```java
/**
 * A warm-blooded egg-laying vertebrate of the class <i>Aves,</i> characterized
 * by a body covering of feathers and forelimbs modified as wings
 * (<a href=
 * "http://www.collinsdictionary.com/dictionary/english/bird">Definition
 * of bird</a> retrieved 2013-04-29).
 *
 * @author Victoria Windsor
 * @version 1.1 2013-04-29
 */
public class NumberedBird
{
    // class fields
    private static int nextNumber = 1;

    // instance fields
    private boolean canFly;
    private String colour;
    private int serialNumber;
    private double wingspan;

    /* constructors */

    /**
     * Constructs a bird with default characteristics.
     */
    public NumberedBird()
    {
        colour = "unknown";
        wingspan = 0;
        canFly = false;
        serialNumber = nextNumber++;
    } // end of constructor NumberedBird()

    /**
     * Constructs a bird with the specified characteristics.
     *
     * @param colour the colour of this bird
     * @param wingspan the non-negative wingspan in centimetres of this bird
     * @param canFly <code>true</code> if this bird can fly; otherwise
     * <code>false</code>
     */
    public NumberedBird(String colour, double wingspan, boolean canFly)
    {
        if (colour.length() > 0)
        {
            this.colour = colour;
        }
        else
        {
            this.colour = "unknown";
        } // end of if (colour.length() > ))

        if (wingspan > 0)
        {
            this.wingspan = wingspan;
        }
        else
        {
            this.wingspan = 0;
        } // end of if (wingspan > 0)
```

```
62
63          this.canFly = canFly;
64          serialNumber = nextNumber++;
65      } // end of constructor NumberedBird(String colour, double wingspan, ...)
66
67      /**
68       * Constructs a bird with the specified colour and wingspan.
69       *
70       * @param colour the colour of this bird
71       * @param wingspan the non-negative wingspan in centimetres of this bird
72       */
73      public NumberedBird(String colour, double wingspan)
74      {
75          if (colour.length() > 0)
76          {
77              this.colour = colour;
78          }
79          else
80          {
81              this.colour = "unknown";
82          } // end of if (colour.length() > ))
83
84          if (wingspan > 0)
85          {
86              this.wingspan = wingspan;
87          }
88          else
89          {
90              this.wingspan = 0;
91          } // end of if (wingspan > 0)
92
93          this.canFly = false;
94          serialNumber = nextNumber++;
95      } // end of constructor NumberedBird(String colour, double wingspan)
96
97      /**
98       * Constructs a bird with the specified colour and flying ability.
99       *
100      * @param colour the colour of this bird
101      * @param canFly <code>true</code> if this bird can fly; otherwise
102      * <code>false</code>
103      */
104     public NumberedBird(String colour, boolean canFly)
105     {
106         if (colour.length() > 0)
107         {
108             this.colour = colour;
109         }
110         else
111         {
112             this.colour = "unknown";
113         } // end of if (colour.length() > ))
114
115         this.wingspan = 0;
116         this.canFly = canFly;
117         serialNumber = nextNumber++;
118     } // end of constructor NumberedBird(String colour, boolean canFly)
119
120     /**
121      * Constructs a bird with the specified colour.
122      *
```

```
123         * @param colour the colour of this bird
124         */
125        public NumberedBird(String colour)
126        {
127            if (colour.length() > 0)
128            {
129                this.colour = colour;
130            }
131            else
132            {
133                this.colour = "unknown";
134            } // end of if (colour.length() > ))
135
136            this.wingspan = 0;
137            this.canFly = false;
138            serialNumber = nextNumber++;
139        } // end of constructor NumberedBird(String colour)
140
141        /**
142         * Constructs a bird with the specified wingspan and flying ability.
143         *
144         * @param wingspan the non-negative wingspan in centimetres of this bird
145         * @param canFly <code>true</code> if this bird can fly; otherwise
146         * <code>false</code>
147         */
148        public NumberedBird(double wingspan, boolean canFly)
149        {
150            this.colour = "unknown";
151
152            if (wingspan > 0)
153            {
154                this.wingspan = wingspan;
155            }
156            else
157            {
158                this.wingspan = 0;
159            } // end of if (wingspan > 0)
160
161            this.canFly = canFly;
162            serialNumber = nextNumber++;
163        } // end of constructor NumberedBird(double wingspan, boolean canFly)
164
165        /**
166         * Constructs a bird with the specified wingspan.
167         *
168         * @param wingspan the non-negative wingspan in centimetres of this bird
169         */
170        public NumberedBird(double wingspan)
171        {
172            this.colour = "unknown";
173
174            if (wingspan > 0)
175            {
176                this.wingspan = wingspan;
177            }
178            else
179            {
180                this.wingspan = 0;
181            } // end of if (wingspan > 0)
182
183            this.canFly = false;
```

```
184         serialNumber = nextNumber++;
185     } // end of constructor NumberedBird(double wingspan)
186 
187     /**
188      * Constructs a bird with the specified flying ability.
189      *
190      * @param canFly <code>true</code> if this bird can fly; otherwise
191      * <code>false</code>
192      */
193     public NumberedBird(boolean canFly)
194     {
195         this.colour = "unknown";
196         this.wingspan = 0;
197         this.canFly = canFly;
198         serialNumber = nextNumber++;
199     } // end of constructor NumberedBird(boolean canFly)
200 
201     /* accessors */
202 
203     /**
204      * Returns the colour of this bird.
205      *
206      * @return the colour of this bird
207      */
208     public String getColour()
209     {
210         return colour;
211     } // end of method getColour()
212 
213     /**
214      * Returns <code>true</code> if this bird can fly; otherwise
215      * <code>false</code>.
216      *
217      * @return <code>true</code> if this bird can fly; otherwise
218      * <code>false</code>
219      */
220     public boolean ableToFly()
221     {
222         return canFly;
223     } // end of method ableToFly()
224 
225     /**
226      * Returns the wingspan in centimetres of this bird.
227      *
228      * @return the wingspan in centimetres of this bird
229      */
230     public double getWingspan()
231     {
232         return wingspan;
233     } // end of method getWingspan()
234 
235     /**
236      * Returns the serial number of this bird.
237      *
238      * @return the serial number of this bird
239      */
240     public int getSerialNumber()
241     {
242         return serialNumber;
243     } // end of method getSerialNumber()
244 
```

```
245     /**
246      * Returns a string representation of this bird.
247      *
248      * @return a string representation of this bird
249      */
250     public String toString()
251     {
252         return
253         getClass().getName()
254         + "[colour: " + colour
255         + ", wingspan: " + wingspan
256         + ", able to fly: " + canFly
257         + ", serial number: " + serialNumber
258         + "]";
259     } // end of method toString()
260 
261     /* mutators */
262 
263     /**
264      * Sets the colour of this bird.
265      *
266      * @param colour the colour of this bird; may not be an empty string
267      */
268     public void setColour(String colour)
269     {
270         if (colour.length() > 0)
271         {
272             this.colour = colour;
273         }
274         else
275         {
276             this.colour = "unknown";
277         } // end of if (colour.length() > ))
278     } // end of method setColour(String colour)
279 
280     /**
281      * Sets the flight ability of this bird.
282      *
283      * @param canFly <code>true</code> if this bird can fly; otherwise
284      * <code>false</code>
285      */
286     public void setFlightAbility(boolean canFly)
287     {
288         this.canFly = canFly;
289     } // end of method setFlightAbility(boolean canFly)
290 
291     /**
292      * Sets the wingspan in centimetres of this bird.
293      *
294      * @param wingspan the non-negative wingspan in centimetres
295      */
296     public void setWingspan(double wingspan)
297     {
298         if (wingspan > 0)
299         {
300             this.wingspan = wingspan;
301         }
302         else
303         {
304             this.wingspan = 0;
305         } // end of if (wingspan > 0)
```

```
306      } // end of method setWingspan(double wingspan)
307
308  } // end of class NumberedBird
```