```java
import java.util.Set;

import components.simplewriter.SimpleWriter;
import components.simplewriter.SimpleWriter1L;

/**
 * HW35
 *
 * @author Sam Espanioly
 */
public final class HW35 {

    /**
     * Default constructor--private to prevent instantiation.
     */
    private HW35() {
        // no code needed here
    }

    /**
     *  * Raises the salary of all the employees in {@code map} whose name
     * starts  * with the given {@code initial} by the given
     * {@code raisePercent}.  *  * @param map  * the name to salary map
     *  * @param initial  * the initial of names of employees to be given a
     * raise  * @param raisePercent  * the raise to be given as a percentage of
     * the current salary  * @updates map  * @requires [the salaries in map are
     * positive] and raisePercent > 0  * @ensures <pre>
      * DOMAIN(map) = DOMAIN(#map)  and
      * [the salaries of the employees in map whose names start with the given
      *  initial have been increased by raisePercent percent (and truncated to
      *  the nearest integer); all other employees have the same salary]
      * </pre>
     */
    private static void giveRaise(components.map.Map<String, Integer> map,
            char initial, int raisePercent) {
        // i don't know this look easy but I don't know
    }

    /**
     *  * Raises the salary of all the employees in {@code map} whose name
     * starts  * with the given {@code initial} by the given
     * {@code raisePercent}.
     *
     * @param map
     *            the name to salary map
     * @param initial
     *             the initial of names of employees to be given a raise
     * @param raisePercent
     *             the raise to be given as a percentage of the current salary
     * @updates map
     * @requires <pre>
      * [the salaries in map are positive]  and  raisePercent > 0  and
      * [the dynamic types of map and of all objects reachable from map
      *  (including any objects returned by operations (such as entrySet() and,
      *  from there, iterator()), and so on, recursively) support all
      *  optional operations]
      * </pre>  * @ensures <pre>
      * DOMAIN(map) = DOMAIN(#map)  and
      * [the salaries of the employees in map whose names start with the given
      *  initial have been increased by raisePercent percent (and truncated to
      *  the nearest integer); all other employees have the same salary]
      * </pre>
     */
    @SuppressWarnings("unused")
    private static void giveRaise(java.util.Map<String, Integer> map,
            char initial, int raisePercent) {
        final double hun = 100.00;
        double raise = raisePercent / hun;
        double p = -1;
        Set<String> names = map.keySet();
        int size = map.size();
        int i = 0;
        while (names.iterator().hasNext() && i < size) {
            String n = names.iterator().next();
            p = map.get(n);
            if (n.charAt(0) == initial) {
                p = map.get(n) * raise;
            }
            int pay = (int) p;
            map.put(n, pay);
            names.iterator().remove();
        }

    }

    /**
     * Main method.
     *
     * @param args
     *            the command line arguments; unused here
     */
    public static void main(String[] args) {
        SimpleWriter out = new SimpleWriter1L();
        out.close();
    }

}
```