TASK 4: REPORT

Matthew Els u21715191

Dominique da Silva u21629944

Tlhalefo Dikolomela u21507792

Tinashe Austin u21564176

Christopher Katranas u19154853

Table of Contents

[INTRODUCTION 2](#_Toc149720372)

[4.1 RESEARCH 2](#_Toc149720373)

[4.2 DESIGN DECISIONS 4](#_Toc149720374)

[4.3 WRITE UP OF DESIGN PATTERNS 4](#_Toc149720375)

[4.3.1 BUILDER 4](#_Toc149720376)

[4.3.2 CHAIN OF RESPONSIBILITY 4](#_Toc149720377)

[4.3.3 STATE 4](#_Toc149720378)

[4.3.4 STRATEGY 4](#_Toc149720379)

[4.3.5 DECORATOR 4](#_Toc149720380)

[4.3.6 COMMAND 4](#_Toc149720381)

[4.3.7 TEMPLATE METHOD 4](#_Toc149720382)

[4.3.8 4](#_Toc149720383)

[4.3.9 4](#_Toc149720384)

[4.3.10 4](#_Toc149720385)

[4.4 ASSUMPTIONS 5](#_Toc149720386)

[4.5 SUPPORTING UML DIAGRAMS 6](#_Toc149720387)

# INTRODUCTION

Our (Concrete pass) implementation of the given assignment focuses on the implementation of a restaurant simulation system. Customers are seated at a table, orders are taken by the waiter, and the food is prepared and served.

The primary components we have focused on, in our system, is the floor where customers interact with the waiters and are managed by the staff, as well as the kitchen where food orders are prepared and dispatched.  
Customer satisfaction also plays a role in this simulation, influencing factors such as tipping of their assigned waiter.

Customers also can create customized orders from a menu and may choose to start tabs for deferred payment.

# 4.1 RESEARCH

GitHub. (n.d.). *Grokking-OOD/object-oriented-design-case-studies/design-a-restaurant-management-system.md at master · wyaadarsh/Grokking-OOD*. [online] Available at: https://github.com/wyaadarsh/Grokking-OOD/blob/master/object-oriented-design-case-studies/design-a-restaurant-management-system.md#class-diagram [Accessed 1 Nov. 2023].

‌

Shahid, M. (2023). *Restaurant-Management-System*. [online] GitHub. Available at: https://github.com/mabbia706/Restaurant-Management-System/tree/master [Accessed 1 Nov. 2023].

‌

www.c-sharpcorner.com. (n.d.). *Food Delivery Application Using Design Patterns*. [online] Available at: https://www.c-sharpcorner.com/article/food-delivery-application-using-with-design-patterns/.

‌

Doyle, W. (2023). *Design Patterns Explained with Food 🥕*. [online] GitHub. Available at: https://github.com/wesdoyle/design-patterns-explained-with-food/tree/main [Accessed 1 Nov. 2023].

‌

Kothari, A. (2018). *Java Builder Design Pattern Example - Java Code Geeks*. [online] Examples Java Code Geeks. Available at: https://examples.javacodegeeks.com/java-development/core-java/java-builder-design-pattern-example/ [Accessed 1 Nov. 2023].

‌

# 4.2 DESIGN DECISIONS

# 4.3 WRITE UP OF DESIGN PATTERNS

## 4.3.1 BUILDER

The Builder design pattern's basic intent is to separate the construction of a complex object from its representation, allowing the same construction process to create different representations.

A pizzeria has many different types of pizzas (representations), however the basic construction remains the same up until toppings are to be chosen for the final product.  
Thus we have chosen the builder design pattern to be able to create many different types of pizzas while simplifying the construction process.

## 4.3.2 CHAIN OF RESPONSIBILITY

The Chain of Responsibility design pattern is intended to create a chain of objects, where each object can process a request and decide whether to pass it to the next object in the chain or to stop processing it. It allows you to decouple the sender of a request from its receiver, providing multiple objects the opportunity to handle the request.  
With this in mind, the decision was made to pass the order made by the table through a chain. Multiple classes have to handle the request in different ways.  
In our design the order made (the request made by the customer table) is passed from the customer table, to the waiter. The waiter then passes it to the kitchen, and eventually the order is passed to the head chef to plate, and then sent back to the waiter to serve the table with their meal.

## 4.3.3 STATE

## 4.3.4 STRATEGY

## 4.3.5 DECORATOR

## 4.3.6 COMMAND

## 4.3.7 TEMPLATE METHOD

## 4.3.8

## 4.3.9

## 4.3.10

# 4.4 ASSUMPTIONS

# 4.5 SUPPORTING UML DIAGRAMS

(The idea is to include all the design patterns under this heading and refer to the figures in 4.3)

(We can just instant reverse our code and use a snipping tool to get each pattern’s UML class diagram)

Possibly also include a link to the entire system design UML Class diagram.