Submission Due: June/13/2022 23:59

Homework 3: Cook the dishes in the kitchen

Prerequisite: Runnable interface, Thread class

For you to do this assignment, you'll need to understand Runnable interface, Thread class and How to pass a task to Thread using Runnable interface.

Overview

You are given a skeleton project called **Homework3-template**, whose file structure is shown in Figure 1. The project is incomplete in its current form, and would run correctly only when **Kitchen.java** and **Order.java** are properly implemented. Your goal in this homework is thus to appropriately design the classes to fix the problem.

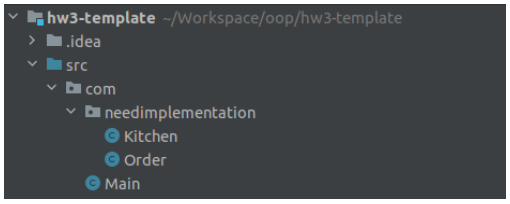

Figure 1: Hierarchy of **Homework3-template**.

In this homework, you are only allowed to modify **Kitchen.java** and **Order.java** while other files should remain as they are.

⚠The task should be carried out by adding code based on the existing code, not modifying the existing code.

Class description

```java
public class Main {
  private static final String CONTENT = "6\n" +
      "john\n" + // 150
      "1\n" +
      "friedrice\n" +
      "mike\n" + // 250
      "1\n" +
      "bibimmyon\n" +
      "din din\n" + // 350
      "3\n" +
      "ramen\n" +
      "ovenroast\n" +
      "egg\n" +
      "hyu undergraduate student 1\n" + // 450
      "3\n" +
      "bibimmyon\n" +
      "friedrice\n" +
      "egg\n" +
      "alexa\n" + // 450
      "3\n" +
      "ramen\n" +
      "friedrice\n" +
      "ovenroast\n" +
      "king sejong\n" + // 750
      "5\n" +
      "egg\n" +
      "ramen\n" +
      "friedrice\n" +
      "ovenroast\n" +
      "bibimmyon";
  public static void main(String[] args) throws InterruptedException {
    Scanner scanner = new Scanner(CONTENT);
    Order[] orders = Order.loadOrder(scanner);
    Kitchen kitchen = new Kitchen();
    for (Order o : orders) {
      kitchen.cook(o); // Cooking should be done by each thread
      println(">>Kitchen took the order from " + o + ", currently order " + Arrays.toString(
    kitchen.getAllUnfinishedOrders()) + " pended"); // Kitchen is aware if cooking is
    finished or not
      Thread.sleep(100);
    }
    while (!kitchen.isAllOrderFinished()) {
      println(">>Waiting for these orders to be end: " + Arrays.toString(kitchen.
    getAllUnfinishedOrders()));
      Thread.sleep(100);
    }
    println(">>All order is cooked!");
  }
  public static synchronized void println(String s) {
    System.out.println(s);
  }
}
```

Figure 2: Main class source code.

Main class

1. The **Main** class is already fully implemented and provided as Figure 2. Description is as follows.
   - **Main** class has **CONTENT** variable, containing order details (Line 2 in Figure 2).
   - Using the **loadOrder** method in the **Order** class, **Main** class parse the **CONTENT** variable, which contains the order details, to get **Order[]** (Line 33 in Figure 2).
   - Then, **Main** class call cook method of **Kitchen** class, which generates separate thread per order (Line 36 in Figure 2).
   - After that, **Main** class prints what order has been received and currently what order has not been processed, using **Kitchen** class (Line 47 in Figure 2).
   - After requesting all orders to **Kitchen**, wait for the orders to complete. While waiting, **Main** class prints order that is not yet completed for every 0.1 second (Line 40-44 in Figure 2).
2. Below is about format of **CONTENT** variable in details.
   - The first line is **The total number of orders**.
   - After that, each **Order details** are repeated as many as **The total number of orders**.
   - **Order details** starts with name of the orderer at first line. **Number of dishes** at second line. At last, dish name is repeated for **number of dishes**.

Kitchen class

1. Three incompletely implemented methods are given, and you must finish implementing methods. Add additional methods if necessary.
   - **public void cook(Order o)** - A separate thread should be able to process the **cook** method in the **Order** class.
   - **public boolean isAllOrderFinished()** - Return as **boolean** type about all the orders received from the **cook** method has all been processed or not.
   - **public Order[] getAllUnfinishedOrders()** - Among the orders passed from the **cook** method, incomplete orders should be returned as **Order[]** type.

   Although you can, it is not necessary to use Thread.State getState() to implement **isAllOrderFinished** and **getAllUnfinishedOrders** method.

Order class

1. Two incompletely implemented methods and one fully implemented method are given, and you must finish implementing methods. Add additional methods if necessary.

- **public static Order[] loadOrder(Scanner scanner)** - Method should parse the total order, read via scanner, and return it as **Order[]**.
- **public void run()** - It should be implemented to invoke the **cook** method in **Order** class for all dishes within the order instance.
- **private void cook(String dish)** - Already fully implemented method. Used **Thread.sleep** method to implement the time consumed by cooking. Care must be taken not to invoke this method by Main thread.

2. Note that, the amount of time consumed per dish is already defined in the **private void cook (String dish)** method, and the name of the dish in the **Order details** always exists in the switch-case statement of the cook method in Order class.

Specification

The content and format of the each line should exactly match as below. If you implemented as requested above, your program should exactly print as below. The order of the lines might not match because this assignment is using Multi-Thread, but that will rarely happen.

```
>>Kitchen took the order from john, currently order [john] pended
>>Kitchen took the order from mike, currently order [john, mike] pended
>>Order from [john] finished
>>Kitchen took the order from din din, currently order [mike, din din] pended
>>Kitchen took the order from hyu undergraduate student 1, currently order [mike, din din,
    hyu undergraduate student 1] pended
>>Order from [mike] finished
>>Kitchen took the order from alexa, currently order [din din, hyu undergraduate student 1,
    alexa] pended
>>Kitchen took the order from king sejong, currently order [din din, hyu undergraduate
    student 1, alexa, king sejong] pended
>>Order from [din din] finished
>>Waiting for these orders to be end: [hyu undergraduate student 1, alexa, king sejong]
>>Waiting for these orders to be end: [hyu undergraduate student 1, alexa, king sejong]
>>Order from [hyu undergraduate student 1] finished
>>Waiting for these orders to be end: [alexa, king sejong]
>>Order from [alexa] finished
>>Waiting for these orders to be end: [king sejong]
>>Waiting for these orders to be end: [king sejong]
>>Waiting for these orders to be end: [king sejong]
>>Waiting for these orders to be end: [king sejong]
>>Order from [king sejong] finished
>>All order is cooked!
```

Scoring Criteria (10 points)

- **(7 points)** Your program should satisfy all the requirements listed so far, ensuring its accurate working.
- **(3 points)** Each class should be properly implemented following its specification.