**Ресторан(1)**

Давай напишем программу для автоматизации работы в ресторане.

Пусть задание сформулировано так — директор ресторана хочет, чтобы:  
1. на каждом столике лежал планшет, через который можно было бы сделать заказ;  
2. пока заказ готовится, на планшете показывалась реклама;  
3. в конце рабочего дня была возможность посмотреть различную статистику:  
а) загрузки повара;  
б) сумму выручки за заказы;  
в) сумму выручки за показы рекламы.

Мы будем разрабатывать ядро, без UI. UI ты потом сможешь дописать самостоятельно, сейчас ограничимся консольным интерфейсом.  
Также мы часть данных захардкодим, ты потом сможешь прикрутить БД, и работать с ней.

Выделим несколько фич, которые будем реализовывать:

Первая:  
— созданный посетителем заказ будет автоматически поступать к повару;  
— повар будет готовить его какое-то время и отмечать приготовленным;  
— после этого официант будет относить его.

Вторая:  
— подобрать нужные рекламные ролики из списка оплаченных;  
— отображение рекламных роликов во время приготовления заказа;  
— максимизировать прибыль от показа рекламы.

Третья:  
— подсчет статистики;  
— отображение статистики директору.

Начнем с первой фичи — автоматизация обработки заказа:

Итак..  
1. Создай класс Restaurant с методом main.

2. Начнем с планшета, создадим класс Tablet, в котором создадим поле final int number — это номер планшета, чтобы можно было однозначно установить, откуда поступил заказ. Номер планшета должен инициализироваться в конструкторе переданным параметром.

3. В классе Tablet создадим метод public void createOrder(), который будет создавать заказ из тех блюд, которые выберет пользователь.  
Для этого создадим класс Order, который поместим в пакет kitchen.

4. В классе Order (*заказ*) должна быть информация, относящаяся к списку выбранных пользователем блюд.  
Т.е. где-то должен быть список всех блюд, и должен быть список выбранных блюд в классе Order.  
В классе Order нужны поля private final Tablet tablet и protected List<Dish> dishes.  
Конструктор должен принимать один параметр типа Tablet и инициализировать поле tablet.

5. Нам нужен класс Dish(*Блюдо*), создадим его в пакете kitchen. Пусть это будет enum со списком блюд: Fish, Steak, Soup, Juice, Water.

6. Чтобы пользователь мог выбрать себе блюда, нужно их все ему отобразить. Для этого в Dish создай метод public static String allDishesToString(), который сформирует строку из всех блюд.

Пример: «*Fish, Steak, Soup, Juice, Water*«. Формируй строку динамически.

**Ресторан(2)**

1. Мы много работаем с консолью. Пора создать единую точку взаимодействия.  
Создай класс ConsoleHelper с единственным BufferedReader, через который будем работать с консолью.  
Запомни, этот класс не хранит никаких данных и состояний, поэтому все методы будут статическими.  
Создай в нем три метода:  
— writeMessage(String message) — для вывода message в консоль  
— String readString() — для чтения строки с консоли  
— List<Dish> getAllDishesForOrder() — просит пользователя выбрать блюдо и добавляет его в список.  
Выведи список всех блюд и попроси пользователя ввести строку — название блюда.  
Введенное ‘exit‘ означает завершение заказа.  
В случае, если введенное блюдо не представлено в меню, выведи сообщение о том, что такого блюда нет и продолжи формирование заказа.  
Исключения ввода-вывода бросай выше, на этом уровне не понятно, что с ними делать.

2. Сделай рефакторинг — работа с консолью должна быть только через класс ConsoleHelper.

3. Вернемся к классу Order: в нем есть ссылка на планшет, и еще есть список выбранных блюд.  
Инициализируй список dishes в конструкторе, вызвав метод getAllDishesForOrder из ConsoleHelper.

4. Перепиши метод toString в классе Order. Пусть он возвращает пустую строку, если нет блюд в заказе, иначе вывод должен быть аналогичным примеру в порядке добавления блюд. Используй ConsoleHelper.  
Также измени метод toString в классе Tablet (*внутри класса* Tablet *нажмите* Alt+Insert -> toString()).  
Пример:  
Your order: [Juice, Fish] of Tablet{number=5}

5. У нас все завязано на работу с консолью. Однако, при возникновении исключений, наше приложение умрет.  
Чтобы узнать причину — добавим в Tablet статическое поле logger типа *java.util.logging.Logger*, инициализированное именем класса (Logger.getLogger(Tablet.class.getName())).

6. В методе createOrder класса Tablet обработаем исключения ввода-вывода.  
Запишем в лог «*Console is unavailable.*«. Уровень лога — SEVERE — это самый серьезный уровень, мы не можем работать.  
Также в методе createOrder класса Tablet должен быть создан новый заказ.

7. Надо начинать тестировать наше приложение.  
Добавьте в main создание планшета и создание заказа — new Tablet(5).createOrder();

Требования:  
1. Метод writeMessage класса ConsoleHelper должен выводить полученную строку в консоль.  
2. Метод readString класса ConsoleHelper должен возвращать строку считанную с консоли.  
3. Метод getAllDishesForOrder класса ConsoleHelper должен возвращать список блюд выбранных пользователем.  
4. В конструкторе класса Order список dishes должен быть инициализирован результатом работы метода getAllDishesForOrder.  
5. Метод toString в классе Order должен возвращать содержимое заказа согласно условию задачи.  
6. Поле logger в классе Tablet должно быть инициализировано с помощью метода Logger.getLogger() с именем текущего класса в качестве параметра.  
7. В методе createOrder класса Tablet должен быть создан новый заказ, в качестве параметра передай текущий планшет.  
8. В случае возникновения IOException в процессе создания заказа - в лог должно быть записано сообщение "Console is unavailable.", уровень - SEVERE.  
9. В методе main класса Restaurant должен быть создан новый планшет и с его помощью создан новый заказ.  
10. Метод toString в классе Tablet должен возвращать информацию о планшете (смотри пример в задании).