

```

1 import java.util.Set;
2
3 import components.simplewriter.SimpleWriter;
4 import components.simplewriter.SimpleWriter1L;
5
6 /**
7  * HW35
8  *
9  * @author Sam Espanioly
10 */
11 public final class HW35 {
12
13     /**
14      * Default constructor--private to prevent instantiation.
15      */
16     private HW35() {
17         // no code needed here
18     }
19
20     /**
21      * * Raises the salary of all the employees in {@code map} whose name
22      * starts * with the given {@code initial} by the given
23      * {@code raisePercent}. * * @param map * the name to salary map
24      * * @param initial * the initial of names of employees to be given a
25      * raise * @param raisePercent * the raise to be given as a percentage of
26      * the current salary * @updates map * @requires [the salaries in map are
27      * positive] and raisePercent > 0 * @ensures <pre>
28      * DOMAIN(map) = DOMAIN(#map) and
29      * [the salaries of the employees in map whose names start with the given
30      * initial have been increased by raisePercent percent (and truncated to
31      * the nearest integer); all other employees have the same salary]
32      * </pre>
33      */
34     private static void giveRaise components.map.Map<String, Integer> map,
35         char initial, int raisePercent {
36         // i don't know this look easy but I don't know
37     }
38
39     /**
40      * * Raises the salary of all the employees in {@code map} whose name
41      * starts * with the given {@code initial} by the given
42      * {@code raisePercent}.
43      *
44      * @param map
45      *         the name to salary map
46      * @param initial
47      *         the initial of names of employees to be given a raise
48      * @param raisePercent
49      *         the raise to be given as a percentage of the current salary
50      * @updates map
51      * @requires <pre>
52      * [the salaries in map are positive] and raisePercent > 0 and
53      * [the dynamic types of map and of all objects reachable from map
54      * (including any objects returned by operations (such as entrySet() and,
55      * from there, iterator()), and so on, recursively) support all
56      * optional operations]
57      * </pre> * @ensures <pre>

```

```
58     * DOMAIN(map) = DOMAIN(#map) and
59     * [the salaries of the employees in map whose names start with the given
60     * initial have been increased by raisePercent percent (and truncated to
61     * the nearest integer); all other employees have the same salary]
62     * </pre>
63     */
64     @SuppressWarnings("unused")
65     private static void giveRaise(java.util.Map<String, Integer> map,
66         char initial, int raisePercent) {
67         final double hun = 100.00;
68         double raise = raisePercent / hun;
69         double p = -1;
70         Set<String> names = map.keySet();
71         int size = map.size();
72         int i = 0;
73         while (names.iterator().hasNext() && i < size) {
74             String n = names.iterator().next();
75             p = map.get(n);
76             if (n.charAt(0) == initial) {
77                 p = map.get(n) * raise;
78             }
79             int pay = (int) p;
80             map.put(n, pay);
81             names.iterator().remove();
82         }
83     }
84 }
85
86 /**
87  * Main method.
88  *
89  * @param args
90  *         the command line arguments; unused here
91  */
92 public static void main(String[] args) {
93     SimpleWriter out = new SimpleWriter1L();
94     out.close();
95 }
96
97 }
98
```