Проектування систем штучного інтелекту. Практична робота 7

Виконав студент групи **ШІДМ-51 Тертишний** Владислав Юрійович

Тема:

Створення діаграм класів для інформаційної системи (ІС) у межах предметної галузі "Діяльність відділу реклами" (варіант №18).

Мета роботи:

Вивчити діаграми класів та їх застосування в процесі проєктування моделі інформаційної системи. Побудувати діаграму класів для відображення архітектури системи та деталізації окремих підсистем.

Завдання

1. Виділити основні класи об'єктів у системі "Діяльність відділу реклами".
2. Побудувати загальну діаграму класів, що демонструє архітектуру системи.
3. Створити одну-дві деталізовані діаграми класів для підсистеми.
4. Вказати для класів:
    - Основні атрибути.
    - Операції.
    - Типи та напрямки асоціацій.

Рішення

1. Опис предметної галузі

Система: Автоматизація роботи відділу реклами.
Мета: Управління рекламними кампаніями, взаємодія із замовниками, розподіл завдань між співробітниками.

---

2. Основні класи системи

1. **Замовник**:
    - Атрибути:
        - `Ім'я`
        - `Контактна інформація`
        - `Тип компанії`
    - Операції:
        - `Оформити замовлення`
        - `Редагувати дані`
2. **Рекламна кампанія**:
    - Атрибути:
        - `Назва`
        - `Бюджет`
        - `Ціль`
        - `Період проведення`
    - Операції:
        - `Запустити кампанію`
        - `Аналіз результатів`
3. **Співробітник**:
    - Атрибути:
        - `Ім'я`
        - `Посада`
        - `Зарплата`
    - Операції:
        - `Призначити завдання`
        - `Редагувати завдання`
4. **Звіт**:
    - Атрибути:
        - `Назва`
        - `Дата створення`
        - `Результати`
    - Операції:
        - `Створити звіт`
        - `Переглянути`

---

3. Загальна діаграма класів

Класи та зв’язки:

- **Замовник** асоціюється з **Рекламною кампанією** (1 замовник -> 0..* кампаній).
- **Рекламна кампанія** пов’язана зі **Співробітниками** (1 кампанія -> 1..* співробітників).

- **Рекламна кампанія** має **Звіти** (1 кампанія -> 0..* звітів).

Додаткові зв’язки:

- Узагальнення: клас **Менеджер** є підкласом класу **Співробітник**.

---

4. Побудова діаграми в UML

1. **Інструмент:** Використовуємо StarUML або Rational Rose.
2. **Етапи створення:**
    - Додати класи та їх атрибути.
    - Визначити зв’язки:
        - **Асоціація:** між замовником і кампанією.
        - **Узагальнення:** менеджер успадковує атрибути співробітника.
        - **Множинність:** визначити кількість об’єктів у зв’язках.
    - Додати операції.

5. Програмна генерація діаграми класів

У рамках практичної роботи було прийнято рішення відійти від традиційного методу створення діаграм через інструменти, такі як StarUML або Rational Rose, та застосувати сучасний підхід програмної генерації діаграм. Для цього використано бібліотеку Python `Graphviz`, яка дозволяє створювати та налаштовувати діаграми програмно.

Опис виконання

1. **Причини відхилення від завдання:**
    - Програмна генерація надає більше гнучкості у налаштуванні вигляду діаграми.
    - Код дозволяє автоматизувати створення діаграм, що особливо корисно для великих систем із численними класами.
    - Інструменти StarUML та Rational Rose можуть мати обмеження у кастомізації зв’язків, шрифтів та стилів, що легко реалізується програмно.
2. **Застосовані налаштування:**
    - **Розташування:** Встановлено напрямок діаграми зверху вниз (`rankdir='TB'`).
    - **Шрифти:** Використано шрифт Arial для покращення читабельності.
    - **Стиль вузлів:** Закруглені кути та формат запису класів із поділом на атрибути й методи.
    - **Зв’язки:** Реалізовано різні типи зв’язків із підписами (асоціація, успадкування).
    - **Роздільна здатність:** Налаштовано DPI для високої якості зображення.

3. **Результат:**
    - Програмно згенеровано діаграму класів для предметної області "Діяльність відділу реклами".
    - Отриманий файл `class_diagram.png` автоматично відкривається після створення.

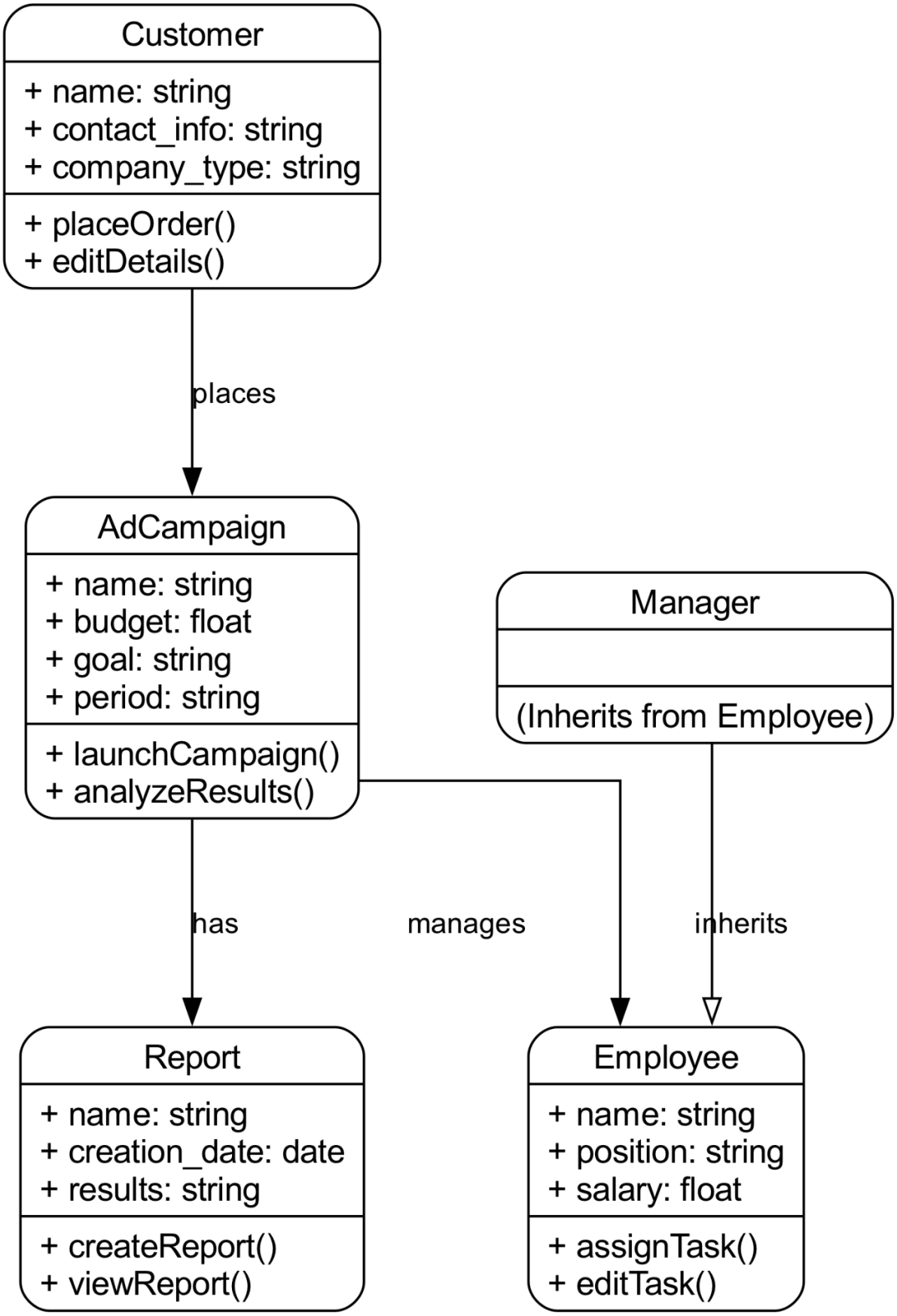
Код для генерації діаграми

```
from graphviz import Digraph


# Ініціалізація графу

dot = Digraph('Class Diagram', format='png')

# Add DPI setting for higher resolution

dot.attr(dpi='300')

dot.attr(rankdir='TB')

# Add global font settings

dot.attr('node', shape='record', style='rounded', fontsize='14',
fontname='Arial')

dot.attr('edge', fontsize='12', fontname='Arial')

dot.graph_attr['splines'] = 'ortho'

dot.graph_attr['nodesep'] = '0.8'

dot.graph_attr['ranksep'] = '1.0'

# Increase size of the entire graph

dot.attr(size='8,8')  # width,height in inches


# Додавання класів

dot.node('Customer', '''{ Customer |

+ name: string\l
```

```
+ contact_info: string\l
+ company_type: string\l|
+ placeOrder()\l
+ editDetails()\l}''')

dot.node('AdCampaign', '''{ AdCampaign |
+ name: string\l
+ budget: float\l
+ goal: string\l
+ period: string\l|
+ launchCampaign()\l
+ analyzeResults()\l}''')

dot.node('Employee', '''{ Employee |
+ name: string\l
+ position: string\l
+ salary: float\l|
+ assignTask()\l
+ editTask()\l}''')

dot.node('Report', '''{ Report |
+ name: string\l
+ creation_date: date\l
+ results: string\l|
```

```
+ createReport()\l

+ viewReport()\l}''')


dot.node('Manager', '''{ Manager |

|

(Inherits from Employee)\l}''')


# Зв'язки між класами

dot.edge('Customer', 'AdCampaign', label='places')

dot.edge('AdCampaign', 'Employee', label='manages')

dot.edge('AdCampaign', 'Report', label='has')

dot.edge('Manager', 'Employee', label='inherits',
arrowhead='onormal')


# Збереження та візуалізація

dot.render('class_diagram', view=True)

print("Diagram generated: class_diagram.png")
```

Результат


Customer
+ name: string
+ contact_info: string
+ company_type: string
+ placeOrder()
+ editDetails()
places
AdCampaign
+ name: string
+ budget: float
+ goal: string
+ period: string
+ launchCampaign()
+ analyzeResults()
Manager
(Inherits from Employee)
has
manages
inherits
Report
+ name: string
+ creation_date: date
+ results: string
+ createReport()
+ viewReport()
Employee
+ name: string
+ position: string
+ salary: float
+ assignTask()
+ editTask()

Висновки щодо підходу

- **Переваги програмної генерації:**
  - Автоматизація створення діаграм, що спрощує процес роботи над великими проєктами.
  - Гнучкість у налаштуванні вигляду та стилю діаграм.
  - Збереження діаграми у форматі високої якості для використання у документації.
- **Обмеження:**
  - Потреба у встановленні Python та бібліотеки `Graphviz`.
  - Необхідність базових навичок програмування.
- **Рекомендації:**
  - Цей підхід можна використовувати для систем, які вимагають частих змін та оновлення архітектури.

Це рішення демонструє ефективність сучасного підходу та може бути основою для розширення функціоналу діаграм.

Контрольні питання

1. **Призначення діаграм класів.**
   - На стадії аналізу: визначити ролі сутностей, які забезпечують необхідну поведінку системи.
   - На стадії проєктування: передати структуру класів, які формують архітектуру системи.
2. **Основні компоненти діаграм класів:**
   - Класи (атрибути, операції).
   - Статичні зв’язки (асоціації, узагальнення, композиції).
3. **Що таке асоціація?**
   Асоціація описує зв’язки між екземплярами класів (наприклад, замовник може оформити кілька замовлень).
4. **Що таке операція класу?**
   Операція реалізує поведінку класу. Наприклад, операція "Запустити кампанію" у класі "Рекламна кампанія".

Висновки

- Діаграма класів забезпечує візуалізацію архітектури системи.
- Виділені основні класи допомагають структурувати функціонал системи.
- Множинність зв’язків дозволяє описати реальні відносини між об’єктами.