Лабораторна робота 3- 4. Структурне моделювання. Класи

Мета: Ознайомлення з основними принципами структурного моделювання програмних систем та побудови класів як основних структурних одиниць програмного забезпечення. Закріплення практичних навичок побудови структурних діаграм (зокрема UML-діаграм класів) та аналізу взаємозв'язків між класами у процесі проектування програмної системи.

1. **Стан дії** 2. **Графічне зображення класів**
1. **Переходи** 2. Визначення атрибутів
1. **Доріжки (секції)** 2. Типи класів
1. **Об'єкти на діаграмі активності** 2. Визначення методів

Теоретичні питння

PlantUML: Діаграма класів — елементи та їх позначення

1. Клас

Клас — це базовий елемент UML, який описує об'єкт або абстракцію об'єкта системи. Клас містить атрибути (стан) і методи (поведінка).
Синтаксис у PlantUML:

```
class ClassName {
   - attribute : Type
   + method()
}
```

2. Атрибути

Атрибути — це змінні, що зберігають стан об'єкта класу. Мають тип і рівень доступу.
Позначення рівня доступу:

+ — публічний

- — приватний

— захищений

~ — пакетний

Приклад у класі:

```
class Student {
   - name : String
   - id : int
}
```

3. Методи

Методи (операції) визначають поведінку об'єкта класу. Можуть бути публічними, приватними, захищеними або пакетними.
Приклад:

```
class Student {
   + register()
   + displayInfo()
}
```

4. Інтерфейс

Інтерфейс задає набір методів, які клас повинен реалізувати, але не містить їхньої реалізації.

```
interface Payable {
   + pay()
}
```

5. Абстрактний клас

Абстрактний клас може містити як реалізовані, так і абстрактні методи. Використовується як шаблон для похідних класів.

```
abstract class Employee {
   + getSalary()
}
```

6. Пакет

Пакет об’єднує класи у логічну підсистему для кращої організації моделі.

```
package "University System" {
   class Student
   class Course
}
```

7. Асоціація

Лінійний зв’язок між двома класами, що відображає логічну взаємодію.

```
Student --> Course : enrolls
```

8. Наслідування (Generalization)

Ієрархічний зв’язок, у якому підклас успадковує атрибути та методи батьківського класу.

```
Person <|-- Student
```

9. Агрегація

Відношення “ціле–частина”, при якому частина може існувати окремо від цілого.

```
Team o-- Player
```

10. Композиція

Тісне відношення “ціле–частина”, при якому частина не може існувати без цілого.

```
Car *-- Engine
```

11. Залежність (Dependency)

Тимчасовий зв’язок, коли один клас використовує інший для виконання певної дії.

```
Driver ..> Car
```

12. Кратність (Multiplicity)

Позначає кількість об’єктів одного класу, що можуть бути пов’язані з об’єктом іншого класу.

```
Student "1" -- "1..*" Course
```

13. Коментарі

Додатковий текст, що пояснює елементи моделі, не змінюючи структуру.

```
' Це коментар
// Альтернативний спосіб коментаря
```

Завдання

Завдання 1. Розглянути наведний приклад та проаналізувати структуру системи та методи опису класів на діаграмі UML у відповідності до структури системи (рис.1)

Завдання 2. Створити діаграму класів системи “Ordering”

```
    @startuml
' Пакет системи замовлень
package "Ordering System" {
    ' Клас клієнта
    class Customer {
        - name : String
        - email : String
        + placeOrder()
        + cancelOrder()
    }

    ' Клас замовлення
    class Order {
        - orderId : int
        - orderDate : Date
        - status : String
        + addItem()
        + removeItem()
        + calculateTotal()
    }

    ' Клас продукту
    class Product {
        - productId : int
        - name : String
        - price : float
        + getDetails()
    }

    ' Клас доставки
    class Delivery {
        - deliveryId : int
        - address : String
        - deliveryDate : Date
        + scheduleDelivery()
    }

    ' Асоціації
    Customer "1" --> "0..*" Order : places >
    Order "1" --> "1..*" Product : contains >
    Order "1" --> "0..1" Delivery : assigned to >
}
@enduml
```

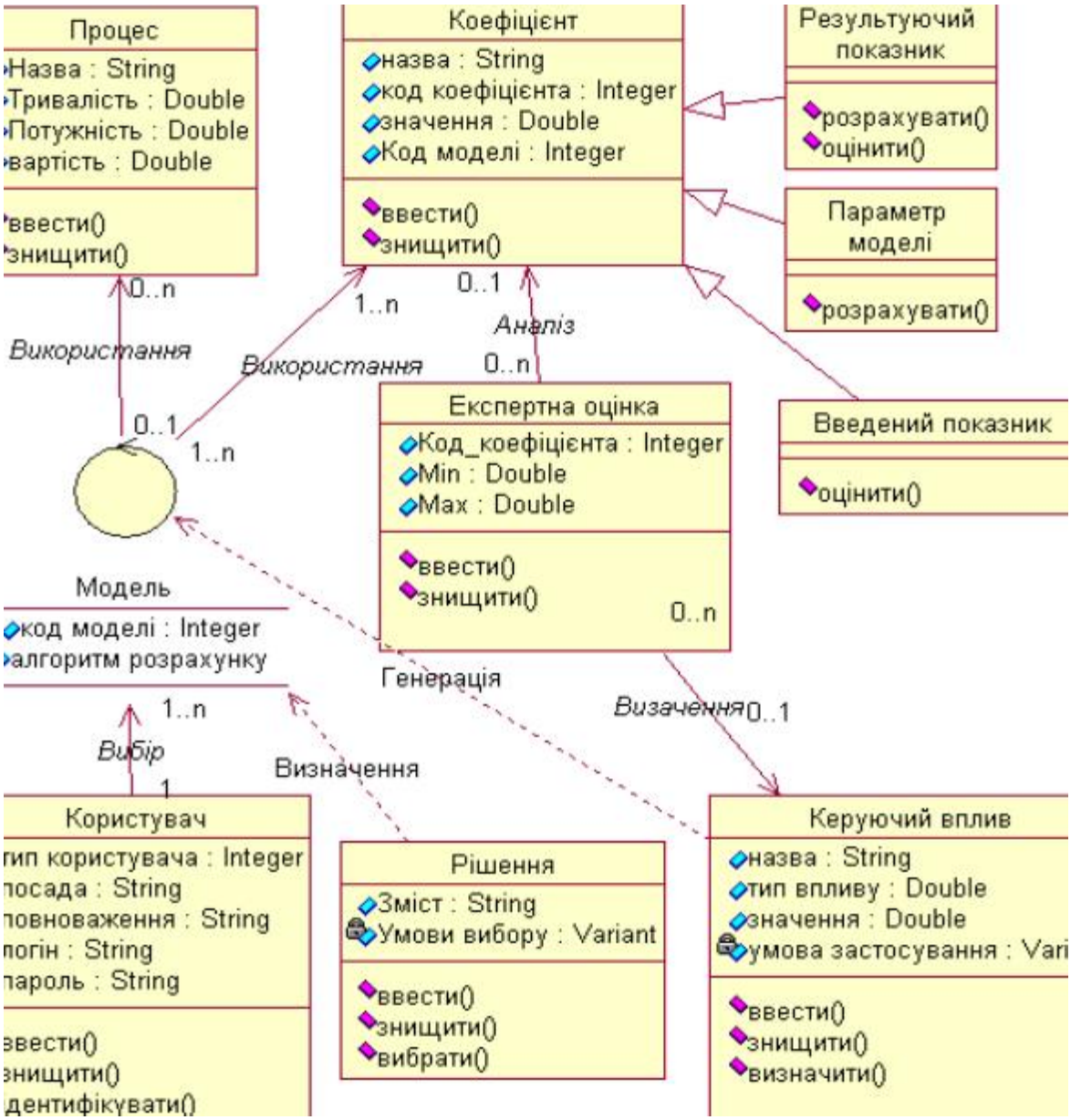

Рис. 1. Приклад діаграми класів (система управління інноваціями)

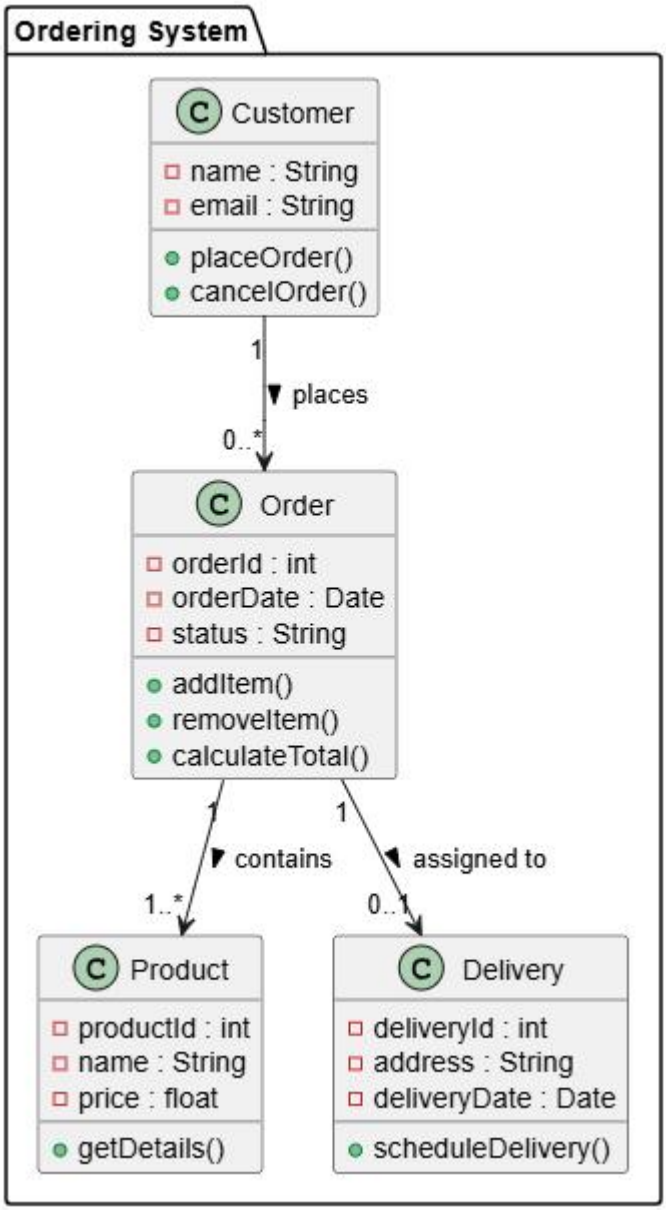
Ordering System
Customer
name : String
email : String
placeOrder()
cancelOrder()
1
places
0..*
Order
orderId : int
orderDate : Date
status : String
addItem()
removeItem()
calculateTotal()
1
contains
1..*
1
assigned to
0..1
Product
productId : int
name : String
price : float
getDetails()
Delivery
deliveryId : int
address : String
deliveryDate : Date
scheduleDelivery()

Рис. 2

Завдання 3. Проаналізувати діаграму класів системи та модіфікувати із використанням патернів GOF.

```
@startuml
package "Ordering System" {

    ' Singleton: Менеджер замовлень
    class OrderManager {
        - instance : OrderManager
        + getInstance() : OrderManager
        + createOrder()
        + cancelOrder()
    }

    ' Customer
    class Customer {
        - name : String
        - email : String
        + placeOrder()
        + cancelOrder()
    }

    ' Order
    class Order {
        - orderId : int
        - orderDate : Date
        - status : String
        + addItem()
        + removeItem()
        + calculateTotal()
    }

    ' Product
    class Product {
        - productId : int
        - name : String
        - price : float
        + getDetails()
    }

    ' Delivery
    class Delivery {
        - deliveryId : int
        - address : String
        - deliveryDate : Date
        + scheduleDelivery()
    }

    ' Strategy Pattern: Різні способи оплати
    interface PaymentStrategy {
        + pay(amount : float)
    }
```

```
    class CreditCardPayment {
        + pay(amount : float)
    }

    class PayPalPayment {
        + pay(amount : float)
    }

    ' Factory Pattern: Створення продуктів
    interface ProductFactory {
        + createProduct(name : String) : Product
    }

    class ConcreteProductFactory {
        + createProduct(name : String) : Product
    }

    ' Асоціації
    Customer "1" --> "0..*" Order : places >
    Order "1" --> "1..*" Product : contains >
    Order "1" --> "0..1" Delivery : assigned to >
    Order --> PaymentStrategy : uses >
    ConcreteProductFactory ..|> ProductFactory
    OrderManager ..> Order : manages >
}
@enduml
```

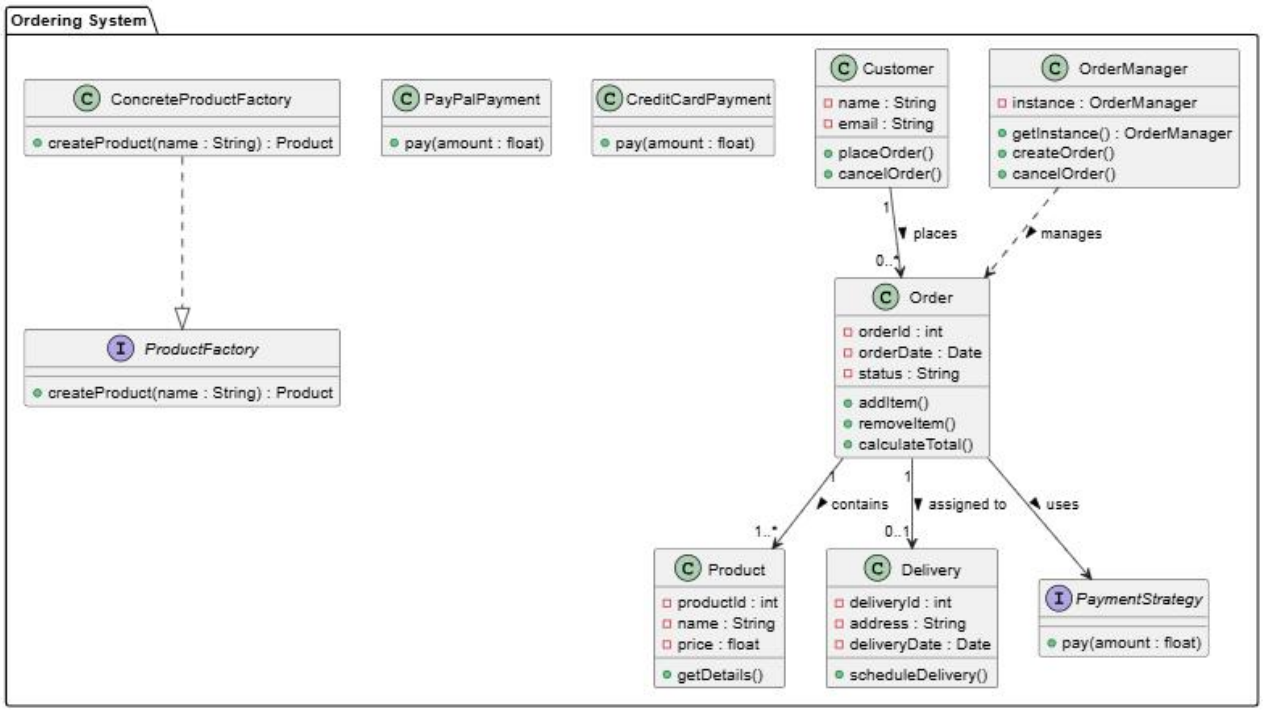

Рис.3
Опис патернів, використаних у діаграмі

Singleton (OrderManager)
Мета: Забезпечує єдиний глобальний доступ до менеджера замовлень.
Використання: У системі замовлень лише один менеджер керує створенням і скасуванням замовлень.
Strategy (PaymentStrategy)
Мета: Дозволяє змінювати алгоритм оплати динамічно без зміни класу Order.
Використання: Клас Order використовує інтерфейс PaymentStrategy. Реалізації CreditCardPayment і PayPalPayment визначають конкретні способи оплати.
Factory (ProductFactory / ConcreteProductFactory)
Мета: Інкапсулює створення продуктів, спрощує додавання нових типів товарів.
Використання: Клас Order не створює об’єкти Product напряму, а використовує фабрику для їх створення.

Пояснення структури:
Customer створює замовлення, пов’язане з Product і може мати Delivery.
OrderManager контролює процес замовлень як Singleton.
PaymentStrategy дозволяє змінювати спосіб оплати без зміни класів замовлень.
ProductFactory інкапсулює створення продуктів.

Завдання 4. Побудувати діаграму класів системи „Smart House”. Для цього:

1. Визначити основні класи (приміщення, датчики, пристрої керування тощо).
2. Ідентифікувати атрибути класів: (+температура[1..*]:=20 тощо)
3. Визначити методи класів(Змінити(температура, температура_задана):bool{Yes;No})
4. Реалізувати зв‘язкі між класами.