Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №6

Технології розроблення програмного забезпечення

«*ШАБЛОНИ «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator»*»

Варіант 29

Виконав:
студент групи ІА-13
Хілько І.А.

Перевірив:
Мягкий М. Ю.

Київ 2023

Тема: Шаблони Abstract factory, Factory method, Memento, Observer, Decorator.

Варіант:
Система для колективних покупок. (State, Chain of responsibility, **Abstract factory**, Mediator, Composite, Client-server).

Система дозволяє створити список групи для колективної покупки, список що потрібно купити з орієнтовною вартістю кожної позиції та орієнтовною загальною вартістю, запланувати хто що буде купляти. Щоб користувач міг відмітити що він купив, за яку суму, з можливістю прикріпити чек. Система дозволяє користувачу вести списки бажаних для нього покупок, з можливістю позначати списки, які будуть доступні для друзів (як списки, що можна подарувати користувачеві). Система дозволяє добавляти інших користувачів в друзі.

Хід роботи:
1. Реалізувати не менше 3-х класів та шаблон Abstract factory.

Abstract factory – це породжувальний патерн проектування, що дає змогу створювати сімейства пов’язаних об’єктів, не прив’язуючись до конкретних класів створюваних об’єктів.

AbstractDatabaseFactory - абстрактний клас фабрики бази даних. Він містить два абстрактних методи: create_connection і create_cursor. Ці методи будуть реалізовані в конкретних фабриках.

SQLiteDatabaseFactory - конкретна фабрика для роботи з SQLite базою даних. Вона наслідує AbstractDatabaseFactory і реалізує методи create_connection та create_cursor для SQLite.

PostgreSQLDatabaseFactory - ще одна конкретна фабрика, але для роботи з PostgreSQL базою даних. Вона також наслідує AbstractDatabaseFactory і має свою власну реалізацію методів create_connection та create_cursor.

CompositeDatabaseFactory - композитна фабрика, яка може об'єднувати кілька інших фабрик. Вона має метод add_factory, який дозволяє додавати інші фабрики до списку. Методи create_connection і create_cursor викликають відповідні методи всіх доданих фабрик, створюючи тим самим з'єднання та курсори для всіх баз даних, які представлені в композитній фабриці.

```
class AbstractDatabaseFactory(abc.ABC):
    @abc.abstractmethod
    def create_connection(self):
        pass
    @abc.abstractmethod
    def create_cursor(self, connection):
        pass

class SQLiteDatabaseFactory(AbstractDatabaseFactory):
    def create_connection(self, database_name='database.db'):
        return sqlite3.connect(database_name)
    def create_cursor(self, connection):
        return connection.cursor()

class PostgreSQLDatabaseFactory(AbstractDatabaseFactory):
    def create_connection(self, database_name='database', user='user',
password='password', host='localhost', port=5432,
                          psycopg2=None):
        return psycopg2.connect(database=database_name, user=user, password=password,
host=host, port=port)
    def create_cursor(self, connection):
        return connection.cursor()

class CompositeDatabaseFactory(AbstractDatabaseFactory):
    def __init__(self):
        self.factories = []
    def add_factory(self, factory):
        self.factories.append(factory)
    def create_connection(self, *args, **kwargs):
        connections = [factory.create_connection(*args, **kwargs) for factory in
self.factories]
        return connections
    def create_cursor(self, connections):
        cursors = [factory.create_cursor(connection) for factory, connection in
zip(self.factories, connections)]
        return cursors
```

Використання цього паттерну дозволяє легко розширювати та змінювати конкретні фабрики без необхідності зміни клієнського коду.

Висновок: Отже, під час виконання лабораторної роботи, я реалізував не менше 3-х класів та шаблон Abstract factory.