[0][from]) / Math.pow(10, units[0][to])) \* input;

} else {

output = (Math.pow(10, units[0][to]) / Math.pow(10, units[0][from])) \* input;

}

return output;

}

public double convertUS(int from, int to, double input) {

double output;

if (from < to) {

output = (units[1][from] / units[1][to]) \* input;

} else {

output = (units[1][to] / units[1][from]) \* input;

}

return output;

}

private double imperialToUS(double input) {

return (1550.00 \* input);

}

private double USToImperial(double input) {

return (6.4516 \* (Math.pow(10, -4)) \* input);

}

}

## DataTransferRate.java

package converter.converters;

/\*\*

\*

\* d = new DataTransferRate(3, 2, 1, 0, 1); d.convert();

\*/

public class DataTransferRate {

/\*\*

\* "Bit per second", "Kilobit per second", "Megabit per second", "Gigabit

\* per second", "Terabit per second", "Byte per second", "Kilobyte per

\* second", "Megabyte per second", "Gigabyte per second", "Terabyte per

\* second"

\*/

private int from, to, type1, type2;

private final double units[][] = {

{0, 3, 6, 9, 12},

{0, 3, 6, 9, 12}

};

private double input, output;

public DataTransferRate(int from, int to, int type1, int type2, double input) {

this.from = from;

this.to = to;

this.type1 = type1;

this.type2 = type2;

this.input = input;

}

public double convert() {

if (type1 == type2) {

if (type1 == 0) {

output = convertBits(from, to);

} else {

output = convertBytes(from, to);

}

} else if (type1 == 0 && type2 == 1) {

input = bitsToBytes(convertBits(from, 0));

output = convertBytes(0, to, input);

} else {

input = bytesToBits(convertBytes(from, 0));

output = convertBits(0, to, input);

}

return output;

}

public double convertBits(int from, int to) {

return (Math.pow(10, units[0][from]) / Math.pow(10, units[0][to])) \* input;

}

public double convertBytes(int from, int to) {

return (Math.pow(10, units[1][from]) / Math.pow(10, units[1][to])) \* input;

}

public double convertBits(int from, int to, double input) {

double output;

if (from < to) {

output = (Math.pow(10, units[0][from]) / Math.pow(10, units[0][to])) \* input;

} else {

output = (Math.pow(10, units[0][to]) / Math.pow(10, units[0][from])) \* input;

}

return output;

}

public double convertBytes(int from, int to, double input) {

double output;

if (from < to) {

output = (Math.pow(10, units[1][from]) / Math.pow(10, units[1][to])) \* input;

} else {

output = (Math.pow(10, units[1][to]) / Math.pow(10, units[1][from])) \* input;

}

return output;

}

private double bitsToBytes(double input) {

return 0.125 \* input;

}

private double bytesToBits(double input) {

return 8 \* input;

}

}

## DigitalStorage.java

package converter.converters;

/\*\*

\*

\* convert d = new DigitalStorage(3, 2, 0, 1, 1); d.convert();

\*/

public class DigitalStorage {

/\*\*

\* "Bit", "Kilobit", "Megabit", "Gigabit", "Terabit", "Petabit", "Byte",

\* "Kilobyte", "Megabyte", "Gigabyte", "Terabyte", "Petabyte"

\*/

private int from, to, type1, type2;

private final double units[][] = {

{0, 3, 6, 9, 12, 15},

{0, 3, 6, 9, 12, 15}

};

private double input, output;

public DigitalStorage(int from, int to, int type1, int type2, double input) {

this.from = from;

this.to = to;

this.type1 = type1;

this.type2 = type2;

this.input = input;

}

public double convert() {

if (type1 == type2) {

if (type1 == 0) {

output = convertBits(from, to);

} else {

output = convertBytes(from, to);

}

} else if (type1 == 0 && type2 == 1) {

input = bitsToBytes(convertBits(from, 0));

output = convertBytes(0, to, input);

} else {

input = bytesToBits(convertBytes(from, 0));

output = convertBits(0, to, input);

}

return output;

}

public double convertBits(int from, int to) {

return (Math.pow(10, units[0][from]) / Math.pow(10, units[0][to])) \* input;

}

public double convertBytes(int from, int to) {

return (Math.pow(10, units[1][from]) / Math.pow(10, units[1][to])) \* input;

}

public double convertBits(int from, int to, double input) {

double output;

if (from < to) {

output = (Math.pow(10, units[0][from]) / Math.pow(10, units[0][to])) \* input;

} else {

output = (Math.pow(10, units[0][to]) / Math.pow(10, units[0][from])) \* input;

}

return output;

}

public double convertBytes(int from, int to, double input) {

double output;

if (from < to) {

output = (Math.pow(10, units[1][from]) / Math.pow(10, units[1][to])) \* input;

} else {

output = (Math.pow(10, units[1][to]) / Math.pow(10, units[1][from])) \* input;

}

return output;

}

private double bitsToBytes(double input) {

return ((Math.pow(2, -3)) \* input);

}

private double bytesToBits(double input) {

return ((Math.pow(2, 3)) \* input);

}

}

## Frequency.java

package converter.converters;

/\*\*

\*

\* convert f = new Frequency(3, 0, 1); f.convert();

\*/

public class Frequency {

/\*\*

\* "Hertz", "Kilohertz", "Megahertz", "Gigahertz"

\*/

private int from, to;

private final double units[] = {0, 3, 6, 9};

private double input, output;

public Frequency(int from, int to, double input) {

this.from = from;

this.to = to;

this.input = input;

}

public double convert() {

if (from < to) {

output = (Math.pow(10, units[to]) / Math.pow(10, units[from])) \* input;

} else {

output = (Math.pow(10, units[from]) / Math.pow(10, units[to])) \* input;

}

return output;

}

}

## Length.java

package converter.converters;

/\*\*

\*

\* convert l = new Length(4, 2, 0, 1, 1); l.convert();

\*/

public class Length {

/\*\*

\* "nanometre", "Micrometre", "millimetre", "Centimetre", "Metre",

\* "Kilometre", "Inch", "Foot", "Yard", "Mile"

\*/

private int from, to, type1, type2;

private final double units[][] = {

{0, 3, 6, 7, 9, 12},

{1, 12, 36, 63360}

};

private double input, output;

public Length(int from, int to, int type1, int type2, double input) {

this.from = from;

this.to = to;

this.type1 = type1;

this.type2 = type2;

this.input = input;

}

public double convert() {

if (type1 == type2) {

if (type1 == 0) {

output = convertImperial(from, to);

} else {

output = convertUS(from, to);

}

} else if (type1 == 0 && type2 == 1) {

input = imperialToUS(convertImperial(from, 3));

output = convertUS(0, to, input);

} else {

input = USToImperial(convertUS(from, 0));

output = convertImperial(0, to, input);

}

return output;

}

public double convertImperial(int from, int to) {

return (Math.pow(10, units[0][from]) / Math.pow(10, units[0][to])) \* input;

}

public double convertUS(int from, int to) {

return (units[1][from] / units[1][to]) \* input;

}

public double convertImperial(int from, int to, double input) {

double output;

if (from < to) {

output = (Math.pow(10, units[0][from]) / Math.pow(10, units[0][to])) \* input;

} else {

output = (Math.pow(10, units[0][to]) / Math.pow(10, units[0][from])) \* input;

}

return output;

}

public double convertUS(int from, int to, double input) {

double output;

if (from < to) {

output = (units[1][from] / units[1][to]) \* input;

} else {

output = (units[1][to] / units[1][from]) \* input;

}

return output;

}

private double imperialToUS(double input) {

return (3.93701 \* Math.pow(10, -1) \* input);

}

private double USToImperial(double input) {

return (25400000 \* input);

}

}

## Temperature.java

package converter.converters;

/\*\*

\*

\* convert t = new Temperature(2, 1, 256); t.convert();

\*/

public class Temperature {

/\*\*

\* "Celsius", "Fahrenheit", "Kelvin"

\*/

private int from, to;

private double input, output;

public Temperature(int from, int to, double input) {

this.from = from;

this.to = to;

this.input = input;

}

public double convert() {

if (from == 0 && to == 1) {

output = ((input \* 9) / 5) + (32.00);

} else if (from == 0 && to == 2) {

output = (input + 273.15);

} else if (from == 1 && to == 0) {

output = ((input - 32.00) \* 5) / (9);

} else if (from == 1 && to == 2) {

input = ((input - 32.00) \* 5) / (9);

output = (input + 273.15);

} else if (from == 2 && to == 0) {

output = (input - 273.15);

} else if (from == 2 && to == 1) {

input = (input - 273.15);

output = ((input \* 9) / 5) + (32.00);

}

return output;

}

}

## Volume.java

package converter.converters;

/\*\*

\* v = new Volume(1, 0, 0, 1, 1); v.convert();

\*/

public class Volume {

/\*\*

\* "Millilitre", "Litre", "Cubic metre", "Cubic inch", "Cubic foot"

\*/

private int from, to, type1, type2;

private final double units[][] = {

{0, 3, 6},

{1, 1728}

};

private double input, output;

public Volume(int from, int to, int type1, int type2, double input) {

this.from = from;

this.to = to;

this.type1 = type1;

this.type2 = type2;

this.input = input;

}

public double convert() {

if (type1 == type2) {

if (type1 == 0) {

output = convertImperial(from, to);

} else {

output = convertUS(from, to);

}

} else if (type1 == 0 && type2 == 1) {

input = imperialToUS(convertImperial(from, 0));

output = convertUS(0, to, input);

} else {

input = USToImperial(convertUS(from, 0));

output = convertImperial(0, to, input);

}

return output;

}

public double convertImperial(int from, int to) {

return (Math.pow(10, units[0][from]) / Math.pow(10, units[0][to])) \* input;

}

public double convertUS(int from, int to) {

return (units[1][from] / units[1][to]) \* input;

}

public double convertImperial(int from, int to, double input) {

double output;

if (from < to) {

output = (Math.pow(10, units[0][from]) / Math.pow(10, units[0][to])) \* input;

} else {

output = (Math.pow(10, units[0][to]) / Math.pow(10, units[0][from])) \* input;

}

return output;

}

public double convertUS(int from, int to, double input) {

double output;

if (from < to) {

output = (units[1][from] / units[1][to]) \* input;

} else {

output = (units[1][to] / units[1][from]) \* input;

}

return output;

}

private double imperialToUS(double input) {

return (0.061023 \* input);

}

private double USToImperial(double input) {

return (16.387052 \* input);

}

}

## CustomValidator.java

package converter.gui.main;

import javafx.beans.DefaultProperty;

import javafx.scene.control.TextArea;

import javafx.scene.control.TextField;

import javafx.scene.control.TextInputControl;

import com.jfoenix.validation.base.ValidatorBase;

@DefaultProperty(value = "icon")

public class CustomValidator extends ValidatorBase {

/\*\*

\* {@inheritDoc}

\*/

@Override

protected void eval() {

if (srcControl.get() instanceof TextInputControl) {

TextInputControl textField = (TextInputControl) srcControl.get();

if (textField.getText() == null || textField.getText().equals("")) {

hasErrors.set(true);

} else {

hasErrors.set(false);

}

// or if it's not double

hasErrors.set(false);

try {

Double.parseDouble(textField.getText());

} catch (NumberFormatException e) {

hasErrors.set(true);

}

}

}

}

## MainController.java

package converter.gui.main;

import javax.annotation.PostConstruct;

import io.datafx.controller.FXMLController;

import io.datafx.controller.flow.FlowException;

import io.datafx.controller.util.VetoException;

import io.datafx.controller.flow.context.FXMLViewFlowContext;

import io.datafx.controller.flow.context.ViewFlowContext;

import javafx.scene.layout.StackPane;

import javafx.fxml.FXML;

import javafx.scene.control.Label;

import com.jfoenix.controls.JFXComboBox;

import com.jfoenix.controls.JFXButton;

import com.jfoenix.controls.JFXTextField;

import com.jfoenix.controls.JFXDialog;

import com.jfoenix.controls.JFXDialog.DialogTransition;

import converter.UnitTypes;

import converter.converters.\*;

@FXMLController(value = "/resources/fxml/Main.fxml", title = "Converter v1.0")

public class MainController {

@FXMLViewFlowContext

private ViewFlowContext context;

@FXML

private StackPane root, mainbody;

@FXML

private JFXComboBox types, inputType, outputType;

@FXML

private JFXTextField inputText, outputText;

@FXML

private JFXButton convert, swap, clear, about, ok;

@FXML

private JFXDialog dialog;

@PostConstruct

public void init() throws FlowException, VetoException {

root.getChildren().remove(dialog);

// init the types combo

for (String type : UnitTypes.types) {

types.getItems().add(new Label(type));

}

// on changing conversion type

types.setOnAction((e) -> {

// remove all children of inputTypes and outoutTypes if present

inputType.getItems().clear();

outputType.getItems().clear();

// add new items according to the current selection index

for (String unit : UnitTypes.units[types.getSelectionModel().getSelectedIndex()]) {

inputType.getItems().add(new Label(unit));

outputType.getItems().add(new Label(unit));

}

});

// validate input and output

inputText.focusedProperty().addListener((o, oldVal, newVal) -> {

inputText.validate();

});

// convert input to output

convert.setOnMouseClicked((e) -> {

Double inputData = 0.0,

outputData = 0.0;

try {

inputData = Double.parseDouble(inputText.getText());

} catch (NumberFormatException err) {

inputText.validate();

} finally {

if (inputData != 0.0) {

int type1 = 0,

type2 = 0,

unitofType = types.getSelectionModel().getSelectedIndex(),

from = inputType.getSelectionModel().getSelectedIndex(),

to = outputType.getSelectionModel().getSelectedIndex();

switch (unitofType) {

case 0:

if (from > 2) {

type1 = 1;

from -= 3;

}

if (to > 2) {

type2 = 1;

to -= 3;

}

Area a = new Area(from, to, type1, type2, inputData);

outputData = a.convert();

break;

case 1:

if (from > 4) {

type1 = 1;

from -= 5;

}

if (to > 4) {

type2 = 1;

to -= 5;

}

DataTransferRate dtr = new DataTransferRate(from, to, type1, type2, inputData);

outputData = dtr.convert();

break;

case 2:

if (from > 5) {

type1 = 1;

from -= 6;

}

if (to > 5) {

type2 = 1;

to -= 6;

}

DigitalStorage ds = new DigitalStorage(from, to, type1, type2, inputData);

outputData = ds.convert();

break;

case 3:

Frequency f = new Frequency(from, to, inputData);

outputData = f.convert();

break;

case 4:

if (from > 5) {

type1 = 1;

from -= 6;

}

if (to > 5) {

type2 = 1;

to -= 6;

}

Length l = new Length(from, to, type1, type2, inputData);

outputData = l.convert();

break;

case 5:

Temperature t = new Temperature(from, to, inputData);

outputData = t.convert();

break;

case 6:

if (from > 2) {

type1 = 1;

from -= 3;

}

if (to > 2) {

type2 = 1;

to -= 3;

}

Volume v = new Volume(from, to, type1, type2, inputData);

outputData = v.convert();

break;

default:

}

outputText.setText(outputData.toString());

}

}

});

// swap units if swapable

swap.setOnMouseClicked((e) -> {

// check if the selected conversion type is in a valid state

if (types.getSelectionModel().getSelectedIndex() > -1) {

int from = inputType.getSelectionModel().getSelectedIndex(),

to = outputType.getSelectionModel().getSelectedIndex();

from += to;

to = from - to;

from -= to;

// now set them

inputType.getSelectionModel().select(from);

outputType.getSelectionModel().select(to);

}

});

// clear input and output

clear.setOnMouseClicked((e) -> {

inputText.clear();

outputText.clear();

});

// show about us dialog

about.setOnMouseClicked((e) -> {

dialog.setTransitionType(DialogTransition.CENTER);

dialog.show(mainbody);

});

// hide about us dialog

ok.setOnMouseClicked((e) -> {

dialog.close();

});

}

}

## Converter.java

package converter;

/\*\*

\*

\* @author Joy Jain

\*/

import com.jfoenix.controls.JFXDecorator;

import com.jfoenix.svg.SVGGlyphLoader;

import converter.gui.main.MainController;

import io.datafx.controller.flow.Flow;

import io.datafx.controller.flow.container.DefaultFlowContainer;

import io.datafx.controller.flow.context.FXMLViewFlowContext;

import io.datafx.controller.flow.context.ViewFlowContext;

import javafx.application.Application;

import javafx.scene.Scene;

import javafx.stage.Stage;

public class Converter extends Application {

@FXMLViewFlowContext

private ViewFlowContext flowContext;

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

launch(args);

}

@Override

public void start(Stage stage) throws Exception {

new Thread(() -> {

try {

SVGGlyphLoader.loadGlyphsFont(Converter.class.getResourceAsStream("/resources/fonts/icomoon.svg"), "icomoon.svg");

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}).start();

Flow flow = new Flow(MainController.class);

DefaultFlowContainer container = new DefaultFlowContainer();

flowContext = new ViewFlowContext();

flowContext.register("Stage", stage);

flow.createHandler(flowContext).start(container);

JFXDecorator decorator = new JFXDecorator(stage, container.getView());

decorator.setCustomMaximize(true);

Scene scene = new Scene(decorator, 600, 500);

scene.getStylesheets().add(Converter.class.getResource("/resources/css/jfoenix-fonts.css").toExternalForm());

scene.getStylesheets().add(Converter.class.getResource("/resources/css/jfoenix-design.css").toExternalForm());

scene.getStylesheets().add(Converter.class.getResource("/resources/css/jfoenix-main-demo.css").toExternalForm());

stage.setTitle("Converter v1.0");

stage.setScene(scene);

stage.show();

}

}

## UnitTypes.java

package converter;

public class UnitTypes {

public static final String types[] = {"Area", "Data Transfer Rate", "Digital Storage", "Frequency", "Length", "Temperature", "Volume"},

units[][] = {

{"Square metre", "Hectare", "Square kilometre", "Square inch", "Square foot", "Square yard", "Acre", "Square mile"},

{"Bit per second", "Kilobit per second", "Megabit per second", "Gigabit per second", "Terabit per second", "Byte per second", "Kilobyte per second", "Megabyte per second", "Gigabyte per second", "Terabyte per second"},

{"Bit", "Kilobit", "Megabit", "Gigabit", "Terabit", "Petabit", "Byte", "Kilobyte", "Megabyte", "Gigabyte", "Terabyte", "Petabyte"},

{"Hertz", "Kilohertz", "Megahertz", "Gigahertz"},

{"nanometre", "Micrometre", "millimetre", "Centimetre", "Metre", "Kilometre", "Inch", "Foot", "Yard", "Mile"},

{"Celsius", "Fahrenheit", "Kelvin"},

{"Millilitre", "Litre", "Cubic metre", "Cubic inch", "Cubic foot"}

};

}

## Main.fxml

<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.\*?>

<?import javafx.scene.control.\*?>

<?import javafx.scene.layout.\*?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.layout.StackPane?>

<?import de.jensd.fx.fontawesome.AwesomeIcon?>

<?import de.jensd.fx.fontawesome.AwesomeIconsStack?>

<?import de.jensd.fx.fontawesome.Icon?>

<?import javafx.scene.shape.SVGPath?>

<?import javafx.scene.control.ProgressBar?>

<?import javafx.scene.shape.Rectangle?>

<?import javafx.scene.layout.GridPane?>

<?import javafx.scene.layout.VBox?>

<?import javafx.scene.shape.Line?>

<?import javafx.scene.layout.Pane?>

<?import javafx.scene.layout.HBox?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.layout.FlowPane?>

<?import com.jfoenix.controls.JFXToolbar?>

<?import com.jfoenix.controls.JFXTextField?>

<?import com.jfoenix.controls.JFXRippler?>

<?import com.jfoenix.controls.JFXListView?>

<?import com.jfoenix.controls.JFXButton?>

<?import com.jfoenix.controls.JFXComboBox?>

<?import com.jfoenix.controls.JFXDialog?>

<?import com.jfoenix.controls.JFXDialogLayout?>

<?import converter.gui.main.CustomValidator?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.layout.BorderPane?>

<?import javafx.geometry.Insets?>

<StackPane fx:id="root" xmlns:fx="http://javafx.com/fxml/1"

xmlns="http://javafx.com/javafx/2.2">

<BorderPane>

<top>

<VBox spacing="20">

<JFXToolbar>

<leftItems>

<Label>Converter v1.0</Label>

</leftItems>

</JFXToolbar>

</VBox>

</top>

<center>

<StackPane id="mainbody" xmlns:fx="http://javafx.com/fxml/1"

xmlns="http://javafx.com/javafx/2.2">

<VBox spacing="20" alignment="CENTER">

<JFXComboBox id="types" promptText="Choose a Conversion">

</JFXComboBox>

<HBox spacing="20" alignment="CENTER">

<JFXTextField maxWidth="50" id="inputText" promptText="Input Unit">

<validators>

<CustomValidator id="inputTextValidator" message="Please enter a number only!">

<Icon awesomeIcon="WARNING" style="-fx-font-family: FontAwesome;" />

</CustomValidator>

</validators>

</JFXTextField>

<Label style="-fx-text-fill:#5264AE;-fx-font-size:26px;">=</Label>

<JFXTextField id="outputText" promptText="Output Unit"></JFXTextField>

<VBox.margin>

<Insets>

<left>25</left>

<right>25</right>

</Insets>

</VBox.margin>

</HBox>

<HBox spacing="40" alignment="CENTER">

<JFXComboBox minWidth="150" id="inputType" promptText="Input Unit">

</JFXComboBox>

<JFXComboBox minWidth="150" id="outputType" promptText="Output Unit">

</JFXComboBox>

</HBox>

<HBox spacing="20" alignment="CENTER">

<JFXButton id="convert" buttonType="RAISED" prefWidth="120" text="Convert" style="-fx-text-fill:WHITE;-fx-background-color:#5264AE;-fx-font-size:14px;">

<HBox.margin>

<Insets>

<top>15</top>

</Insets>

</HBox.margin>

</JFXButton>

<JFXButton id="swap" buttonType="RAISED" prefWidth="120" text="Swap" style="-fx-text-fill:WHITE;-fx-background-color:#4CAF50;-fx-font-size:14px;">

<HBox.margin>

<Insets>

<top>15</top>

</Insets>

</HBox.margin>

</JFXButton>

<JFXButton id="clear" buttonType="RAISED" prefWidth="120" text="Clear" style="-fx-text-fill:WHITE;-fx-background-color:#F44336;-fx-font-size:14px;">

<HBox.margin>

<Insets>

<top>15</top>

</Insets>

</HBox.margin>

</JFXButton>

<JFXButton id="about" buttonType="RAISED" prefWidth="120" text="About" style="-fx-text-fill:WHITE;-fx-background-color:#5264AE;-fx-font-size:14px;">

<HBox.margin>

<Insets>

<top>15</top>

</Insets>

</HBox.margin>

</JFXButton>

</HBox>

<StackPane.margin>

<Insets>

<top>-160</top>

</Insets>

</StackPane.margin>

</VBox>

</StackPane>

</center>

</BorderPane>

<JFXDialog id="dialog">

<JFXDialogLayout>

<heading>

<Label>Thanks for using!</Label>

</heading>

<body>

<Label>

This Project was built by Joy Jain and Jayesh

Using Java 8, Javafxml 2.0 and Jfoenix material design theme

On the Netbeans IDE 2.0 and Atom 1.x

Also this project requires a minimum of JRE 8u60

</Label>

</body>

<actions>

<JFXButton id="ok" styleClass="dialog-accept">ACCEPT

</JFXButton>

</actions>

</JFXDialogLayout>

</JFXDialog>

</StackPane>

# Output Samples











