```java
import javax.swing.*;

/**
 * ME 35401 - Spring Calculator
 *
 * This program receives inputs of end type, material, wire diameter, outer diameter, free length, and solid length.
 * It then outputs the pitch, number of total coils, number of active coils, spring rate, force needed to compress
 * to solid length, and factor of safety when the spring is compressed to this length. Simple GUI elements are
 * used for input.
 *
 * @author Brendan Whittemore, Lab Section 006
 *
 * @version April 14, 2022
 *
 */

public class SpringCalculator {
    private static final String[] endTypeOptions =
            {"Plain", "Plain and ground", "Squared or closed", "Squared and ground"};

    private static final String[] materialTypeOptions =
            {"Music wire (ASTM No. A228)", "Hard-drawn wire (ASTM No. A227)", "Chrome-vanadium wire (ASTM No. A232)",
                    "Chrome-silicon wire (ASTM No. A401)", "302 stainless wire (ASTM No. A313)",
                    "Phosphor-bronze wire (ASTM No. B159)"};

    private static final String[] peenTypeOptions = {"Peened", "Unpeened"};

    public static void main(String[] args) {
        // Receive input from the user with a simple GUI
        showWelcomeMessageDialog();
        String endType = showEndTypeInputDialog();
        String material = showMaterialTypeInputDialog();
        boolean peened = showPeenTypeInputDialog();
        double wireDiameter = showWireDiameterInputDialog();
        double coilDiameter = showOuterDiameterInputDialog() - wireDiameter;
        double freeLength = showFreeLengthInputDialog();
        double solidLength = showSolidLengthInputDialog();
        double minForce = showMinForceInputDialog();
        double maxForce = showMaxForceInputDialog();

        // Calculate material characteristics
        double[] materialInfo = calculateMaterialInfo(material, wireDiameter);
        double ultimateTensileStrength = materialInfo[0];
        double yieldStrength = materialInfo[1];
        double yieldStrengthShear = materialInfo[2];
        double E = materialInfo[3];
        double G = materialInfo[4];

        // Calculate dimensional characteristics
        double[] dimensionalInfo = calculateDimensionalInfo(endType, wireDiameter, freeLength, solidLength);
        double totalCoils = dimensionalInfo[0];
        double activeCoils = dimensionalInfo[1];
        double pitch = dimensionalInfo[2];

        // Calculate spring rate
        double springRate = calculateSpringRate(wireDiameter, G, coilDiameter, activeCoils);

        // Calculate force to compress to solid
```

```java
length and factor of safety for static yielding at this point
        double forceToSolid = calculateForceToSolid(springRate, freeLength, solidLength);
        double factorOfSafetyAtForceToSolid = calculateFactorOfSafetyAtForceToSolid(forceToSolid, yieldStrengthShear,
                coilDiameter, wireDiameter);

        // Calculate the static load factor of safety or cyclic load factor of safety for infinite life
        if (Math.abs(maxForce - minForce) < 0.000001d) {
            double factorOfSafety = calculateStaticFactorOfSafety(minForce, yieldStrengthShear, coilDiameter,
                    wireDiameter);
            showStaticFinalValuesMessageDialog(pitch, totalCoils, activeCoils, springRate, forceToSolid,
                    factorOfSafetyAtForceToSolid, factorOfSafety);
        } else {
            double factorOfSafety = calculateFatigueFactorOfSafety(coilDiameter, wireDiameter, minForce, maxForce,
                    ultimateTensileStrength, peened);
            showFatigueFinalValuesMessageDialog(pitch, totalCoils, activeCoils, springRate, forceToSolid,
                    factorOfSafetyAtForceToSolid, factorOfSafety);
        }
    }

    /* Calculates Sut (psi), Sy (psi), Sys (psi), E (psi), and G(psi)) */
    public static double[] calculateMaterialInfo(String material, double wireDiameter) {
        double A = 0;
        double m = 0;
        double ultimateTensileStrength = 0;
        double yieldStrength = 0;
        double yieldStrengthShear = 0;
        double E = 0;
        double G = 0;

        switch (material) {
            case "Music wire (ASTM No. A228)" -> {
                A = 201;
                m = 0.145;
                ultimateTensileStrength = (A / (Math.pow(wireDiameter, m))) * 1000;
                yieldStrength = 0.65 * ultimateTensileStrength;
                yieldStrengthShear = 0.45 * ultimateTensileStrength;
                if (wireDiameter <= 0.032) {
                    E = 29.5 * 1000000;
                    G = 12.0 * 1000000;
                } else if (wireDiameter > 0.032 && wireDiameter <= 0.063) {
                    E = 29.0 * 1000000;
                    G = 11.85 * 1000000;
                } else if (wireDiameter > 0.063 && wireDiameter <= 0.125) {
                    E = 28.5 * 1000000;
                    G = 11.75 * 1000000;
                } else if (wireDiameter > 0.125) {
                    E = 28.0 * 1000000;
                    G = 11.6 * 1000000;
                }
            }
            case "Hard-drawn wire (ASTM No. A227)" -> {
                A = 140;
                m = 0.190;
                ultimateTensileStrength = (A / (Math.pow(wireDiameter, m))) * 1000;
                yieldStrength = 0.6 * ultimateTensileStrength;
                yieldStrengthShear = 0.45 * ultimateTensileStrength;
                if (wireDiameter <= 0.032) {
                    E = 28.8 * 1000000;
                    G = 11.7 * 1000000;
                } else if (wireDiameter > 0.032 && wireDiameter <= 0.063) {
                    E = 28.7 * 1000000;
                    G = 11.6 * 1000000;
                } else if (wireDiameter > 0.063 && wireDiameter <= 0.125) {
                    E = 28.6 * 1000000;
                    G = 11.5 * 1000000;
                } else if (wireDiameter > 0.125) {
                    E = 28.5 * 1000000;
                    G = 11.4 * 1000000;
                }
            }
            case "Chrome-vanadium wire (ASTM No. A232)" -> {
                A = 169;
                m = 0.168;
                ultimateTensileStrength = (A / (Math.pow(wireDiameter, m))) * 1000;
                yieldStrength = 0.88 * ultimateTensileStrength;
                yieldStrengthShear = 0.65 * ultimateTensileStrength;
                E = 29.5 * 1000000;
                G = 11.2 * 1000000;
            }
            case "Chrome-silicon wire (ASTM No. A401)" -> {
                A = 202;
                m = 0.108;
                ultimateTensileStrength = (A / (Math.pow(wireDiameter, m))) * 1000;
                yieldStrength = 0.85 * ultimateTensileStrength;
                yieldStrengthShear = 0.65 * ultimateTensileStrength;
                E = 29.5 * 1000000;
                G = 11.2 * 1000000;
            }
            case "302 stainless wire (ASTM No. A313)" -> {
                if (wireDiameter > 0.013 && wireDiameter <= 0.1) {
                    A = 169;
                    m = 0.146;
                } else if (wireDiameter > 0.1 && wireDiameter <= 0.2) {
                    A = 128;
                    m = 0.263;
                } else if (wireDiameter > 0.2 && wireDiameter <= 0.4) {
                    A = 90;
                    m = 0.478;
                }
                ultimateTensileStrength = (A / (Math.pow(wireDiameter, m))) * 1000;
                yieldStrength = 0.65 * ultimateTensileStrength;
                yieldStrengthShear = 0.45 * ultimateTensileStrength;
                E = 28.0 * 1000000;
                G = 10.0 * 1000000;
            }
            case "Phosphor-bronze wire (ASTM No. B159)" -> {
                if (wireDiameter > 0.004 && wireDiameter <= 0.022) {
                    A = 145;
                    m = 0;
                } else if (wireDiameter > 0.022 && wireDiameter <= 0.075) {
                    A = 121;
                    m = 0.028;
                } else if (wireDiameter > 0.075 && wireDiameter <= 0.3) {
                    A = 110;
                    m = 0.064;
                }
                ultimateTensileStrength = (A / (Math.pow(wireDiameter, m))) * 1000;
                yieldStrength = 0.75 * ultimateTensileStrength;
                yieldStrengthShear = 0.45 * ultimateTensileStrength;
                E = 15.0 * 1000000;
                G = 6.0 * 1000000;
            }
        }

        return new double[] {ultimateTensileStrength, yieldStrength, yieldStrengthShear, E, G};
    }

    /* Calculates Nt (coils), Na (coils), and p (in)) */
    public static double[] calculateDimensionalInfo(String endType, double wireDiameter,
                                                    double freeLength, double solidLength) {
        double totalCoils = 0;
        double activeCoils = 0;
        double pitch = 0;

        switch (endType) {
            case "Plain" -> {
                totalCoils = (solidLength / wireDiameter) - 1;
                activeCoils = totalCoils;
                pitch = (freeLength - wireDiameter) / activeCoils;
            }
            case "Plain and ground" -> {
                totalCoils = solidLength / wireDiameter;
                activeCoils = totalCoils - 1;
                pitch = freeLength / (activeCoils + 1);
            }
            case "Squared or closed" -> {
                totalCoils = (solidLength / wireDiameter) - 1;
                activeCoils = totalCoils - 2;
                pitch = (freeLength - (3 * wireDiameter)) / activeCoils;
            }
            case "Squared and ground" -> {
                totalCoils = solidLength / wireDiameter;
                activeCoils = totalCoils - 2;
                pitch = (freeLength - (2 * wireDiameter)) / activeCoils;
            }
        }

        return new double[] {totalCoils, activeCoils, pitch};
    }

    /* Calculates k (lbf/in) */
    public static double calculateSpringRate(double wireDiameter, double G,
                                             double coilDiameter, double activeCoils) {
        return (Math.pow(wireDiameter, 4) * G) / (8 * Math.pow(coilDiameter, 3) * activeCoils);
    }

    /* Calculates F (lbf) with a deflection of (Lo - Ls) */
    public static double calculateForceToSolid(double springRate, double freeLength, double solidLength) {
        return springRate * (freeLength - solidLength);
    }

    /* Calculates n at solid length */
    public static double calculateFactorOfSafetyAtForceToSolid(double forceToSolid, double yieldStrengthShear,
                                                               double coilDiameter, double wireDiameter
```

```java
) {
        double springIndex = coilDiameter / wireDiameter;
        double bergstrasserFactor = ((4 * springIndex) + 2) / ((4 * springIndex) - 3);
        double shearStress = bergstrasserFactor
                * ((8 * forceToSolid * coilDiameter) / (Math.PI * Math.pow(wireDiameter, 3)));

        return yieldStrengthShear / shearStress;
    }

    /* Calculates the factor of safety for a static load */
    public static double calculateStaticFactorOfSafety(double minForce, double yieldStrengthShear,
                                                       double coilDiameter, double wireDiameter) {
        double springIndex = coilDiameter / wireDiameter;
        double bergstrasserFactor = ((4 * springIndex) + 2) / ((4 * springIndex) - 3);
        double shearStress = bergstrasserFactor
                * ((8 * minForce * coilDiameter) / (Math.PI * Math.pow(wireDiameter, 3)));

        return yieldStrengthShear / shearStress;
    }

    /* Calculates the factor of safety for a cyclic load */
    public static double calculateFatigueFactorOfSafety(double coilDiameter, double wireDiameter, double minForce,
                                                        double maxForce, double ultimateTensileStrength,
                                                        boolean peened) {
        double springIndex = coilDiameter / wireDiameter;
        double bergstrasserFactor = ((4 * springIndex) + 2) / ((4 * springIndex) - 3);
        double forceAmplitude = (maxForce - minForce) / 2;
        double forceMean = (maxForce + minForce) / 2;
        double shearStressAmplitude = bergstrasserFactor
                * ((8 * forceAmplitude * coilDiameter) / (Math.PI * Math.pow(wireDiameter, 3)));
        double shearStressMean = bergstrasserFactor
                * ((8 * forceMean * coilDiameter) / (Math.PI * Math.pow(wireDiameter, 3)));
        double Ssu = 0.67 * ultimateTensileStrength;
        double Sse;
        if (peened) {
            Sse = (57.5 * 1000) / (1 - ((77.5 * 1000) / (Ssu)));
        } else {
            Sse = (35 * 1000) / (1 - ((55 * 1000) / (Ssu)));
        }
        return 1 / ((shearStressAmplitude / Sse) + (shearStressMean / Ssu));
    }

    /**
     * All methods below this point have to do with displaying the GUI elements only (no calculations)
     */

    /* Shows a welcome method dialog */
    public static void showWelcomeMessageDialog() {
        JOptionPane.showMessageDialog(null, "Welcome to the Spring Calculator!",
                "Spring Calculator", JOptionPane.INFORMATION_MESSAGE);
    }

    /* Gets end type from the user */
    public static String showEndTypeInputDialog() {
        String endType;

        do {
            endType = (String) JOptionPane.showInputDialog(null, "Select your end type",
                    "Spring Calculator", JOptionPane.QUESTION_MESSAGE, null, endTypeOptions,
                    endTypeOptions[0]);
            if (endType == null) {
                JOptionPane.showMessageDialog(null, "Invalid choice",
                        "Spring Calculator", JOptionPane.ERROR_MESSAGE);
            }
        } while (endType == null);

        return endType;
    }

    /* Gets material type from the user */
    public static String showMaterialTypeInputDialog() {
        String materialType;

        do {
            materialType = (String) JOptionPane.showInputDialog(null, "Select your material type",
                    "Spring Calculator", JOptionPane.QUESTION_MESSAGE, null, materialTypeOptions,
                    materialTypeOptions[0]);
            if (materialType == null) {
                JOptionPane.showMessageDialog(null, "Invalid choice",
                        "Spring Calculator", JOptionPane.ERROR_MESSAGE);
            }
        } while (materialType == null);

        return materialType;
    }

    /* Gets peen type from the user */
    public static boolean showPeenTypeInputDialog() {
        String peenType;

        do {
            peenType = (String) JOptionPane.showInputDialog(null, "Select your peen type",
                    "Spring Calculator", JOptionPane.QUESTION_MESSAGE, null, peenTypeOptions,
                    peenTypeOptions[0]);
            if (peenType == null) {
                JOptionPane.showMessageDialog(null, "Invalid choice",
                        "Spring Calculator", JOptionPane.ERROR_MESSAGE);
            }
        } while (peenType == null);

        return peenType.equals("Peened");
    }

    /* Gets wire diameter from the user */
    public static double showWireDiameterInputDialog() {
        String wireDiameter;

        do {
            wireDiameter = JOptionPane.showInputDialog(null, "Enter the wire diameter (in)",
                    "Spring Calculator", JOptionPane.QUESTION_MESSAGE);

            try {
                if (Double.parseDouble(wireDiameter) < 0) {
                    throw new NumberFormatException();
                }
            } catch (Exception e) {
                wireDiameter = null;
            }

            if ((wireDiameter == null) || (wireDiameter.isEmpty())) {
                JOptionPane.showMessageDialog(null, "Invalid input",
                        "Spring Calculator", JOptionPane.ERROR_MESSAGE);
            }
        } while ((wireDiameter == null) || (wireDiameter.isEmpty()));

        return Double.parseDouble(wireDiameter);
    }

    /* Gets outer diameter from the user */
    public static double showOuterDiameterInputDialog() {
        String outerDiameter;

        do {
            outerDiameter = JOptionPane.showInputDialog(null, "Enter the outer diameter (in)",
                    "Spring Calculator", JOptionPane.QUESTION_MESSAGE);

            try {
                if (Double.parseDouble(outerDiameter) < 0) {
                    throw new NumberFormatException();
                }
            } catch (Exception e) {
                outerDiameter = null;
            }

            if ((outerDiameter == null) || (outerDiameter.isEmpty())) {
                JOptionPane.showMessageDialog(null, "Invalid input",
                        "Spring Calculator", JOptionPane.ERROR_MESSAGE);
            }
        } while ((outerDiameter == null) || (outerDiameter.isEmpty()));

        return Double.parseDouble(outerDiameter);
    }

    /* Gets free Length from the user */
    public static double showFreeLengthInputDialog() {
        String freeLength;

        do {
            freeLength = JOptionPane.showInputDialog(null, "Enter the free length (in)",
                    "Spring Calculator", JOptionPane.QUESTION_MESSAGE);

            try {
                if (Double.parseDouble(freeLength) < 0) {
                    throw new NumberFormatException();
                }
            } catch (Exception e) {
                freeLength = null;
            }

            if ((freeLength == null) || (freeLength.isEmpty())) {
                JOptionPane.showMessageDialog(null, "Invalid input",
                        "Spring Calculator", JOptionPane.ERROR_MESSAGE);
            }
        } while ((freeLength == null) || (freeLength.isEmpty()));

        return Double.parseDouble(freeLength);
    }

    /* Gets solid length from the user */
    public static double showSolidLengthInputDialog() {
        String solidLength;

        do {
            solidLength = JOptionPane.showInputDialog(null, "Enter the solid length (in)",
                    "Spring Calculator", JOptionPane.QUESTION_MESSAGE);

            try {
                if (Double.parseDouble(solidLength) < 0) {
                    throw new NumberFormatException();
                }
            } catch (Exception e) {
                solidLength = null;
            }

            if ((solidLength == null) || (solidLength.isEmpty())) {
                JOptionPane.showMessageDialog(null, "Invalid input",
                        "Spring Calculator", JOptionPane.ERROR_MESSAGE);
            }
        } while ((solidLength == null) || (solidLength.isEmpty()));

        return Double.parseDouble(solidLength);
    }

    /* Gets min force from the user */
    public static double showMinForceInputDialog() {
        String minForce;

        do {
            minForce = JOptionPane.showInputDialog(null, "Enter the min force (lbf)",
                    "Spring Calculator", JOptionPane.QUESTION_MESSAGE);

            try {
                Double.parseDouble(minForce);
            } catch (Exception e) {
                minForce = null;
            }

            if ((minForce == null) || (minForce.isEmpty())) {
                JOptionPane.showMessageDialog(null, "Invalid input",
                        "Spring Calculator", JOptionPane.ERROR_MESSAGE);
            }
        } while ((minForce == null) || (minForce.isEmpty()));

        return Double.parseDouble(minForce);
    }

    /* Gets max force from the user */
    public static double showMaxForceInputDialog() {
        String maxForce;

        do {
            maxForce = JOptionPane.showInputDialog(null, "Enter the max force (lbf)",
                    "Spring Calculator", JOptionPane.QUESTION_MESSAGE);

            try {
                Double.parseDouble(maxForce);
            } catch (Exception e) {
                maxForce = null;
            }

            if ((maxForce == null) || (maxForce.isEmpty())) {
                JOptionPane.showMessageDialog(null, "Invalid input",
                        "Spring Calculator", JOptionPane.ERROR_MESSAGE);
            }
        } while ((maxForce == null) || (maxForce.isEmpty()));

        return Double.parseDouble(maxForce);
    }

    /* Shows the final values dialog for the static case */
    public static void showStaticFinalValuesMessageDialog(double pitch,
                                                          double totalCoils,
                                                          double activeCoils,
                                                          double springRate,
                                                          double forceToSolid,
                                                          double factorOfSafetyAtForceToSolid,
                                                          double factorOfSafety) {
        String message = String.format("Spring Values\n\n" +
                    "Pitch: %.3f in\n" +
                    "Total Coils: %.3f coils\n" +
                    "Active Coils: %.3f coils\n" +
                    "Spring Rate: %.3f lbf/in\n\n" +
                    "Force to Compress to Solid Length: %.3f lbf\n" +
                    "Factor of Safety for Static Yielding at Solid Length: %.1f\n\n" +
                    "Factor of Safety for Static
```

```java
Load: %.1f", pitch, totalCoils, activeCoils, springRate,
                    forceToSolid, factorOfSafetyAtForceToSolid, factorOfSafety);

        JOptionPane.showMessageDialog(null, message, "Spring Calculator",
                JOptionPane.INFORMATION_MESSAGE);
    }

    /* Shows the final values dialog for the cyclic case */
    public static void showFatigueFinalValuesMessageDialog(double pitch,
                                                            double totalCoils,
                                                            double activeCoils,
                                                            double springRate,
                                                            double forceToSolid,
                                                            double factorOfSafetyAtForceToSolid,
                                                            double factorOfSafety) {
        String message = String.format("Spring Values\n\n" +
                                "Pitch: %.3f in\n" +
                                "Total Coils: %.3f coils\n" +
                                "Active Coils: %.3f coils\n" +
                                "Spring Rate: %.3f lbf/in\n\n" +
                                "Force to Compress to Solid Length: %.3f lbf\n" +
                                "Factor of Safety for Static Yielding at Solid Length: %.1f\n\n" +
                                "Factor of Safety for Infinite Life for Cyclic Load: %.1f", pitch,
totalCoils, activeCoils,
                        springRate, forceToSolid, factorOfSafetyAtForceToSolid, factorOfSafety);

        JOptionPane.showMessageDialog(null, message, "Spring Calculator",
                JOptionPane.INFORMATION_MESSAGE);
    }
}

```