APPENDICES

APPENDIX A – SOURCE CODE

```
import java.awt.*;
import java.awt.event.*;
public class CurrencyConverter extends Frame implements ActionListener {
    Label lblAmount, lblFrom, lblTo, lblResult;
    TextField txtAmount, txtResult;
    Choice choiceFrom, choiceTo;
    Button btnConvert, btnClear, btnExit;
    // Exchange rates (relative to USD)
    double[] rates = {
        1.0,    // USD - United States Dollar
        0.85,   // EUR - Euro
        109.5,  // JPY - Japanese Yen
        0.75,   // GBP - British Pound
        1.5,    // AUD - Australian Dollar
        1.34,   // CAD - Canadian Dollar
        0.91,   // CHF - Swiss Franc
        7.15,   // CNY - Chinese Yuan
        7.8,    // HKD - Hong Kong Dollar
        1.25,   // NZD - New Zealand Dollar
        0.75,   // SEK - Swedish Krona
        0.7,    // NOK - Norwegian Krone
        1.37,   // SGD - Singapore Dollar
        20.0,   // MXN - Mexican Peso
        12.0,   // ZAR - South African Rand
        82.0,   // INR - Indian Rupee
```

```
    11.5,   // BRL - Brazilian Real
    26.0,   // TRY - Turkish Lira
    100.0,  // RUB - Russian Ruble
    1320.0, // KRW - South Korean Won
    3.67    // AED - UAE Dirham
};
String[] currencies = {
    "USD", "EUR", "JPY", "GBP", "AUD", "CAD", "CHF",
    "CNY", "HKD", "NZD", "SEK", "NOK", "SGD",
    "MXN", "ZAR", "INR", "BRL", "TRY", "RUB", "KRW", "AED"
};
public CurrencyConverter() {
    setTitle("Currency Converter");
    setSize(600, 400);
    setLayout(null);
    setVisible(true);
    // Labels
    lblAmount = new Label("Amount:");
    lblAmount.setBounds(50, 50, 100, 30);
    add(lblAmount);
    lblFrom = new Label("From:");
    lblFrom.setBounds(50, 100, 100, 30);
    add(lblFrom);
    lblTo = new Label("To:");
    lblTo.setBounds(50, 150, 100, 30);
    add(lblTo);
    lblResult = new Label("Result:");
    lblResult.setBounds(50, 200, 100, 30);
    add(lblResult);
```

```
// Text fields
txtAmount = new TextField();
txtAmount.setBounds(150, 50, 250, 30);
add(txtAmount);
txtResult = new TextField();
txtResult.setBounds(150, 200, 250, 30);
txtResult.setEditable(false);
add(txtResult);
// Choice dropdowns
choiceFrom = new Choice();
choiceTo = new Choice();
for (String currency : currencies) {
    choiceFrom.add(currency);
    choiceTo.add(currency);
}
choiceFrom.setBounds(150, 100, 250, 30);
choiceTo.setBounds(150, 150, 250, 30);
add(choiceFrom);
add(choiceTo);
// Buttons
btnConvert = new Button("Convert");
btnConvert.setBounds(50, 300, 100, 30);
btnConvert.addActionListener(this);
add(btnConvert);
btnClear = new Button("Clear");
btnClear.setBounds(200, 300, 100, 30);
btnClear.addActionListener(this);
add(btnClear);
btnExit = new Button("Exit");
```

```
    btnExit.setBounds(350, 300, 100, 30);
    btnExit.addActionListener(this);
    add(btnExit);
    // Window closing
    addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            dispose();
        }
    });
}
@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == btnConvert) {
        try {
            double amount = Double.parseDouble(txtAmount.getText());
            int fromIndex = choiceFrom.getSelectedIndex();
            int toIndex = choiceTo.getSelectedIndex();
            // Conversion logic
            double result = amount * (rates[toIndex] / rates[fromIndex]);
            txtResult.setText(String.format("%.2f", result));
        } catch (NumberFormatException ex) {
            txtResult.setText("Invalid Input");
        }
    } else if (e.getSource() == btnClear) {
        txtAmount.setText("");
        txtResult.setText("");
        choiceFrom.select(0);
        choiceTo.select(0);
    } else if (e.getSource() == btnExit) {
```

```
            dispose();
        }
    }
    public static void main(String[] args) {
        new CurrencyConverter();
    }
}
```