```java
import java.util.Scanner;
import java.util.regex.Pattern;

import components.map.Map;
import components.map.Map1L;
import components.simplereader.SimpleReader;
import components.simplereader.SimpleReader1L;
import components.simplewriter.SimpleWriter;
import components.simplewriter.SimpleWriter1L;

/**
 * Simple pizza order manager: inputs orders from a file and computes and
 * displays the total price for each order.
 *
 * @author Put your name here
 *
 */
public final class PizzaOrderManager {

    /**
     * Private constructor so this utility class cannot be instantiated.
     */
    private PizzaOrderManager() {
    }

    /**
     * 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 not ',']
     * </pre>
     * @ensures [priceMap contains word -> price mapping from file fileName]
     */
    private static void getPriceMap(String fileName,
            Map<String, Integer> priceMap) {
        assert fileName != null : "Violation of: fileName is not null";
        assert priceMap != null : "Violation of: priceMap is not null";
        /*
         * Note: Precondition not checked!
         */
        Scanner sc = new Scanner(fileName);
        String item = "";
        Integer price = 0;
        while (sc.hasNextLine()) {
            // i did not know how to do this part correctly but
            // i looked for a sequence but the program did not catch it
            // while debugging the variable item would be null for some reason
            item = sc.findInLine(Pattern.compile("[a-z]+"));
            sc.findInLine(",");
            // same thing here
            String tempValue = sc.findInLine(Pattern.compile("[0-9]+"));
            // I forgot the name of the call to check if string can be int
            //if(Integer.canparseInt?)else file has an error
            if (tempValue != null) {
                price = Integer.parseInt(tempValue);
                if (!priceMap.hasKey(item)) {
                    priceMap.add(tempValue, price);
                }
            }
        }
    }

    /**
     * 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
     * @return the total price (in cents)
     * @updates input
     * @requires <pre>
     * input.is_open and
     * [input.content begins with a pizza order consisting
```

```
of a size
     *  (something defined in sizePriceMap) on the first line, followed
     *  by zero or more toppings (something defined in toppingPriceMap)
     *  each on a separate line, followed by an empty line]
     * </pre>
     * @ensures <pre>
     * input.is_open and
     * #input.content = [one pizza order (as described
     *              in the requires clause)] * input.content and
     * getOneOrder = [total price (in cents) of that pizza order]
     * </pre>
     */
    private static int getOneOrder(SimpleReader input,
            Map<String, Integer> sizePriceMap,
            Map<String, Integer> toppingPriceMap) {
        assert input != null : "Violation of: input is not null";
        assert sizePriceMap != null : "Violation of: sizePriceMap is not null";
        assert toppingPriceMap != null : "Violation of: toppingPriceMap is not null";
        assert input.isOpen() : "Violation of: input.is_open";
        /*
         * Note: Rest of precondition not checked!
         */
        int price = 0;
        // could not figure out how to stop when there's a new line
```

```java
//        while (!input.nextLine().equals("\r\n")) {
        String order = input.nextLine();
        if (sizePriceMap.hasKey(order)) {
            price = price + sizePriceMap.value(order);
        }
        if (toppingPriceMap.hasKey(order)) {
            price = price + toppingPriceMap.value(order);
//            }
        }
        return price;
    }

    /**
     * Output the given price formatted in dollars and cents.
     *
     * @param output
     *            the output stream
     * @param price
     *            the price to output
     * @updates output
     * @requires output.is_open = true and 0 <= price
     * @ensures <pre>
     * output.is_open and
     * output.content = #output.content *
     *  [display of price, where price is in cents but
     *   display is formatted in dollars and cents]
     * </pre>
     */
    private static void putPrice(SimpleWriter output, int price) {
        assert output != null : "Violation of: output is not null";
        assert output.isOpen() : "Violation of:
```

```java
output.is_open";
        assert 0 <= price : "Violation of: 0 <= price";

        // TODO - fill in body
        int cents = price % 100;
        int dollas = price / 100;
        output.println("$" + dollas + "." + cents);

    }

    /**
     * Main method.
     *
     * @param args
     *            the command line arguments
     */
    public static void main(String[] args) {
        SimpleReader in = new SimpleReader1L("data/orders.txt");
        SimpleWriter out = new SimpleWriter1L();
        Map<String, Integer> sizeMenu = new Map1L<String, Integer>();
        Map<String, Integer> toppingMenu = new Map1L<String, Integer>();
        int orderNumber = 1;
        /*
         * Get menus of sizes with prices and toppings with prices
         */
        getPriceMap("data/sizes.txt", sizeMenu);
        getPriceMap("data/toppings.txt", toppingMenu);
        /*
         * Output heading for report of pizza orders
         */
        out.println();
        out.println("Order");
        out.println("Number Price");
        out.println("------ ------");
        /*
         * Process orders, one at a time, from input file
         */
        while (!in.atEOS()) {
            // another attempt of picking one order at a time
//            if (!in.equals('\r\n')) {
            int price = getOneOrder(in, sizeMenu, toppingMenu);
            out.print(orderNumber + "      ");
            putPrice(out, price);
            out.println();
            orderNumber++;
//            } else {
//                out.println();
//            }

        }
        out.println();
        /*
         * Close input and output streams
         */
        in.close();
        out.close();
    }

}
```