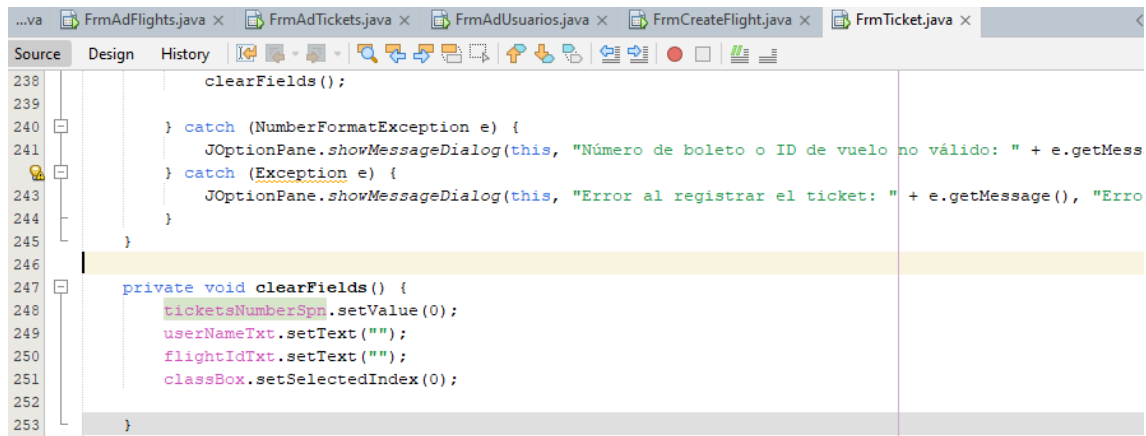


HW21 - Find pitfalls in the Project

1. Method overloading



```
238      clearFields();
239
240      } catch (NumberFormatException e) {
241          JOptionPane.showMessageDialog(this, "Número de boleto o ID de vuelo no válido: " + e.getMess
242      } catch (Exception e) {
243          JOptionPane.showMessageDialog(this, "Error al registrar el ticket: " + e.getMessage(), "Erro
244      }
245  }
246
247  private void clearFields() {
248      ticketsNumberSpin.setValue(0);
249      userNameTxt.setText("");
250      flightIdTxt.setText("");
251      classBox.setSelectedIndex(0);
252  }
253  }
```

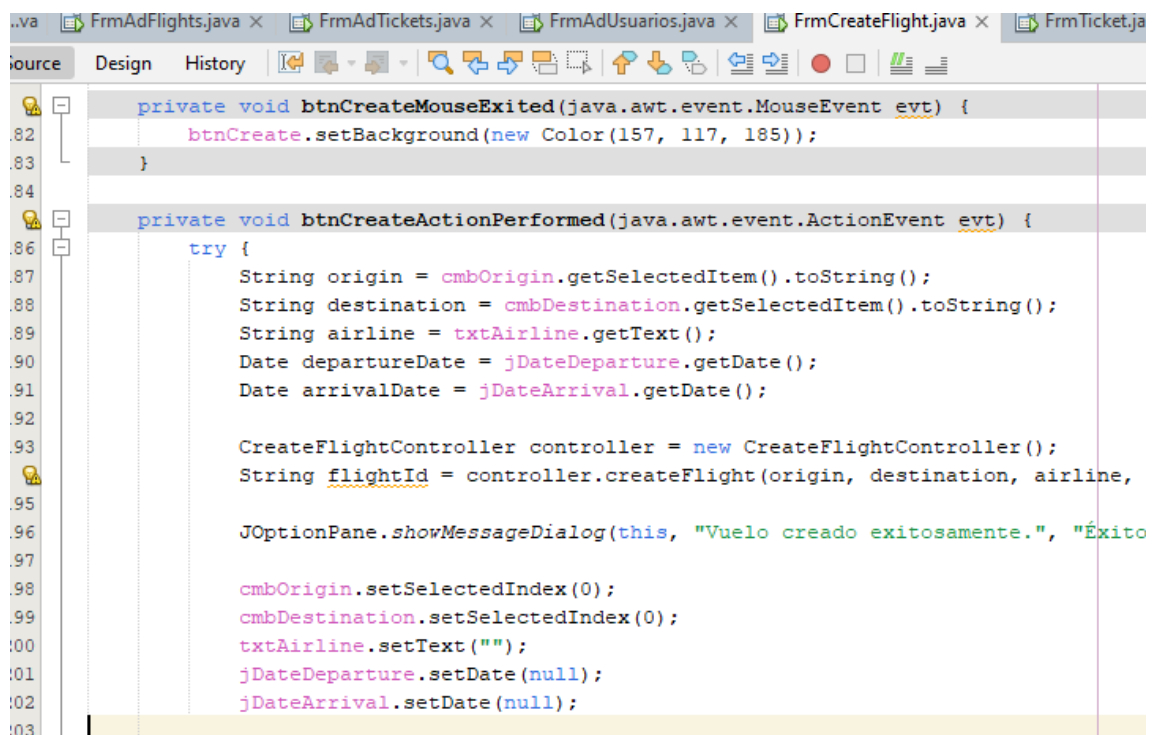
Explanation:

In this code the button is used to perform actions that should not be its own.

Possible solution:

Create a class with generic functions and only call them in the button action code

2. Duplicate Code



```
private void btnCreateMouseClicked(java.awt.event.MouseEvent evt) {
    btnCreate.setBackground(new Color(157, 117, 185));
}

private void btnCreateActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String origin = cmbOrigin.getSelectedItem().toString();
        String destination = cmbDestination.getSelectedItem().toString();
        String airline = txtAirline.getText();
        Date departureDate = jDateDeparture.getDate();
        Date arrivalDate = jDateArrival.getDate();

        CreateFlightController controller = new CreateFlightController();
        String flightId = controller.createFlight(origin, destination, airline,

        JOptionPane.showMessageDialog(this, "Vuelo creado exitosamente.", "Éxito

        cmbOrigin.setSelectedIndex(0);
        cmbDestination.setSelectedIndex(0);
        txtAirline.setText("");
        jDateDeparture.setDate(null);
        jDateArrival.setDate(null);
    }
}
```

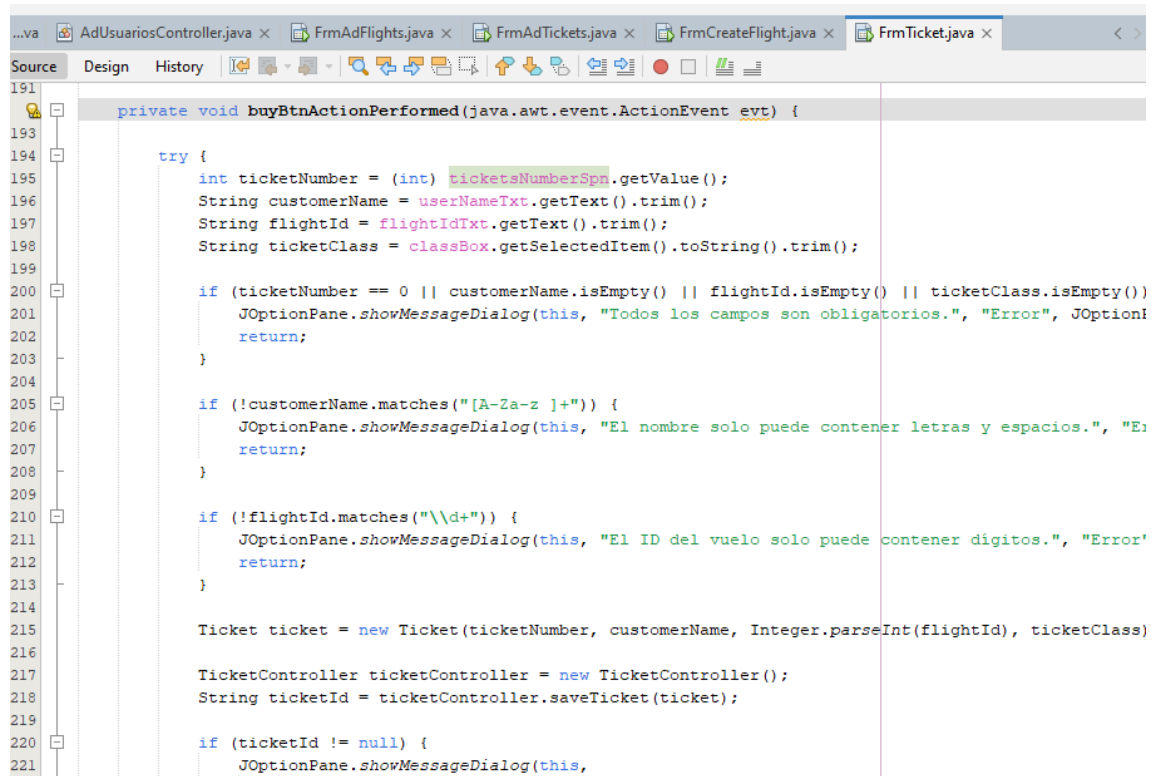
Explanation:

In this code a function to clean cells of another class is duplicated

Possible solution:

Create a generic class so you can reuse code instead of duplicating it

3. Long Methods



```
191 private void buyBtnActionPerformed(java.awt.event.ActionEvent evt) {
193
194     try {
195         int ticketNumber = (int) ticketsNumberSpn.getValue();
196         String customerName = userNameTxt.getText().trim();
197         String flightId = flightIdTxt.getText().trim();
198         String ticketClass = classBox.getSelectedItem().toString().trim();
199
200         if (ticketNumber == 0 || customerName.isEmpty() || flightId.isEmpty() || ticketClass.isEmpty())
201             JOptionPane.showMessageDialog(this, "Todos los campos son obligatorios.", "Error", JOptionPane.
202                 return;
203         }
204
205         if (!customerName.matches("[A-Za-z ]+")) {
206             JOptionPane.showMessageDialog(this, "El nombre solo puede contener letras y espacios.", "E
207                 return;
208         }
209
210         if (!flightId.matches("\\d+")) {
211             JOptionPane.showMessageDialog(this, "El ID del vuelo solo puede contener digitos.", "Error'
212                 return;
213         }
214
215         Ticket ticket = new Ticket(ticketNumber, customerName, Integer.parseInt(flightId), ticketClass)
216
217         TicketController ticketController = new TicketController();
218         String ticketId = ticketController.saveTicket(ticket);
219
220         if (ticketId != null) {
221             JOptionPane.showMessageDialog(this,
```

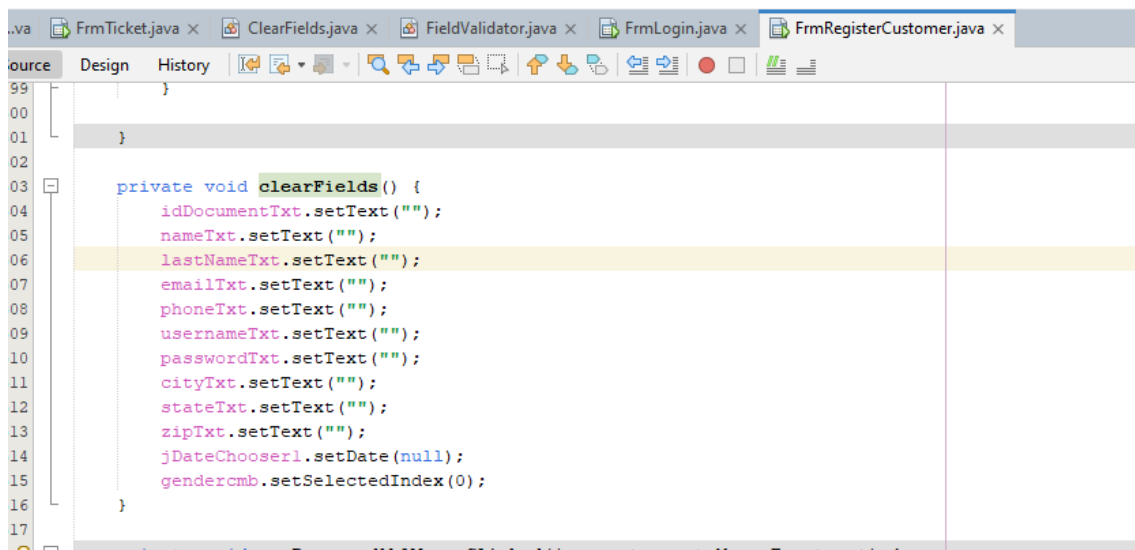
Explanation:

In this code the button is overloaded to validate if all the fields are completed and that they have the correct data type.

Possible solution:

Create a generic class in utils that performs that action and only call it on the button

4. Duplicate Code



```
99 }
100
101 }
102
103 private void clearFields() {
104     idDocumentTxt.setText("");
105     nameTxt.setText("");
106     lastNameTxt.setText("");
107     emailTxt.setText("");
108     phoneTxt.setText("");
109     usernameTxt.setText("");
110     passwordTxt.setText("");
111     cityTxt.setText("");
112     stateTxt.setText("");
113     zipTxt.setText("");
114     jDateChooser1.setDate(null);
115     gendercmb.setSelectedIndex(0);
116 }
117
118 private void seePasswordBtnMouseClicked(java.awt.event.MouseEvent evt) {
```

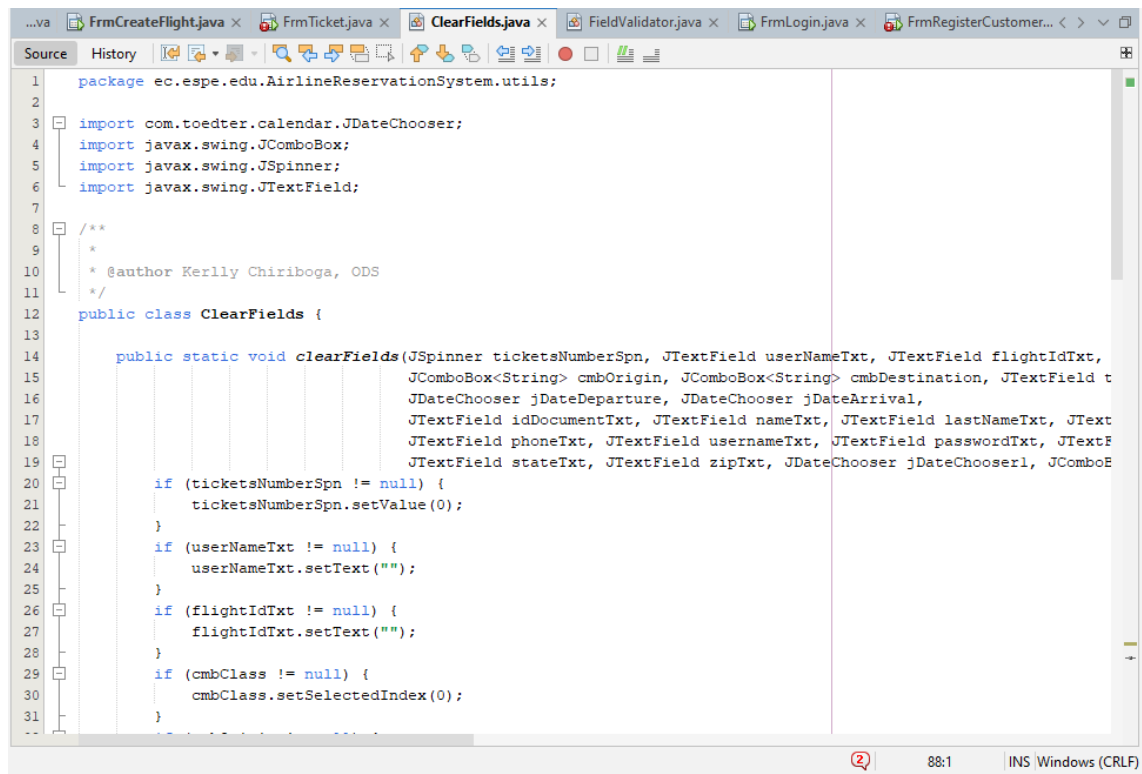
Explanation:

In this code a function to clean cells of another class is duplicated

Possible solution:

Create a generic class in utils that performs that action and only call it on the button

5. Long Methods



```
1 package ec.espe.edu.AirlineReservationSystem.utils;
2
3 import com.toedter.calendar.JDateChooser;
4 import javax.swing.JComboBox;
5 import javax.swing.JSpinner;
6 import javax.swing.JTextField;
7
8 /**
9  *
10  * @author Kerlly Chiriboga, ODS
11  */
12 public class ClearFields {
13
14     public static void clearFields(JSpinner ticketsNumberSpn, JTextField userNameTxt, JTextField flightIdTxt,
15                                   JComboBox<String> cmbOrigin, JComboBox<String> cmbDestination, JTextField t
16                                   JDateChooser jDateDeparture, JDateChooser jDateArrival,
17                                   JTextField idDocumentTxt, JTextField nameTxt, JTextField lastNameTxt, JText
18                                   JTextField phoneTxt, JTextField usernameTxt, JTextField passwordTxt, JTextF
19                                   JTextField stateTxt, JTextField zipTxt, JDateChooser jDateChooser1, JComboE
20
21     if (ticketsNumberSpn != null) {
22         ticketsNumberSpn.setValue(0);
23     }
24     if (userNameTxt != null) {
25         userNameTxt.setText("");
26     }
27     if (flightIdTxt != null) {
28         flightIdTxt.setText("");
29     }
30     if (cmbClass != null) {
31         cmbClass.setSelectedIndex(0);
32     }
33 }
```

Explanation:

This class has a method that is too long and is not efficient when reusing it.

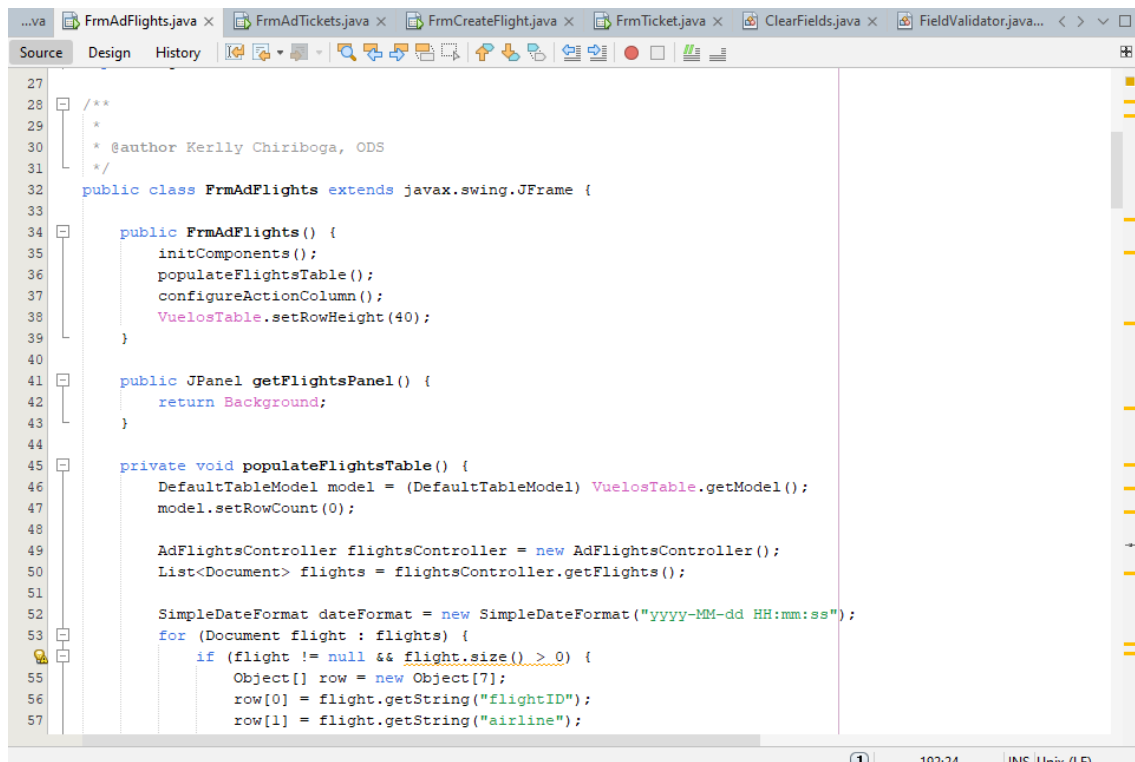
Possible solution:

Break this method into smaller methods that perform the same actions as the original.

6. Duplicate Code and Overloading Class

```
...va  FrmAdTickets.java x  FrmCreateFlight.java x  FrmTicket.java x  ClearFields.java x  FieldValidator.java x  Fr
Source  Design  History  [Icons]
20  public class FrmAdTickets extends javax.swing.JFrame {
21
22      private AdTicketsController ticketController;
23
24      public FrmAdTickets() {
25          initComponents();
26          ticketController = new AdTicketsController();
27          populateTicketsTable();
28          configureActionColumn();
29          TicketsTable.setRowHeight(40);
30      }
31
32      public JPanel getTicketsPanel() {
33          return Background;
34      }
35
36      private void populateTicketsTable() {
37          DefaultTableModel model = (DefaultTableModel) TicketsTable.getModel();
38          model.setRowCount(0);
39
40          List<Document> tickets = ticketController.getTickets();
41
42          for (Document ticket : tickets) {
43              if (ticket != null && ticket.size() > 0) {
44                  Object[] row = new Object[6];
45                  row[0] = ticket.getInteger("Number of Ticket");
46                  row[1] = ticket.getInteger("Ticket ID");
47                  row[2] = ticket.getString("Customer Name");
48                  row[3] = ticket.getInteger("Id Flight");
49                  row[4] = ticket.getString("Ticket Class");
50              }
          }
      }
  }
```

```
27  /**
28   *
29   * @author Kerlly Chiriboga, ODS
30   */
31
32  public class FrmAdUsuarios extends javax.swing.JFrame {
33
34      private AdUsuariosController usuariosController;
35
36      public FrmAdUsuarios() {
37          initComponents();
38          usuariosController = new AdUsuariosController();
39          populateUsuariosTable();
40          configureActionColumn();
41          UsuariosTable.setRowHeight(40);
42      }
43
44      public JPanel getUsuariosPanel() {
45          return Background;
46      }
47
48      private void populateUsuariosTable() {
49          DefaultTableModel model = (DefaultTableModel) UsuariosTable.getModel();
50          model.setRowCount(0);
51
52          List<Document> usuarios = usuariosController.getUsuarios();
53
54          for (Document usuario : usuarios) {
55              if (usuario != null && usuario.size() > 0) {
56                  Object[] row = new Object[6];
57                  row[0] = usuario.getInteger("Number of User");
58                  row[1] = usuario.getInteger("User ID");
59                  row[2] = usuario.getString("Customer Name");
60                  row[3] = usuario.getInteger("Id Flight");
61                  row[4] = usuario.getString("User Class");
62              }
          }
      }
  }
```



```
27
28 /**
29  *
30  * @author Kerlly Chiriboga, ODS
31  */
32 public class FrmAdFlights extends javax.swing.JFrame {
33
34     public FrmAdFlights() {
35         initComponents();
36         populateFlightsTable();
37         configureActionColumn();
38         VuelosTable.setRowHeight(40);
39     }
40
41     public JPanel getFlightsPanel() {
42         return Background;
43     }
44
45     private void populateFlightsTable() {
46         DefaultTableModel model = (DefaultTableModel) VuelosTable.getModel();
47         model.setRowCount(0);
48
49         AdFlightsController flightsController = new AdFlightsController();
50         List<Document> flights = flightsController.getFlights();
51
52         SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
53         for (Document flight : flights) {
54             if (flight != null && flight.size() > 0) {
55                 Object[] row = new Object[7];
56                 row[0] = flight.getString("flightID");
57                 row[1] = flight.getString("airline");
```

Explanation:

In the code of these three classes, the code for handling tables in the interface is repeated and a screen that is only for viewing is overloaded.

Possible solution:

Take the code out of there and create new classes in utils to only handle a call from the screen.