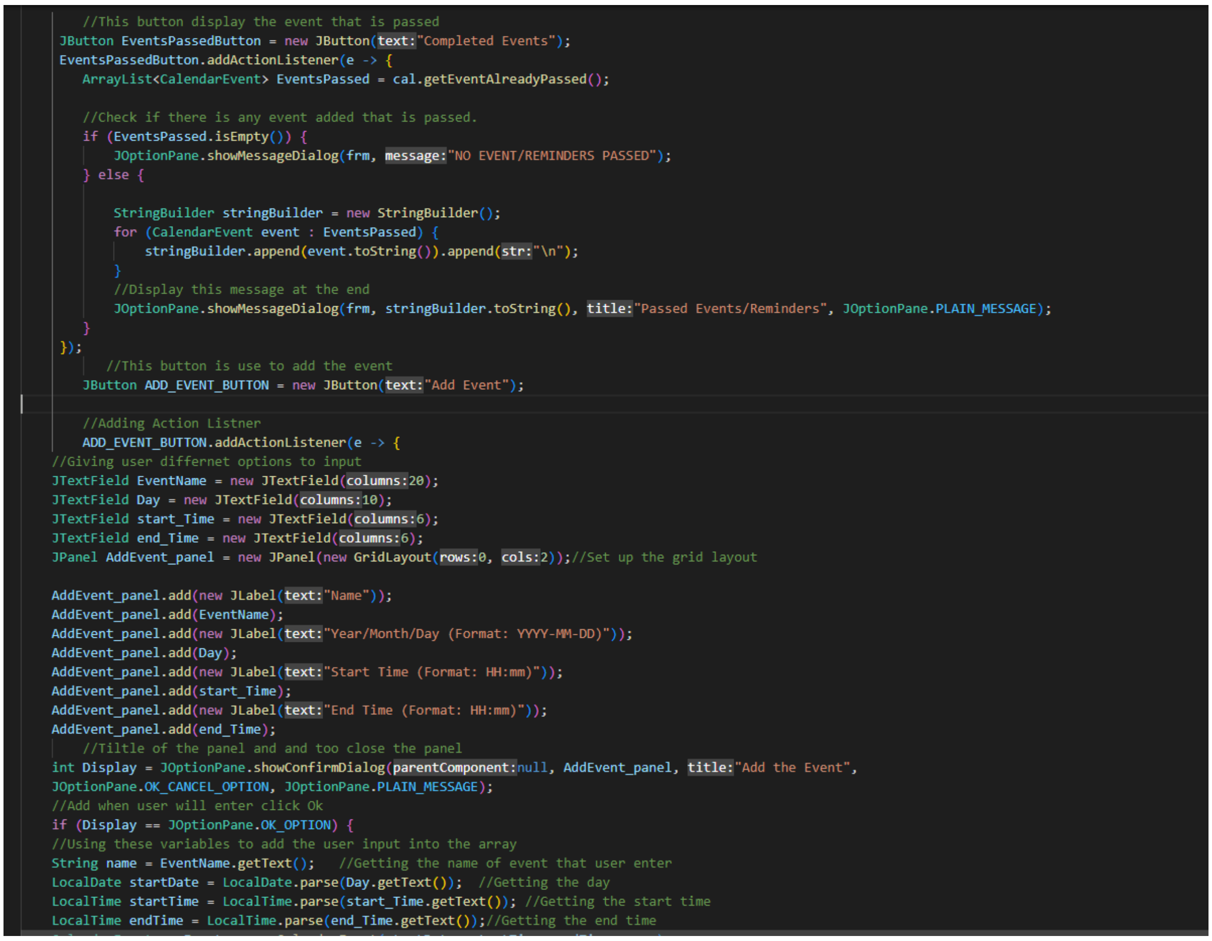
Iteration 3 - Refactoring Document

Event Methods:

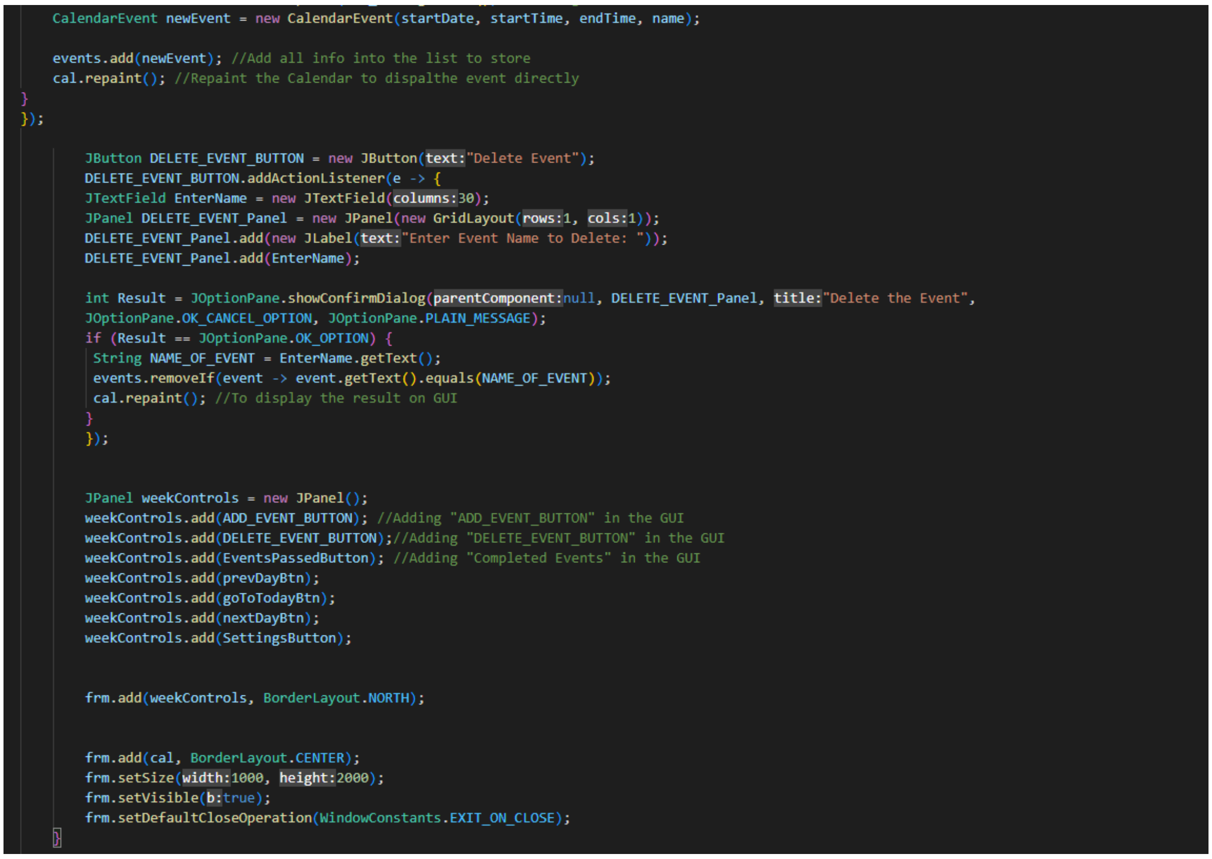
Code smell: Bloaters

Before Refactoring the events functionality had a bunch of methods which were extensively long and as we wanted to add more features or troubleshoot it would have been hard to do so. After refactoring, we made 1 method for all the event functionality so it would be easier for us to maintain or troubleshoot when needed to do so. It also helped us make a singular button on our GUI for all the features.

Before Refactoring:

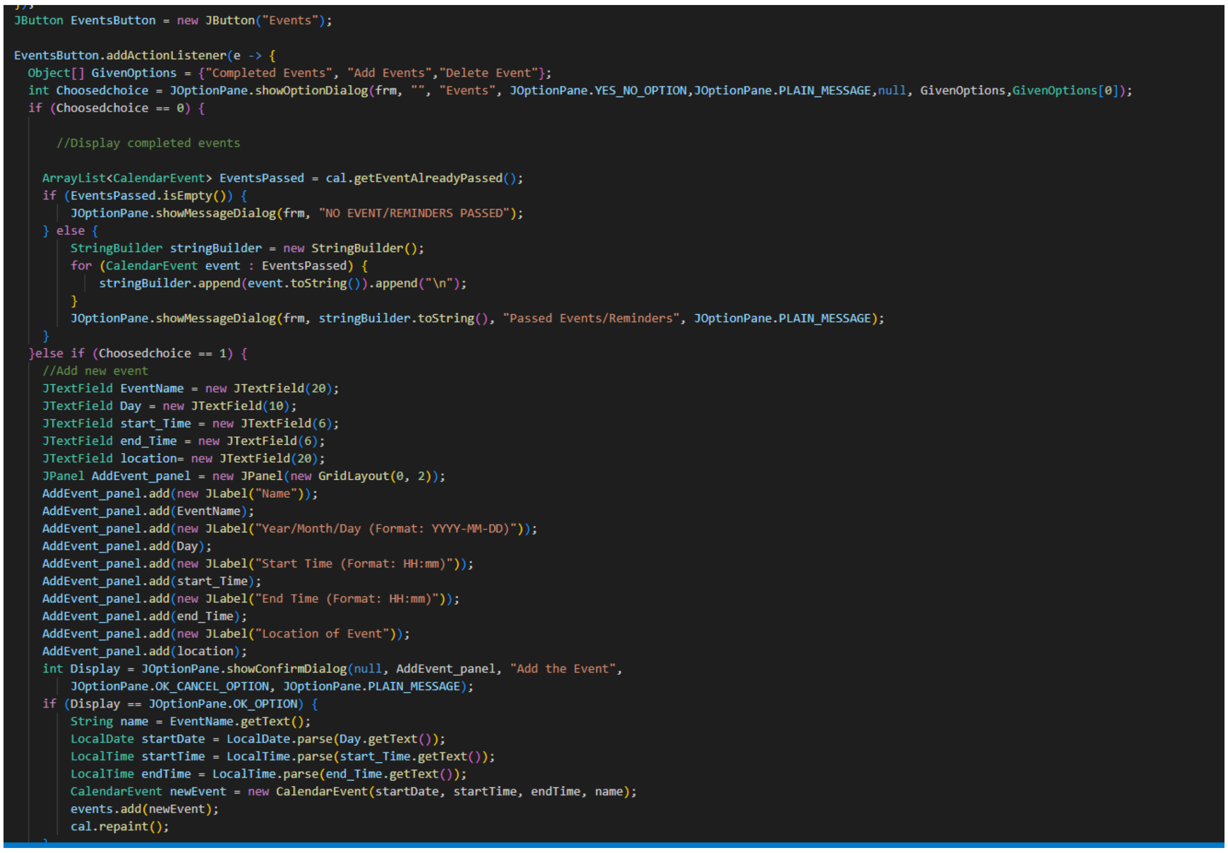


```java
    //This button display the event that is passed
  JButton EventsPassedButton = new JButton(text:"Completed Events");
  EventsPassedButton.addActionListener(e -> {
    ArrayList<CalendarEvent> EventsPassed = cal.getEventAlreadyPassed();

    //Check if there is any event added that is passed.
    if (EventsPassed.isEmpty()) {
        JOptionPane.showMessageDialog(frm, message:"NO EVENT/REMINDERS PASSED");
    } else {

        StringBuilder stringBuilder = new StringBuilder();
        for (CalendarEvent event : EventsPassed) {
            stringBuilder.append(event.toString()).append(str:"\n");
        }
        //Display this message at the end
        JOptionPane.showMessageDialog(frm, stringBuilder.toString(), title:"Passed Events/Reminders", JOptionPane.PLAIN_MESSAGE);
    }
  });
      //This button is use to add the event
    JButton ADD_EVENT_BUTTON = new JButton(text:"Add Event");

    //Adding Action Listner
    ADD_EVENT_BUTTON.addActionListener(e -> {
 //Giving user differnet options to input
 JTextField EventName = new JTextField(columns:20);
 JTextField Day = new JTextField(columns:10);
 JTextField start_Time = new JTextField(columns:6);
 JTextField end_Time = new JTextField(columns:6);
 JPanel AddEvent_panel = new JPanel(new GridLayout(rows:0, cols:2));//Set up the grid layout

 AddEvent_panel.add(new JLabel(text:"Name"));
 AddEvent_panel.add(EventName);
 AddEvent_panel.add(new JLabel(text:"Year/Month/Day (Format: YYYY-MM-DD)"));
 AddEvent_panel.add(Day);
 AddEvent_panel.add(new JLabel(text:"Start Time (Format: HH:mm)"));
 AddEvent_panel.add(start_Time);
 AddEvent_panel.add(new JLabel(text:"End Time (Format: HH:mm)"));
 AddEvent_panel.add(end_Time);
     //Tiltle of the panel and and too close the panel
 int Display = JOptionPane.showConfirmDialog(parentComponent:null, AddEvent_panel, title:"Add the Event",
 JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);
 //Add when user will enter click Ok
 if (Display == JOptionPane.OK_OPTION) {
 //Using these variables to add the user input into the array
 String name = EventName.getText();   //Getting the name of event that user enter
 LocalDate startDate = LocalDate.parse(Day.getText());  //Getting the day
 LocalTime startTime = LocalTime.parse(start_Time.getText()); //Getting the start time
 LocalTime endTime = LocalTime.parse(end_Time.getText());//Getting the end time
```



```java
        CalendarEvent newEvent = new CalendarEvent(startDate, startTime, endTime, name);

        events.add(newEvent); //Add all info into the list to store
        cal.repaint(); //Repaint the Calendar to dispalthe event directly
    }
    });

        JButton DELETE_EVENT_BUTTON = new JButton(text:"Delete Event");
        DELETE_EVENT_BUTTON.addActionListener(e -> {
        JTextField EnterName = new JTextField(columns:30);
        JPanel DELETE_EVENT_Panel = new JPanel(new GridLayout(rows:1, cols:1));
        DELETE_EVENT_Panel.add(new JLabel(text:"Enter Event Name to Delete: "));
        DELETE_EVENT_Panel.add(EnterName);

        int Result = JOptionPane.showConfirmDialog(parentComponent:null, DELETE_EVENT_Panel, title:"Delete the Event",
        JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);
        if (Result == JOptionPane.OK_OPTION) {
         String NAME_OF_EVENT = EnterName.getText();
         events.removeIf(event -> event.getText().equals(NAME_OF_EVENT));
         cal.repaint(); //To display the result on GUI
        }
        });


        JPanel weekControls = new JPanel();
        weekControls.add(ADD_EVENT_BUTTON); //Adding "ADD_EVENT_BUTTON" in the GUI
        weekControls.add(DELETE_EVENT_BUTTON);//Adding "DELETE_EVENT_BUTTON" in the GUI
        weekControls.add(EventsPassedButton); //Adding "Completed Events" in the GUI
        weekControls.add(prevDayBtn);
        weekControls.add(goToTodayBtn);
        weekControls.add(nextDayBtn);
        weekControls.add(SettingsButton);


        frm.add(weekControls, BorderLayout.NORTH);


        frm.add(cal, BorderLayout.CENTER);
        frm.setSize(width:1000, height:2000);
        frm.setVisible(b:true);
        frm.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    }
```

After Refactoring:



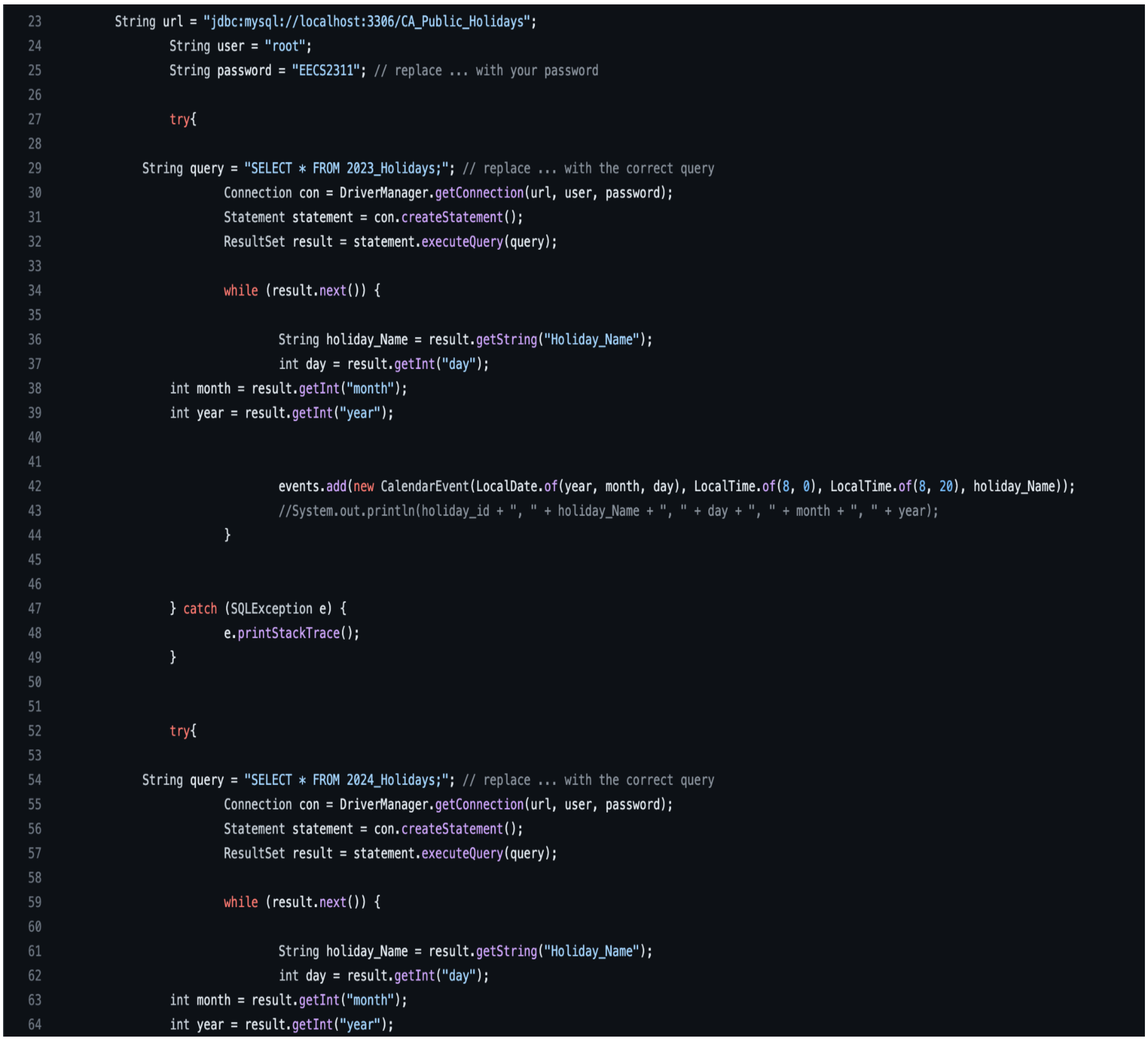
```java
JButton EventsButton = new JButton("Events");

EventsButton.addActionListener(e -> {
  Object[] GivenOptions = {"Completed Events", "Add Events","Delete Event"};
  int Choosedchoice = JOptionPane.showOptionDialog(frm, "", "Events", JOptionPane.YES_NO_OPTION,JOptionPane.PLAIN_MESSAGE,null, GivenOptions,GivenOptions[0]);
  if (Choosedchoice == 0) {

      //Display completed events

    ArrayList<CalendarEvent> EventsPassed = cal.getEventAlreadyPassed();
    if (EventsPassed.isEmpty()) {
        JOptionPane.showMessageDialog(frm, "NO EVENT/REMINDERS PASSED");
    } else {
        StringBuilder stringBuilder = new StringBuilder();
        for (CalendarEvent event : EventsPassed) {
            stringBuilder.append(event.toString()).append("\n");
        }
        JOptionPane.showMessageDialog(frm, stringBuilder.toString(), "Passed Events/Reminders", JOptionPane.PLAIN_MESSAGE);
    }
  }else if (Choosedchoice == 1) {
    //Add new event
    JTextField EventName = new JTextField(20);
    JTextField Day = new JTextField(10);
    JTextField start_Time = new JTextField(6);
    JTextField end_Time = new JTextField(6);
    JTextField location= new JTextField(20);
    JPanel AddEvent_panel = new JPanel(new GridLayout(0, 2));
    AddEvent_panel.add(new JLabel("Name"));
    AddEvent_panel.add(EventName);
    AddEvent_panel.add(new JLabel("Year/Month/Day (Format: YYYY-MM-DD)"));
    AddEvent_panel.add(Day);
    AddEvent_panel.add(new JLabel("Start Time (Format: HH:mm)"));
    AddEvent_panel.add(start_Time);
    AddEvent_panel.add(new JLabel("End Time (Format: HH:mm)"));
    AddEvent_panel.add(end_Time);
    AddEvent_panel.add(new JLabel("Location of Event"));
    AddEvent_panel.add(location);
    int Display = JOptionPane.showConfirmDialog(null, AddEvent_panel, "Add the Event",
        JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);
    if (Display == JOptionPane.OK_OPTION) {
        String name = EventName.getText();
        LocalDate startDate = LocalDate.parse(Day.getText());
        LocalTime startTime = LocalTime.parse(start_Time.getText());
        LocalTime endTime = LocalTime.parse(end_Time.getText());
        CalendarEvent newEvent = new CalendarEvent(startDate, startTime, endTime, name);
        events.add(newEvent);
        cal.repaint();
```

Canadian Public Holidays SQL Implementation:

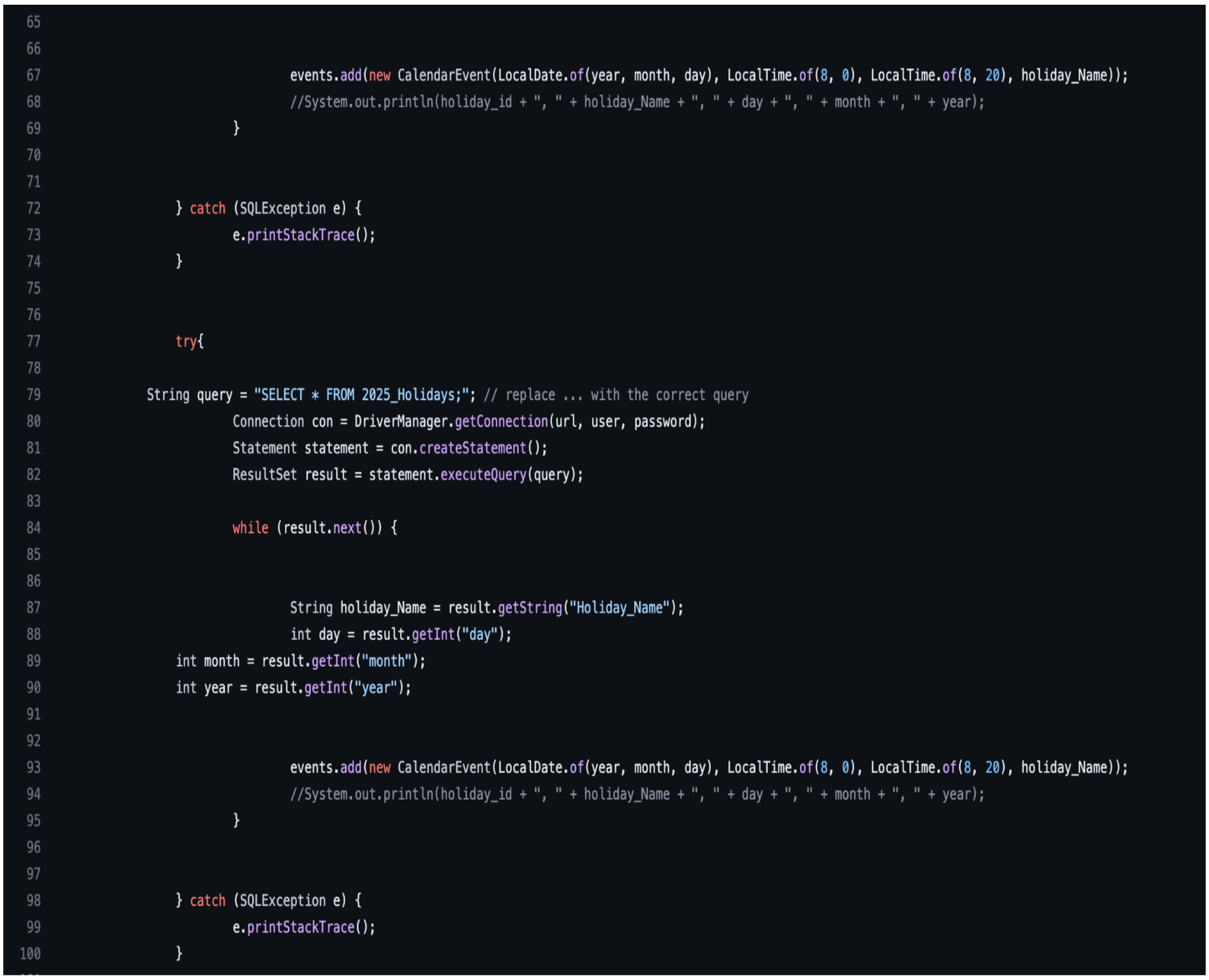
Code Smell: Dispensables

The SQL implementation for displaying all Canadian Public Holidays on the calendar was refactored to fix the ‘dispensables’ code smell. An SQL database containing 3 tables for all the holidays in 2023, 2024 and 2025 was created. To have this implemented in our Java program, we had to make use of JDBC. The initial implementation was repetitive and lengthy, as the same piece of code was repeated for each table, with the only difference being the SQL query. This was refactored to fix the code smell and to make it more efficient. Rather than repeating the same code different times, a for loop was implemented which allowed us to replicate the same functionality in fewer lines of code.

Before Refactoring:

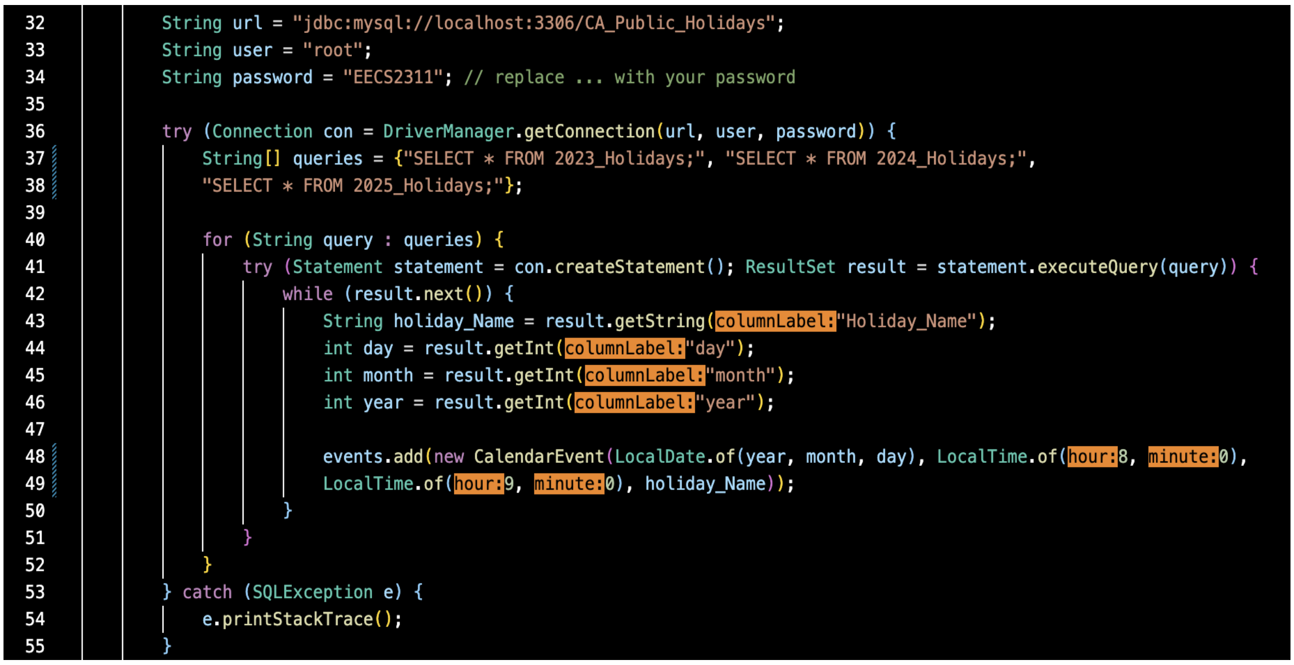


```java
    String url = "jdbc:mysql://localhost:3306/CA_Public_Holidays";
        String user = "root";
        String password = "EECS2311"; // replace ... with your password

        try{

      String query = "SELECT * FROM 2023_Holidays;"; // replace ... with the correct query
            Connection con = DriverManager.getConnection(url, user, password);
            Statement statement = con.createStatement();
            ResultSet result = statement.executeQuery(query);

            while (result.next()) {

                String holiday_Name = result.getString("Holiday_Name");
                int day = result.getInt("day");
        int month = result.getInt("month");
        int year = result.getInt("year");


                events.add(new CalendarEvent(LocalDate.of(year, month, day), LocalTime.of(8, 0), LocalTime.of(8, 20), holiday_Name));
                //System.out.println(holiday_id + ", " + holiday_Name + ", " + day + ", " + month + ", " + year);
            }


        } catch (SQLException e) {
            e.printStackTrace();
        }


        try{

      String query = "SELECT * FROM 2024_Holidays;"; // replace ... with the correct query
            Connection con = DriverManager.getConnection(url, user, password);
            Statement statement = con.createStatement();
            ResultSet result = statement.executeQuery(query);

            while (result.next()) {

                String holiday_Name = result.getString("Holiday_Name");
                int day = result.getInt("day");
        int month = result.getInt("month");
        int year = result.getInt("year");
```



```java


                events.add(new CalendarEvent(LocalDate.of(year, month, day), LocalTime.of(8, 0), LocalTime.of(8, 20), holiday_Name));
                //System.out.println(holiday_id + ", " + holiday_Name + ", " + day + ", " + month + ", " + year);
            }


        } catch (SQLException e) {
            e.printStackTrace();
        }


        try{

      String query = "SELECT * FROM 2025_Holidays;"; // replace ... with the correct query
            Connection con = DriverManager.getConnection(url, user, password);
            Statement statement = con.createStatement();
            ResultSet result = statement.executeQuery(query);

            while (result.next()) {


                String holiday_Name = result.getString("Holiday_Name");
                int day = result.getInt("day");
        int month = result.getInt("month");
        int year = result.getInt("year");


                events.add(new CalendarEvent(LocalDate.of(year, month, day), LocalTime.of(8, 0), LocalTime.of(8, 20), holiday_Name));
                //System.out.println(holiday_id + ", " + holiday_Name + ", " + day + ", " + month + ", " + year);
            }


        } catch (SQLException e) {
            e.printStackTrace();
        }
```

After Refactoring:



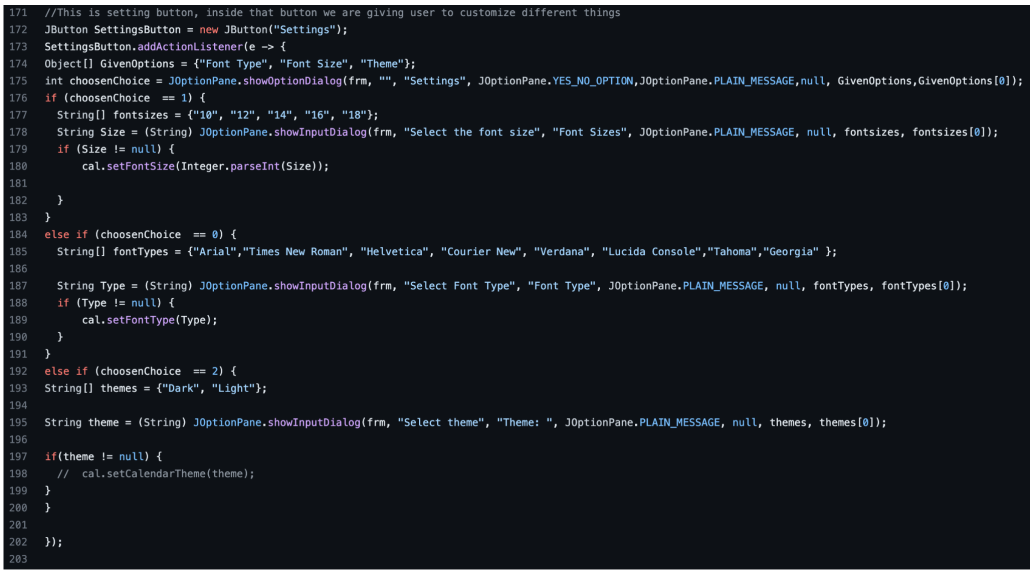
```java
String url = "jdbc:mysql://localhost:3306/CA_Public_Holidays";
String user = "root";
String password = "EECS2311"; // replace ... with your password

try (Connection con = DriverManager.getConnection(url, user, password)) {
    String[] queries = {"SELECT * FROM 2023_Holidays;", "SELECT * FROM 2024_Holidays;",
    "SELECT * FROM 2025_Holidays;"};

    for (String query : queries) {
        try (Statement statement = con.createStatement(); ResultSet result = statement.executeQuery(query)) {
            while (result.next()) {
                String holiday_Name = result.getString(columnLabel:"Holiday_Name");
                int day = result.getInt(columnLabel:"day");
                int month = result.getInt(columnLabel:"month");
                int year = result.getInt(columnLabel:"year");

                events.add(new CalendarEvent(LocalDate.of(year, month, day), LocalTime.of(hour:8, minute:0),
                LocalTime.of(hour:9, minute:0), holiday_Name));
            }
        }
    }
} catch (SQLException e) {
    e.printStackTrace();
}
```

Font Size / Font type / Export (Settings button):

Code smell: Bloaters

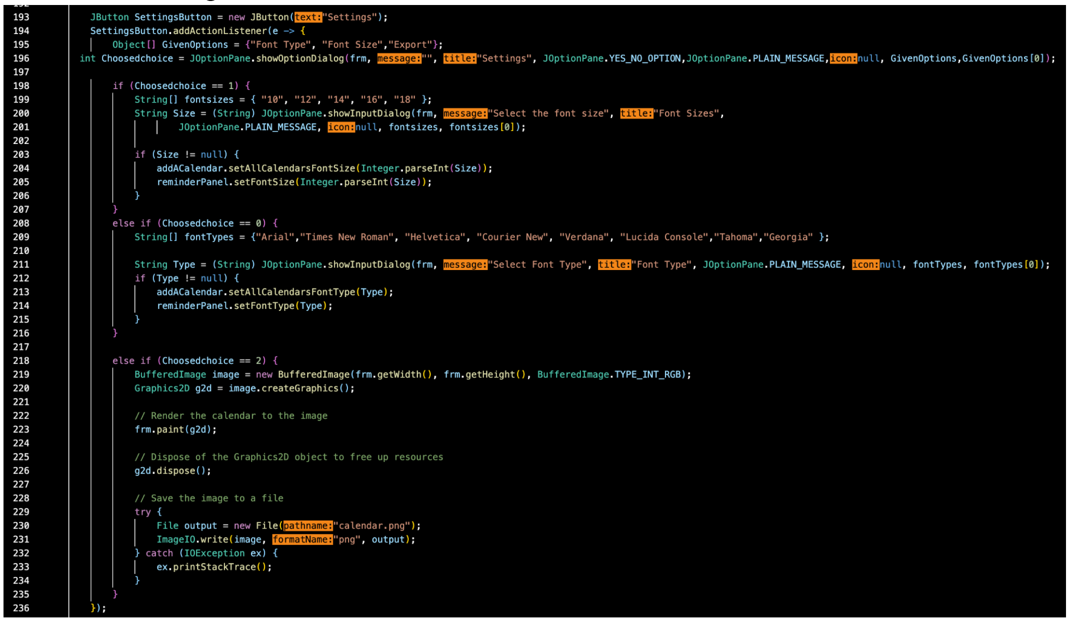
Before refactoring, we had a bunch of methods for each font size, font type and export feature which were quite long pieces of code. They also took up a bunch of space on the GUI because each of these functions needed its own separate button to access them. After refactoring we made it so that all of these functions would fall under one method class “Settings” that incorporated each of these functionalities into one button. This helped us save a lot of space on the GUI to add more buttons and made it so it would be easy for us making it so that we didn't have to worry about making more buttons instead we can just add it in the Settings button for any future features.

Before Refactoring:



```java
//This is setting button, inside that button we are giving user to customize different things
JButton SettingsButton = new JButton("Settings");
SettingsButton.addActionListener(e -> {
Object[] GivenOptions = {"Font Type", "Font Size", "Theme"};
int choosenChoice = JOptionPane.showOptionDialog(frm, "", "Settings", JOptionPane.YES_NO_OPTION,JOptionPane.PLAIN_MESSAGE,null, GivenOptions,GivenOptions[0]);
if (choosenChoice  == 1) {
  String[] fontsizes = {"10", "12", "14", "16", "18"};
  String Size = (String) JOptionPane.showInputDialog(frm, "Select the font size", "Font Sizes", JOptionPane.PLAIN_MESSAGE, null, fontsizes, fontsizes[0]);
  if (Size != null) {
      cal.setFontSize(Integer.parseInt(Size));

  }
}
else if (choosenChoice  == 0) {
  String[] fontTypes = {"Arial","Times New Roman", "Helvetica", "Courier New", "Verdana", "Lucida Console","Tahoma","Georgia" };

  String Type = (String) JOptionPane.showInputDialog(frm, "Select Font Type", "Font Type", JOptionPane.PLAIN_MESSAGE, null, fontTypes, fontTypes[0]);
  if (Type != null) {
      cal.setFontType(Type);
  }
}
else if (choosenChoice  == 2) {
String[] themes = {"Dark", "Light"};

String theme = (String) JOptionPane.showInputDialog(frm, "Select theme", "Theme: ", JOptionPane.PLAIN_MESSAGE, null, themes, themes[0]);

if(theme != null) {
  //  cal.setCalendarTheme(theme);
}
}

});

```

After Refactoring:



```java
  JButton SettingsButton = new JButton(text:"Settings");
  SettingsButton.addActionListener(e -> {
      Object[] GivenOptions = {"Font Type", "Font Size","Export"};
int Choosedchoice = JOptionPane.showOptionDialog(frm, message:"", title:"Settings", JOptionPane.YES_NO_OPTION,JOptionPane.PLAIN_MESSAGE,icon:null, GivenOptions,GivenOptions[0]);

      if (Choosedchoice == 1) {
          String[] fontsizes = { "10", "12", "14", "16", "18" };
          String Size = (String) JOptionPane.showInputDialog(frm, message:"Select the font size", title:"Font Sizes",
                  JOptionPane.PLAIN_MESSAGE, icon:null, fontsizes, fontsizes[0]);

          if (Size != null) {
              addACalendar.setAllCalendarsFontSize(Integer.parseInt(Size));
              reminderPanel.setFontSize(Integer.parseInt(Size));
          }
      }
      else if (Choosedchoice == 0) {
          String[] fontTypes = {"Arial","Times New Roman", "Helvetica", "Courier New", "Verdana", "Lucida Console","Tahoma","Georgia" };

          String Type = (String) JOptionPane.showInputDialog(frm, message:"Select Font Type", title:"Font Type", JOptionPane.PLAIN_MESSAGE, icon:null, fontTypes, fontTypes[0]);
          if (Type != null) {
              addACalendar.setAllCalendarsFontType(Type);
              reminderPanel.setFontType(Type);
          }
      }

      else if (Choosedchoice == 2) {
          BufferedImage image = new BufferedImage(frm.getWidth(), frm.getHeight(), BufferedImage.TYPE_INT_RGB);
          Graphics2D g2d = image.createGraphics();

          // Render the calendar to the image
          frm.paint(g2d);

          // Dispose of the Graphics2D object to free up resources
          g2d.dispose();

          // Save the image to a file
          try {
              File output = new File(pathname:"calendar.png");
              ImageIO.write(image, formatName:"png", output);
          } catch (IOException ex) {
              ex.printStackTrace();
          }
      }
  });
```