```java
/*
 * Title:         Number Conversion System
 * Developed by:  Md. An Nahian Prince
 * ID:            12105007
 * Availability:  Converts and performs arithmetic on
custom/predefined bases
 *                (Binary, Decimal, Octal, Hexadecimal).
 * Key Features:
 * - Custom base number operations
 * - Base conversions (Binary, Decimal, Octal, Hexadecimal,
Custom)
 * - Addition and subtraction in any base
 * - Fractional number support
 */

package application;

import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.scene.text.Font;
import javafx.stage.Stage;

public class NumberConversionSystem extends Application {

    @Override
    /*****************************
     *
     * Primary or First Window
     *
     *****************************/

    public void start(Stage primaryStage) {
        primaryStage.setTitle("Number Conversion System");

        // Main layout
```

```
        VBox mainLayout = new VBox(50);
        // 50 means vertical gap of every button in 50
pixels

        // (top, right, bottom, left)
        mainLayout.setPadding(new Insets(20, 20, 20, 20));
        mainLayout.setAlignment(Pos.CENTER);

        // primary dialogue box bg
        mainLayout.setStyle("-fx-background-color: linear-
gradient(to bottom, #ffd4c2, #ffe5d1\r\n"
                + ");");
        // Serene Lavender Gradient

        // Show in title bar
        Label titleLabel = new Label("Number Conversion
System");
        titleLabel.setFont(Font.font("Arial", 28));
        // Customize title font

        // Custom Base Button
        Button customBaseButton = new Button("Use Custom
Base");
        setButtonStyle(customBaseButton, "#1B5E20", 35);
        // deep green
        // Customize button style

        // when click then called "openCustomBaseSelection"
function for open that
        // window
        /*
         * Passing primaryStage as a parameter lets
openCustomBaseSelection manage
         * the main window (stage) for scene transitions
while keeping the design
         * modular
         */
        customBaseButton.setOnAction(e ->
openCustomBaseSelection(primaryStage));
```

```
        // This is the option to press button in first
window

        // Default Base Button
        Button defaultBaseButton = new Button("Use Default
Base");
        setButtonStyle(defaultBaseButton, "#4A148C", 36);
        // Vibrant orange
        // Customize button style

        // when click then called "openDefaultBaseSelection"
function for open that
        // window
        /*
         * Passing primaryStage as a parameter lets
openCustomBaseSelection manage
         * the main window (stage) for scene transitions
while keeping the design
         * modular
         */
        defaultBaseButton.setOnAction(e ->
openDefaultBaseSelection(primaryStage));
        // This is the option to press button in first
window

        /*
         * For Add a Universal Converter
         */

        // Universal Base Converter Button
        Button universalBaseButton = new Button("Universal
Base Converter");
        setButtonStyle(universalBaseButton, "#0D47A1", 30);
        universalBaseButton.setOnAction(e ->
openUniversalBaseConverter(primaryStage));

        /*
         * here modify the hierarchical method so that which
is the root node find
```

```
         * title is the root and other button is children
         */
        mainLayout.getChildren().addAll(titleLabel,
customBaseButton, defaultBaseButton, universalBaseButton);

        // Main scene
        /*
         * Primary Dialogue Box
         * Width x Height
         */
        Scene mainScene = new Scene(mainLayout, 450, 500);
        primaryStage.setScene(mainScene);
        primaryStage.show();
        // show the primary screen
    }

    /*****************************
     *
     * If press "Use Custom Base"
     * Open Second Display
     *
     *****************************/

    private void openCustomBaseSelection(Stage primaryStage)
{
        VBox selectionLayout = new VBox(15);
        selectionLayout.setPadding(new Insets(20));
        selectionLayout.setAlignment(Pos.CENTER);
        selectionLayout.setStyle("-fx-background-color:
linear-gradient(to bottom, #e0c3fc, #8ec5fc);");
        // Pastel Purple Gradient
        // Customize layout background color

        // create addition button
        Button addButton = new Button("Addition");
        setButtonStyle(addButton, "#2E7D32", 35);
        // Customize button style

        /*
```

```
         * when click addition button then called
"openCustomBaseWindow" function for
         * this window
         *
         * Send Parameters:
         * primaryStage : Manage the main window cause here
use scene transitions
         * Add : Send as string so that check which button
is clicked
         */
        addButton.setOnAction(e ->
openCustomBaseWindow(primaryStage, "Add"));

        // create subtraction button
        Button subtractButton = new Button("Subtraction");
        setButtonStyle(subtractButton, "#0D47A1", 35);
        // Customize button style

        /*
         * when click subtract button then called
"openCustomBaseWindow" function for
         * this window
         * Send Parameters:
         * primaryStage : Manage the main window cause here
use scene transitions
         * Subtract : Send as string so that check which
button is clicked
         */
        subtractButton.setOnAction(e ->
openCustomBaseWindow(primaryStage, "Subtract"));

        // create convert button
        Button convertButton = new Button("Convert");
        setButtonStyle(convertButton, "#F57C00", 35);
        // Customize button style

        /*
         * when click convert button then called
"openCustomBaseWindow" function for
```

```
         * this window
         * Send Parameters:
         * primaryStage : Manage the main window cause here
use scene transitions
         * Convert : Send as string so that check which
button is clicked
         */
        convertButton.setOnAction(e ->
openCustomBaseWindow(primaryStage, "Convert"));

        // create back button
        Button backButton = new Button("Back");
        setButtonStyle(backButton, "#f44336", 35);
        // Customize button style

        /*
         * when click back button then go the first window(i
am stay in second window)
         */
        backButton.setOnAction(e -> start(primaryStage));

        /*
         * here modify the hierarchical method so that which
is the root node find
         * addButton is the root and other button is
children
         */
        selectionLayout.getChildren().addAll(addButton,
subtractButton, convertButton, backButton);

        // set this window width X height
        Scene selectionScene = new Scene(selectionLayout,
400, 380);
        primaryStage.setScene(selectionScene);
        // This replaces the first window with the second
window (the custom base
        // selection scene).
    }
```

```
    /*********************************
     *
     * Else If press "Use Default Base"
     * Open Second Display
     *
     *********************************/

    private void openDefaultBaseSelection(Stage
primaryStage) {
        VBox selectionLayout = new VBox(15);
        // distance between every label, button etc. at 15
pixels

        selectionLayout.setPadding(new Insets(20));
        selectionLayout.setAlignment(Pos.CENTER);
        selectionLayout.setStyle("-fx-background-color:
linear-gradient(to bottom, #fafcc2, #fefbd8);");
        // Pastel Purple Gradient
        // Customize layout background color

        // create button for addition
        Button addButton = new Button("Addition");
        setButtonStyle(addButton, "#2E7D32", 35);
        // Customize button style

        /*
         * when click addition button then called
"openDefaultBaseWindow" function for
         * this window
         *
         * Send Parameters:
         * primaryStage : Manage the main window cause here
use scene transitions
         * Add : Send as string so that check which button
is clicked
         */
        addButton.setOnAction(e ->
openDefaultBaseWindow(primaryStage, "Add"));
```

```
        // create subtraction button
        Button subtractButton = new Button("Subtraction");
        setButtonStyle(subtractButton, "#0D47A1", 35);
        // Customize button style

        /*
         * when click subtract button then called
"openDefaultBaseWindow" function for
         * this window
         * Send Parameters:
         * primaryStage : Manage the main window cause here
use scene transitions
         * Subtract : Send as string so that check which
button is clicked
         */
        subtractButton.setOnAction(e ->
openDefaultBaseWindow(primaryStage, "Subtract"));

        // create convert button
        Button convertButton = new Button("Convert");
        setButtonStyle(convertButton, "#F57C00", 35);
        // Customize button style

        /*
         * when click convert button then called
"openDefaultBaseWindow" function for
         * this window
         * Send Parameters:
         * primaryStage : Manage the main window cause here
use scene transitions
         * Convert : Send as string so that check which
button is clicked
         */
        convertButton.setOnAction(e ->
openDefaultBaseWindow(primaryStage, "Convert"));

        // create back button
        Button backButton = new Button("Back");
        setButtonStyle(backButton, "#f44336", 35);
```

```
        // Customize button style

        /*
         * when click back button then go the first window(i
am stay in second window)
         */
        backButton.setOnAction(e -> start(primaryStage));

        /*
         * here modify the hierarchical method so that which
is the root node find
         * addButton is the root and other button is
children
         */
        selectionLayout.getChildren().addAll(addButton,
subtractButton, convertButton, backButton);

        // set this window width X height
        Scene selectionScene = new Scene(selectionLayout,
400, 380);
        primaryStage.setScene(selectionScene);
        // This replaces the first window with the second
window (the default base
        // selection scene).
    }

    /**************************************
     *
     * Else press "Universal Base Converter"
     * Open Second Display
     *
     **************************************/
    private void openUniversalBaseConverter(Stage
primaryStage) {
        VBox layout = new VBox(15);
        layout.setPadding(new Insets(20));
        layout.setAlignment(Pos.CENTER);
        layout.setStyle("-fx-background-color: linear-
gradient(to bottom, #b7eaff, #94dfff);");
```

```
        Label fromBaseLabel = new Label("From Base:");
        fromBaseLabel.setStyle("-fx-font-size: 17px; " + // Set font size
                               "-fx-font-family: 'Arial Rounded MT Bold'; " + // Set font to Arial Rounded MT Bold
                               "-fx-text-fill: blue;"); // Set text color to blue

        TextField fromBaseInput = new TextField();
        fromBaseInput.setPromptText("Enter original base");
        fromBaseInput.setStyle("-fx-font-size: 17px; " + // Set font size
                               "-fx-font-family: 'Arial Rounded MT Bold'; " + // Set font to Arial Rounded MT Bold
                               "-fx-text-fill: blue;"); // Set text color to blue


        Label numberLabel = new Label("Number to Convert:");
        numberLabel.setStyle("-fx-font-size: 17px; " + // Set font size
                               "-fx-font-family: 'Arial Rounded MT Bold'; " + // Set font to Arial Rounded MT Bold
                               "-fx-text-fill: #006400;"); // Set text color to dark green

        TextField numberInput = new TextField();
        numberInput.setPromptText("Enter number");
        numberInput.setStyle("-fx-font-size: 17px; " + // Set font size
                               "-fx-font-family: 'Arial Rounded MT Bold'; " + // Set font to Arial Rounded MT Bold
                               "-fx-text-fill: #006400;"); // Set text color to dark green


        Label toBaseLabel = new Label("To Base:");
```

```
        toBaseLabel.setStyle("-fx-font-size: 17px; " + // Set font size
                             "-fx-font-family: 'Arial Rounded MT Bold'; " + // Set font to Arial Rounded MT Bold
                             "-fx-text-fill: #4B0082;"); // Set text color to deep indigo

        TextField toBaseInput = new TextField();
        toBaseInput.setPromptText("Enter target base");
        toBaseInput.setStyle("-fx-font-size: 17px; " + // Set font size
                             "-fx-font-family: 'Arial Rounded MT Bold'; " + // Set font to Arial Rounded MT Bold
                             "-fx-text-fill: #4B0082;"); // Set text color to deep indigo


        Button convertButton = new Button("Convert");
        setButtonStyle(convertButton, "#4CAF50", 20);
        Label resultLabel = new Label();

        convertButton.setOnAction(e -> {
            try {
                int fromBase = Integer.parseInt(fromBaseInput.getText());
                int toBase = Integer.parseInt(toBaseInput.getText());
                String number = numberInput.getText();
                String result = convertNumberBetweenBases(number, fromBase, toBase);
                resultLabel.setText("Result: " + result);
                // Set the result label color to blue to match the "From Base" label
                resultLabel.setStyle("-fx-font-size: 22px; -fx-text-fill: blue; -fx-font-family: 'Arial Rounded MT Bold';"); // Set to blue
            } catch (NumberFormatException ex) {
                resultLabel.setText("Invalid base or number format.");
```

```
                resultLabel.setStyle("-fx-font-size: 17px; -
fx-text-fill: red; -fx-font-family: 'Arial Rounded MT
Bold';");
            }
        });


        Button backButton = new Button("Back");
        setButtonStyle(backButton, "#f44336", 20);
        backButton.setOnAction(e -> start(primaryStage));

        layout.getChildren().addAll(fromBaseLabel,
fromBaseInput, numberLabel, numberInput, toBaseLabel,
toBaseInput, convertButton, resultLabel, backButton);

        Scene converterScene = new Scene(layout, 400, 430);
        primaryStage.setScene(converterScene);
    }


    private String convertNumberBetweenBases(String number,
int fromBase, int toBase) {
        String[] parts = number.split("\\.");
        int integerPart = Integer.parseInt(parts[0],
fromBase);
        StringBuilder result = new
StringBuilder(Integer.toString(integerPart,
toBase).toUpperCase());

        if (parts.length > 1) {
            double fractionalPart = 0;
            for (int i = 0; i < parts[1].length(); i++) {
                int digitValue =
Character.digit(parts[1].charAt(i), fromBase);
                fractionalPart += digitValue /
Math.pow(fromBase, i + 1);
            }

            double fractionalResult = 0;
```

```
            StringBuilder fractionalStr = new
StringBuilder(".");
            while (fractionalPart != 0 &&
fractionalStr.length() < 10) {
                fractionalPart *= toBase;
                integerPart = (int) fractionalPart;

fractionalStr.append(Integer.toString(integerPart,
toBase).toUpperCase());
                fractionalPart -= integerPart;
            }

            result.append(fractionalStr);
        }

        return result.toString();
    }

    /*********************************************
     *
     * This is also Second Display
     * When press button then Third Display Open
     * Third Display For Custom Base
     *
     *********************************************/

    private void openCustomBaseWindow(Stage primaryStage,
String operation) {
        VBox customLayout = new VBox(15);
        customLayout.setPadding(new Insets(20));
        customLayout.setAlignment(Pos.CENTER);
        customLayout.setStyle("-fx-background-color:
#f5f5f5;");
        // Customize layout background color

        // create a label for show "Enter Your Base:"
        Label baseLabel = new Label("Enter Your Base:");
        baseLabel.setStyle("-fx-font-size: 17px; -fx-text-
fill: #0D47A1; -fx-font-family: 'Arial Rounded MT Bold';");
```

```
        // Change font size, color, and font

        // use textfield to input first base number
        TextField baseInput = new TextField();
        baseInput.setPromptText("Enter base (e.g., 2 for
Binary, 8 for Octal)");
        baseInput.setStyle("-fx-font-size: 17px; -fx-text-
fill: #000000; -fx-font-family: 'Arial Rounded MT Bold';");
        // Font size, text color, and font for input

        // create a label for show "Enter First Base
Number:"
        Label numberLabel1 = new Label("Enter First Base
Number:");
        numberLabel1.setStyle("-fx-font-size: 17px; -fx-
text-fill: #0D47A1; -fx-font-family: 'Arial Rounded MT
Bold';");
        // Change font size, color, and font

        // use textfield to input first base number
        TextField numberInput1 = new TextField();
        numberInput1.setPromptText("Enter first base
number");
        numberInput1.setStyle("-fx-font-size: 17px; -fx-
text-fill: #000000; -fx-font-family: 'Arial Rounded MT
Bold';");
        // Font size, text color, and font for input

        // create a label for show "Enter Second Base
Number:"
        Label numberLabel2 = new Label("Enter Second Base
Number:");
        numberLabel2.setStyle("-fx-font-size: 17px; -fx-
text-fill: #0D47A1; -fx-font-family: 'Arial Rounded MT
Bold';");
        // Change font size, color, and font

        // use textfield to input second base number
        TextField numberInput2 = new TextField();
```

```
        numberInput2.setPromptText("Enter second base
number");
        numberInput2.setStyle("-fx-font-size: 17px; -fx-
text-fill: #000000; -fx-font-family: 'Arial Rounded MT
Bold';");
        // Font size, text color, and font for input

        /*
         * For Dropbox Menu
         * here store some string so that user can easily
select output format
         */
        ComboBox<String> targetBaseBox = new ComboBox<>();
        targetBaseBox.getItems().addAll("Decimal", "Binary",
"Octal", "Hexadecimal", "Custom");
        targetBaseBox.setPromptText("Select target base");

        // Change font, color, and size
        targetBaseBox.setStyle(
                "-fx-font-size: 17px; " + // Set font size
                        "-fx-font-family: 'Arial Rounded MT
Bold'; " + // Set font to Arial Rounded MT Bold
                        "-fx-text-fill: #0D47A1;" // Set
text color to deep blue
        );

        /*
         * Create a Result name label for showing output
         */
        Label resultLabel = new Label();

        /*
         * operation button means:
         * Add
         * Subtract
         * Convert
         */
        Button actionButton = new Button(operation);
        setButtonStyle(actionButton, "#4CAF50", 20);
```

```
        // Customize button style

        // create back button for custom base 3rd window
        Button backButton = new Button("Back");
        setButtonStyle(backButton, "#f44336", 20);
        // Customize button style

        /*
         * When "Convert" is selected in the second window
and the Convert button is
         * clicked:
         * The third window displays these elements:
         *
         * baseLabel = Label prompting "Enter Your Base:"
         * baseInput = TextField for inputting the base
(Integer/Double, initially a
         * String to be parsed later)
         * numberLabel1 = Label prompting "Enter First Base
Number:"
         * numberInput1 = TextField for the first base
number (Integer/Double, initially
         * a String to be parsed later)
         * targetBaseBox = Dropdown box to select the target
base (e.g., Decimal,
         * Binary, Octal, Hexadecimal, or Custom)
         * actionButton = Button to trigger the Convert
operation
         * resultLabel = Label to display the conversion
result
         *
         */
        if (operation.equals("Convert")) {
            customLayout.getChildren().addAll(baseLabel,
baseInput, numberLabel1, numberInput1, targetBaseBox,
                    actionButton, resultLabel);

            /*
             * Now called "convertNumber" function and send
some arguments in third window
```

```
             * convert button
             *
             * baseInput = text field for input base:
Integer/Double [Note: TextField input
             * as String then convert!]
             *
             * numberInput1 = Enter First Base Number:
Integer/Double [Note: TextField input
             * as String then convert!]
             *
             * targetBaseBox = Select target base (dropdown
box)
             *
             * resultLabel = Result : Output result as
Double format always
             *
             */
            actionButton.setOnAction(e ->
convertNumber(baseInput, numberInput1, targetBaseBox,
resultLabel));
        }

        else {
            /*
             * Else click "Add Button" or "Subtract Button"
in Custom Base 2nd Window
             * Section
             * Note: Add & Subtract button's Argument is
same just operation is different
             *
             * baseLabel = Enter Your Base
             * baseInput = text field for input base:
Integer/Double [Note: TextField input
             * as String then convert!]
             * numberLabel1 = Enter First Base Number: Label
             * numberInput1 = Enter First Base Number:
Integer/Double [Note: TextField input
             * as String then convert!]
```

```
             * numberLabel2 = Enter Second Base Number:
Label
             * numberInput2 = Enter Second Base Number:
Integer/Double [Note: TextField
             * input as String then convert!]
             * targetBaseBox = Select target base (dropdown
box)
             * actionButton = Convert button
             * resultLabel = Result: Output result as Double
format always
             */

            customLayout.getChildren().addAll(baseLabel,
baseInput, numberLabel1, numberInput1, numberLabel2,
                    numberInput2, targetBaseBox,
actionButton, resultLabel);

            /*
             * Now called "performOperation" function and
send some arguments in third
             * window add/subtract button
             *
             * baseInput = text field for input base:
Integer/Double [Note: TextField input
             * as String then convert!]
             *
             * numberInput1 = Enter First Base Number:
Integer/Double [Note: TextField input
             * as String then convert!]
             *
             * numberInput2 = Enter Second Base Number:
Integer/Double [Note: TextField
             * input as String then convert!]
             *
             * targetBaseBox = Select target base (dropdown
box)
             *
             * resultLabel = Result : Output result as
Double format always
```

```
             *
             */
            actionButton.setOnAction(e ->
performOperation(baseInput, numberInput1, numberInput2,
targetBaseBox,
                    resultLabel, operation));
        }

        /*
         * when click back button then go the first window(i
am stay in second window
         * also until press any button)
         */
        backButton.setOnAction(e ->
openCustomBaseSelection(primaryStage));
        customLayout.getChildren().add(backButton);
        // is needed to add the backButton to the visual
layout of the third window

        Scene customScene = new Scene(customLayout, 400,
460);
        primaryStage.setScene(customScene);
    }

    /**********************************************
     *
     * This is also Second Display
     * When press button then Third Display Open
     * Third Display For Custom Base
     *
     **********************************************/

    private void openDefaultBaseWindow(Stage primaryStage,
String operation) {
        VBox defaultLayout = new VBox(15);
        defaultLayout.setPadding(new Insets(20));
        defaultLayout.setAlignment(Pos.CENTER);
        defaultLayout.setStyle("-fx-background-color:
#e0ffff;");
```

```
        // Customize layout background color

        ComboBox<String> fromBox = new ComboBox<>();
        fromBox.getItems().addAll("Binary", "Decimal",
"Octal", "Hexadecimal");
        fromBox.setPromptText("From");

        // Change font, color, fill, and size
        fromBox.setStyle(
                "-fx-font-size: 17px; " + // Set font size
                        "-fx-font-family: 'Arial Rounded MT
Bold'; " + // Set font to Arial Rounded MT Bold
                        "-fx-text-fill: #0D47A1;" // Set
text color to deep blue
        );

        ComboBox<String> toBox = new ComboBox<>();
        toBox.getItems().addAll("Binary", "Decimal",
"Octal", "Hexadecimal");
        toBox.setPromptText("To");
        // Change font, color, fill, and size
        toBox.setStyle(
                "-fx-font-size: 17px; " + // Set font size
                        "-fx-font-family: 'Arial Rounded MT
Bold'; " + // Set font to Arial Rounded MT Bold
                        "-fx-text-fill: #0D47A1;" // Set
text color to deep blue
        );

        // first number label and text field
        TextField numberInput1 = new TextField();
        numberInput1.setPromptText("Enter first number");
        // Apply custom font, color, and size
        numberInput1.setStyle("-fx-font-size: 17px; " +
// Set font size
                "-fx-font-family: 'Arial Rounded MT Bold'; "
+ // Set font to Arial Rounded MT Bold
                "-fx-text-fill: blue;"); // Set text color
to blue
```

```
        // second number label and text field
        TextField numberInput2 = new TextField();
        numberInput2.setPromptText("Enter second number (for
addition/subtraction)");
        // Apply custom font, color, and size
        numberInput2.setStyle("-fx-font-size: 17px; " + //
Set font size
                "-fx-font-family: 'Arial Rounded MT Bold'; "
+ // Set font to Arial Rounded MT Bold
                "-fx-text-fill: #006400;"); // Set text
color to dark green

        // create a label for showing result
        Label resultLabel = new Label();

        /*
         * operation button means:
         * Add
         * Subtract
         * Convert
         */
        Button actionButton = new Button(operation);
        setButtonStyle(actionButton, "#4CAF50", 25); //
Customize button style

        // back button
        Button backButton = new Button("Back");
        setButtonStyle(backButton, "#f44336", 20); //
Customize button style

        /*
         * When "Convert" is selected in the second window
and the Convert button is
         * clicked:
         * The third window displays these elements:
         *
         * fromBox = Dropdown box to select the input base
(e.g., Binary, Decimal,
```

```
         * Octal, Hexadecimal)
         * toBox = Dropdown box to select the target base
(e.g., Binary, Decimal, Octal,
         * Hexadecimal)
         * numberInput1 = TextField for entering the number
to convert (Integer/Double,
         * initially a String to be parsed later)
         * actionButton = Button to trigger the Convert
operation
         * resultLabel = Label to display the conversion
result in the selected target
         * base
         *
         */

        if (operation.equals("Convert")) {
            defaultLayout.getChildren().addAll(fromBox,
toBox, numberInput1, actionButton, resultLabel);

            actionButton.setOnAction(e ->
convertNumber(fromBox, toBox, numberInput1, resultLabel));
            /*
             * Now calls the "convertNumber" function and
sends arguments in the third
             * window Convert button:
             *
             * fromBox = Dropdown box to select the input
base (e.g., Binary, Decimal,
             * Octal, Hexadecimal)
             *
             * toBox = Dropdown box to select the target
base (e.g., Binary, Decimal, Octal,
             * Hexadecimal)
             *
             * numberInput1 = TextField for input number:
Integer/Double [Note: TextField
             * input as String then convert!]
             *
```

```
             * resultLabel = Label to display the conversion
result: Output always in Double
             * format
             *
             */

        } else {
            /*
             * When "Add Button" or "Subtract Button" is
clicked in Default Base 2nd Window
             * Section:
             * Note: Add & Subtract button's arguments are
the same; only the operation
             * differs.
             *
             * fromBox = Dropdown box to select the input
number's base (e.g., Binary,
             * Decimal, Octal, Hexadecimal)
             * toBox = Dropdown box to select the target
base for output (e.g., Binary,
             * Decimal, Octal, Hexadecimal)
             * numberInput1 = TextField for entering the
first number (Integer/Double)
             * [Note: TextField input as String then parsed]
             * numberInput2 = TextField for entering the
second number (Integer/Double)
             * [Note: TextField input as String then parsed]
             * actionButton = Button to trigger the
operation (Add or Subtract)
             * resultLabel = Label to display the result in
the target base as Double format
             */

            defaultLayout.getChildren().addAll(fromBox,
toBox, numberInput1, numberInput2, actionButton,
resultLabel);

            actionButton.setOnAction(
```

```
                    e -> performOperation(fromBox, toBox,
numberInput1, numberInput2, resultLabel, operation));
            /*
             * Now calls the "performOperation" function
when the Add/Subtract button is
             * clicked in the third window.
             * The following arguments are passed:
             *
             * fromBox = Dropdown to select the base of the
input numbers (e.g., Binary,
             * Decimal, Octal, Hexadecimal).
             *
             * toBox = Dropdown to select the target base
for the result (e.g., Binary,
             * Decimal, Octal, Hexadecimal).
             *
             * numberInput1 = TextField for the first number
in the selected base
             * (Integer/Double, parsed from String).
             *
             * numberInput2 = TextField for the second
number in the selected base
             * (Integer/Double, parsed from String).
             *
             * resultLabel = Label to display the result of
the operation (formatted as
             * Double).
             *
             * operation = The operation to perform ("Add"
or "Subtract").
             */

        }

        /*
         * when click back button then go the first window(i
am stay in second window
         * also until press any button)
         */
```

```
        backButton.setOnAction(e ->
openDefaultBaseSelection(primaryStage));
        defaultLayout.getChildren().add(backButton);
        // is needed to add the backButton to the visual
layout of the third window

        // third window width x height
        Scene defaultScene = new Scene(defaultLayout, 400,
400);
        primaryStage.setScene(defaultScene);
        // show the window
    }

    /**********************************************************
     *
     * Styling method to easily set button color and font
     * Button color font and fill change
     *
     **********************************************************/
    private void setButtonStyle(Button button, String color,
int fontSize) {
        button.setStyle("-fx-background-color: " + color +
"; -fx-text-fill: white;");
        button.setFont(Font.font("Arial", fontSize));
    }

    /**********************************
     *
     * 1. performOperation Method:
     *
     **********************************/

    private void performOperation(TextField baseInput,
TextField numberInput1, TextField numberInput2,
            ComboBox<String> targetBaseBox, Label
resultLabel, String operation) {
        try {

            /*
```

```
            * Convert the input base (entered as a string)
to an integer.
            * parseInt is a built-in function that converts
a string to an integer.
            */
           int base =
Integer.parseInt(baseInput.getText());

           /*
            * Convert the first number (entered as a
string) to a double.
            * parseFractional is a custom function that
converts a string to a double,
            * considering both the integer and fractional
parts of the number.
            */
           double num1 =
parseFractional(numberInput1.getText(), base);

           /*
            * Convert the second number (entered as a
string) to a double.
            * parseFractional is a custom function that
converts a string to a double,
            * considering both the integer and fractional
parts of the number.
            */
           double num2 =
parseFractional(numberInput2.getText(), base);

           /*
            * Use a ternary operator to perform the
operation (Add or Subtract).
            * The result will be displayed in double
format.
            */
           double result = operation.equals("Add") ? num1 +
num2 : num1 - num2;
```

```
            /*
             * If "Custom" is selected in the target base
dropdown, use the entered base.
             * Otherwise, use the getTargetBase function to
get the selected target base
             * (e.g., Decimal, Binary, Octal, Hexadecimal).
             */
            int targetBase =
targetBaseBox.getValue().equals("Custom") ? base :
getTargetBase(targetBaseBox.getValue());

            // Set the result text with a consistent style
showing the operation and the
            // conversion result
            resultLabel.setText("Result (" + operation + "):
" + convertFractional(result, targetBase));

            // Apply consistent styling to the result label
(blue text, Arial Rounded MT
            // Bold font, font size 17)
            resultLabel.setStyle("-fx-font-size: 17px; -fx-
text-fill: blue; -fx-font-family: 'Arial Rounded MT
Bold';");

        } catch (Exception e) {
            // If an error occurs (invalid input or base),
display an error message with red
            // text
            resultLabel.setText("Invalid input or base.");
            resultLabel.setStyle("-fx-font-size: 17px; -fx-
text-fill: red; -fx-font-family: 'Arial Rounded MT Bold';");
        }
    }

    /**********************************************
     *
     * 2. performOperation with ComboBox Method:
     *
     **********************************************/
```

```java
    private void performOperation(ComboBox<String> fromBox,
ComboBox<String> toBox, TextField numberInput1,
            TextField numberInput2, Label resultLabel,
String operation) {
        try {

            /*
             * Retrieve the selected input base from the
dropdown (fromBox).
             * Convert the selected base name (e.g.,
"Binary", "Decimal", "Octal", or
             * "Hexadecimal") into its corresponding integer
value.
             * If "Custom" is selected, ensure the input
base is entered and validated
             * beforehand.
             */
            int fromBase =
getTargetBase(fromBox.getValue());

            /*
             * Parse the first number (entered as a String
in numberInput1) into a double.
             * This function (parseFractional) supports both
integer and fractional parts of
             * the number.
             * The conversion is based on the input base
(fromBase).
             */
            double num1 =
parseFractional(numberInput1.getText(), fromBase);

            /*
             * Parse the second number (entered as a String
in numberInput2) into a double.
             * Like num1, this also supports fractional
values.
             * The conversion uses the same input base
(fromBase).
```

```
             */
            double num2 =
parseFractional(numberInput2.getText(), fromBase);

            /*
             * Perform the specified operation (either "Add"
or "Subtract") on the parsed
             * numbers.
             * If the operation is "Add", the result will be
the sum of num1 and num2.
             * If the operation is "Subtract", the result
will be the difference (num1 -
             * num2).
             * The ternary operator is used here for concise
conditional logic.
             */
            double result = operation.equals("Add") ? num1 +
num2 : num1 - num2;

            /*
             * Retrieve the target base for the result
conversion from the dropdown (toBox).
             * Convert the selected target base name (e.g.,
"Binary", "Decimal", "Octal", or
             * "Hexadecimal")
             * into its corresponding integer value using
the getTargetBase method.
             * If the target base is invalid, an exception
will be thrown.
             */
            int targetBase =
getTargetBase(toBox.getValue());

            /*
             * Set the text of the resultLabel to display
the result of the operation.
             * The text includes:
             * - The type of operation performed (e.g.,
"Add" or "Subtract").
```

```
             * - The converted result in the target base,
formatted as a string.
             * - The result is obtained by converting the
numerical value (result) into the
             * selected target base using the
convertFractional method.
             */
            resultLabel.setText("Result (" + operation + "):
" + convertFractional(result, targetBase));

            /*
             * Apply consistent styling to the resultLabel
for better readability and UI
             * uniformity.
             * - Font size: 17px
             * - Text color: Blue (#0000FF) to indicate
success or result output
             * - Font family: 'Arial Rounded MT Bold' for a
clean, professional look
             */
            resultLabel.setStyle("-fx-font-size: 17px; -fx-
text-fill: blue; -fx-font-family: 'Arial Rounded MT
Bold';");

        } catch (Exception e) {
            /*
             * Set the text of the resultLabel to display an
error message.
             * The message "Invalid input or base." is shown
when the user provides
             * incorrect or unsupported input,
             * such as:
             * - Non-numeric or out-of-range base values.
             * - Invalid numbers for the selected base.
             * This ensures the user is informed about the
issue and can correct their
             * input.
             */
            resultLabel.setText("Invalid input or base.");
```

```
            /*
             * Apply consistent styling to the resultLabel
for error messages.
             * - Font size: 17px for readability.
             * - Text color: Red (#FF0000) to clearly
indicate an error.
             * - Font family: 'Arial Rounded MT Bold' for a
clean and professional
             * appearance.
             * This styling differentiates error messages
from successful operation results.
             */
            resultLabel.setStyle("-fx-font-size: 17px; -fx-
text-fill: red; -fx-font-family: 'Arial Rounded MT Bold';");

        }
    }

    /**********************************************
     *
     * Convert text field string to Double number
     *
     **********************************************/

    private double parseFractional(String number, int base)
{
        /*
         * Split the input number (as a String) into its
integer and fractional parts.
         * The input is divided at the decimal point (".")
using a regular expression
         * ("\\.") as the delimiter.
         * - parts[0]: The integer part of the number
(before the decimal point).
         * - parts[1]: The fractional part of the number
(after the decimal point), if
         * present.
```

```
         * If no decimal point exists in the input, only the
integer part will be
         * present.
         */
        String[] parts = number.split("\\.");

        /*
         * Parse the integer part of the number (parts[0])
into an integer value.
         * This uses the specified base to correctly
interpret the number.
         * For example:
         * - Base 2 interprets "10" as binary (2 in
decimal).
         * - Base 16 interprets "10" as hexadecimal (16 in
decimal).
         * This conversion ensures that the integer part is
understood in the given
         * base.
         */
        int integerPart = Integer.parseInt(parts[0], base);

        /*
         * Initialize the fractional part of the number as
0.
         * This will be calculated only if a fractional part
(parts[1]) exists in the
         * input.
         * If there is no fractional part in the input, the
value remains 0.
         */
        double fractionalPart = 0;

        /*
         * Check if the input number contains a fractional
part.
         * This is determined by verifying if the length of
the array (parts) is greater
         * than 1.
```

```
         * - parts[1] represents the fractional part of the
input (if it exists).
         * If the input does not contain a decimal point,
this block is skipped.
         */
        if (parts.length > 1) {
            /*
             * Iterate over each character in the fractional
part (parts[1]).
             * This loop processes the digits of the
fractional part, one by one.
             * - i: The index of the current digit in the
fractional part.
             */
            for (int i = 0; i < parts[1].length(); i++) {
                /*
                 * Convert the current digit (character) of
the fractional part into its
                 * numerical value.
                 * This uses the base to correctly interpret
the digit.
                 * For example:
                 * - In base 16, 'A' is interpreted as 10.
                 * - In base 8, '7' is interpreted as 7.
                 */
                int digitValue =
Character.digit(parts[1].charAt(i), base);

                /*
                 * Add the contribution of the current digit
to the fractional part.
                 * - The value of the digit is divided by
the base raised to the power of its
                 * position (i + 1).
                 * - For example, in base 10:
                 * - The first fractional digit contributes
digitValue / 10^1.
                 * - The second fractional digit contributes
digitValue / 10^2.
```

```
                * This formula ensures that the fractional
digits are properly weighted.
                */
               fractionalPart += digitValue /
Math.pow(base, i + 1);
            }
        }

        /*
         * Return the final result as the sum of the integer
part and the fractional
         * part.
         * The integer part contributes the whole number
portion of the value.
         * The fractional part (if present) adds the decimal
component to the result.
         */
        return integerPart + fractionalPart;

    }

    /**********************************
     *
     * Fractional number calculation
     *
     **********************************/

    private String convertFractional(double number, int
base) {
        /*
         * Extract the integer part of the input number.
         * The number is explicitly cast to an integer,
truncating any fractional
         * component.
         * For example:
         * - If the input number is 12.34, integerPart will
be 12.
         */
        int integerPart = (int) number;
```

```
        /*
         * Calculate the fractional part of the input
number.
         * Subtract the integer part from the original
number to isolate the fractional
         * component.
         * For example:
         * - If the input number is 12.34, fractionalPart
will be 0.34.
         */
        double fractionalPart = number - integerPart;

        /*
         * Convert the integer part of the number to a
string in the specified base.
         * The integer part is formatted according to the
base (e.g., binary, octal,
         * hexadecimal).
         * The result is converted to uppercase to ensure
consistency in output (e.g.,
         * "A" instead of "a" for hexadecimal).
         */
        String integerResult = Integer.toString(integerPart,
base).toUpperCase();

        /*
         * Initialize a StringBuilder to construct the
fractional part of the result.
         * Start with a period (".") to separate the
fractional part from the integer
         * part.
         * This will hold the fractional component formatted
in the specified base.
         */
        StringBuilder fractionalResult = new
StringBuilder(".");

        /*
```

```
        * Loop to convert the fractional part of the number
into the specified base.
        * The loop iterates up to 10 times to limit the
precision to 10 digits in the
        * fractional part.
        * - Each iteration calculates one digit of the
fractional part in the target
        * base.
        */
       for (int i = 0; i < 10; i++) { // Limit to 10 digits
for precision
           /*
            * Multiply the fractional part by the base to
shift the next significant digit
            * to the left of the decimal point.
            * For example:
            * - In base 10, multiplying 0.25 by 10 shifts
the next significant digit to
            * 2.5.
            */
           fractionalPart *= base;

           /*
            * Extract the integer part of the shifted
fractional value.
            * This gives the next digit in the target base.
            * For example:
            * - If the fractionalPart is 2.5, digitValue
will be 2.
            */
           int digitValue = (int) fractionalPart;

           /*
            * Convert the extracted digit to its
corresponding character in the target
            * base.
            * - For base 16, digitValue 10 would become
'A'.
```

```
            * Append the digit character to the
fractionalResult StringBuilder.
            */

fractionalResult.append(Character.forDigit(digitValue,
base));

           /*
            * Subtract the extracted digit from the
fractional part to isolate the
            * remaining fraction.
            * For example:
            * - If fractionalPart is 2.5, subtracting 2
results in 0.5.
            */
           fractionalPart -= digitValue;

           /*
            * If the fractional part becomes 0, break out
of the loop early.
            * This ensures the loop doesn't run
unnecessarily when there are no remaining
            * fractional digits.
            */
           if (fractionalPart == 0)
               break;
       }

       /*
        * Combine the integer and fractional parts into the
final result string.
        * Ensure the fractional part is also converted to
uppercase for consistency.
        * - integerResult: The integer part converted to
the target base.
        * - fractionalResult: The fractional part converted
to the target base,
        * prefixed by a period.
```

```
         * Return the combined result string in uppercase
format.
         */
        return integerResult +
fractionalResult.toString().toUpperCase();

    }

    /*****************************************
     *
     * 3. convertNumber Method with TextField:
     *
     *****************************************/

    private void convertNumber(TextField baseInput,
TextField numberInput, ComboBox<String> targetBaseBox,
            Label resultLabel) {
        /*
         * Attempt to parse the input values and perform the
base conversion.
         * This block handles potential exceptions caused by
invalid input or
         * unsupported operations.
         */
        try {
            /*
             * Retrieve the base for the input number from
the baseInput text field.
             * The input is parsed as an integer to ensure
it is a valid numeric base.
             * If the base is invalid (e.g., non-numeric or
out of range), an exception will
             * be thrown.
             */
            int base =
Integer.parseInt(baseInput.getText());

            /*
```

```
            * Parse the input number (entered as a String
in numberInput) into a double
            * value.
            * This includes handling both the integer and
fractional parts of the number,
            * if present.
            * The parsing is performed based on the
retrieved input base.
            */
            double number =
parseFractional(numberInput.getText(), base);

            /*
            * Determine the target base for the conversion.
            * If the user selects "Custom" in the
targetBaseBox dropdown, use the same
            * input base.
            * Otherwise, retrieve the target base (e.g.,
Binary, Decimal, Octal,
            * Hexadecimal) using the getTargetBase method.
            */
            int targetBase =
targetBaseBox.getValue().equals("Custom") ? base :
getTargetBase(targetBaseBox.getValue());

            /*
            * Convert the parsed input number to the target
base and format it as a string.
            * The conversion uses the convertFractional
method to handle both integer and
            * fractional components.
            * The formatted result is displayed in the
resultLabel.
            */
            resultLabel.setText("Converted Value: " +
convertFractional(number, targetBase));

            /*
```

```
            * Apply consistent styling to the resultLabel
for displaying the converted
            * value.
            * - Font size: 17px for readability.
            * - Text color: Blue (#0000FF) to indicate
successful conversion.
            * - Font family: 'Arial Rounded MT Bold' for a
professional appearance.
            */
           resultLabel.setStyle("-fx-font-size: 17px; -fx-
text-fill: blue; -fx-font-family: 'Arial Rounded MT
Bold';");

        } /*
           * Catch any exceptions that occur during the
execution of the try block.
           * This handles scenarios such as:
           * - Non-numeric or invalid input in the baseInput
or numberInput fields.
           * - Unsupported or unrecognized base values.
           * - Logical errors in parsing or conversion.
           * The catch block ensures the application does
not crash and provides user
           * feedback.
           */
        catch (Exception e) {
            /*
             * Set an error message in the resultLabel to
inform the user of the issue.
             * The message "Invalid input or base."
indicates that the problem lies in the
             * provided base or number input, prompting the
user to correct it.
             */
            resultLabel.setText("Invalid input or base.");

            /*
             * Apply consistent styling to the resultLabel
for error messages.
```

```
            * - Font size: 17px for visibility.
            * - Text color: Red (#FF0000) to indicate an
error state.
            * - Font family: 'Arial Rounded MT Bold' for a
professional look.
            * This styling visually distinguishes error
messages from successful results.
            */
           resultLabel.setStyle("-fx-font-size: 17px; -fx-
text-fill: red; -fx-font-family: 'Arial Rounded MT Bold';");
        }

    }

    /*****************************************
     *
     * 4. convertNumber Method with ComboBox:
     *
     *****************************************/

    private void convertNumber(ComboBox<String> fromBox,
ComboBox<String> toBox, TextField numberInput,
           Label resultLabel) {
        /*
         * Attempt to perform base conversion based on user
inputs for the source and
         * target bases.
         * The try block ensures proper handling of valid
inputs and triggers an
         * exception
         * if invalid inputs or errors occur during parsing
or conversion.
         */
        try {
            /*
             * Retrieve the source base (fromBase) selected
by the user from the dropdown
             * (fromBox).
```

```
             * The getTargetBase method converts the
dropdown selection into its
             * corresponding integer base value.
             * For example:
             * - "Binary" becomes 2.
             * - "Decimal" becomes 10.
             * If the input base is invalid, an exception is
thrown.
             */
            int fromBase =
getTargetBase(fromBox.getValue());

            /*
             * Parse the input number from the numberInput
text field.
             * The parseFractional method converts the
string into a double value based on
             * the source base.
             * This includes both the integer and fractional
parts of the number, if
             * present.
             */
            double number =
parseFractional(numberInput.getText(), fromBase);

            /*
             * Retrieve the target base (targetBase)
selected by the user from the dropdown
             * (toBox).
             * The getTargetBase method converts the
dropdown selection into its
             * corresponding integer base value.
             * For example:
             * - "Hexadecimal" becomes 16.
             * - "Octal" becomes 8.
             * If the target base is invalid, an exception
is thrown.
             */
```

```
            int targetBase =
getTargetBase(toBox.getValue());

            /*
             * Convert the parsed input number (in the
source base) into the target base.
             * The convertFractional method handles the
conversion and formats the result
             * as a string, preserving both integer and
fractional parts.
             * The result is displayed in the resultLabel
for the user to view.
             */
            resultLabel.setText("Converted Value: " +
convertFractional(number, targetBase));

            /*
             * Apply consistent styling to the resultLabel
for displaying the converted
             * value.
             * - Font size: 17px for readability.
             * - Text color: Blue (#0000FF) to indicate
successful conversion.
             * - Font family: 'Arial Rounded MT Bold' for a
professional appearance.
             */
            resultLabel.setStyle("-fx-font-size: 17px; -fx-
text-fill: blue; -fx-font-family: 'Arial Rounded MT
Bold';");
        } catch (Exception e) {
            /*
             * Catch any exceptions that occur during the
execution of the try block.
             * These exceptions can result from:
             * - Invalid or unsupported base values.
             * - Non-numeric or incorrectly formatted input
numbers.
             * - Logical errors in parsing or conversion.
```

```
             * Display an error message in the resultLabel
to inform the user of the issue.
             */
            resultLabel.setText("Invalid input or base.");

            /*
             * Apply consistent styling to the resultLabel
for error messages.
             * - Font size: 17px for visibility.
             * - Text color: Red (#FF0000) to emphasize the
error state.
             * - Font family: 'Arial Rounded MT Bold' for a
professional look.
             * This styling differentiates error messages
from successful results.
             */
            resultLabel.setStyle("-fx-font-size: 17px; -fx-
text-fill: red; -fx-font-family: 'Arial Rounded MT Bold';");
        }

    }

    /************************
     *
     * Target Base
     *
     ************************/
    private int getTargetBase(String baseName) {
        return switch (baseName) {
            case "Binary" -> 2;
            case "Octal" -> 8;
            case "Decimal" -> 10;
            case "Hexadecimal" -> 16;
            default -> throw new
IllegalArgumentException("Invalid base");
        };
    }

    /*******************
```

```
     *
     * Main Function
     *
     ******************/
    public static void main(String[] args) {
        launch(args);
    }
}
```