```
 2 *  Licensed to the Apache Software Foundation (ASF) under one or more
17 package com.google;
18
19 import java.util.ArrayList;
20 import java.util.Collection;
21 import java.util.List;
22
23 /**
24  * Audit - Rules - Performance.
25  *
26  * @author  lihg
27  * @version 2013-6-11
28  */
29 public class Performance {
30     /**
31      * <pre>
32      * 1, Append <b>single character</b>.
33      * 2, <b>Concatenation</b> in <font color="red"><b>appending</b></font> method.
34      * 3, <font color="red"><b>Replace</b></font> <b>synchronized classes</b>.
35      * </pre>
36      */
37     public void appendChar() {
38         StringBuilder sb = new StringBuilder(); // ! StringBuffer
39         String userName = "Bert";
40         sb.append("Hello ").append(userName)
41               .append('!');
42     }
43
44     /**
45      * Debugging code.
46      *
47      * @param s
48      * @return
49      */
50     public static long parseLong(String s) {
51         try {
52             return Long.parseLong(s, 10);
53         } catch (NumberFormatException nfe) {
54 //          out.println("Non number: " + s);
55             return -1;
56         }
57     }
58
59     /**
60      * Define <b>initial capacity</b>.
61      */
62     public static void initialCapacity() {
63         List<String> list = new ArrayList<String>(3);
64         list.add("1");
65         list.add("2");
66         list.add("3");
67     }
68
69     /**
70      * <pre>
71      * 1, Don't use concatenation to convert to String.
72      * 2, <font color="red"><b>Temporary</b></font> <b>object creation</b>.
73      * </pre>
74      *
75      * @param i
76      * @return
```

```
77      */
78     public static String valueOf() {
79         int count = 5;
80 //      return count + "";
81 
82         // 重载：即使参数类型有变化，也无需改动代码
83 //      return String.valueOf(count);
84         /*
85          * 当参数类型有变化时，需改动代码实现；否则，可能导致Bug。
86          * 如int -> long
87          */
88         return Integer.toString(count);
89     }
90 
91     /**
92      * <font color="red"><b>Efficient</b></font> expression.
93      *
94      * @param s
95      * @return
96      */
97     public static long parseInt(String s) {
98         try {
99             int i = Integer.parseInt(s, 10);
100            return i;
101        } catch (NumberFormatException nfe) {
102            return -1;
103        }
104    }
105 
106    /**
107     * <b>Empty string</b> detection.
108     *
109     * @param s
110     * @return
111     */
112    public static boolean isEmpty(String s) {
113        return s == null || s.isEmpty();
114    }
115 
116    /**
117     * <font color="red"><b>Inefficient</b></font> use of <b>toArray()</b>.
118     *
119     * @param c
120     * @return
121     */
122    public static String[] toArray(Collection<String> c) {
123 //      return c.toArray(new String[0]);
124        return c.toArray(new String[c.size()]);
125    }
126 
127    /** <font color="red"><b>Reusable</b></font> <b>Immutable</b>. */
128    public static final String EMPTY_STRING = "";
129 
130    /**
131     * <b>String concatenation</b> <font color="red"><b>in loop</b></font>.
132     *
133     * @param path
134     * @return
135     */
136    public static String concatePath(String[] path) {
137        StringBuilder sb = new StringBuilder();
```

```
138         for (String p : path) {
139             sb.append(p).append('.');
140         }
141         return sb.toString();
142     }
143 
144     /**
145      * Use <b>arraycopy()</b> rather than a loop.
146      */
147     public static void arraycopy() {
148         String[] a1 = new String[3];
149         String[] a2 = new String[3];
150         System.arraycopy(a1, 0, a2, 0, a2.length);
151     }
152 
153     /**
154      * <pre>
155      * 1, Use available <b>constants</b>.
156      * 2, Use <b>valueOf()</b> to wrap primitives.
157      * </pre>
158      *
159      * @param i
160      * @return
161      */
162     public static Integer newInteger(int i) {
163         return Integer.valueOf(i);
164     }
165 
166     /**
167      * Use <b>char</b> rather than string, it can improve performance.
168      *
169      * @param s
170      * @return
171      */
172     public static int searchForChar(String s) {
173         return s.lastIndexOf('/');
174     }
175 }
176 
```