

```

1 import javax.swing.*;
2
3 /**
4  * ME 35401 - Spring Calculator
5  *
6  * This program receives inputs of end type, material
7  * , wire diameter, outer diameter, free length, and
8  * solid length.
9  * It then outputs the pitch, number of total coils,
10 * number of active coils, spring rate, force needed to
11 * compress
12 * to solid length, and factor of safety when the
13 * spring is compressed to this length. Simple GUI
14 * elements are
15 * used for input.
16 *
17 * @author Brendan Whittemore, Lab Section 006
18 *
19 * @version April 14, 2022
20 *
21 */
22
23 public class SpringCalculator {
24     private static final String[] endTypeOptions =
25         {"Plain", "Plain and ground", "Squared or
26         closed", "Squared and ground"};
27
28     private static final String[] materialTypeOptions
29         =
30         {"Music wire (ASTM No. A228)", "Hard-
31         drawn wire (ASTM No. A227)", "Chrome-vanadium wire (
32         ASTM No. A232)",
33         "Chrome-silicon wire (ASTM No.
34         A401)", "302 stainless wire (ASTM No. A313)",
35         "Phosphor-bronze wire (ASTM No.
36         B159)"};
37
38     private static final String[] peenTypeOptions = {
39         "Peened", "Unpeened"};
40
41     public static void main(String[] args) {

```

```

29      // Receive input from the user with a simple
    GUI
30      showWelcomeMessageDialog();
31      String endType = showEndTypeInputDialog();
32      String material = showMaterialTypeInputDialog
    ();
33      boolean peened = showPeenTypeInputDialog();
34      double wireDiameter =
    showWireDiameterInputDialog();
35      double coilDiameter =
    showOuterDiameterInputDialog() - wireDiameter;
36      double freeLength = showFreeLengthInputDialog
    ();
37      double solidLength =
    showSolidLengthInputDialog();
38      double minForce = showMinForceInputDialog();
39      double maxForce = showMaxForceInputDialog();
40
41      // Calculate material characteristics
42      double[] materialInfo = calculateMaterialInfo
    (material, wireDiameter);
43      double ultimateTensileStrength = materialInfo
    [0];
44      double yieldStrength = materialInfo[1];
45      double yieldStrengthShear = materialInfo[2];
46      double E = materialInfo[3];
47      double G = materialInfo[4];
48
49      // Calculate dimensional characteristics
50      double[] dimensionalInfo =
    calculateDimensionalInfo(endType, wireDiameter,
    freeLength, solidLength);
51      double totalCoils = dimensionalInfo[0];
52      double activeCoils = dimensionalInfo[1];
53      double pitch = dimensionalInfo[2];
54
55      // Calculate spring rate
56      double springRate = calculateSpringRate(
    wireDiameter, G, coilDiameter, activeCoils);
57
58      // Calculate force to compress to solid

```

```

58 length and factor of safety for static yielding at
   this point
59     double forceToSolid = calculateForceToSolid(
        springRate, freeLength, solidLength);
60     double factorOfSafetyAtForceToSolid =
        calculateFactorOfSafetyAtForceToSolid(forceToSolid,
        yieldStrengthShear,
61         coilDiameter, wireDiameter);
62
63     // Calculate the static load factor of safety
   or cyclic load factor of safety for infinite life
64     if (Math.abs(maxForce - minForce) < 0.000001d
        ) {
65         double factorOfSafety =
        calculateStaticFactorOfSafety(minForce,
        yieldStrengthShear, coilDiameter,
66         wireDiameter);
67         showStaticFinalValuesMessageDialog(pitch
        , totalCoils, activeCoils, springRate, forceToSolid,
68         factorOfSafetyAtForceToSolid,
        factorOfSafety);
69     } else {
70         double factorOfSafety =
        calculateFatigueFactorOfSafety(coilDiameter,
        wireDiameter, minForce, maxForce,
71         ultimateTensileStrength, peened);
72         showFatigueFinalValuesMessageDialog(pitch
        , totalCoils, activeCoils, springRate, forceToSolid,
73         factorOfSafetyAtForceToSolid,
        factorOfSafety);
74     }
75 }
76
77 /* Calculates Sut (psi), Sy (psi), Sys (psi), E (
   psi), and G(psi)) */
78 public static double[] calculateMaterialInfo(
    String material, double wireDiameter) {
79     double A = 0;
80     double m = 0;
81     double ultimateTensileStrength = 0;
82     double yieldStrength = 0;

```

```

83         double yieldStrengthShear = 0;
84         double E = 0;
85         double G = 0;
86
87         switch (material) {
88             case "Music wire (ASTM No. A228)" -> {
89                 A = 201;
90                 m = 0.145;
91                 ultimateTensileStrength = (A / (Math
92 .pow(wireDiameter, m))) * 1000;
93                 yieldStrength = 0.65 *
94 ultimateTensileStrength;
95                 yieldStrengthShear = 0.45 *
96 ultimateTensileStrength;
97                 if (wireDiameter <= 0.032) {
98                     E = 29.5 * 1000000;
99                     G = 12.0 * 1000000;
100                } else if (wireDiameter > 0.032 &&
101 wireDiameter <= 0.063) {
102                    E = 29.0 * 1000000;
103                    G = 11.85 * 1000000;
104                } else if (wireDiameter > 0.063 &&
105 wireDiameter <= 0.125) {
106                    E = 28.5 * 1000000;
107                    G = 11.75 * 1000000;
108                } else if (wireDiameter > 0.125) {
109                    E = 28.0 * 1000000;
110                    G = 11.6 * 1000000;
111                }
112            }
113            case "Hard-drawn wire (ASTM No. A227)"
114 -> {
115                 A = 140;
116                 m = 0.190;
117                 ultimateTensileStrength = (A / (Math
118 .pow(wireDiameter, m))) * 1000;
119                 yieldStrength = 0.6 *
120 ultimateTensileStrength;
121                 yieldStrengthShear = 0.45 *
122 ultimateTensileStrength;
123                 if (wireDiameter <= 0.032) {

```

```

115         E = 28.8 * 1000000;
116         G = 11.7 * 1000000;
117     } else if (wireDiameter > 0.032 &&
wireDiameter <= 0.063) {
118         E = 28.7 * 1000000;
119         G = 11.6 * 1000000;
120     } else if (wireDiameter > 0.063 &&
wireDiameter <= 0.125) {
121         E = 28.6 * 1000000;
122         G = 11.5 * 1000000;
123     } else if (wireDiameter > 0.125) {
124         E = 28.5 * 1000000;
125         G = 11.4 * 1000000;
126     }
127 }
128     case "Chrome-vanadium wire (ASTM No.
A232)" -> {
129         A = 169;
130         m = 0.168;
131         ultimateTensileStrength = (A / (Math
.pow(wireDiameter, m))) * 1000;
132         yieldStrength = 0.88 *
ultimateTensileStrength;
133         yieldStrengthShear = 0.65 *
ultimateTensileStrength;
134         E = 29.5 * 1000000;
135         G = 11.2 * 1000000;
136     }
137     case "Chrome-silicon wire (ASTM No. A401
)" -> {
138         A = 202;
139         m = 0.108;
140         ultimateTensileStrength = (A / (Math
.pow(wireDiameter, m))) * 1000;
141         yieldStrength = 0.85 *
ultimateTensileStrength;
142         yieldStrengthShear = 0.65 *
ultimateTensileStrength;
143         E = 29.5 * 1000000;
144         G = 11.2 * 1000000;
145     }

```

```

146         case "302 stainless wire (ASTM No. A313
    )" -> {
147             if (wireDiameter > 0.013 &&
    wireDiameter <= 0.1) {
148                 A = 169;
149                 m = 0.146;
150             } else if (wireDiameter > 0.1 &&
    wireDiameter <= 0.2) {
151                 A = 128;
152                 m = 0.263;
153             } else if (wireDiameter > 0.2 &&
    wireDiameter <= 0.4) {
154                 A = 90;
155                 m = 0.478;
156             }
157             ultimateTensileStrength = (A / (Math
    .pow(wireDiameter, m))) * 1000;
158             yieldStrength = 0.65 *
    ultimateTensileStrength;
159             yieldStrengthShear = 0.45 *
    ultimateTensileStrength;
160             E = 28.0 * 1000000;
161             G = 10.0 * 1000000;
162         }
163         case "Phosphor-bronze wire (ASTM No.
    B159)" -> {
164             if (wireDiameter > 0.004 &&
    wireDiameter <= 0.022) {
165                 A = 145;
166                 m = 0;
167             } else if (wireDiameter > 0.022 &&
    wireDiameter <= 0.075) {
168                 A = 121;
169                 m = 0.028;
170             } else if (wireDiameter > 0.075 &&
    wireDiameter <= 0.3) {
171                 A = 110;
172                 m = 0.064;
173             }
174             ultimateTensileStrength = (A / (Math
    .pow(wireDiameter, m))) * 1000;

```

```

175         yieldStrength = 0.75 *
ultimateTensileStrength;
176         yieldStrengthShear = 0.45 *
ultimateTensileStrength;
177         E = 15.0 * 1000000;
178         G = 6.0 * 1000000;
179     }
180 }
181
182     return new double[] {ultimateTensileStrength
, yieldStrength, yieldStrengthShear, E, G};
183 }
184
185     /* Calculates Nt (coils), Na (coils), and p (in
)) */
186     public static double[] calculateDimensionalInfo(
String endType, double wireDiameter,
187     double freeLength, double solidLength) {
188         double totalCoils = 0;
189         double activeCoils = 0;
190         double pitch = 0;
191
192         switch (endType) {
193             case "Plain" -> {
194                 totalCoils = (solidLength /
wireDiameter) - 1;
195                 activeCoils = totalCoils;
196                 pitch = (freeLength - wireDiameter
) / activeCoils;
197             }
198             case "Plain and ground" -> {
199                 totalCoils = solidLength /
wireDiameter;
200                 activeCoils = totalCoils - 1;
201                 pitch = freeLength / (activeCoils +
1);
202             }
203             case "Squared or closed" -> {
204                 totalCoils = (solidLength /
wireDiameter) - 1;

```

```

205         activeCoils = totalCoils - 2;
206         pitch = (freeLength - (3 *
    wireDiameter)) / activeCoils;
207     }
208     case "Squared and ground" -> {
209         totalCoils = solidLength /
    wireDiameter;
210         activeCoils = totalCoils - 2;
211         pitch = (freeLength - (2 *
    wireDiameter)) / activeCoils;
212     }
213 }
214
215     return new double[] {totalCoils, activeCoils
    , pitch};
216 }
217
218     /* Calculates k (lbf/in) */
219     public static double calculateSpringRate(double
    wireDiameter, double G,
220
    double
    coilDiameter, double activeCoils) {
221         return (Math.pow(wireDiameter, 4) * G) / (8
    * Math.pow(coilDiameter, 3) * activeCoils);
222     }
223
224     /* Calculates F (lbf) with a deflection of (Lo
    - Ls) */
225     public static double calculateForceToSolid(
    double springRate, double freeLength, double
    solidLength) {
226         return springRate * (freeLength -
    solidLength);
227     }
228
229     /* Calculates n at solid length */
230     public static double
    calculateFactorOfSafetyAtForceToSolid(double
    forceToSolid, double yieldStrengthShear,
231
    double coilDiameter, double wireDiameter

```



```

231 ) {
232     double springIndex = coilDiameter /
    wireDiameter;
233     double bergstrasserFactor = ((4 *
    springIndex) + 2) / ((4 * springIndex) - 3);
234     double shearStress = bergstrasserFactor
235         * ((8 * forceToSolid * coilDiameter
    ) / (Math.PI * Math.pow(wireDiameter, 3)));
236
237     return yieldStrengthShear / shearStress;
238 }
239
240 /* Calculates the factor of safety for a static
    load */
241     public static double
    calculateStaticFactorOfSafety(double minForce,
    double yieldStrengthShear,
242
    double coilDiameter, double wireDiameter) {
243         double springIndex = coilDiameter /
    wireDiameter;
244         double bergstrasserFactor = ((4 *
    springIndex) + 2) / ((4 * springIndex) - 3);
245         double shearStress = bergstrasserFactor
246             * ((8 * minForce * coilDiameter) / (
    Math.PI * Math.pow(wireDiameter, 3)));
247
248         return yieldStrengthShear / shearStress;
249     }
250
251 /* Calculates the factor of safety for a cyclic
    load */
252     public static double
    calculateFatigueFactorOfSafety(double coilDiameter,
    double wireDiameter, double minForce,
253
    double maxForce, double ultimateTensileStrength,
254
    boolean peened) {
255         double springIndex = coilDiameter /
    wireDiameter;

```

```

256         double bergstrasserFactor = ((4 *
        springIndex) + 2) / ((4 * springIndex) - 3);
257         double forceAmplitude = (maxForce - minForce
        ) / 2;
258         double forceMean = (maxForce + minForce) / 2
        ;
259         double shearStressAmplitude =
        bergstrasserFactor
260             * ((8 * forceAmplitude *
        coilDiameter) / (Math.PI * Math.pow(wireDiameter, 3
        ))));
261         double shearStressMean = bergstrasserFactor
262             * ((8 * forceMean * coilDiameter
        ) / (Math.PI * Math.pow(wireDiameter, 3)));
263         double Ssu = 0.67 * ultimateTensileStrength;
264         double Sse;
265         if (peened) {
266             Sse = (57.5 * 1000) / (1 - ((77.5 * 1000
        ) / (Ssu)));
267         } else {
268             Sse = (35 * 1000) / (1 - ((55 * 1000
        ) / (Ssu)));
269         }
270         return 1 / ((shearStressAmplitude / Sse) + (
        shearStressMean / Ssu));
271     }
272
273     /**
274      * All methods below this point have to do with
        displaying the GUI elements only (no calculations)
275      */
276
277     /* Shows a welcome method dialog */
278     public static void showWelcomeMessageDialog() {
279         JOptionPane.showMessageDialog(null, "Welcome
        to the Spring Calculator!",
280             "Spring Calculator", JOptionPane.
        INFORMATION_MESSAGE);
281     }
282
283     /* Gets end type from the user */

```

```

284     public static String showEndTypeInputDialog() {
285         String endType;
286
287         do {
288             endType = (String) JOptionPane.
showInputDialog(null, "Select your end type",
289                 "Spring Calculator", JOptionPane
.QUESTION_MESSAGE, null, endTypeOptions,
290                 endTypeOptions[0]);
291             if (endType == null) {
292                 JOptionPane.showMessageDialog(null,
"Invalid choice",
293                 "Spring Calculator",
JOptionPane.ERROR_MESSAGE);
294             }
295             } while (endType == null);
296
297             return endType;
298         }
299
300         /* Gets material type from the user */
301         public static String showMaterialTypeInputDialog
() {
302             String materialType;
303
304             do {
305                 materialType = (String) JOptionPane.
showInputDialog(null, "Select your material type",
306                     "Spring Calculator", JOptionPane
.QUESTION_MESSAGE, null, materialTypeOptions,
307                     materialTypeOptions[0]);
308                 if (materialType == null) {
309                     JOptionPane.showMessageDialog(null,
"Invalid choice",
310                     "Spring Calculator",
JOptionPane.ERROR_MESSAGE);
311                 }
312                 } while (materialType == null);
313
314                 return materialType;
315             }

```

```

316
317     /* Gets peen type from the user */
318     public static boolean showPeenTypeInputDialog
319     () {
320         String peenType;
321
322         do {
323             peenType = (String) JOptionPane.
324             showInputDialog(null, "Select your peen type",
325                             "Spring Calculator", JOptionPane
326                             .QUESTION_MESSAGE, null, peenTypeOptions,
327                             peenTypeOptions[0]);
328             if (peenType == null) {
329                 JOptionPane.showMessageDialog(null,
330                 "Invalid choice",
331                 "Spring Calculator",
332                 JOptionPane.ERROR_MESSAGE);
333             }
334             while (peenType == null);
335
336             return peenType.equals("Peened");
337         }
338
339     /* Gets wire diameter from the user */
340     public static double showWireDiameterInputDialog
341     () {
342         String wireDiameter;
343
344         do {
345             wireDiameter = JOptionPane.
346             showInputDialog(null, "Enter the wire diameter (in)"
347                             ,
348                             "Spring Calculator", JOptionPane
349                             .QUESTION_MESSAGE);
350
351             try {
352                 if (Double.parseDouble(wireDiameter
353                                     ) < 0) {
354                     throw new NumberFormatException
355                     ();
356                 }
357             }
358         }
359     }

```

```

346         } catch (Exception e) {
347             wireDiameter = null;
348         }
349
350         if ((wireDiameter == null) || (
wireDiameter.isEmpty())) {
351             JOptionPane.showMessageDialog(null,
"Invalid input",
352                                     "Spring Calculator",
JOptionPane.ERROR_MESSAGE);
353         }
354     } while ((wireDiameter == null) || (
wireDiameter.isEmpty()));
355
356     return Double.parseDouble(wireDiameter);
357 }
358
359 /* Gets outer diameter from the user */
360 public static double
showOuterDiameterInputDialog() {
361     String outerDiameter;
362
363     do {
364         outerDiameter = JOptionPane.
showInputDialog(null, "Enter the outer diameter (in
)",
365                 "Spring Calculator", JOptionPane
.QUESTION_MESSAGE);
366
367         try {
368             if (Double.parseDouble(outerDiameter
) < 0) {
369                 throw new NumberFormatException
();
370             }
371         } catch (Exception e) {
372             outerDiameter = null;
373         }
374
375         if ((outerDiameter == null) || (
outerDiameter.isEmpty())) {

```

```

376         JOptionPane.showMessageDialog(null,
    "Invalid input",
377         "Spring Calculator",
    JOptionPane.ERROR_MESSAGE);
378     }
379     } while ((outerDiameter == null) || (
    outerDiameter.isEmpty()));
380
381     return Double.parseDouble(outerDiameter);
382 }
383
384     /* Gets free Length from the user */
385     public static double showFreeLengthInputDialog
    () {
386         String freeLength;
387
388         do {
389             freeLength = JOptionPane.showInputDialog
    (null, "Enter the free length (in)",
390             "Spring Calculator", JOptionPane
    .QUESTION_MESSAGE);
391
392             try {
393                 if (Double.parseDouble(freeLength
    ) < 0) {
394                     throw new NumberFormatException
    ();
395                 }
396             } catch (Exception e) {
397                 freeLength = null;
398             }
399
400             if ((freeLength == null) || (freeLength.
    isEmpty())) {
401                 JOptionPane.showMessageDialog(null,
    "Invalid input",
402                 "Spring Calculator",
    JOptionPane.ERROR_MESSAGE);
403             }
404             } while ((freeLength == null) || (freeLength
    .isEmpty()));

```

```

405
406         return Double.parseDouble(freeLength);
407     }
408
409     /* Gets solid length from the user */
410     public static double showSolidLengthInputDialog
411     () {
412         String solidLength;
413         do {
414             solidLength = JOptionPane.
415             showInputDialog(null, "Enter the solid length (in)",
416                             "Spring Calculator", JOptionPane
417                             .QUESTION_MESSAGE);
418             try {
419                 if (Double.parseDouble(solidLength
420 ) < 0) {
421                     throw new NumberFormatException
422                     ();
423                 }
424             } catch (Exception e) {
425                 solidLength = null;
426             }
427             if ((solidLength == null) || (
428             solidLength.isEmpty())) {
429                 JOptionPane.showMessageDialog(null,
430 "Invalid input",
431 "Spring Calculator",
432 JOptionPane.ERROR_MESSAGE);
433             }
434             } while ((solidLength == null) || (
435             solidLength.isEmpty()));
436
437         return Double.parseDouble(solidLength);
438     }
439
440     /* Gets min force from the user */
441     public static double showMinForceInputDialog() {
442         String minForce;

```

```

437
438         do {
439             minForce = JOptionPane.showInputDialog(
177         null, "Enter the min force (lbf)",
440                 "Spring Calculator", JOptionPane
178         .QUESTION_MESSAGE);
441
442             try {
443                 Double.parseDouble(minForce);
444             } catch (Exception e) {
445                 minForce = null;
446             }
447
448             if ((minForce == null) || (minForce.
179             isEmpty())) {
449                 JOptionPane.showMessageDialog(null,
180                 "Invalid input",
450                     "Spring Calculator",
181                 JOptionPane.ERROR_MESSAGE);
451             }
452             } while ((minForce == null) || (minForce.
182             isEmpty()));
453
454             return Double.parseDouble(minForce);
455         }
456
457         /* Gets max force from the user */
458         public static double showMaxForceInputDialog() {
459             String maxForce;
460
461             do {
462                 maxForce = JOptionPane.showInputDialog(
183             null, "Enter the max force (lbf)",
463                     "Spring Calculator", JOptionPane
184             .QUESTION_MESSAGE);
464
465             try {
466                 Double.parseDouble(maxForce);
467             } catch (Exception e) {
468                 maxForce = null;
469             }

```



```

470
471         if ((maxForce == null) || (maxForce.
isEmpty())) {
472             JOptionPane.showMessageDialog(null,
"Invalid input",
473             "Spring Calculator",
JOptionPane.ERROR_MESSAGE);
474         }
475     } while ((maxForce == null) || (maxForce.
isEmpty()));
476
477     return Double.parseDouble(maxForce);
478 }
479
480     /* Shows the final values dialog for the static
case */
481     public static void
showStaticFinalValuesMessageDialog(double pitch,
482
double totalCoils,
483
double activeCoils,
484
double springRate,
485
double forceToSolid,
486
double factorOfSafetyAtForceToSolid,
487
double factorOfSafety) {
488         String message = String.format("Spring
Values\n\n" +
489             "Pitch: %.3f in\n" +
490             "Total Coils: %.3f coils\n" +
491             "Active Coils: %.3f coils\n" +
492             "Spring Rate: %.3f lbf/in\n\n" +
493             "Force to Compress to Solid
Length: %.3f lbf\n" +
494             "Factor of Safety for Static
Yielding at Solid Length: %.1f\n\n" +
495             "Factor of Safety for Static

```

```

495 Load: %.1f", pitch, totalCoils, activeCoils,
    springRate,
496             forceToSolid,
    factorOfSafetyAtForceToSolid, factorOfSafety);
497
498 JOptionPane.showMessageDialog(null, message
    , "Spring Calculator",
499             JOptionPane.INFORMATION_MESSAGE);
500     }
501
502     /* Shows the final values dialog for the cyclic
    case */
503     public static void
    showFatigueFinalValuesMessageDialog(double pitch,
504
505         double totalCoils,
506
507         double activeCoils,
508
509         double springRate,
510
511         double forceToSolid,
512
513         double factorOfSafetyAtForceToSolid,
514
515         double factorOfSafety) {
516         String message = String.format("Spring
    Values\n\n" +
517
518             "Pitch: %.3f in\n" +
519             "Total Coils: %.3f coils\n"
520         +
521             "Active Coils: %.3f coils\n"
522         +
523             "Spring Rate: %.3f lbf/in\n\n"
524             "Force to Compress to Solid
    Length: %.3f lbf\n" +
525             "Factor of Safety for Static
    Yielding at Solid Length: %.1f\n\n" +
526             "Factor of Safety for
    Infinite Life for Cyclic Load: %.1f", pitch,

```

```
517 totalCoils, activeCoils,
518             springRate, forceToSolid,
        factorOfSafetyAtForceToSolid, factorOfSafety);
519
520     JOptionPane.showMessageDialog(null, message
        , "Spring Calculator",
521             JOptionPane.INFORMATION_MESSAGE);
522     }
523 }
524
```