МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТУ “ЛЬВІВСЬКА ПОЛІТЕХНІКА“

Кафедра систем штучного інтелекту

Розрахунково-графічна робота
Частина 2

з дисципліни

«Системний аналіз»

Виконав:

Студент групи КН-214

Добрій Назарій

Прийняла:

Бойко.Н.І.

Львів – 2021р.

Тема: реалізація патернів для предметної області “Туристична компанія”.

Мета роботи: навчитися реалізовувати патерни: “Стратегія”, “Спостерігач”, “Декоратор”, “Фабричний метод”, “Одинак” для предметної області “Туристична компанія”.

1. Реалізувати патерн «Стратегія» на основі предметної області “Туристична компанія”.

Стратегія – поведінковий патерн проектування, який визначає сімейство схожих алгоритмів і розміщує кожен з них у власному класі. Після цього алгоритми можна замінити один на інших прямо під час виконання програми.

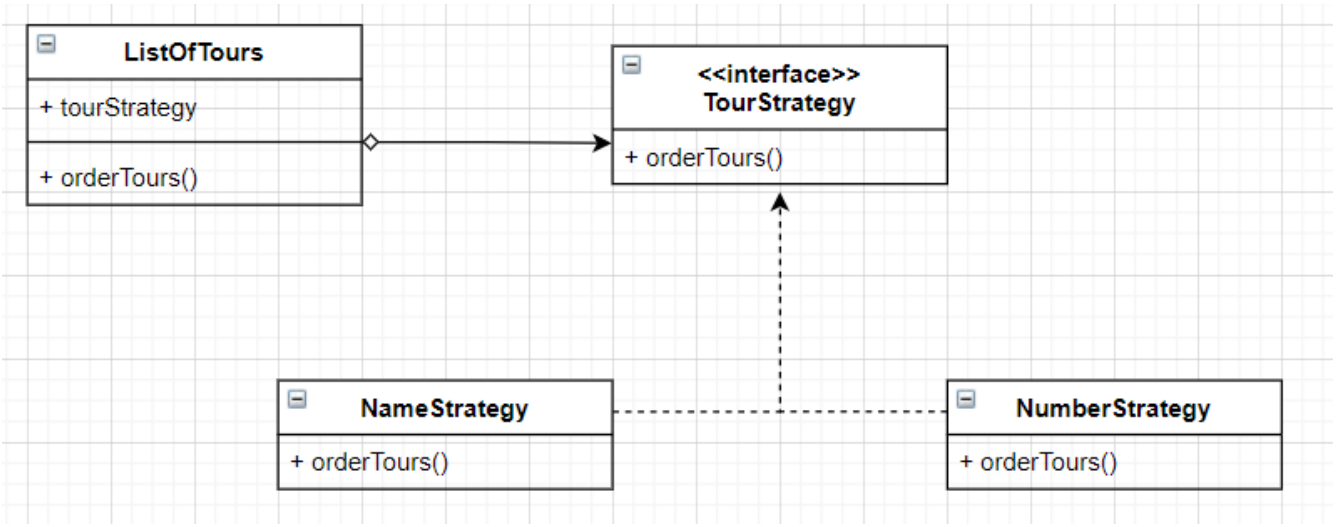

Рис. 1.1. UML-діаграма патерну “Стратегія”

Код програми на мові Python:

```
from abc import ABC, abstractmethod


class TourStrategy(ABC):
    @abstractmethod
    def order(self, list):
        pass


class OrderByNameStrategy(TourStrategy):
    def order(self, list):
        print("Ordering by client's last name...")
        list.sort(key = lambda x: x.client, reverse=False)
        for e in list:
            e.showOrder()


class OrderByNumberStrategy(TourStrategy):
    def order(self, list):
```

```
        print("Ordering by order's number")
        list.sort(key=lambda x: x.number, reverse=False)
        for e in list:
            e.showOrder()


class Order:
    def __init__(self, number, client, date, desc):
        self.number = number
        self.client = client
        self.date = date
        self.desc = desc

    def showOrder(self):
        print('==============================')
        print("Номер замовлення: " + str(self.number) )
        print("Клієнт: " + self.client)
        print("Дата: " + self.date)
        print("Опис: " + self.desc)
        print('==============================')


class ListOfTours:
    def __init__(self):
        self.orders = []

    def addOrder(self, order):
        self.orders.append(order)

    def orderOrders(self, strategy: TourStrategy):
        strategy.order(self.orders)


order1 = Order(66, "Назар", "2021-05-12", "опис тура")
order2 = Order(55, "Віталій", "2021-05-15", "опис тура")
order3 = Order(111, "Роман", "2021-05-06", "опис тура")

list = ListOfTours()

list.addOrder(order1)
list.addOrder(order2)
list.addOrder(order3)

print("Виберіть тип сортування: (1 - за прізвищем Клієнта, 2 - за номером
замовлення)")
inp = int(input())
if inp == 1:
    list.orderOrders(OrderByNameStrategy())
if inp == 2:
    list.orderOrders(OrderByNumberStrategy())
else:
    pass
```

Рис. 1.2. Код реалізації програми.

```
Виберіть тип сортування: (1 - за прізвищем Клієнта, 2 - за номером замовлення)
1
Ordering by client's last name...
==============================
Номер замовлення: 55
Клієнт: Віталій
Дата: 2021-05-15
Опис: опис тура
==============================
==============================
Номер замовлення: 66
Клієнт: Назар
Дата: 2021-05-12
Опис: опис тура
==============================
==============================
Номер замовлення: 111
Клієнт: Роман
Дата: 2021-05-06
Опис: опис тура
==============================

Process finished with exit code 0
```

Рис. 1.3. Сортування за прізвищем клієнта.

```
Виберіть тип сортування: (1 - за прізвищем Клієнта, 2 - за номером замовлення)
2
Ordering by order's number
==============================
Номер замовлення: 55
Клієнт: Віталій
Дата: 2021-05-15
Опис: опис тура
==============================
==============================
Номер замовлення: 66
Клієнт: Назар
Дата: 2021-05-12
Опис: опис тура
==============================
==============================
Номер замовлення: 111
Клієнт: Роман
Дата: 2021-05-06
Опис: опис тура
==============================

Process finished with exit code 0
```

Рис. 1.4. Сортування за номером замовлення клієнта.

2. Реалізувати патерн «Спостерігач» на основі предметної області “Туристична компанія”.

Спостерігач — це поведінковий патерн проектування, який створює механізм підписки, що дає змогу одним об’єктам стежити й реагувати на події, які відбуваються в інших об’єктах.

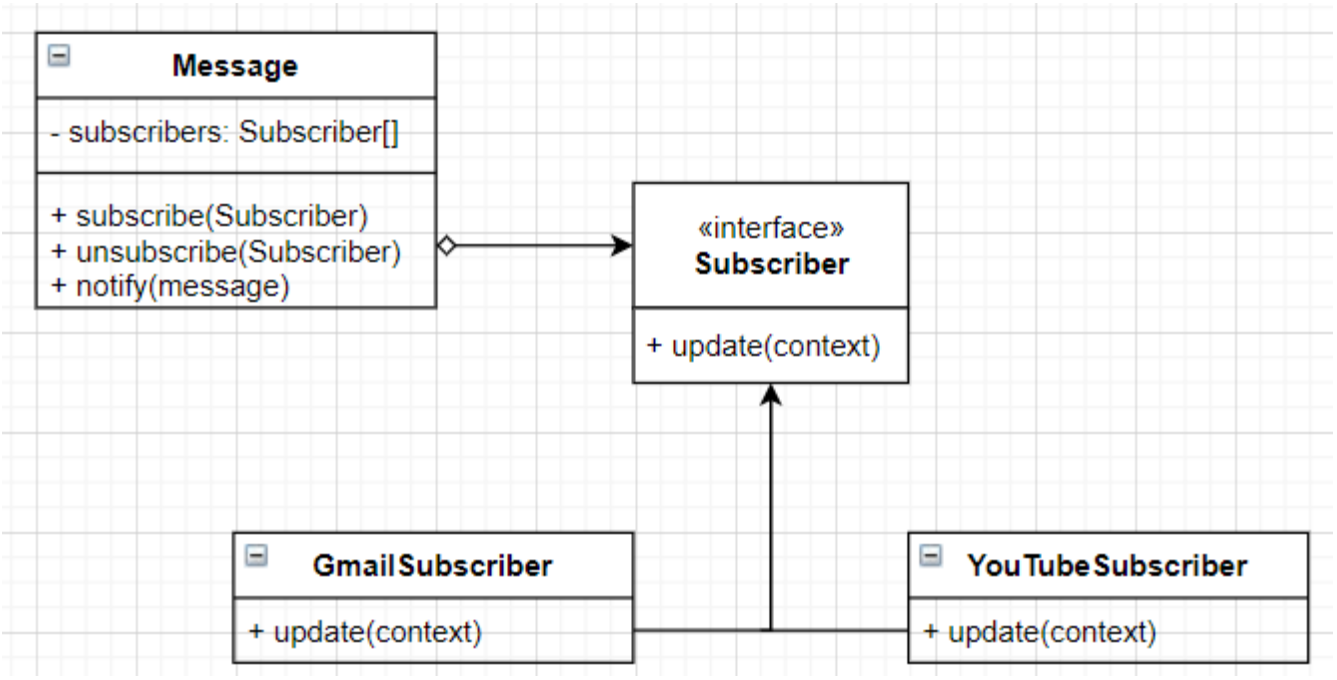

Рис. 2.1. UML-діаграма патерна “Спостерігач”

Код програми на мові Python:

```
from abc import ABC, abstractmethod


class Subscriber(ABC):
    @abstractmethod
    def update(self, message):
        pass


class YouTubeSubscriber(Subscriber):
    def __init__(self, name):
        self.name = name

    def update(self, message):
        print('Користувач {} отримав повідомлення "{}" через
YouTube'.format(self.name, message))


class GmailSubscriber(Subscriber):
    def __init__(self, name):
        self.name = name
```

```
    def update(self, message):
        print('Користувач {} отримав повідомлення "{}" через пошту
gmail.com'.format(self.name, message))


class Notificator:
    def __init__(self):
        self.subscribers = set()

    def subscribe(self, sub: Subscriber):
        self.subscribers.add(sub)
        print('{} підписався'.format(sub.name))

    def unsubscribe(self, sub):
        self.subscribers.discard(sub)
        print('{} відписався'.format(sub.name))

    def notify(self, message):
        for subs in self.subscribers:
            subs.update(message)


notifiactor = Notificator()

sub1 = GmailSubscriber("nazardobriy@gmail.com")
sub2 = YouTubeSubscriber("Nazar Dobriy")

notifiactor.subscribe(sub1)
notifiactor.subscribe(sub2)

notifiactor.notify("Виконання підписки продовжено на місяць!")

notifiactor.unsubscribe(sub2)

notifiactor.notify("Успішна підписка на місяць!")
```

Рис. 2.2. Код реалізації програми.

```
nazardobriy@gmail.com підписався
Nazar Dobriy підписався
Користувач Nazar Dobriy отримав повідомлення "Виконання підписки продовжено на місяць!" через YouTube
Користувач nazardobriy@gmail.com отримав повідомлення "Виконання підписки продовжено на місяць!" через пошту gmail.com
Nazar Dobriy відписався
Користувач nazardobriy@gmail.com отримав повідомлення "Успішна підписка на місяць!" через пошту gmail.com

Process finished with exit code 0
```

Рис. 2.3. Виконання програми.

3. Реалізувати патерн «Декоратор» на основі предметної області “Туристична компанія”.

Декоратор — це структурний патерн проектування, що дає змогу динамічно додавати об’єктам нову функціональність, загортаючи їх у корисні «обгортки».

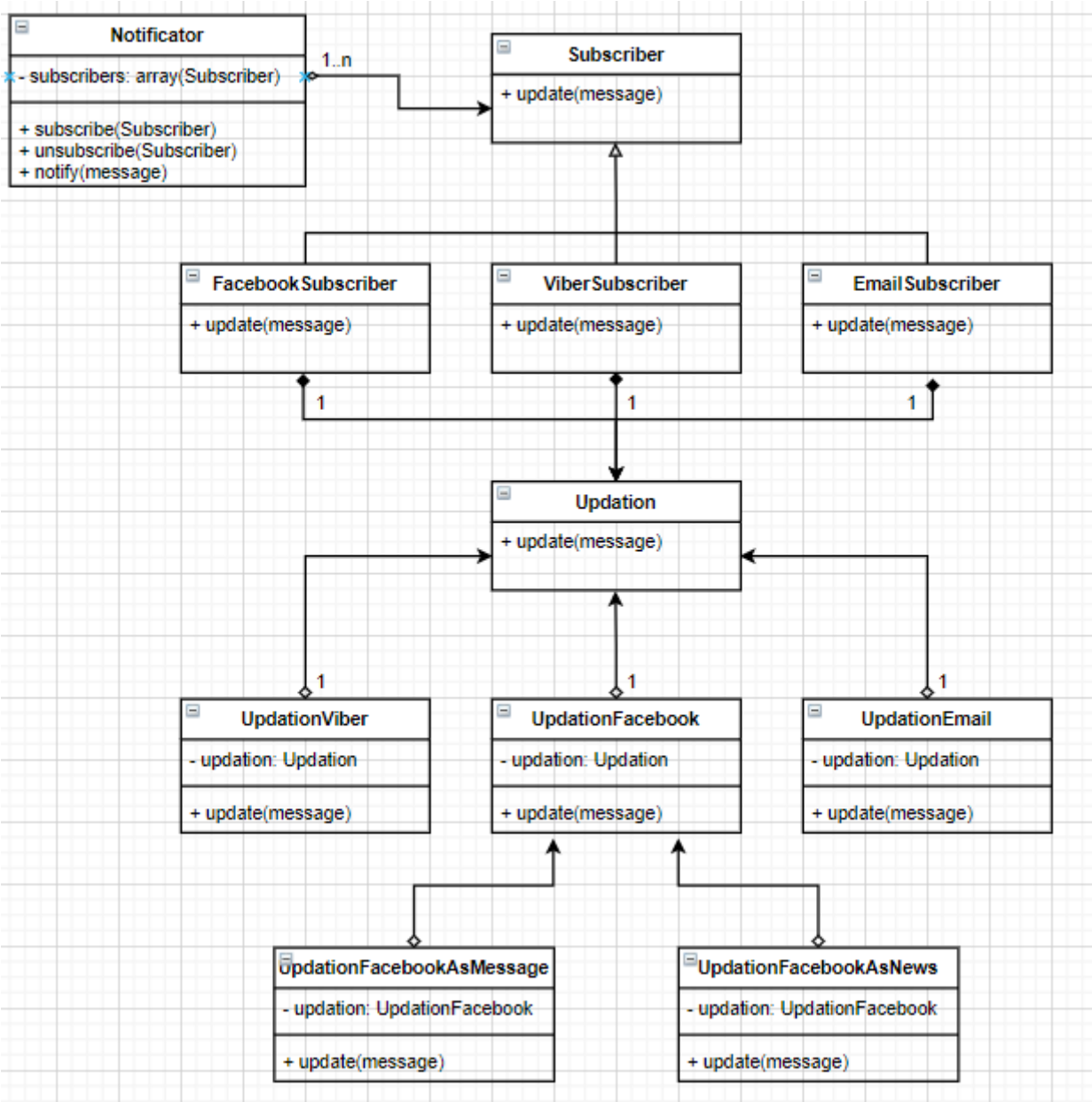

Рис. 3.1. UML-діаграма патерну “Декоратор”

Код програми на мові Python:

```
from abc import ABC, abstractmethod


class Updation():
    def __init__(self, message, name):
```

```
        self.message = message
        self.name = name

    def update(self):
        return 'Користувач {} отримав повідомлення "{}" '.format(self.name,
self.message)


class UpdationFacebook():
    def __init__(self, updation):
        self.updation = updation

    def update(self):
        return self.updation.update() + " через Facebook"


class UpdationFacebookAsNews():
    def __init__(self, updation):
        self.updation = updation

    def update(self):
        return self.updation.update() + ", як новину"


class UpdationFacebookAsMessage():

    def __init__(self, updation):
        self.updation = updation

    def update(self):
        return self.updation.update() + ", як повідомлення"


class UpdationViber():
    def __init__(self, updation):
        self.updation = updation

    def update(self):
        return self.updation.update() + " через Вайбер"


class UpdationEmail():
    def __init__(self, updation):
        self.updation = updation

    def update(self):
        return self.updation.update() + " через електронну пошту"


class Subscriber(ABC):
    @abstractmethod
    def update(self, message):
        pass


class FacebookSubscriber(Subscriber):
    def __init__(self, name, type):
        self.name = name
        self.type = type

    def update(self, message):
        updation = UpdationFacebook(Updation(message, self.name))
```

```
        if self.type == "повідомлення":
            updation = UpdationFacebookAsMessage(updation)
        elif self.type == 'новина':
            updation = UpdationFacebookAsNews(updation)

        print(updation.update())


class ViberSubscriber(Subscriber):
    def __init__(self, name):
        self.name = name

    def update(self, message):
        updation =  UpdationViber(Updation(message, self.name))

        print(updation.update())


class EmailSubscriber(Subscriber):
    def __init__(self, name):
        self.name = name

    def update(self, message):
        updation = UpdationEmail(Updation(message, self.name))

        print(updation.update())




class Notificator:
    def __init__(self):
        self.subscribers = set()

    def subscribe(self, sub: Subscriber):
        self.subscribers.add(sub)
        print('{} підписався'.format(sub.name))
    def unsubscribe(self, sub):
        self.subscribers.discard(sub)
        print('{} відписався'.format(sub.name))

    def notify(self, message):
        for subs in self.subscribers:
            subs.update(message)


notifiactor = Notificator()

sub1 = EmailSubscriber("nazardobriy@gmail.com")
sub2 = FacebookSubscriber("Nazar Dobriy", 'новина')
sub3 = ViberSubscriber("Nazar Dobriy")

notifiactor.subscribe(sub1)

notifiactor.subscribe(sub2)

notifiactor.subscribe(sub3)

notifiactor.notify("Виконання замовлення №5 продовжено на день!")

notifiactor.unsubscribe(sub3)
```

```
notifiactor.notify("Замовлення №5 виконано!")
```

Рис. 3.2. Код реалізації програми.

```
Nazar Dobriy підписався
Користувач Nazar Dobriy отримав повідомлення "Виконання замовлення №5 продовжено на день!"  через Facebook, як новину
Користувач Nazar Dobriy отримав повідомлення "Виконання замовлення №5 продовжено на день!"  через Вайбер
Користувач nazardobriy@gmail.com отримав повідомлення "Виконання замовлення №5 продовжено на день!"  через електронну пошту
Nazar Dobriy відписався
Користувач Nazar Dobriy отримав повідомлення "Замовлення №5 виконано!"  через Facebook, як новину
Користувач nazardobriy@gmail.com отримав повідомлення "Замовлення №5 виконано!"  через електронну пошту

Process finished with exit code 0
```

Рис. 3.3. Виконання програми.

4. Реалізувати патерн «Фабричний метод» на основі предметної області “Туристична компанія”.

Фабричний метод — це породжувальний патерн проектування, який визначає загальний інтерфейс для створення об’єктів у суперкласі, дозволяючи підкласам змінювати тип створюваних об’єктів.

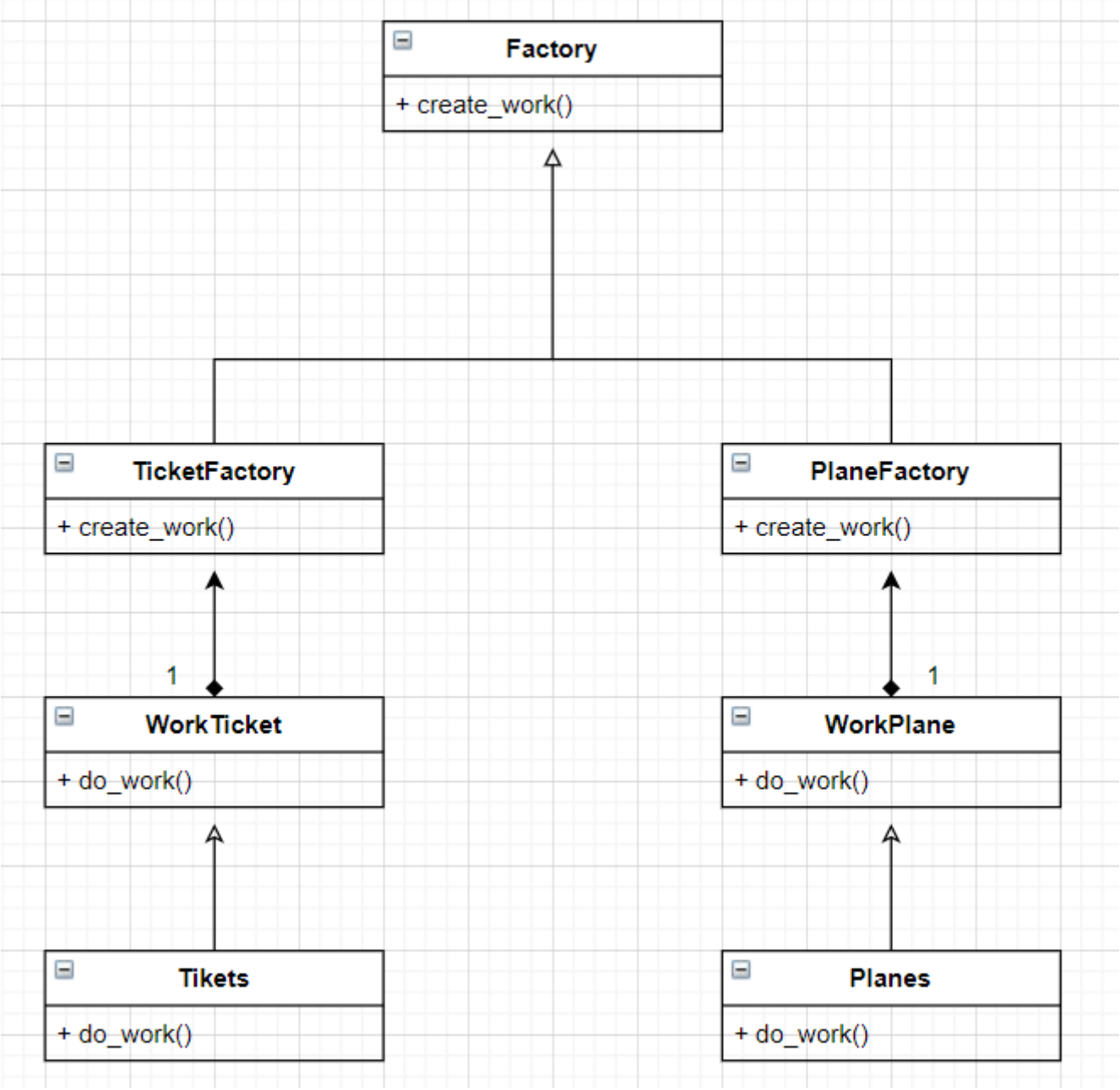

Рис. 4.1. UML-діаграма патерну “Фабричний метод”

Код програми на мові Python:

```
from abc import ABC, abstractmethod


class WorkTikcet(ABC):
    @abstractmethod
    def do_work(self):
        pass


class WorkPlane(ABC):
    @abstractmethod
    def do_work(self):
        pass


class Tickets(WorkTikcet):
    def do_work(self):
        print("Видаємо квиток для туру")


class Planes(WorkPlane):
    def do_work(self):
        print("Видаємо літак для подорожі")


class Factory(ABC):
    @abstractmethod
    def create_work(self, type):
        pass


class TicketFactory(Factory):
    def create_work(self, type):
        return Tickets()


class PlaneFactory(Factory):
    def create_work(self, type):
        return Planes()


cover_factory = TicketFactory()
page_factory = PlaneFactory()


work_cover = cover_factory.create_work('Створення квитка')

work_cover.do_work()

work_page = page_factory.create_work('Створення літака')

work_page.do_work()
```

Рис. 4.2. Код реалізації програми.

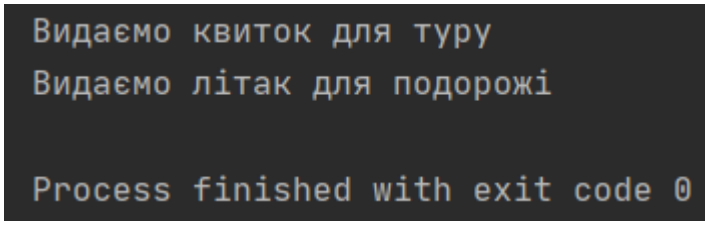

Рис. 4.3. Виконання програми.

5. Реалізувати патерн «Одинак» на основі предметної області “Туристична компанія”.

Одинак — це породжувальний патерн проектування, який гарантує, що клас має лише один екземпляр, та надає глобальну точку доступу до нього.

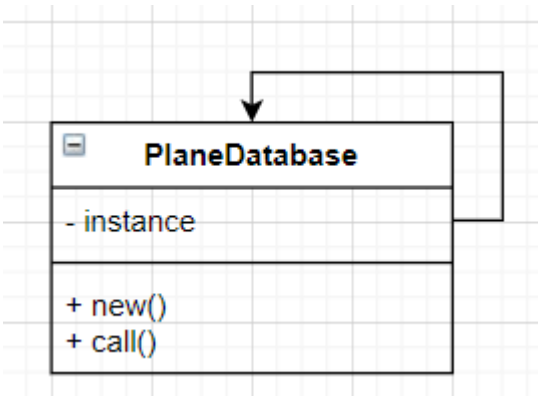

Рис. 5.1 UML-діаграма патерну «Одинак»

Код програми на мові Python:

```
class PlaneDataBase:
    __instance = None

    def __new__(cls, server_url, server_name):
        if cls.__instance is None:
            cls.__instance = super().__new__(cls)
        cls.__instance.name = server_name
        cls.__instance.url = server_url
        return cls.__instance

    def __call__(cls):
        print('Сервер бази даних літаків {} запущений за адресою {}'
              .format(cls.__instance.name, cls.__instance.url))



db = PlaneDataBase("http://127.0.0.1:5000/", "PlaneDB#1")
db()
db2 = PlaneDataBase("http://127.0.0.1:3000/", "PlaneDB#2")
db2()
db()
```

Рис. 5.2. Код реалізації програми.

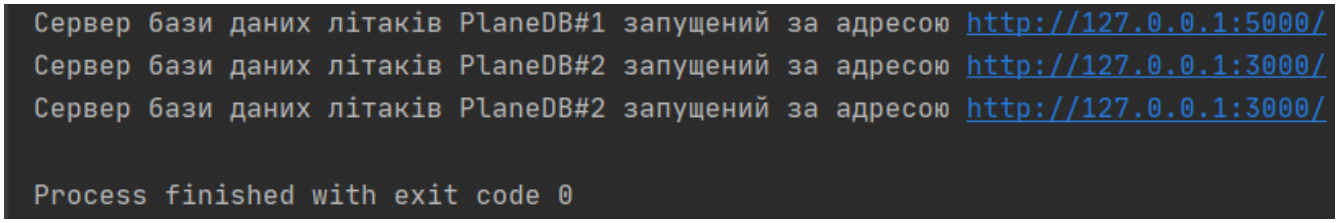

Рис. 5.3. Виконання програми.

Висновок: я навчився реалізовувати патерни: “Стратегія”, “Спостерігач”, “Декоратор”, “Фабричний метод”, “Одинак” для предметної області “Туристична компанія”.