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 №2
З дисципліни «Технології розроблення програмного забезпечення»
Тема: **«ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ. СЦЕНАРІЇ ВАРІАНТІВ ВИКОРИСТАННЯ. ДІАГРАМИ UML. ДІАГРАМИ КЛАСІВ. КОНЦЕПТУАЛЬНА МОДЕЛЬ СИСТЕМИ»**
Flex Automatical tool

Виконав:
Студент групи ІА-22
Сидорін Д.О.

Перевірив:
Мягкий М. Ю.

Київ-2025

Зміст

Тема:

Діаграма варіантів використання. Сценарії варіантів використання. Діаграми uml. Діаграми класів. Концептуальна модель системи

Мета:

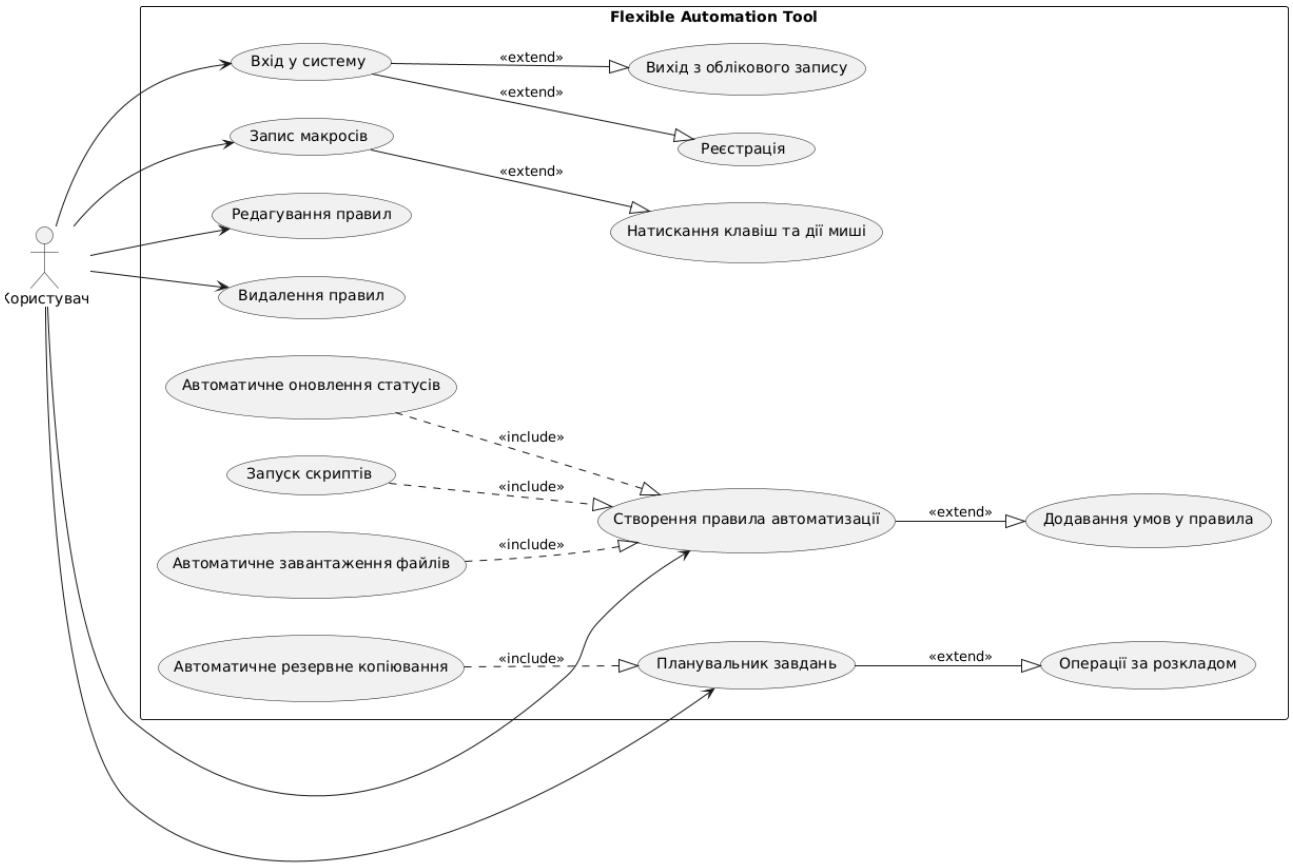
Дослідити та застосувати UML-діаграми для моделювання варіантів використання і концептуальної структури даних системи, зосереджуючись на діаграмах класів, прецедентів та опису їхніх сценаріїв використання.

Завдання:

Розробити візуальний додаток для автоматизації, що включає функції для створення та управління кількома автоматичними завданнями (у вкладках). Користувач повинен мати можливість автоматично налаштовувати і виконувати дії, такі як завантаження нових файлів, оновлення статусів в месенджерах, виконання макросів. У додатку має бути можливість додавати категорії для завдань (терміновість, типи дій) та автоматично виконувати їх згідно з заданими правилами. Також повинна бути можливість візуалізації завдань у вигляді "карт пам'яті", де кожне завдання або макрос буде представлено у вигляді вузла з можливістю додавати вкладені файли, зображення, відео (з попереднім переглядом).

Хід роботи

1. Схема прецедентів



2. Деталі сценаріїв використання

1. Створення правила автоматизації (U2)

Короткий опис:
Цей сценарій описує процес створення нового правила автоматизації, яке включає налаштування тригера та визначення умов дії.

Передумова:
Користувач повинен бути авторизований у системі, а інтерфейс управління автоматизацією повинен бути доступний.

Основний сценарій:

1. Користувач вибирає опцію "Створити нове правило".
2. Система запитує параметри для нового правила (тип тригера, умови тригера, тип і параметри дії).
3. Користувач вводить необхідні дані.
4. Користувач натискає "Зберегти".
5. Система зберігає нове правило та підтверджує успішне створення.

Альтернативний сценарій:
• Якщо введені некоректні дані (наприклад, неправильний формат URL або неіснуючий тип дії):

1. Система відображає повідомлення про помилку.
2. Користувач виправляє помилку та повторює дію.

Винятки:

- Якщо користувач не заповнив обов'язкові поля або ввів некоректні дані, система не дозволить зберегти правило.

Післяумова:
Новостворене правило автоматизації зберігається в системі, готове до подальшого використання.

---

2. Автоматичне завантаження файлів (U8)

Короткий опис:
Цей сценарій описує автоматичний процес завантаження файлів із визначеного джерела відповідно до налаштувань правила автоматизації.

Передумова:
Користувач має налаштоване правило автоматизації з тригером для завантаження файлів.

Основний сценарій:

1. Користувач налаштовує правило автоматизації, яке включає автоматичне завантаження файлів.
2. Система перевіряє, чи є нові файли для завантаження.
3. Якщо файли є, система завантажує їх з вказаного джерела (наприклад, FTP-сервер).
4. Система надає повідомлення про успішне завантаження файлів.

Альтернативний сценарій:
• Якщо система не змогла підключитися до джерела файлів (наприклад, через помилку з'єднання):

1. Система виводить повідомлення про помилку.
2. Користувач може спробувати повторно підключитись або налаштувати інше джерело.

Винятки:

- Якщо джерело для завантаження файлів не доступне або має помилки з'єднання, система не зможе виконати завантаження файлів.

Післяумова:
Файли завантажуються в систему або користувач отримує повідомлення про помилку при невдачі.

---

3. Автоматичне оновлення статусів (U9)

Короткий опис:
Цей сценарій описує процес автоматичного оновлення статусів в системі залежно від визначених умов та тригерів.

Передумова:
Користувач має налаштоване правило автоматизації для зміни статусів в системі.

Основний сценарій:

1. Користувач створює правило автоматизації, яке передбачає автоматичне оновлення статусів.
2. Система перевіряє умови для оновлення статусу (наприклад, зміна статусу за певний час).
3. Якщо умови виконуються, система автоматично оновлює статус в залежності від налаштувань користувача.
4. Користувач отримує підтвердження про успішне оновлення статусу.

Альтернативний сценарій:
• Якщо система не змогла оновити статус через помилку на сервері або вказану некоректну інформацію:

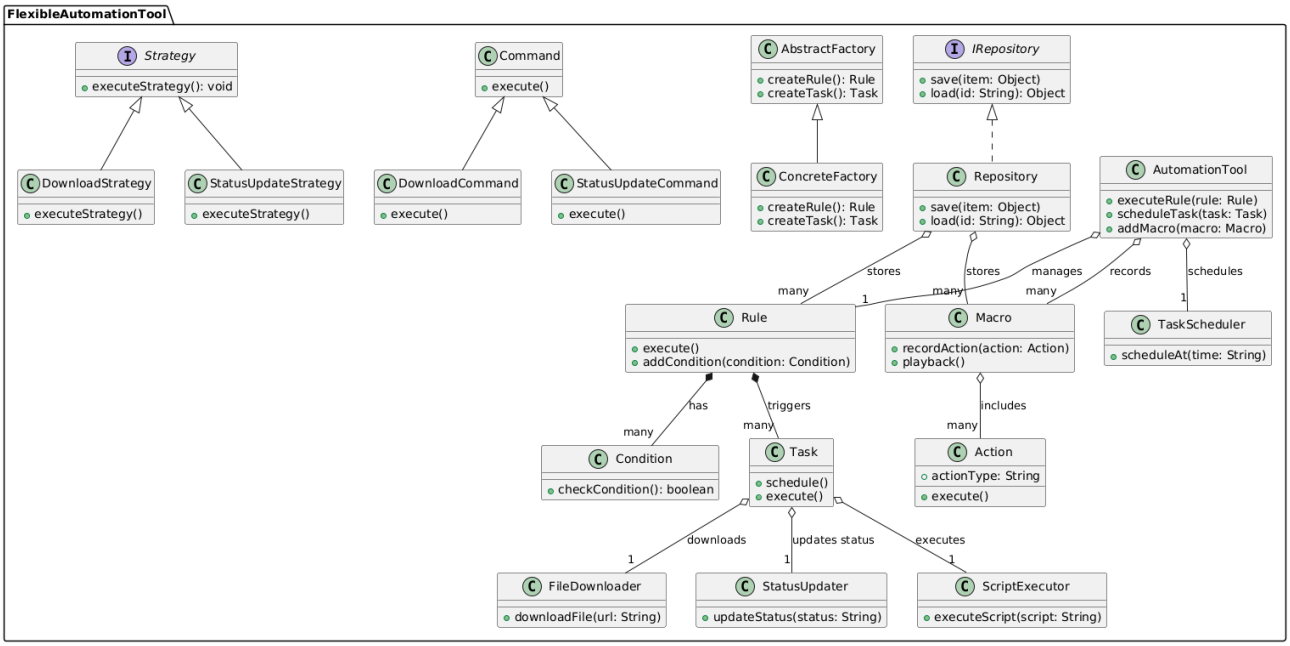
1. Система виводить повідомлення про помилку.
2. Користувач має можливість перевірити налаштування та виправити помилку.

Винятки:

- Якщо тригер не спрацьовує або статус не може бути оновлений через помилку з системою (наприклад, відсутність з'єднання), система виведе повідомлення про помилку.

Післяумова:
Статус оновлюється, або користувач отримує повідомлення про помилку і можливість виправити налаштування.

3. Схема класів



1. **utomationTool**:
   - Основний клас, який керує всіма процесами автоматизації. Він виконує правила (executeRule), планує завдання (scheduleTask) і додає макроси (addMacro).
2. **Rule**:
   - Представляє правило автоматизації, яке включає умови та завдання. Може мати одну або кілька умов (Condition) і завдань (Task), які потрібно виконати, коли умови виконуються.
3. **Condition**:
   - Клас, який перевіряє умови для виконання правил. Умова може бути будь-якою перевіркою (наприклад, час чи активність), яка має метод checkCondition() для визначення, чи виконуються умови.
4. **Task**:
   - Визначає завдання, яке виконується в рамках правила. Завдання може бути планованим для виконання у певний час або може бути негайно виконано. Клас має методи для планування (schedule) і виконання (execute) завдання.
5. **Macro**:
   - Клас для запису і відтворення макросів, що складаються з послідовності дій. Макроси можуть записувати натискання клавіш і дії миші за допомогою класу Action, а також відтворювати їх.
6. **Action**:
   - Описує конкретні дії, які можуть бути записані в макросі. Кожна дія має тип (наприклад, натискання клавіші або рух миші) і метод execute(), що дозволяє виконати її.
7. **FileDownloader**:

- Клас для завантаження файлів через інтернет. Може бути використаний в рамках завдань, які включають завантаження файлів.

8. **StatusUpdater**:
   - Клас для оновлення статусів в різних комунікаторах або програмах, наприклад, зміна статусу в Skype.
9. **ScriptExecutor**:
   - Клас для виконання скриптів. Він дозволяє автоматично виконувати скрипти в рамках завдань автоматизації.
10. **TaskScheduler**:

- Клас, що відповідає за планування завдань на певний час або за розкладом.

11. **IRepository (Інтерфейс)**:

- Інтерфейс для репозиторіїв, що дозволяє зберігати та завантажувати об'єкти (наприклад, правила і макроси) з бази даних чи іншого сховища.

12. **Repository**:

- Реалізація інтерфейсу IRepository, яка зберігає і завантажує об'єкти, такі як правила та макроси.

Взаємозв’язки:

AutomationTool:

- Клас **AutomationTool** керує виконанням **Rule** через метод executeRule().
- Клас **AutomationTool** планує виконання **Task** через метод scheduleTask().
- Клас **AutomationTool** додає **Macro** через метод addMacro().
- Клас **AutomationTool** взаємодіє з **TaskScheduler** для планування завдань.

Rule:

- **Rule** може мати кілька **Condition**, що перевіряються через метод addCondition().
- **Rule** містить список **Task**, що виконуються після виконання умови через методи addTask() (якщо такий є).
- Коли умови виконуються, **Rule** тригерить виконання **Task**.

Condition:

- **Condition** перевіряє свої умови за допомогою методу checkCondition(), визначаючи, чи повинні виконуватися завдання **Task**.

Task:

- **Task** виконує операції, такі як **FileDownloader**, **StatusUpdater** та **ScriptExecutor**.
- **Task** використовує **FileDownloader** для завантаження файлів.
- **Task** використовує **StatusUpdater** для оновлення статусу.
- **Task** використовує **ScriptExecutor** для виконання скриптів.

 Macro:

- **Macro** містить **Action**, які записуються через метод recordAction().
- **Macro** може відтворювати дії через метод playback().

 Action:

- **Action** визначає конкретні дії, які виконуються в макросах (наприклад, натискання клавіші чи рух миші).

 FileDownloader:

- **FileDownloader** виконує завдання завантаження файлів для **Task**.

 StatusUpdater:

- **StatusUpdater** оновлює статуси в різних комунікаторах або програмах, коли це потрібно **Task**.

 ScriptExecutor:

- **ScriptExecutor** виконує скрипти в рамках завдань **Task**.

 TaskScheduler:

- **TaskScheduler** відповідає за планування завдань у майбутньому через метод scheduleAt().

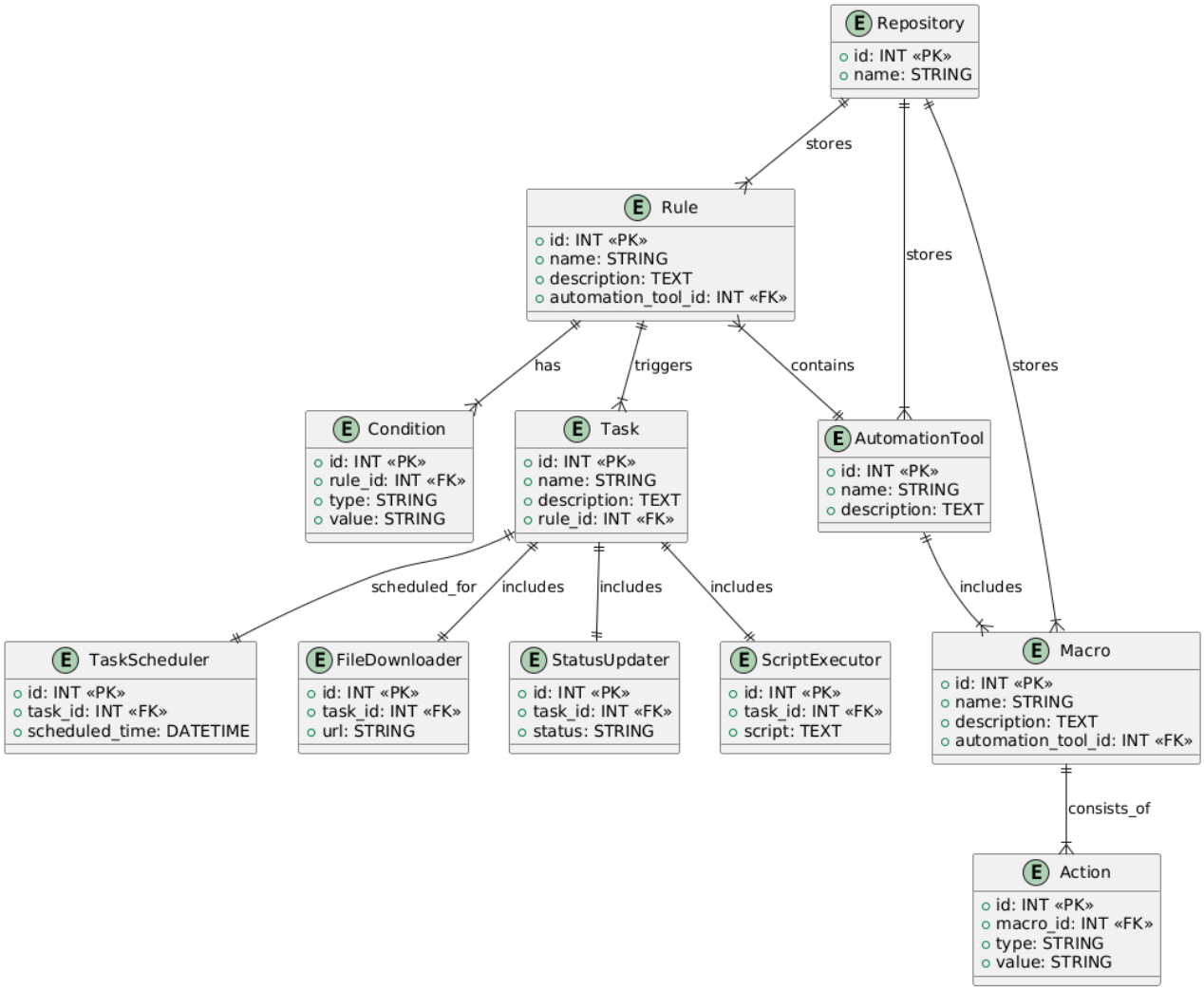
 IRepository (Інтерфейс):

- **IRepository** визначає методи для збереження та завантаження об'єктів.
- **Repository** реалізує інтерфейс **IRepository** і надає методи збереження і завантаження правил і макросів.

 Repository:

- **Repository** зберігає об'єкти **Rule** та **Macro**.
- **Repository** дозволяє **AutomationTool** зберігати та завантажувати правила і макроси.

4. Структура бази даних



1. AutomationTool

a) ID (Primary key) - унікальний ідентифікатор інструменту автоматизації
b) Name - назва інструменту автоматизації (STRING)
c) Description - детальний опис інструменту автоматизації (TEXT)

---

2. Rule

a) ID (Primary key) - унікальний ідентифікатор правила (INT)
b) Name - назва правила автоматизації (STRING)
c) Description - детальний опис правила (TEXT)
d) Automation_tool_id (Foreign key) - зовнішній ключ до таблиці **AutomationTool**, що вказує на інструмент автоматизації, який містить це правило (INT)

---

3. Condition

a) ID (Primary key) - унікальний ідентифікатор умови (INT)
b) Rule_id (Foreign key) - зовнішній ключ до таблиці **Rule**, що вказує на правило, до якого належить ця умова (INT)
c) Type - тип умови (STRING)
d) Value - значення, яке потрібно перевірити для умови (STRING)

---

4. Task

a) ID (Primary key) - унікальний ідентифікатор завдання (INT)
b) Name - назва завдання (STRING)
c) Description - детальний опис завдання (TEXT)
d) Rule_id (Foreign key) - зовнішній ключ до таблиці **Rule**, що вказує на правило, яке активує це завдання (INT)

---

5. Macro

a) ID (Primary key) - унікальний ідентифікатор макросу (INT)
b) Name - назва макросу (STRING)
c) Description - детальний опис макросу (TEXT)
d) Automation_tool_id (Foreign key) - зовнішній ключ до таблиці **AutomationTool**, що вказує на інструмент автоматизації, який містить цей макрос (INT)

---

6. Action

a) ID (Primary key) - унікальний ідентифікатор дії (INT)
b) Macro_id (Foreign key) - зовнішній ключ до таблиці **Macro**, що вказує на макрос, до якого належить ця дія (INT)
c) Type - тип дії (наприклад, натискання клавіші, рух миші) (STRING)
d) Value - значення для дії (наприклад, клавіша для натискання, координати для руху миші) (STRING)

---

7. FileDownloader

a) ID (Primary key) - унікальний ідентифікатор завдання завантаження файлів (INT)
b) Task_id (Foreign key) - зовнішній ключ до таблиці **Task**, що вказує на завдання, яке включає завантаження файлів (INT)
c) URL - адреса файлу для завантаження (STRING)

---

8. StatusUpdater

a) ID (Primary key) - унікальний ідентифікатор завдання для оновлення статусу (INT)

b) Task_id (Foreign key) - зовнішній ключ до таблиці **Task**, що вказує на завдання, яке включає оновлення статусу (INT)
c) Status - новий статус для оновлення в комунікаторі (STRING)

---

9. ScriptExecutor

a) ID (Primary key) - унікальний ідентифікатор завдання для виконання скрипта (INT)
b) Task_id (Foreign key) - зовнішній ключ до таблиці **Task**, що вказує на завдання, яке включає виконання скрипта (INT)
c) Script - текст скрипта для виконання (TEXT)

---

10. TaskScheduler

a) ID (Primary key) - унікальний ідентифікатор завдання для планувальника (INT)
b) Task_id (Foreign key) - зовнішній ключ до таблиці **Task**, що вказує на завдання, яке планується на виконання (INT)
c) Scheduled_time - час, на який заплановане виконання завдання (DATETIME)

---

11. Repository

a) ID (Primary key) - унікальний ідентифікатор сховища (INT)
b) Name - назва сховища (STRING)

Висновки:

Робота над цією лабораторною задачею дозволила розвинути навички моделювання складних інформаційних систем за допомогою діаграм прецедентів, класів і баз даних, а також детально розглянути всі етапи розробки від планування до реалізації логіки взаємодії між компонентами системи. В результаті було створено базову архітектуру інструменту автоматизації, що може бути використана для подальшого розвитку та вдосконалення системи.

Код: https://github.com/Lepseich/trpz/tree/main/lab2/files