LAND AREA CALCULATION AND SURVEY SYSTEM

CODING:

POINT:

```
public class Point {
    private double x;
    private double y;

    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }

    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }

    @Override
    public String toString() {
        return "(" + x + ", " + y + ")";
    }
}
```

LAND PARCEL:

```
import java.util.List;

public class LandParcel {
    private List<Point> vertices;

    public LandParcel(List<Point> vertices) {
        this.vertices = vertices;
    }

    public double calculateArea() {
        double area = 0.0;
        int n = vertices.size();
        for (int i = 0; i < n; i++) {
            Point p1 = vertices.get(i);
            Point p2 = vertices.get((i + 1) % n);
            area += p1.getX() * p2.getY() - p2.getX() * p1.getY();
        }
```

```
        return Math.abs(area) / 2.0;
    }

    public List<Point> getVertices() {
        return vertices;
    }

    @Override
    public String toString() {
        return "LandParcel with vertices: " + vertices.toString();
    }
}
```

SURVEY SYSTEM:

```
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class SurveySystem {
    private List<LandParcel> parcels;

    public SurveySystem() {
        parcels = new ArrayList<>();
    }

    public void addParcel(LandParcel parcel) {
        parcels.add(parcel);
        saveParcelToDatabase(parcel);
    }

    public double getTotalLandArea() {
        double totalArea = 0.0;
        for (LandParcel parcel : parcels) {
            totalArea += parcel.calculateArea();
        }
        return totalArea;
    }

    public List<LandParcel> getParcels() {
        return parcels;
    }

    private void saveParcelToDatabase(LandParcel parcel) {
        try (Connection conn = DatabaseConnection.getConnection()) {
```

```
            // Insert parcel data
            String insertParcelSQL = "INSERT INTO LandParcel
(parcel_name, area) VALUES (?, ?)";
            try (PreparedStatement parcelStmt =
conn.prepareStatement(insertParcelSQL,
Statement.RETURN_GENERATED_KEYS)) {
                parcelStmt.setString(1, "Parcel " + (parcels.size()));
                parcelStmt.setDouble(2, parcel.calculateArea());
                parcelStmt.executeUpdate();

                // Get the generated parcel ID
                ResultSet keys = parcelStmt.getGeneratedKeys();
                int parcelId = keys.next() ? keys.getInt(1) : -1;

                // Insert each point for this parcel
                String insertPointSQL = "INSERT INTO Point (parcel_id, x, y)
VALUES (?, ?, ?)";
                try (PreparedStatement pointStmt =
conn.prepareStatement(insertPointSQL)) {
                    for (Point point : parcel.getVertices()) {
                        pointStmt.setInt(1, parcelId);
                        pointStmt.setDouble(2, point.getX());
                        pointStmt.setDouble(3, point.getY());
                        pointStmt.executeUpdate();
                    }
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    // Method to load parcels from the database
    public void loadParcelsFromDatabase() {
        parcels.clear();
        try (Connection conn = DatabaseConnection.getConnection()) {
            String selectParcelSQL = "SELECT * FROM LandParcel";
            try (Statement stmt = conn.createStatement(); ResultSet rs =
stmt.executeQuery(selectParcelSQL)) {
                while (rs.next()) {
                    int parcelId = rs.getInt("id");
                    double area = rs.getDouble("area");
```

```
                // Load points for each parcel
                String selectPointsSQL = "SELECT * FROM Point WHERE
parcel_id = ?";
                try (PreparedStatement pointStmt =
conn.prepareStatement(selectPointsSQL)) {
                    pointStmt.setInt(1, parcelId);
                    ResultSet pointsRs = pointStmt.executeQuery();

                    List<Point> points = new ArrayList<>();
                    while (pointsRs.next()) {
                        double x = pointsRs.getDouble("x");
                        double y = pointsRs.getDouble("y");
                        points.add(new Point(x, y));
                    }

                    // Create parcel and add to list
                    parcels.add(new LandParcel(points));
                }
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
  }
}
```

MAIN:

```
import javax.swing.*; // For JFrame, JPanel, JLabel, JButton,
JOptionPane, and SwingUtilities
import java.awt.*;    // For Color, BorderLayout, Graphics
import java.awt.event.*; // For MouseAdapter and MouseEvent
import java.util.ArrayList;

public class Main extends JFrame {
   private ArrayList<Point> points;
   private SurveySystem surveySystem;
   private JLabel areaLabel;
   private JPanel canvas;

   public Main() {
      points = new ArrayList<>();
      surveySystem = new SurveySystem();
      surveySystem.loadParcelsFromDatabase(); // Load parcels from the
database on startup
```

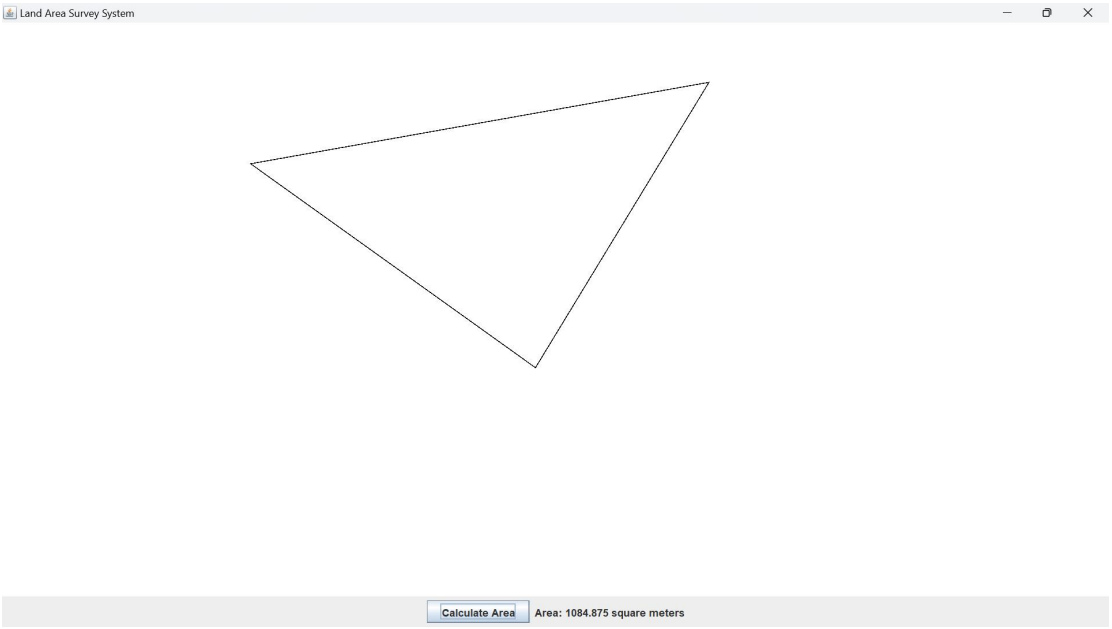
```
        setTitle("Land Area Survey System");
        setSize(600, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Canvas for drawing parcels
        canvas = new JPanel() {
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                g.setColor(Color.BLACK);
                for (int i = 0; i < points.size(); i++) {
                    Point p1 = points.get(i);
                    Point p2 = points.get((i + 1) % points.size());
                    g.drawLine((int)p1.getX(),    (int)p1.getY(),    (int)p2.getX(),
(int)p2.getY());
                }
            }
        };
        canvas.setBackground(Color.WHITE);
        canvas.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                points.add(new Point(e.getX(), e.getY()));
                canvas.repaint();
            }
        });

        JButton calculateAreaButton = new JButton("Calculate Area");
        calculateAreaButton.addActionListener(e                                    ->
calculateAndDisplayArea());

        areaLabel = new JLabel("Area: ");

        add(canvas, BorderLayout.CENTER);
        JPanel controlPanel = new JPanel();
        controlPanel.add(calculateAreaButton);
        controlPanel.add(areaLabel);
        add(controlPanel, BorderLayout.SOUTH);
    }

    private static final double SCALE = 0.5; // 1 pixel = 0.5 meters
```

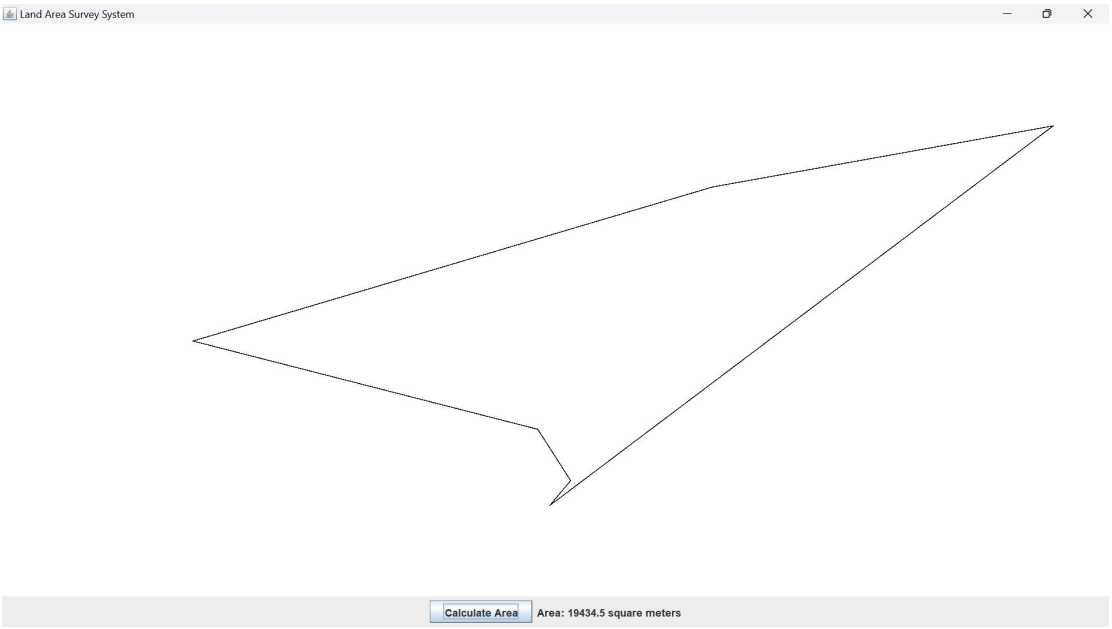
```
private void calculateAndDisplayArea() {
    if (points.size() >= 3) {
        LandParcel parcel = new LandParcel(new ArrayList<>(points));
        surveySystem.addParcel(parcel);

        // Calculate scaled area in square meters
        double area = parcel.calculateArea() * SCALE * SCALE;
        areaLabel.setText("Area: " + area + " square meters"); // Display area with units

        points.clear();
        canvas.repaint();
    } else {
        JOptionPane.showMessageDialog(this, "Need at least 3 points to form a land parcel.");
    }
}


    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            Main frame = new Main();
            frame.setVisible(true);
        });
    }
}
```

DATABASE:

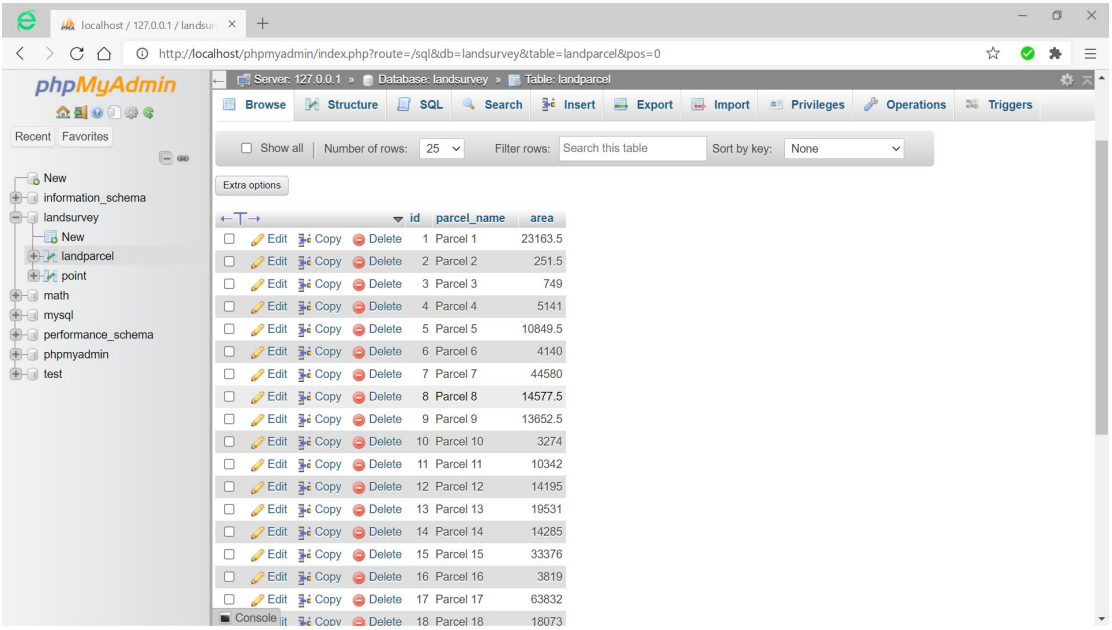
```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/LandSurvey";
    private static final String USER = "root";
    private static final String PASSWORD = "";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

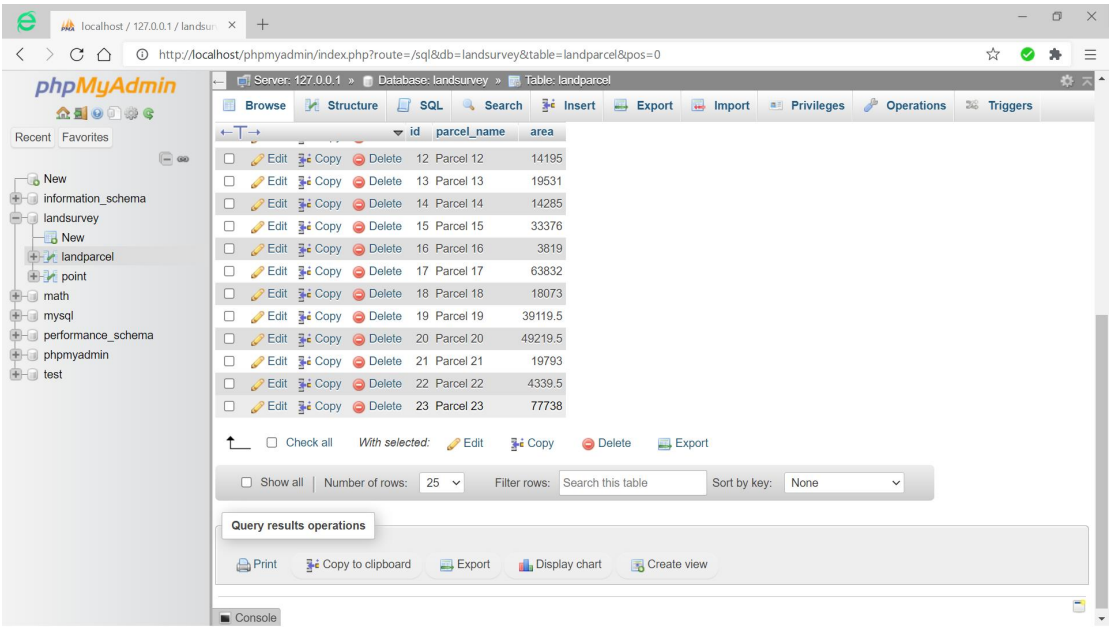
OUTPUT:


Land Area Survey System
Calculate Area
Area: 1084.875 square meters


Land Area Survey System
Calculate Area
Area: 19434.5 square meters

DATABASE:


localhost / 127.0.0.1 / landsur
http://localhost/phpmyadmin/index.php?route=/sql&db=landsurvey&table=landparcel&pos=0
phpMyAdmin
Recent Favorites
New
information_schema
landsurvey
New
landparcel
point
math
mysql
performance_schema
phpmyadmin
test
Server: 127.0.0.1 » Database: landsurvey » Table: landparcel
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options
id parcel_name area
Edit Copy Delete 1 Parcel 1 23163.5
Edit Copy Delete 2 Parcel 2 251.5
Edit Copy Delete 3 Parcel 3 749
Edit Copy Delete 4 Parcel 4 5141
Edit Copy Delete 5 Parcel 5 10849.5
Edit Copy Delete 6 Parcel 6 4140
Edit Copy Delete 7 Parcel 7 44580
Edit Copy Delete 8 Parcel 8 14577.5
Edit Copy Delete 9 Parcel 9 13652.5
Edit Copy Delete 10 Parcel 10 3274
Edit Copy Delete 11 Parcel 11 10342
Edit Copy Delete 12 Parcel 12 14195
Edit Copy Delete 13 Parcel 13 19531
Edit Copy Delete 14 Parcel 14 14285
Edit Copy Delete 15 Parcel 15 33376
Edit Copy Delete 16 Parcel 16 3819
Edit Copy Delete 17 Parcel 17 63832
Console
Edit Copy Delete 18 Parcel 18 18073


localhost / 127.0.0.1 / landsur
http://localhost/phpmyadmin/index.php?route=/sql&db=landsurvey&table=landparcel&pos=0
phpMyAdmin
Recent Favorites
New
information_schema
landsurvey
New
landparcel
point
math
mysql
performance_schema
phpmyadmin
test
Server: 127.0.0.1 » Database: landsurvey » Table: landparcel
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
id parcel_name area
Edit Copy Delete 12 Parcel 12 14195
Edit Copy Delete 13 Parcel 13 19531
Edit Copy Delete 14 Parcel 14 14285
Edit Copy Delete 15 Parcel 15 33376
Edit Copy Delete 16 Parcel 16 3819
Edit Copy Delete 17 Parcel 17 63832
Edit Copy Delete 18 Parcel 18 18073
Edit Copy Delete 19 Parcel 19 39119.5
Edit Copy Delete 20 Parcel 20 49219.5
Edit Copy Delete 21 Parcel 21 19793
Edit Copy Delete 22 Parcel 22 4339.5
Edit Copy Delete 23 Parcel 23 77738
Check all With selected: Edit Copy Delete Export
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Query results operations
Print Copy to clipboard Export Display chart Create view
Console