VehicleRental SystemGUI

```
package vehiclerentalsystemwithgui;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.List;

// Abstract class representing a Vehicle
abstract class Vehicle {

    private final String vehicleId;
    private final String brand;
    private final String model;
    private final double basePricePerDay;
    private boolean isAvailable;

    public Vehicle(String vehicleId, String brand, String model, double basePricePerDay) {
        this.vehicleId = vehicleId;
        this.brand = brand;
        this.model = model;
        this.basePricePerDay = basePricePerDay;
        this.isAvailable = true;
    }

    public String getVehicleId() {
        return vehicleId;
    }

    public String getBrand() {
        return brand;
    }

    public String getModel() {
        return model;
    }

    public double getBasePricePerDay() {
        return basePricePerDay;
    }

    public boolean isAvailable() {
        return isAvailable;
    }

    public void rent() {
        isAvailable = false;
    }
    public void returnVehicle() {
        isAvailable = true;
    }

    public double calculatePrice(int rentalDays) {
        return basePricePerDay * rentalDays;
    }

    // Abstract method to be implemented by subclasses
    public abstract void vehicleType();
}

// Concrete class representing a Car
class Car extends Vehicle {

    public Car(String vehicleId, String brand, String model, double basePricePerDay) {
        super(vehicleId, brand, model, basePricePerDay);
    }

    @Override
    public void vehicleType() {
        System.out.println("Vehicle Type: Car");
    }
}

// Abstract class representing a User
abstract class User {

    private final String userId;
    private final String name;

    public User(String userId, String name) {
        this.userId = userId;
        this.name = name;
    }

    public String getUserId() {
        return userId;
    }

    public String getName() {
        return name;
    }
}

// Concrete class representing a Customer
class Customer extends User {

    public Customer(String userId, String name) {
        super(userId, name);
    }
}
```

```
// Class representing a Rental
class Rental {

   private final Vehicle vehicle;
   private final Customer customer;
   private final int days;

   public Rental(Vehicle vehicle, Customer customer, int days) {
      this.vehicle = vehicle;
      this.customer = customer;
      this.days = days;
   }

   public Vehicle getVehicle() {
      return vehicle;
   }

   public Customer getCustomer() {
      return customer;
   }

   public int getDays() {
      return days;
   }
}

// Custom exception for rental errors
class RentalException extends Exception {

   public RentalException(String message) {
      super(message);
   }
}

// Runnable class for simulating rentals in a separate thread
class RentalTask implements Runnable {

   private final CarRentalSystem rentalSystem;
   private final Vehicle vehicle;
   private final Customer customer;
   private final int days;

   public RentalTask(CarRentalSystem rentalSystem, Vehicle vehicle, Customer customer, int days) {
      this.rentalSystem = rentalSystem;
      this.vehicle = vehicle;
      this.customer = customer;
      this.days = days;
   }

   @Override
   public void run() {
      try {
         rentalSystem.rentVehicle(vehicle, customer, days);
```

```
        } catch (RentalException e) {
            System.out.println(e.getMessage());
        }
    }
}

// Class representing the Car Rental System
class CarRentalSystem {

    private final List<Vehicle> vehicles;
    private final List<Customer> customers;
    private final List<Rental> rentals;

    public CarRentalSystem() {
        vehicles = new ArrayList<>();
        customers = new ArrayList<>();
        rentals = new ArrayList<>();
    }

    public void addVehicle(Vehicle vehicle) {
        vehicles.add(vehicle);
    }

    public void addCustomer(Customer customer) {
        customers.add(customer);
    }

    public void rentVehicle(Vehicle vehicle, Customer customer) throws RentalException {
        rentVehicle(vehicle, customer, 1);
    }

    public void rentVehicle(Vehicle vehicle, Customer customer, int days) throws RentalException {
        if (vehicle.isAvailable()) {
            vehicle.rent();
            rentals.add(new Rental(vehicle, customer, days));
            System.out.println("Vehicle rented successfully.");
        } else {
            throw new RentalException("Vehicle is not available for rent.");
        }
    }

    public void returnVehicle(Vehicle vehicle) throws RentalException {
        if (!vehicle.isAvailable()) {
            vehicle.returnVehicle();
            Rental rentalToRemove = null;
            for (Rental rental : rentals) {
                if (rental.getVehicle() == vehicle) {
                    rentalToRemove = rental;
                    break;
                }
            }
            if (rentalToRemove != null) {
                rentals.remove(rentalToRemove);
            } else {
                throw new RentalException("Vehicle was not rented.");
            }
        } else {
            throw new RentalException("Vehicle is already available.");
        }
    }

    // Method to set up the GUI
    public void setupGUI() {
        JFrame frame = new JFrame("Vehicle Rental System");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 400);

        JPanel panel = new JPanel(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);
        gbc.fill = GridBagConstraints.HORIZONTAL;

        JLabel titleLabel = new JLabel("Vehicle Rental System");
        titleLabel.setFont(new Font("Arial", Font.BOLD, 24));
        titleLabel.setHorizontalAlignment(SwingConstants.CENTER);

        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.gridwidth = 3;
        panel.add(titleLabel, gbc);

        JButton rentButton = new JButton("Rent a Vehicle");
        JButton returnButton = new JButton("Return a Vehicle");
        JButton exitButton = new JButton("Exit");

        gbc.gridwidth = 1;
        gbc.gridy = 1;
        panel.add(rentButton, gbc);

        gbc.gridx = 1;
        panel.add(returnButton, gbc);

        gbc.gridx = 2;
        panel.add(exitButton, gbc);

        frame.add(panel, BorderLayout.CENTER);

        rentButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                rentVehicleGUI();
            }
        });

        returnButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                returnVehicleGUI();
            }
        });

        exitButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        });

        frame.setVisible(true);
    }

    private void rentVehicleGUI() {
        JFrame rentFrame = new JFrame("Rent a Vehicle");
        rentFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        rentFrame.setSize(500, 400);

        JPanel rentPanel = new JPanel(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);
        gbc.fill = GridBagConstraints.HORIZONTAL;

        JLabel nameLabel = new JLabel("Enter your name:");
        JTextField nameField = new JTextField();

        JLabel vehicleLabel = new JLabel("Select a Vehicle:");
        JComboBox<String> vehicleComboBox = new JComboBox<>();
        for (Vehicle vehicle : vehicles) {
            if (vehicle.isAvailable()) {
                vehicleComboBox.addItem(vehicle.getVehicleId() + " - " + vehicle.getBrand() + " " + vehicle.getModel());
            }
        }

        JLabel daysLabel = new JLabel("Enter number of days:");
        JTextField daysField = new JTextField();

        JButton rentConfirmButton = new JButton("Confirm");
        JButton rentCancelButton = new JButton("Cancel");

        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.gridwidth = 1;
        rentPanel.add(nameLabel, gbc);
        gbc.gridx = 1;
        gbc.gridwidth = 2;
        rentPanel.add(nameField, gbc);

        gbc.gridx = 0;
        gbc.gridy = 1;
        gbc.gridwidth = 1;
        rentPanel.add(vehicleLabel, gbc);
        gbc.gridx = 1;
        gbc.gridwidth = 2;
        rentPanel.add(vehicleComboBox, gbc);

        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.gridwidth = 1;
        rentPanel.add(daysLabel, gbc);
        gbc.gridx = 1;
        gbc.gridwidth = 2;
        rentPanel.add(daysField, gbc);

        gbc.gridx = 1;
        gbc.gridy = 3;
        gbc.gridwidth = 1;
        rentPanel.add(rentConfirmButton, gbc);
        gbc.gridx = 2;
        rentPanel.add(rentCancelButton, gbc);

        rentFrame.add(rentPanel, BorderLayout.CENTER);
        rentFrame.setVisible(true);

        rentConfirmButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String customerName = nameField.getText();
                String selectedVehicleInfo = (String) vehicleComboBox.getSelectedItem();
                if (selectedVehicleInfo != null && !             customerName.isEmpty() &&
!daysField.getText().isEmpty()) {
                    String vehicleId = selectedVehicleInfo.split(" - ")[0];
                    int rentalDays = Integer.parseInt(daysField.getText());

                    Customer newCustomer = new Customer("CUS" + (customers.size() + 1), customerName);
                    addCustomer(newCustomer);

                    Vehicle selectedVehicle = null;
                    for (Vehicle vehicle : vehicles) {
                        if (vehicle.getVehicleId().equals(vehicleId)) {
                            selectedVehicle = vehicle;
                            break;
                        }
                    }

                    if (selectedVehicle != null) {
                        double totalPrice = selectedVehicle.calculatePrice(rentalDays);
                        int response = JOptionPane.showConfirmDialog(null,
                                String.format(
                                        "Rental Information\n"
                                        + "Customer ID: %s\n"
                                        + "Customer Name: %s\n"
                                        + "Vehicle: %s %s\n"
                                        + "Rental Days: %d\n"
                                + "Total Price: $%.2f\n\nConfirm rental?",
                                newCustomer.getUserId(),
                                newCustomer.getName(),
                                selectedVehicle.getBrand(),
                                selectedVehicle.getModel(),
                                rentalDays,
                                totalPrice
                            ),
                            "Confirm Rental",
                            JOptionPane.YES_NO_OPTION
                        );

                        if (response == JOptionPane.YES_OPTION) {
                            Thread rentalThread = new Thread(new RentalTask(CarRentalSystem.this, selectedVehicle,
newCustomer, rentalDays));
                            rentalThread.start();
                            rentFrame.dispose();
                        }
                    }
                } else {
                    JOptionPane.showMessageDialog(rentFrame, "Please fill all fields correctly.", "Error",
JOptionPane.ERROR_MESSAGE);
                }
            }
        });

        rentCancelButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                rentFrame.dispose();
            }
        });
    }

    private void returnVehicleGUI() {
        JFrame returnFrame = new JFrame("Return a Vehicle");
        returnFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        returnFrame.setSize(400, 200);

        JPanel returnPanel = new JPanel(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);
        gbc.fill = GridBagConstraints.HORIZONTAL;

        JLabel vehicleLabel = new JLabel("Enter Vehicle ID:");
        JTextField vehicleField = new JTextField();

        JButton returnConfirmButton = new JButton("Confirm");
        JButton returnCancelButton = new JButton("Cancel");

        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.gridwidth = 1;
        returnPanel.add(vehicleLabel, gbc);
        gbc.gridx = 1;
        gbc.gridwidth = 2;
        returnPanel.add(vehicleField, gbc);

        gbc.gridx = 1;
        gbc.gridy = 1;
        gbc.gridwidth = 1;
        returnPanel.add(returnConfirmButton, gbc);
        gbc.gridx = 2;
        returnPanel.add(returnCancelButton, gbc);

        returnFrame.add(returnPanel, BorderLayout.CENTER);
        returnFrame.setVisible(true);

        returnConfirmButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String vehicleId = vehicleField.getText();
                Vehicle vehicleToReturn = null;

                for (Vehicle vehicle : vehicles) {
                    if (vehicle.getVehicleId().equals(vehicleId) && !vehicle.isAvailable()) {
                        vehicleToReturn = vehicle;
                        break;
                    }
                }

                if (vehicleToReturn != null) {
                    try {
                        returnVehicle(vehicleToReturn);
                        JOptionPane.showMessageDialog(returnFrame, "Vehicle returned successfully.", "Success",
JOptionPane.INFORMATION_MESSAGE);
                        returnFrame.dispose();
                    } catch (RentalException ex) {
                        JOptionPane.showMessageDialog(returnFrame, ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
                    }
                } else {
                    JOptionPane.showMessageDialog(returnFrame, "Invalid vehicle ID or vehicle is not rented.", "Error",
JOptionPane.ERROR_MESSAGE);
                }
            }
        });

        returnCancelButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                returnFrame.dispose();
            }
        });
    }
}
```

```
public class Main {

    public static void main(String[] args) {
        CarRentalSystem rentalSystem = new CarRentalSystem();

        Vehicle car1 = new Car("C001", "Toyota", "Camry", 60.0);
        Vehicle car2 = new Car("C002", "Honda", "Accord", 70.0);
        Vehicle car3 = new Car("C003", "Mahindra", "Thar", 150.0);
        Vehicle car4 = new Car("C004", "Nissan", "GTR", 90.0);
        Vehicle car5 = new Car("C005", "Hero", "Bicycle", 70.0);
        Vehicle car6 = new Car("C006", "Tata", "Truck", 150.0);
        Vehicle car7 = new Car("C007", "Tata", "Mini-Truck", 120.0);
        Vehicle car8 = new Car("C008", "Helicopter", "Chopper", 1000.0);
        Vehicle car9 = new Car("C009", "Ford", "Fusion", 80.0);
        Vehicle car10 = new Car("C010", "Chevrolet", "Camaro", 110.0);
        Vehicle car11 = new Car("C011", "BMW", "X5", 200.0);
        Vehicle car12 = new Car("C012", "Audi", "A4", 180.0);
        Vehicle car13 = new Car("C013", "Mercedes-Benz", "S-Class", 250.0);
        Vehicle car14 = new Car("C014", "Volkswagen", "Golf", 90.0);
        Vehicle car15 = new Car("C015", "Subaru", "Impreza", 100.0);
        Vehicle car16 = new Car("C016", "Lamborghini", "Aventador", 1500.0);
        Vehicle car17 = new Car("C017", "Ferrari", "488 GTB", 1600.0);
        Vehicle car18 = new Car("C018", "Porsche", "911", 180.0);
        Vehicle car19 = new Car("C019", "Maserati", "GranTurismo", 220.0);
        Vehicle car20 = new Car("C020", "Bugatti", "Chiron", 3000.0);

        rentalSystem.addVehicle(car1);
        rentalSystem.addVehicle(car2);
        rentalSystem.addVehicle(car3);
        rentalSystem.addVehicle(car4);
        rentalSystem.addVehicle(car5);
        rentalSystem.addVehicle(car6);
        rentalSystem.addVehicle(car7);
        rentalSystem.addVehicle(car8);
        rentalSystem.addVehicle(car9);
        rentalSystem.addVehicle(car10);
        rentalSystem.addVehicle(car11);
        rentalSystem.addVehicle(car12);
        rentalSystem.addVehicle(car13);
        rentalSystem.addVehicle(car14);
        rentalSystem.addVehicle(car15);
        rentalSystem.addVehicle(car16);
        rentalSystem.addVehicle(car17);
        rentalSystem.addVehicle(car18);
        rentalSystem.addVehicle(car19);
        rentalSystem.addVehicle(car20);

        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                rentalSystem.setupGUI();
            }
        });
    }
}
```

Data_COM_Conversion_GUI

```
package gui;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;

public class Data_Com extends javax.swing.JFrame {

    private String A;
    private String B;

    public Data_Com() {
        initComponents();
    }

    private void initComponents() {
        JPanel jPanel = new JPanel();
        JButton resetButton = new JButton();
        JButton calculateButton = new JButton();
        JTextField resultField = new JTextField();
        JLabel outputLabel = new JLabel();
        JTextField inputField2 = new JTextField();
        JLabel inputLabel2 = new JLabel();
        JTextField flagField = new JTextField();
        JLabel flagLabel = new JLabel();
        JComboBox<String> combobox = new JComboBox<>();
        JLabel titleLabel = new JLabel();
        JTextField inputField1 = new JTextField();
        JLabel inputLabel1 = new JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        resetButton.setFont(new java.awt.Font("Tahoma", 1, 18));
        resetButton.setText("RESET");
        resetButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                resetActionPerformed(evt, inputField1, inputField2, resultField, flagField);
            }
        });

        calculateButton.setFont(new java.awt.Font("Tahoma", 1, 18));
        calculateButton.setText("CALCULATE");
        calculateButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                calculateActionPerformed(evt, combobox, inputField1, inputField2, flagField, resultField);
            }
        });

        outputLabel.setFont(new java.awt.Font("Tahoma", 1, 14));
        outputLabel.setText("OUTPUT:");

        inputLabel2.setFont(new java.awt.Font("Tahoma", 1, 14));
        inputLabel2.setText("INPUT:");

        flagLabel.setFont(new java.awt.Font("Tahoma", 1, 14));
        flagLabel.setText("FLAG:");

        combobox.setFont(new java.awt.Font("Tahoma", 1, 18));
        combobox.setModel(new javax.swing.DefaultComboBoxModel<>(new String[]{
            "Select an Option", "Hamming Distance", "Parity Check", "Bit Stuffing", "Bit Destuffing", "Character
Stuffing", "Character Destuffing"
        }));

        titleLabel.setFont(new java.awt.Font("Tahoma", 1, 18));
        titleLabel.setText("BDCLCI");

        inputLabel1.setFont(new java.awt.Font("Tahoma", 1, 14));
        inputLabel1.setText("INPUT:");

        GroupLayout jPanelLayout = new GroupLayout(jPanel);
        jPanel.setLayout(jPanelLayout);
        jPanelLayout.setHorizontalGroup(
            jPanelLayout.createParallelGroup(GroupLayout.Alignment.LEADING)
                .addGroup(jPanelLayout.createSequentialGroup()
                    .addGap(30, 30, 30)
                    .addGroup(jPanelLayout.createParallelGroup(GroupLayout.Alignment.LEADING)
                        .addComponent(inputLabel1)
                        .addComponent(inputLabel2)
                        .addComponent(outputLabel))
                    .addGap(18, 18, 18)
                    .addGroup(jPanelLayout.createParallelGroup(GroupLayout.Alignment.LEADING, false)
                        .addComponent(inputField1, GroupLayout.DEFAULT_SIZE, 300, Short.MAX_VALUE)
                        .addComponent(inputField2)
                        .addComponent(resultField))
                    .addGap(30, 30, 30)
                    .addComponent(flagLabel)
                    .addGap(18, 18, 18)
                    .addComponent(flagField, GroupLayout.PREFERRED_SIZE, 300, GroupLayout.PREFERRED_SIZE)
                    .addContainerGap(30, Short.MAX_VALUE))
                .addGroup(jPanelLayout.createSequentialGroup()
                    .addGap(150, 150, 150)
                    .addComponent(resetButton, GroupLayout.PREFERRED_SIZE, 150, GroupLayout.PREFERRED_SIZE)
                    .addGap(50, 50, 50)
                    .addComponent(calculateButton, GroupLayout.PREFERRED_SIZE, 150,
GroupLayout.PREFERRED_SIZE)
```

```
                .addContainerGap(GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addGroup(GroupLayout.Alignment.TRAILING, jPanelLayout.createSequentialGroup()
                .addContainerGap(GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(jPanelLayout.createParallelGroup(GroupLayout.Alignment.LEADING)
                    .addGroup(GroupLayout.Alignment.TRAILING, jPanelLayout.createSequentialGroup()
                        .addComponent(titleLabel, GroupLayout.PREFERRED_SIZE, 200, GroupLayout.PREFERRED_SIZE)
                        .addGap(345, 345, 345))
                    .addGroup(GroupLayout.Alignment.TRAILING, jPanelLayout.createSequentialGroup()
                        .addComponent(combobox, GroupLayout.PREFERRED_SIZE, 400, GroupLayout.PREFERRED_SIZE)
                        .addGap(250, 250, 250))))
        );
        jPanelLayout.setVerticalGroup(
            jPanelLayout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(jPanelLayout.createSequentialGroup()
                .addGap(30, 30, 30)
                .addComponent(titleLabel)
                .addGap(30, 30, 30)
                .addComponent(combobox, GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE,
GroupLayout.PREFERRED_SIZE)
                .addGap(30, 30, 30)
                .addGroup(jPanelLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(inputLabel1)
                    .addComponent(inputField1, GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE,
GroupLayout.PREFERRED_SIZE)
                    .addComponent(flagLabel)
                    .addComponent(flagField, GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE,
GroupLayout.PREFERRED_SIZE))
                .addGap(30, 30, 30)
                .addGroup(jPanelLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(inputLabel2)
                    .addComponent(inputField2, GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE,
GroupLayout.PREFERRED_SIZE))
                .addGap(30, 30, 30)
                .addGroup(jPanelLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(outputLabel)
                    .addComponent(resultField, GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE,
GroupLayout.PREFERRED_SIZE))
                .addGap(30, 30, 30)
                .addGroup(jPanelLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(resetButton)
                    .addComponent(calculateButton))
                .addContainerGap(30, Short.MAX_VALUE))
        );

        GroupLayout layout = new GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addComponent(jPanel, GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addComponent(jPanel, GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }

    private void resetActionPerformed(ActionEvent evt, JTextField inputField1, JTextField inputField2, JTextField
resultField, JTextField flagField) {
        inputField1.setText("");
        inputField2.setText("");
        resultField.setText("");
        flagField.setText("");
    }

    private void calculateActionPerformed(ActionEvent evt, JComboBox<String> combobox, JTextField
inputField1, JTextField inputField2, JTextField flagField, JTextField resultField) {
        String selectedOption = combobox.getSelectedItem().toString();
        switch (selectedOption) {
            case "Hamming Distance":
                handleHammingDistance(inputField1, inputField2, resultField);
                break;
            case "Parity Check":
                handleParityCheck(inputField1, inputField2, resultField);
                break;
            case "Bit Stuffing":
                handleBitStuffing(inputField1, flagField, resultField);
                break;
            case "Bit Destuffing":
                handleBitDestuffing(inputField1, flagField, resultField);
                break;
            case "Character Stuffing":
                handleCharacterStuffing(inputField1, flagField, resultField);
                break;
            case "Character Destuffing":
                handleCharacterDestuffing(inputField1, flagField, resultField);
                break;
            default:
                JOptionPane.showMessageDialog(this, "Please select a valid option.");
                break;
        }
    }

    private void handleHammingDistance(JTextField inputField1, JTextField inputField2, JTextField resultField) {
        A = inputField1.getText();
        B = inputField2.getText();
        if (A.length() != B.length()) {
            resultField.setText("Not Possible");
        } else {
            int dist = 0;
            for (int i = 0; i < A.length(); i++) {
                if (A.charAt(i) != B.charAt(i)) {
                    dist++;
                }
            }
            resultField.setText(Integer.toString(dist));
        }
    }

    private void handleParityCheck(JTextField inputField1, JTextField inputField2, JTextField resultField) {
        // Add parity check logic here
    }

    private void handleBitStuffing(JTextField inputField1, JTextField flagField, JTextField resultField) {
        // Example bit stuffing logic
        String input = inputField1.getText();
        String flag = flagField.getText();
        String stuffed = input.replaceAll("11111", "111110"); // Bit stuffing example
        resultField.setText(flag + stuffed + flag);
    }

    private void handleBitDestuffing(JTextField inputField1, JTextField flagField, JTextField resultField) {
        // Example bit destuffing logic
        String input = inputField1.getText();
        String flag = flagField.getText();
        String destuffed = input.replaceAll("111110", "11111"); // Bit destuffing example
        resultField.setText(destuffed);
    }

    private void handleCharacterStuffing(JTextField inputField1, JTextField flagField, JTextField resultField) {
        // Example character stuffing logic
        String input = inputField1.getText();
        String flag = flagField.getText();
        String stuffed = input.replace(flag, flag + "ESC"); // Character stuffing example
        resultField.setText(flag + stuffed + flag);
    }

    private void handleCharacterDestuffing(JTextField inputField1, JTextField flagField, JTextField resultField) {
        // Example character destuffing logic
        String input = inputField1.getText();
        String flag = flagField.getText();
        String destuffed = input.replace(flag + "ESC", flag); // Character destuffing example
        resultField.setText(destuffed);
    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Data_Com().setVisible(true);
            }
        });
    }

    // Variables declaration
    private JComboBox<String> combobox;
    private JButton calculateButton;
    private JButton resetButton;
    private JLabel outputLabel;
```

```
    private JLabel inputLabel1;
    private JLabel inputLabel2;
    private JLabel flagLabel;
    private JLabel titleLabel;
    private JTextField resultField;
    private JTextField inputField1;
    private JTextField inputField2;
    private JTextField flagField;

}
```