**题目：**

小王准备使用面向对象的方法设计一个快餐店的简单游戏，游戏中有顾客、服务员、菜品和厨师。每个顾客都有一个服务员帮助点菜，并且可以点多个菜；每道菜都由指定厨师制作，不同的菜可能由不同的厨师制作；顾客跟服务员点完菜后，服务员通知后厨做菜。请你针对上面的描述，帮助小王选择合适的设计模式进行设计。

1)简要说明你的设计思路和选择的模式。

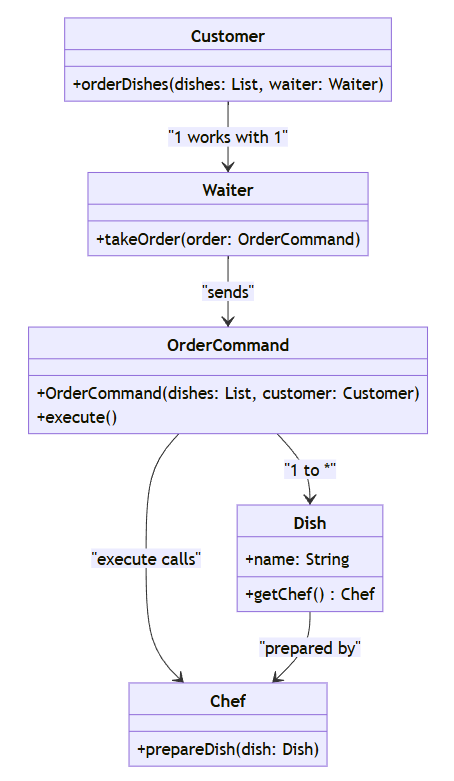
2)给出你设计的UML类图和实现代码。

**设计思路：**

根据描述，可以将该快餐店游戏设计为一个系统，其中顾客、服务员、菜品和厨师都对应为独立的类。为了解决各个角色之间的职责划分和交互流程，可以采用**命令模式**。

命令模式将请求封装为对象，从而使得我们可以将不同的请求（点菜、做菜）参数化，并使得请求的发送者（服务员）和接收者（厨师）解耦。顾客下单后将每个订单封装为一个“命令”，由服务员发送给厨师。厨师接收命令后，按照命令内容制作相应的菜品。

**类图：**



|  |
| --- |
| **代码** |
| import java.util.ArrayList;  import java.util.List;  *// 顾客类*  class Customer {      private String name;      public Customer(String name) {          this.name = name;      }      public void orderDishes(List<Dish> dishes, Waiter waiter) {          OrderCommand order = new OrderCommand(dishes, this);          waiter.takeOrder(order);      }      public String getName() {          return name;      }  }  *// 服务员类*  class Waiter {      public void takeOrder(OrderCommand order) {          order.execute();      }  }  *// 菜品类*  class Dish {      private String name;      private Chef chef;      public Dish(String name, Chef chef) {          this.name = name;          this.chef = chef;      }      public void prepare() {          chef.prepareDish(this);      }      public String getName() {          return name;      }  }  *// 厨师类*  class Chef {      private String name;      public Chef(String name) {          this.name = name;      }      public void prepareDish(Dish dish) {          System.out.println(name + " is preparing " + dish.getName());      }      public String getName() {          return name;      }  }  *// 订单命令类*  class OrderCommand {      private List<Dish> dishes;      private Customer customer;      public OrderCommand(List<Dish> dishes, Customer customer) {          this.dishes = dishes;          this.customer = customer;      }      public void execute() {          System.out.println("Order for " + customer.getName() + " is being processed.");          for (Dish dish : dishes) {              dish.prepare();          }      }  }  *// 测试类*  public class RestaurantSimulation {      public static void main(String[] args) {  *// 创建厨师*          Chef chef1 = new Chef("Chef A");          Chef chef2 = new Chef("Chef B");  *// 创建菜品*          Dish dish1 = new Dish("Burger", chef1);          Dish dish2 = new Dish("Pasta", chef2);  *// 创建顾客和服务员*          Customer customer = new Customer("Alice");          Waiter waiter = new Waiter();  *// 顾客点菜*          List<Dish> dishes = new ArrayList<>();          dishes.add(dish1);          dishes.add(dish2);          customer.orderDishes(dishes, waiter);      }  } |