```java
import components.map.Map;
import components.simplereader.SimpleReader;
import components.simplereader.SimpleReader1L;

/**
 * cooooool homework yaaaaaaayyy
 *
 * @author G. Farmer
 */
public final class Homework21 {

    /**
     * No argument constructor--private to prevent instantiation.
     */
    private Homework21() {
        // no code needed here
    }

    /**
     * Inputs a "menu" of words (items) and their prices from the given file and
     * stores them in the given {@code Map}.
     *
     * @param fileName
     *            the name of the input file
     * @param priceMap
     *            the word -> price map
     * @replaces priceMap
     * @requires <pre>
     * [file named fileName exists but is not open, and has the
     *  format of one "word" (unique in the file) and one price (in cents)
     *  per line, with word and price separated by ','; the "word" may
     *  contain whitespace but no ',']
     * </pre>
     * @ensures [priceMap contains word -> price mapping from file fileName]
     */
    private static void getPriceMap(String fileName,
            Map<String, Integer> priceMap) {

        SimpleReader file = new SimpleReader1L(fileName);
        String[] namePrice;

        String next = file.nextLine();

        while (!next.equals("")) {
            namePrice = next.split("[,]", 0);
            priceMap.add(namePrice[0], Integer.valueOf(namePrice[1]));
            next = file.nextLine();
        }
    }

    /**
     * Input one pizza order and compute and return the total price.
     *
     * @param input
     *            the input stream
     * @param sizePriceMap
     *            the size -> price map
     * @param toppingPriceMap
     *            the topping -> price map
```

```
60      * @return the total price (in cents)
61      * @updates input
62      * @requires <pre>
63      * input.is_open and
64      * [input.content begins with a pizza order consisting of a size
65      *  (something defined in sizePriceMap) on the first line, followed
66      *  by zero or more toppings (something defined in toppingPriceMap)
67      *  each on a separate line, followed by an empty line]
68      * </pre>
69      * @ensures <pre>
70      * input.is_open and
71      * #input.content = [one pizza order (as described
72      *              in the requires clause)] * input.content and
73      * getOneOrder = [total price (in cents) of that pizza order]
74      * </pre>
75      */
76     private static int getOneOrder(SimpleReader input,
77             Map<String, Integer> sizePriceMap,
78             Map<String, Integer> toppingPriceMap) {
79
80         String next = input.nextLine();
81         int total = 0;
82         int i = 0;
83
84         while (!next.equals("")) {
85             if (i == 0) {
86                 total += sizePriceMap.value(next);
87             } else {
88                 total += toppingPriceMap.value(next);
89             }
90         }
91
92         return 0;
93     }
94
95 }
96
```