```
import javax.swing.*;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.sql.*;
import java.util.Vector;

public class Main {
    private Connection conn;
    private JFrame frame;
    private JComboBox<String> comboBox;
    private JTable table;
    private DefaultTableModel tableModel;
    private JProgressBar progressBar;

    public Main() {
        // Establecer conexión a la base de datos PostgreSQL
        connectDB();

        // Crear la interfaz gráfica
        createGUI();
    }

    private void connectDB() {
        try {
            String url = "jdbc:postgresql://localhost:5432/formula1";
            String user = "postgres";
            String password = "miguel";
            conn = DriverManager.getConnection(url, user, password);
            System.out.println("Conexión establecida con PostgreSQL.");
        } catch (SQLException e) {
```

```
            e.printStackTrace();
        }
    }

    private void createGUI() {
        frame = new JFrame("Tabla de Constructores y Puntos por Año de Carrera");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(800, 400);

        // Combo box para seleccionar el año de carrera
        comboBox = new JComboBox<>();
        populateComboBox();
        comboBox.addActionListener(e -> {
            // Cuando se seleccione un año, actualizar la tabla de constructores y puntos
            updateTableInBackground();
        });

        // Barra de progreso para mostrar mientras se carga la tabla
        progressBar = new JProgressBar();
        progressBar.setStringPainted(true);

        // Tabla para mostrar los datos de constructores y puntos
        tableModel = new DefaultTableModel();
        table = new JTable(tableModel);
        JScrollPane scrollPane = new JScrollPane(table);

        // Centrar el contenido de las celdas
        DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();
        centerRenderer.setHorizontalAlignment(JLabel.CENTER);
        table.setDefaultRenderer(Object.class, centerRenderer);
```

```
        frame.getContentPane().setLayout(new BorderLayout());
        frame.getContentPane().add(comboBox, BorderLayout.NORTH);
        frame.getContentPane().add(progressBar, BorderLayout.SOUTH);
        frame.getContentPane().add(scrollPane, BorderLayout.CENTER);

        frame.setVisible(true);
    }

    private void populateComboBox() {
        try {
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT DISTINCT year FROM races ORDER BY year
DESC");
            while (rs.next()) {
                comboBox.addItem(rs.getString("year"));
            }
            rs.close();
            stmt.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void updateTableInBackground() {
        String selectedYear = (String) comboBox.getSelectedItem();
        if (selectedYear != null) {
            // Crear un SwingWorker para ejecutar la consulta en segundo plano
            SwingWorker<Void, Void> worker = new SwingWorker<Void, Void>() {
                @Override
                protected Void doInBackground() throws Exception {
                    try {
```

```
                // Consulta para obtener los constructores y sus puntos totales para el año seleccionado
                String query = "SELECT c.name AS constructor_name, SUM(cs.points) AS total_points " +
                        "FROM constructors c " +
                        "JOIN constructor_standings cs ON c.constructor_id = cs.constructor_id " +
                        "JOIN races r ON cs.race_id = r.race_id " +
                        "WHERE r.year = ? " +
                        "GROUP BY constructor_name " +
                        "ORDER BY total_points DESC";

                PreparedStatement pstmt = conn.prepareStatement(query);
                pstmt.setInt(1, Integer.parseInt(selectedYear));
                ResultSet rs = pstmt.executeQuery();

                // Obtener columnas
                Vector<String> columnNames = new Vector<>();
                columnNames.add("Constructor Name");
                columnNames.add("Total Points");

                // Obtener filas
                Vector<Vector<Object>> data = new Vector<>();
                while (rs.next()) {
                    Vector<Object> row = new Vector<>();
                    row.add(rs.getString("constructor_name"));
                    row.add(rs.getDouble("total_points"));
                    data.add(row);
                }

                // Actualizar modelo de la tabla en el hilo de eventos de Swing
                SwingUtilities.invokeLater(() -> {
                    tableModel.setDataVector(data, columnNames);
```

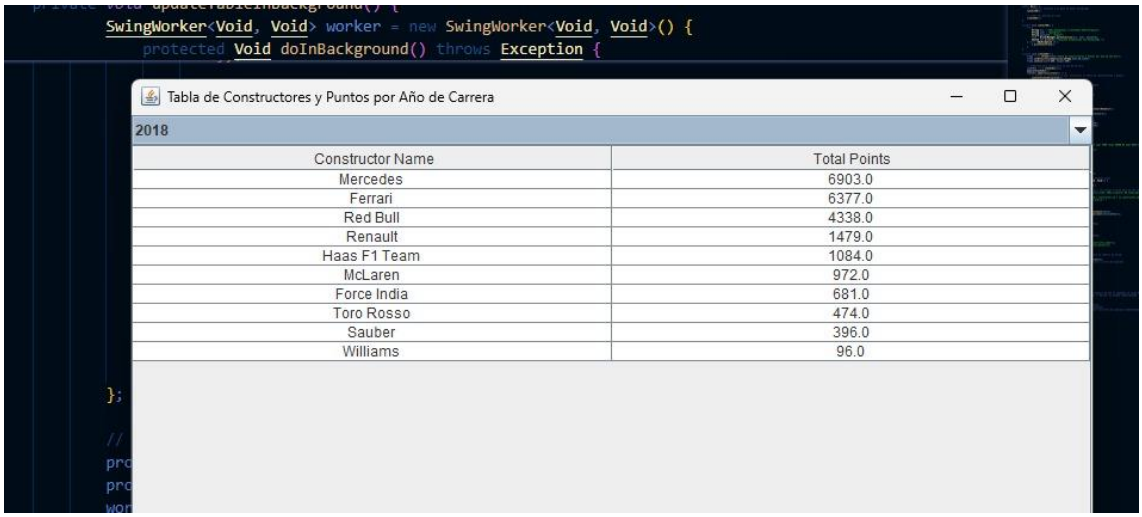
```
                    progressBar.setValue(100); // Completa la barra de progreso
                });

                rs.close();
                pstmt.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return null;
        }

        @Override
        protected void done() {
            // Aquí puedes realizar acciones adicionales después de que se completa la carga de
datos
            progressBar.setIndeterminate(false); // Detener el estado indeterminado
        }
    };

    // Iniciar el SwingWorker y mostrar la barra de progreso
    progressBar.setValue(0); // Reiniciar la barra de progreso
    progressBar.setIndeterminate(true); // Mostrar una barra de progreso indeterminada
    worker.execute();
  }
}

public static void main(String[] args) {
  SwingUtilities.invokeLater(Main::new);
}
}
```


SwingWorker<Void, Void> worker = new SwingWorker<Void, Void>() {
protected Void doInBackground() throws Exception {
Tabla de Constructores y Puntos por Año de Carrera
2018
Constructor Name
Total Points
Mercedes
6903.0
Ferrari
6377.0
Red Bull
4338.0
Renault
1479.0
Haas F1 Team
1084.0
McLaren
972.0
Force India
681.0
Toro Rosso
474.0
Sauber
396.0
Williams
96.0
};