

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**  
**КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**



**ЗВІТ**  
**про виконання лабораторної роботи №6**

**Виконав:** Климок Н.І.  
**студент групи ПЗ - 16**

---

*(дата виконання)*

Львів - 2022

**Тема.** Опис предметної області з використанням UML.

**Мета.** Навчитися створювати об'єктну модель програмної системи.

## ТЕОРЕТИЧНІ ВІДОМОСТІ








Яке основне призначення UML?

3. *UML – це уніфікована мова моделювання. Її мета це змодельовати систему і подати у відповідній діаграмі. Ця мова має певні правила, отже вона конкретно описує систему, тому це можливість для людей передати інформацію, яка буде зрозумілою для всіх. Наведу приклад: коли деяка група програмістів розробляє якусь велику систему, а супроводжувати її збирається інша група, то якраз для цієї іншої групи буде важко зорієнтуватись, але натомість якраз використовується мова UML для моделювання цієї системи, що значно спрощує її розуміння.*

8. Яка історія виникнення мови UML?

*В основу UML покладено декілька об'єктно-орієнтованих методів, кожен із яких був орієнтований на підтримку окремих етапів об'єктно-орієнтованого аналізу та проектування (ООАП): - метод Граді Буча, що одержав умовну назву Booch; - метод Джеймса Румбаха, що одержав назву Object Modeling Technique; - метод Айвара Джекобсона, Object-Oriented Software Engineering – OOSE. Історія розвитку UML датується 1994 роком, коли почалася уніфікація та інтеграція вищевказаних методів їх авторами. Проект уніфікованого методу (Unified Method) версії 0.8 був опублікований в жовтні 1995 року.*

14. Які є відношення між класами, як вони зображаються?

Узагальнення , залежність , асоціація   
, агрегація , композиція , реалізація 

## ПОСТАНОВКА ЗАВДАННЯ

Згідно індивідуального варіанту провести аналіз предметної області. Для виконання завдання:

1. Скласти словник предметної області.
2. Побудувати UML-діаграму класів на концептуальному рівні засобами програми Visio. Зобразити коментарі на схемі. Вказати відношення між сутностями (узагальнення, звичайна асоціація, агрегація, композиція, залежність) із обов'язковим зазначенням їх характеристик (кратність, назва асоціації і т.п.).
3. Побудувати UML-діаграму конкретних класів на рівні реалізації засобами програми Visio. Чітко вказати усі поля та методи класів з відповідними модифікаторами доступу, а також усі необхідні відношення між класами.
4. Оформити звіт.

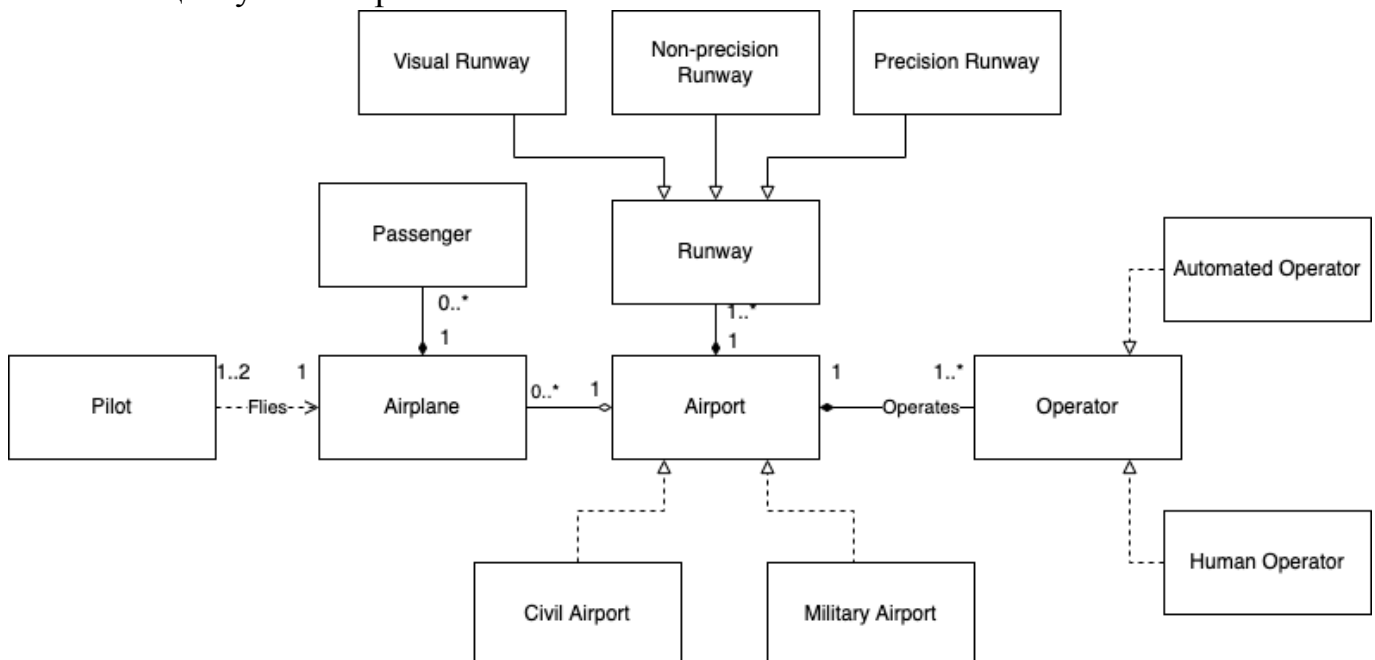
Варіант: інформаційна система “Аеропорт”.

## ВИКОНАННЯ ЗАВДАННЯ

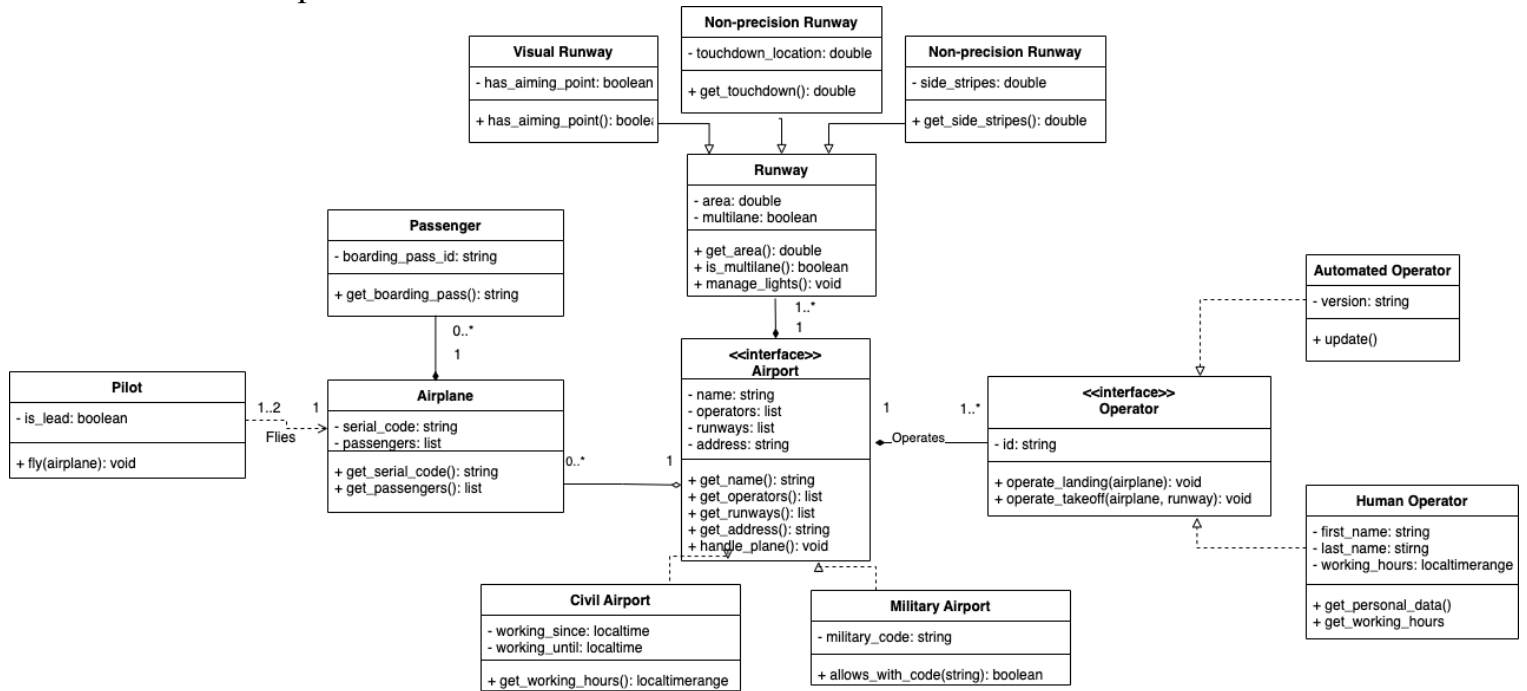
1. Словник предметної області “Аеропорт”
  - 1) Airplane – літак, є складовою сутності Airport (відношення типу частина-ціле), містить властивості: серійний номер (serial\_code), список пасажирів (passengers).
  - 2) Airport – інтерфейс, реалізується класами Civil Airport і Military Airport. Містить поля name, operators, runways, address і методи доступу до них.
  - 3) Civil Airport – реалізація інтерфейсу Airport, містить поля working\_since, working\_until і метод working\_hours() для отримання робочих годин аеропорта.
  - 4) Military Airport – реалізація інтерфейсу Airport, військовий аеропорт. Містить поле military\_code і метод для доступу до нього.
  - 5) Runway – конкретний клас для взлітної смуги. Розширюється класами Visual Runway, Non-precision Runway і Precision Runway. Містить поля area (площа) і multilane (чи багатосмужний), а також методи для доступу до полів.
  - 6) Passenger – клас пасажирів. Містить поле boarding\_pass\_id і метод для доступу до нього. Має композитне відношення “частина-ціле” з класом Airplane.

- 7) Pilot – клас пілота. Містить поле `is_lead`, щоб визначити, чи є пілот головним. `Airplane` залежить від `Pilot`.
- 8) Visual Runway – розширення класу `Runway`. Містить поле `has_aiming_point`, щоб визначити, чи містить смуга прицільну точку.
- 9) Non-precision Runway – розширення класу `Runway`. Містить поле `touchdown_location`, щоб визначити, чи містить смуга чітку локацію приземлення.
- 10) Precision Runway – розширення класу `Runway`. Містить поле `side_stripes`, щоб визначити розміщення бокових смуг.
- 11) Operator – інтерфейс оператора. Містить методи `operate_landing()`, `operate_takeoff()` для роботи з приземленням на злетом літаків, а також поле `id` для ідентифікації оператора.
- 12) Automated operator – реалізація класу `Operator`. Містить `version` та метод `update()` для оновлення версії автоматизованого оператора.
- 13) Human Operator – реалізація класу `Operator`. Містить поля `first_name`, `last_name`, `working_hours`, а також методи `get_personal_data()` для роботи з особистою інформацією та `get_working_hours()` для отримання робочих годин людського оператора.

## 2. Концептуальний рівень



### 3. Рівень реалізації



### Висновок

Я навчився створювати об'єктну модель програмної системи.