

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**КУРСОВА РОБОТА**

з дисципліни “Програмування 1”

тема “**Шаблони проєктування в ООП. Піцерія**”

|  |  |  |
| --- | --- | --- |
| Виконала  студентка IІ курсу  групи КП-02  Гаценко М.А. |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  Керівник роботи  доцент, к.т.н. Заболотня Т.М.. |

Київ 2022

**ЗМІСТ**

|  |  |
| --- | --- |
| **Вступ** | 3 |
| 1. **Структурно-алгоритмічна організація програмної системи** | 5 |
| 1.1 Модульна організація програми | 5 |
| 1.2 Функціональні характеристики | 6 |
| 1. **Програмна реалізація системи за допомогою шаблонів проєктування** | 8 |
| 2.1 Обґрунтування вибору та опис шаблонів проєктування для програмної реалізації | 8 |
| 2.2 Опис результатів роботи програми | 15 |
| **Висновки** |  |

**ВСТУП**

Дана курсова робота призначена розробці програмного забезпечення для піцерії. Сьогодні ні одному бізнесу неможливо обійтися без свого власного додатку, адже ми живемо у час діджиталізації. Саме з метою розробки прототипу такого додатку, була написана ця робота. Завдяки даному додатку, як клієнти, так і робітники зможуть більше ефективно керувати своєю комунікацією з піцерією: клієнти зможуть робити замовлення прямо в своєму гаджеті, а для робітників представлений зручний інтерфейс для запису замовлення клієнтів та подальшої роботи з ними.

Через використання мови програмування С# та фреймворку .Net Framework, даний застосунок є дійсно актуальним. Є можливість підтримувати його роками, без втрати його продуктивності та сучасності.

Тематика курсової роботи була обрана через те, що результати абстрагування об’єктів у цій предметній галузі дозволяють застосувати вивчені принципи та методи об’єктно-орієнтованого програмування для створення програмного забезпечення, зокрема шаблони проєктування.

*Об’єктом* дослідження є процес роботи піцерії.

*Метою роботи* є розроблення програмного забезпечення додатку з використанням шаблонів проєктування.

Для досягнення визначеної мети необхідно виконати такі *завдання*:

* абстрагувати об’єкти предметної галузі;
* розробити структурну організацію ПЗ за допомогою застосування основних принципів ООП та шаблонів проєктування;
* визначити та описати функціональні характеристики програми;
* обґрунтувати вибір шаблонів проєктування, використаних для побудови програми;
* розробити логіку та дизайн додатку;
* виконати реалізацію програмного забезпечення відповідно до вимог технічного завдання;
* виконати тестування розробленої програми;
* оформити документацію з курсової роботи.

Розроблення програмного забезпечення складається з двох основних модулів: модуль інтерфейсу користувача та модуль бізнес логіки.

Реалізовані шаблони проєктування: Міст, Стан, Шаблонний метод, Команда та Декоратор.

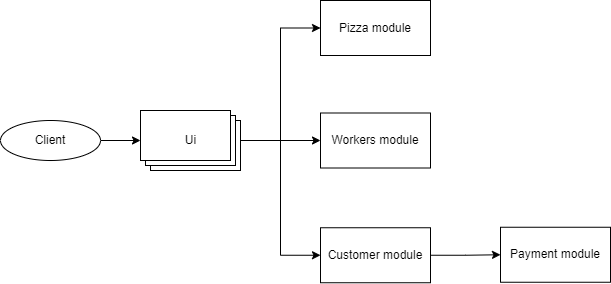
До функціональних можливостей програми належать: створення та редагування піц, сплата рахунку, призначення шефа на кухні тощо.

Для функціонування розробленої програми необхідно забезпечити наявність на комп’ютері 64 Кб вільного дискового простору.

Розроблене програмне забезпечення може бути використане у компаніях, які працюють з піцеріями.

Пояснювальна записка складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел (6-ти найменувань, з них 2 – іноземною мовою). Робота містить 19 рисунок. Загальний обсяг роботи – 24 друкованих сторінки.

1. **Структурно-алгоритмічна організація програмної системи**
   1. **Модульна організація програми**



*Рис. 1.1.1. Модульна організація програми*

Модуль **UI** є посередником між бізнес логікою додатку та користувачем. Іншими словами, даний модуль є інтерфейсом користувача. У нього входять наступні класи:

* HomeForm – клас головного вікна, яке бачить користувач при старті програми. Від даного вікна користувач може доступатися до всіх інших частин програми.
* CreateBasePizzaForm – клас вікна, яке дозволяє створити основу для піци. Це перше вікно, яке бачить користувач після натискання на кнопку «Add pizza» на головному вікні.
* PickPizzaForm – клас вікна, що дозволяє вибрати тип піци.
* PickToppingsForm – клас вікна, що дозволяє додати чи забрати різні начинки піци.
* MenuForm – клас вікна меню піцерії. Користувач може побачити всі позиції і їх ціни, які наявні у меню.
* PickChefForm – клас вікна, що дозволяє вибрати шефа, який буде готувати піци. Від типу шефа залежить якість піци.
* SelectPaymentTypeForm – клас вікна для вибору типу оплати, яке активне тільки при наявності хоча б однієї позиції в замовленні.
* ErrorPopupForm – клас вікна, що показує помилку.

**Pizza module** керує створенням та редагуванням піц. Він складається з наступних класів:

* Pizza – абстрактний клас піци, який дозволяє створювати екземпляри класів
* Класи, що наслідують клас Pizza
* Decorator – клас для додавання інгредієнтів до піци
* Dough – абстрактний клас для отримання екземпляра тіста
* Класи, що наслідують клас Dough
* Ingredients – абстрактний клас, який є батьком для наступних класів:
  + Dough
  + Sause
  + Stuffing
* Sause – абстрактний клас для отримання екземпляру соусу
* Класи, що наслідують клас Sause
* Stuffing – абстрактний клас для отримання екземпляру начинки

**Workers module** керує працівниками. Він складається з абстрактного класу працівників та конкретних класів шеф поварів та касирів.

**Customer module** керує клієнтами, їх оплатою замовлень тощо. Він складається з абстрактного класу клієнтів.

**Payment module** керує оплатою замовлень. Він складається з абстрактного класу способів оплати та конкретних класів типів оплати.

* 1. **Функціональні характеристики**

Щоб запустити додаток піцерії, користувач повинен мати файл Pizzeria.exe, який автоматично відкриє вікно додатку, яким користувач зможе користуватися.

Після запуску, користувача зустрічає головне вікно з наступними кнопками:

* Menu – подивитися меню піцерії, яке містить список піц з цінами (ціни указані для піци з тістом з борошна та з томатним соусом) та список можливих начинок, які можна додавати до будь-якої піци. Для перегляду меню піц, треба натиснути кнопку «Pizzas», а для перегляду меню начинок – кнопку «Toppings»
* Add pizza – додати піцу до замовлення. Спочатку користувач повинен вибрати тип тіста та соус (обов’язково), потім – вид піци (обов’язково), після чого є можливість додати начинку або зробити піцу безлактозною за бажанням
* Pay – сплатити замовлення за допомогою кредитної картки, дебітової картки та готівки. Для вибору типу сплати, потрібно натиснути одну з трьох кнопок:
  + Credit card
  + Debit card
  + Cash

Зверху від цих кнопок знаходиться сума замовлення, а знизу треба ввести баланс користувача. Якщо була вибрана готівка, як тип оплати, на вікно виведеться решта, яку треба віддати клієнту, після чого для виходу треба натиснути кнопку «Ok».

* Pick chef – призначити шефа, який зараз готує на кухні. Для вибору шефа, треба натиснути на одну з трьох кнопок:
  + Top chef
  + Regular chef
  + Trainee chef

Окрім кнопок, у головному меню також є список замовлення, в якому відображаються назви піц та їх начинки.

1. **Програмна реалізація системи за допомогою шаблонів проєктування**
2. **Обґрунтування вибору та опис шаблонів проєктування** **для програмної реалізації**
3. **Міст**

Це структурний шаблон проєктування, що дозволяє розділяти абстракцію і реалізацію таким чином, щоб вони могли змінюватися незалежно. Шаблон «Міст» використовує інкапсуляцію, агрегування та успадкування для того, щоб розділити відповідальність між класами. Діаграма класів представлена на рис. 2.1.1.

Шаблон працює так, що у клас, поведінку якого ми хочемо змінити, ми додаємо поле абстрактного класу, який буде мати нащадків, що зможуть змінювати логіку методу батьківського класу. При виклику основного класу, нам потрібно дати йому об’єкт моста, після чого, при виклику методу з основного класу, об’єкт моста викликатиме свою реалізацію цього методу.



*Рис. 2.1.1. Діаграма класів, які входять до шаблону «Міст»*

*Учасники шаблону:*

* **Client** – клієнт (частина коду, програма), який звертається до шаблону
* **Customer** – абстракція, яка містить екземпляр імплементору. Має метод «PayForOder», який викликає відповідний метод імплементора, що знаходиться всередині абстракції.
* **PaymentType** – абстрактний клас, що має поле балансу користувача, метод перегляду цього балансу та метод сплати.
* **Specific** **payment** **type** – набір класів, що наслідують клас PaymentType і реалізують свої методи «Pay»

*Результати використання конструкції:*

* Користувач може сплатити будь-яким видом оплати
* Відсутність перевірок на вид оплати, а значить і швидше виконання
* Можливість додавати нові види оплати
* Висока абстракція

*Обґрунтування використання шаблону:*

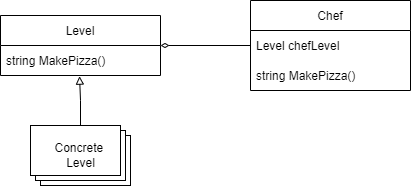
Шаблон міст було обрано через те, що клієнти повинні платити за замовлення, проте у кожного окремого клієнта може бути свій спосіб оплати. Щоб не додавати зайві методи або класи-наслідники, керування якими ставало би складніше і складніше зі збільшенням кількості методів оплати, було вирішено сам спосіб оплати та реалізацію методу Pay винести у окремий клас, який матиме нащадків, що зможуть наділять даний метод конкретною реалізацією.

Тобто, даний шаблон був обраний для можливості динамічної та непомітної імплементації різних способів оплати, якими може скористатися клієнт. При цьому, кожен спосіб має свою реалізацію, що включає у себе комісію, яку платить клієнт. Крім того, даний шаблон дозволить додавати нові способи оплати у майбутньому без ризиків пошкодження роботоздатності програми.

1. **Стан**

Це поведінковий шаблон, який дозволяє об’єкту варіювати свою поведінку залежно від внутрішнього стану. Стан передбачає реалізацію поведінки, асоційованої з певним станом об’єкту, а також забезпечення зміни поведінки відповідно до зміни внутрішнього стану.

Шаблон працює так, що спочатку створюється абстрактний клас стану, після чого – його нащадки, кожен з яких реалізує окрему поведінку в залежності від того стану, який він презентує. Також повинен бути клас контексту, який матиме поле стану та метод, логіка якого буде залежати від певного стану. Для зміни логіки в цьому методі за допомогою об’єкту стану викликається його метод.



*Рис. 2.1.2. Діаграма класів, які входять до шаблону «Стан»*

*Учасники шаблону:*

* **Level** (state) – абстрактний клас, який є рівнем навичок, що може мати шеф повар. Даний клас має абстрактний метод MakePizza, який повинні перевизначати наслідники
* **Concrete level –** наслідники абстрактного класу Level, які реалізують конкретну поведінку методу MakePizza
* **Chef** (context) – клас, який наслідує абстрактний клас Worker. Даний клас має в собі метод MakePizza, що викликає реалізацію з класів-наслідників класу Level завдяки використанню поля chefLevel.

*Результати використання конструкції:*

* Програма не витрачає час на зайві перевірки того, який рівень навичок у шефа, який зараз знаходиться на кухні.
* З’являється можливість динамічно змінювати стан об’єкту прямо під час виконання програми
* Додавання нових станів становиться легшою операцією

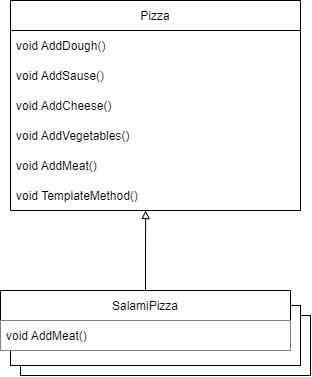
*Обґрунтування використання шаблону:*

Цей шаблон було обрано через необхідність мати окрему поведінку для різних рівнів шеф кухарів: головний шеф буде готувати неймовірну піцу, звичайний шеф – нормальну тощо. В залежності від рівня шефа, піца матиме різну якість. Даний шаблон був обраний з метою динамічної підстановки потрібних алгоритмів до методу MakePizza, класу Chef. Шаблон стан дозволяє уникнути зайвих перевірок щодо того, якого рівня навички шефа, а також дозволяє міняти поведінку одного й того самого шефа прямо під час виконання програми.

1. **Шаблонний метод**

Поведінковий шаблон, який визначає функціональність конкретних методів в рамках лише абстрактних сутностей. Визначає основу алгоритму та дозволяє підкласам перевизначити деякі кроки алгоритму, не змінюючи структуру в цілому.

Шаблонний метод працює так, що у абстрактному класі створюються методи-кроки, які потрібні для створення об’єкту і основний метод, який збирає ці кроки у одному місці. Для створення об’єкта викликається основний метод, який, в свою чергу, викликає методи-кроки. Класи-наслідники абстрактного класу можуть перевизначати логіку методів-кроків, але не логіку основного метода.



*Рис. 2.1.3. Діаграма класів, які входять до шаблону «Шаблонний метод»*

*Учасники шаблону*:

* **Pizza** – абстрактний клас піци, який містить 5 методів-кроків для додавання інгредієнтів, та один шаблонний метод, який викликає ці методи
* **Concrete pizza** – класи-наслідники, які можуть перевизначати методи-кроки свого батьківського класа.

*Результати використання конструкції:*

* Можливість легко і точно перевизначати конкретні кроки у приготуванні піци

*Обґрунтування використання шаблону:*

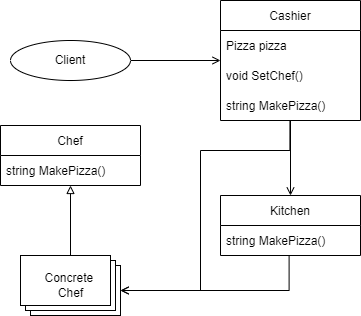
Шаблонний метод було обрано через те, що створення піци є процесом, який складається з декількох кроків: приготування тіста, додавання соусу, сиру, м’яса та овочів. Через те, що у нас є класи-наслідники класу Pizza, які репрезентують конкретний вид піци, нам потрібно перевизначати ці кроки: одна піца буде мати ковбасу, інша взагалі буде вегетаріанською. Через розподілення процесу додавання інгедієнтів на кроки, набагато легше слідкувати, щоб у вегетаріанську піцу не потрапило м’ясо, у безлактозну – звичайні сири тощо. Замість того, щоб мати один метод додавання всіх інгредієнтів, де можна легко заплутатися та пропустити певний інгредієнт, набагато легше розділити створення на декілька кроків і перевизначати тільки те, що потрібно. Крім того, при наявності одного методу, буде постійне повторювання коду, тоді коли шаблонний метод дозволяє змінювати тільки окремі кроки.

Тобто, даний шаблон було вибрано для легшого керування створенням об’єктів піци, а також для уникнення дублікатів коду у одних й тих же самих методах різних класів.

1. **Команда**

Це поведінковий шаблон, який інкапсулює різні алгоритми в єдину сутність, завдяки чому можна параметризувати клієнтів різними запитами, вести історію виконаних операцій та підтримувати скасування операцій.

Шаблон «Команда» працює так, що клієнт звертається до invoker із певним запитом. Клас invoker містить посилання на команду, метод встановлення виконавця і основний метод, що буде виконувати команда. Сама команда теж містить основний метод і посилання на виконавця, який і викликається під час виконання основного методу. Клас виконавця та його наслідники реалізують конкретну поведінку основного метода.



*Рис. 2.1.4. Діаграма класів, які входять до шаблонів «Команда»*

*Учасники шаблону*:

* **Client** – клієнт (частина коду, програма), який звертається до шаблону
* **Cashier** (Invoker) – клас, який приймає запит на команду (створення піци). Цей клас містить посилання на команду, метод установлення шефа та метод створення піци
* **Kitchen** (Command) – клас-команда, який містить метод створення піци
* **Concrete pizza** – набір класів-наслідників класу Pizza
* **Chef** (Executor) – виконавець команди. Цей клас містить метод створення піци
* **Specific chef** – набір класів-наслідників класу Chef

*Результати використання конструкції:*

* Передача об’єкту команди як параметра
* Висока абстракція класів-виконавців
* Можливість додавати нові клас-виконавці без ризиків ушкодити програму

*Обґрунтування використання шаблону:*

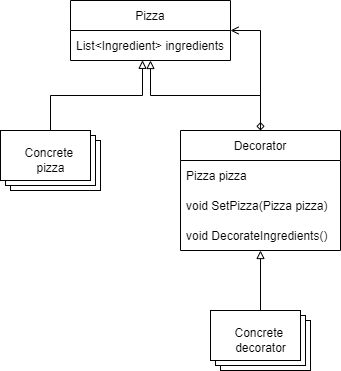
Шаблон «Команда» був обраний через те, що касиру, який приймає замовлення, потрібно передавати вказівки кухарам щодо того, що і як робити. Кухари, в свою чергу, будуть виконувати запити касира в залежності від їх навичок (встановлення навичок кухарів виконується завдяки шаблону Стан, що був розглянутий вище).

Тобто, даний шаблон було використано для передачі команди виконавцям з більшою абстракцією. Використання цього шаблону дозволить передавати запити на створення піци як параметр.

1. **Декоратор**

Це структурний шаблон проєктування, який призначений для динамічного підключення додаткових станів та поведінки до об’єкта та є гнучкою альтернативою практиці створення підкласів з метою розширення функціональності.

Шаблон працює так, що цільовий об’єкт розміщується у іншому об’єкті-обгортці, який запускає базову поведінку об’єкта, а потім додає до результату щось своє.



*Рис. 2.1.5. Діаграма класів, які входять до шаблону «Декоратор»*

*Учасники шаблону*:

* **Pizza** (component) – абстрактний клас-компонент, що буде декоруватися
* **Concrete pizza** – класи-наслідники класу Pizza
* **Decorator** – абстрактний клас-наслідник декоратор, що додає та видаляє інгредієнти з піци
* **Concrete decorator** – класи-наслідники класу Decorator

*Результати використання конструкції:*

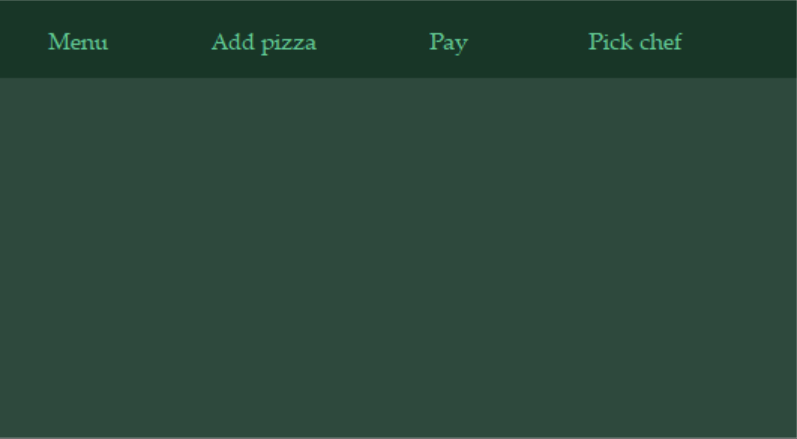
* Додавання інгредієнтів до піци виноситься окремо, що дозволяє без перешкод маніпулювати вмістом піци під час її створення
* Додавання нових декораторів є простою задачею

*Обґрунтування використання шаблону:*

Даний шаблон було обрано через те, що під час створення піци у користувача є можливість маніпулювати її начинками. Шаблон «Декоратор» спрямований на динамічну зміну об’єкту. Під час додавання чи прибирання інгредієнтів піци змінюється і ціна. Дані дії було б доволі складно зробити без використання цього шаблону, адже до кожного типу піци довелося би робити свої класи-наслідники або методи для додавання кожного окремого інгредієнту, що збільшило би код у декілька разів і було б незручно у використані. Саме через вищезазначені проблеми було вирішено використати шаблон «Декоратор».

1. **Опис результатів роботи програми**

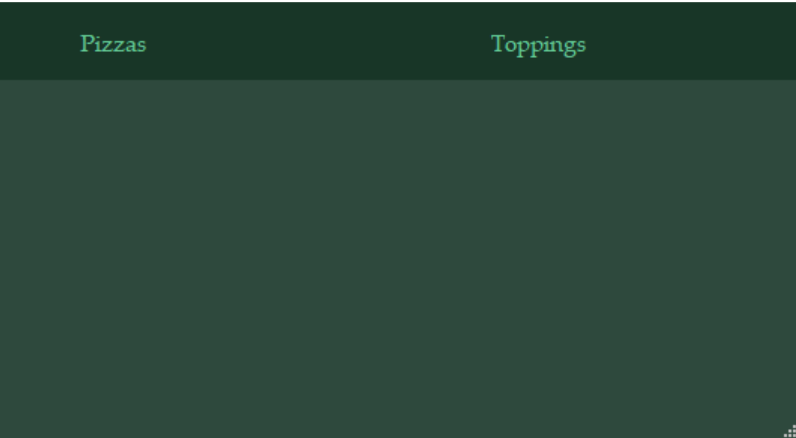
При запуску додатку нас зустрічає наступне вікно:



*Рис. 2.3.1. Головна сторінка*

Вікно головної сторінки має чотири кнопки: Menu, Add pizza, Pay та Pick chef. Крім цих кнопок, дана сторінка має текстове поле, яке можна побачити при додаванні хоча б однієї позиції до замовлення.

При активації кнопки «Menu», користувач побачить наступну сторінку:

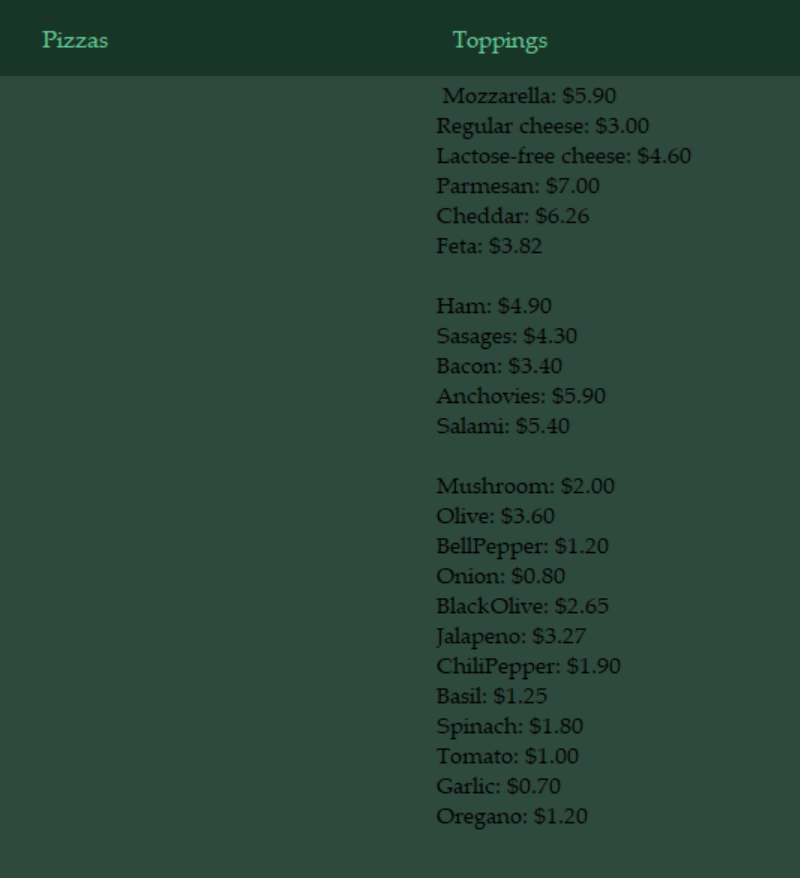


*Рис. 2.3.2. Вікно сторінки menu*

Дана сторінка містить дві кнопки: Pizzas та Toppings. При натисканні кнопки Pizzas, користувач побачить меню піц зі всіма позиціями та їх цінами. При натисканні кнопки Toppings, користувач побачить меню начинок зі всіма позиціями та їх цінами:

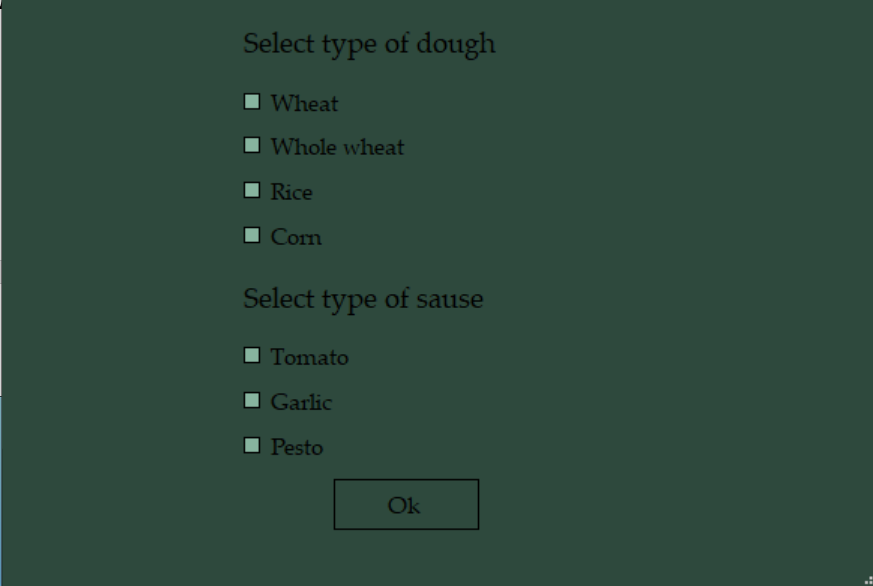


*Рис. 2.3.3. Вікно меню піц*



*Рис. 2.3.4. Вікно меню начинок*

Для додавання піци до замовлення, на головній сторінці треба натиснути на кнопку «Add pizza»:

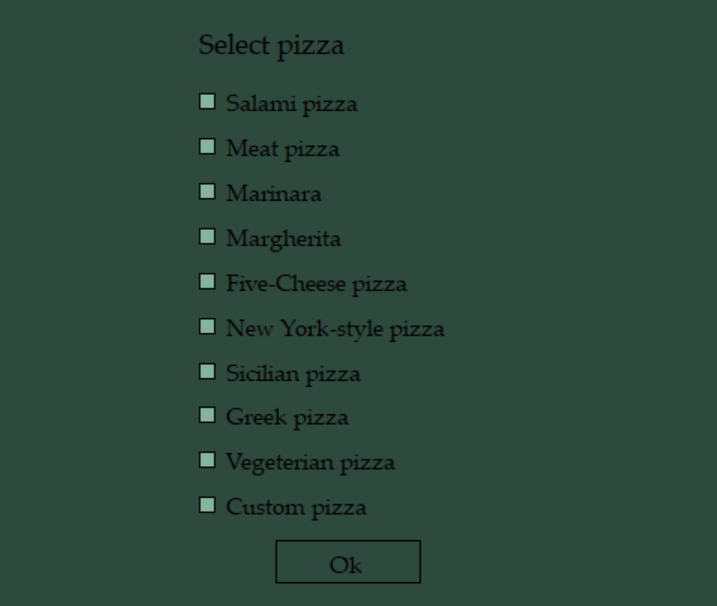


*Рис. 2.3.5. Вікно вибору тіста та соусу*

На цій сторінці користувач повинен вбрати по одній позиції з двох категорій: тіста, на якому буде готуватися піца та соус, який піде на піцу. При виборі одного пункту, інші пункти з тієї ж категорії деактивуються, щоб унеможливити вибір одночасно декількох видів тіста та соусів.

Якщо користувач натисне кнопку Ok, не вибравши позиції хоча в одній категорії, він побачить вікно помилки і буде відправлений на головну сторінку.

Після обов’язкового вибору одного виду тіста та одного виду соусу, користувач повинен натиснути кнопку Ok, яка перенесе його на окреме вікно вибору типу піци:



*Рис. 2.3.6. Вікно вибору виду піци*

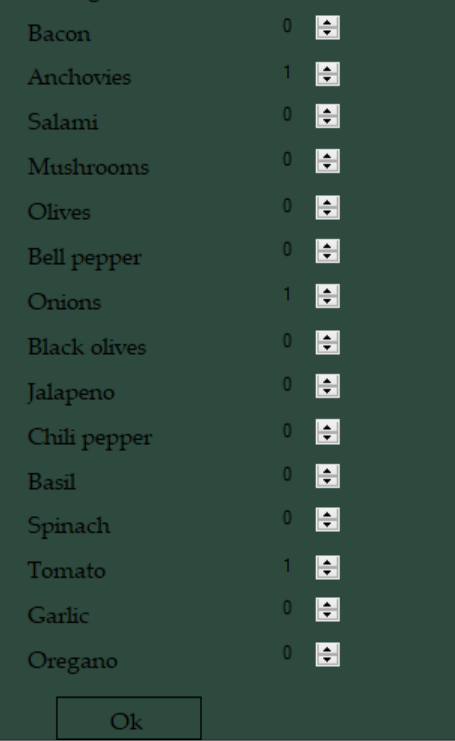
На цій сторінці користувач повинен вбрати одну позицію. При виборі одного пункту, інші пункти деактивуються, щоб унеможливити вибір одночасно декількох видів піц.

Якщо користувач натисне кнопку Ok, не вибравши позиції, він побачить вікно помилки і буде відправлений на головну сторінку.

Після обов’язкового вибору виду піци, користувач повинен натиснути кнопку Ok, яка перенесе його на окреме вікно додавання чи видалення начинок:



*Рис. 2.3.7. Вікно додавання та видалення начинок з піци*

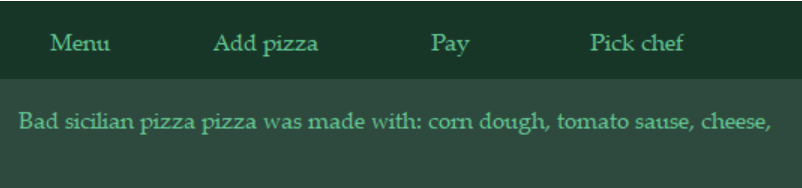


*Рис. 2.3.8. Вікно додавання та видалення начинок з піци*

Для додавання інгредієнтів, користувачу потрібно натиснути на стрілочку вгору напроти потрібного інгредієнта. Для видалення інгредієнту, користувачу потрібно натиснути на стрілочку вниз напроти потрібного інгредієнта.

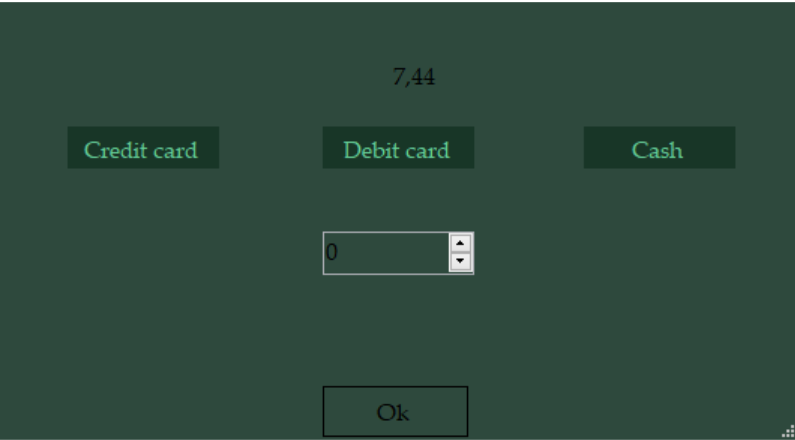
Для того, щоб зробити піцу безлактозною, користувач повинен натиснути на чек бокс «Lactose-free pizza».

Після підтвердження вибору піци шляхом натискання кнопки «Ok», на головній сторінці користувач побачить своє замовлення:



*Рис. 2.3.9. Головне вікно з замовленням*

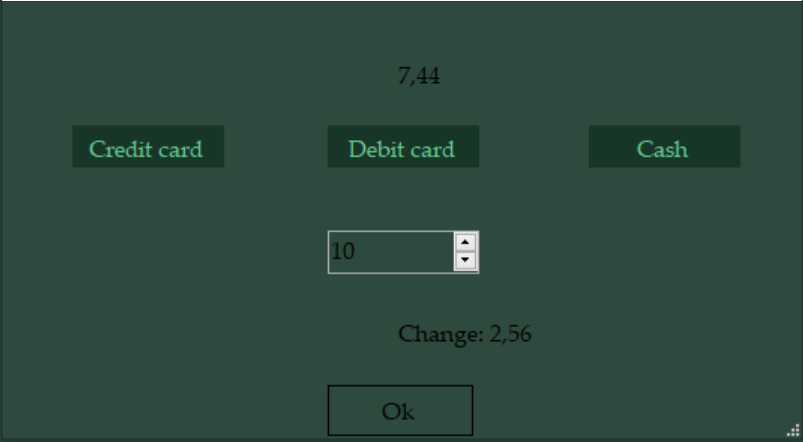
Для оплати замовлення, потрібно натиснути кнопку «Pay», що перекине нас на наступне вікно:



*Рис. 2.3.10. Вікно сплати замовлення*

Для сплати, користувач повинен внести суму, яка є на балансі клієнта та натиснути на спосіб оплати. Якщо користувач вибере готівку, то на екрані з’явиться сума решти, після чого користувач повинен буде натиснути кнопку Ok. Якщо ж оплата буде проходити карткою, одразу після вибору користувача переправить на головну сторінку.

Якщо при виборі готівки або дебетової картки ввести баланс, що є менше за суму замовлення, програма видасть помилку і відправить користувача на головну сторінку.



*Рис. 2.3.11. Вікно сплати замовлення з рештою*

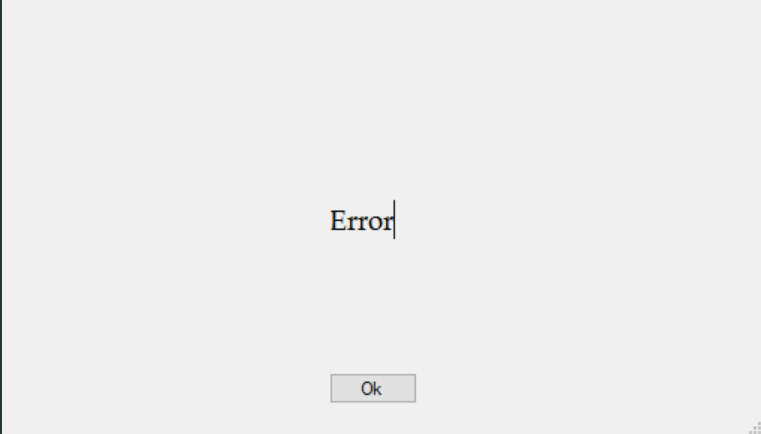
Для вибору шефу, потрібно натиснути на кнопку «Pick chef»:



*Рис. 2.3.12. Помилка доступу*

Вибір шефу впливає на якість піци. Якщо шефа не було обрано, якість піци буде поганою.

Також є вікно помилки:



*Рис. 2.3.13. Вікно помилки*

**ВИСНОВКИ**

Метою даної курсової роботи було розроблення програмного забезпечення додатку з використанням шаблонів проєктування. Підставою для розроблення стало завдання на виконання курсової роботи з дисципліни «Програмування 1» студенткою ІІ курсу кафедри ПЗКС НТУУ «КПІ ім. І.Сікорського».

Для досягнення поставленої мети у повному обсязі виконано завдання, визначені у аркуші завдання на курсову роботу; розроблено графічні матеріали; реалізовано всі вимоги до програмного продукту, програмного та апаратного забезпечення, наведені у технічному завданні; створено відповідну документацію.

Розроблене програмне забезпечення дозволяє керувати програмою піцерії, що включає до себе: додавання піц до замовлень, додавання та віднімання інгредієнтів під час створення замовлення, оплата замовлення, визначення шеф кухара на кухні та перегляд меню піцерії.

Програму створено на основі використання шаблонів проєктування: зокрема до структури розробленого програмного забезпечення входить реалізація п’яти шаблонів, які належать до різних груп шаблонів проєктування.

Для розроблення програмного забезпечення використано мову програмування С#. Платформи: Windows, Visual Studio Code.

Перспективним напрямком подальшого дослідження даної тематики є розширення функціоналу даного додатку шляхом додання підтримки багатопотоковості та асинхронних обробників.

**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. Приёмы объектно-ориентированного проектирования. Паттерны проектирования / Эрих Гамма, Ричард Хелм, Ральф Джонсон, Джон Влиссидес. – 1994. – 395 с.
2. Шаблоны проектирования. Новый подход к объектно-ориентированному анализу и проектированию / Алан Шаллоуей, Джеймс Р. Тротт. — М. : «Вильямс», 2002. — 288 с.
3. Крэг Ларман. Применение UML 2.0 и шаблонов проектирования. — М.: «Вильямс», 2006. — С. 736.
4. Пышкин, Е.В. Основные концепции и механизмы объектно-ориентированного программирования [Текст] / Е.В. Пышкин – СПб. : БХВ-Петербург, 2005. – 640 с.
5. Design Patterns [Електронний ресурс]: [Веб-сайт]. - Режим доступу: https://sourcemaking.com/design\_patterns (дата звернення 26.04.2022) - Назва з екрана
6. Software Design Patterns [Електронний ресурс]: [Веб-сайт]. - Режим доступу: https://www.geeksforgeeks.org/software-design-patterns (дата звернення 26.04.2022) - Назва з екрана