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
Діаграма варіантів використання. Сценарії варіантів використання. Діаграми UML. Діаграми класів. Концептуальна модель системи
Варіант: 25. Installer generator

Виконав:

Студент групи ІА-22

Птачик Р.С.

Перевірив:

Мягкий М.Ю.

Київ 2024

Зміст

Тема: Діаграма варіантів використання. Сценарії варіантів використання. Діаграми UML. Діаграми класів. Концептуальна модель системи.

Мета: Ознайомитися з теоретичними відомостями. Намалювати діаграму прецедентів, діаграму класів. Написати основні прецеденти. Розробити основні класи та структуру БД.

Хід роботи:

Тема:

25. Installer generator (iterator, builder, factory method, bridge, interpreter, client-server)

Генератор інсталяційних пакетів повинен мати якийсь спосіб налаштування файлів, що входять в установку, установки вікон з інтерактивними можливостями (галочка - створити ярлик на робочому столі; ввести в текстове поле деякі дані, наприклад, ліцензійний ключ і т.д.). Генератор повинен вивести один файл .exe або .msi.

Короткі теоретичні відомості

UML – це мова для візуального моделювання, яка використовується для специфікації, візуалізації проектування та документування компонентів ПЗ, бізнес-процесів тощо. Мова UML є потужним засобом моделювання, який використовується для будування концептуальних, логічних та графічних моделей складних систем.

Діаграма – це графічне представлення сукупності елементів моделі у формі зв'язкового графа. В UML існують наступні види канонічних діаграм:

- варіантів використання (use case diagram)
- класів (class diagram)
- кооперації (collaboration diagram)
- послідовності (sequence diagram)
- станів (statechart diagram)
- діяльності (activity diagram)
- компонентів (component diagram)
- розгортання (deployment diagram)

Діаграма варіантів використання

Діаграма варіантів використання – це UML діаграма за допомогою якої можна продемонструвати вимоги до системи в графічному вигляді. Ця діаграма є початковою концептуальною моделлю системи. Вона не описує внутрішню структуру системи.

Діаграма варіантів використання потрібна для:

1. Визначення загальної межі функціональності проектованої системи;
2. Сформулювати загальні вимоги до функціональної поведінки проектованої системи
3. Розробка вихідної концептуальної моделі системи;
4. Створення основи для виконання аналізу, проектування, розробки та тестування.

Основні компоненти діаграми: **прецеденти** (use case) – послуги, які надає система, **актори** (actor) – об’єкт, суб’єкт або система, який взаємодіє з системою; **зв’язки** (relationship) – відношення між прецедентами та акторами.

Діаграма класів. Концептуальна модель системи

Діаграми класів використовують при моделюванні програмних систем. Вони є формою опису системи, демонструючи її структуру, але не відображаючи поведінку об’єктів. Діаграми класів містять: класи, інтерфейси та відносини між ними.

Клас – це основний будівельний блок ПС. Клас має назву, атрибути та операції. На діаграмі зображується як прямокутник, який розділений на 3 області:

- Верхня – назва класу;
- Середня – опис атрибутів;
- Нижня – назви операцій, які надає клас.

Атрибути класу визначають склад та структуру даних, які зберігаються в об’єктах цього класу. Атрибути містять назву та тип, який визначає які дані він представляє. Для атрибутів можна задавати видимість:

- Відкритий (public) – атрибут видно для будь-якого іншого класу (об'єкта);

- Захищений (protected) – атрибут видно для нащадків цього класу;
- Закритий (private) – атрибут не видно зовнішніми класами (об'єктами) та може використовуватися лише об'єктом, що його містить.

Операції класу - визначення запитів, які повинні виконувати об'єкти даного класу. Кожна операція містить ім’я операції, тип значення, який повертається та список параметрів, який може бути порожнім. Для операцій можна задавати «видимість». Закриті операції є внутрішніми для об'єктів класу та недоступні з інших об'єктів. Інші утворюють інтерфейсну частину класу і є засобом інтеграції класу до ПС.

Діаграми класів зазвичай показують асоціації та об’єднання:

- **Асоціація** – найбільш загальний вид зв'язку між двома класами системи. Відображає використання одного класу іншим за допомогою певної якості або поля.
- **Об’єднання** (**успадкування**) - використовується щоб показати зв'язок між класом-батьком та класом-нащадком. Воно вводиться на діаграму, коли виникає різновид якогось класу, а також у тих випадках, коли в системі виявляються кілька класів, які мають подібну поведінку.

Логічна структура бази даних. Нормальні форми

Існує 2 моделі БД – логічна та фізична:

- **Логічна модель бази даних** є структурою таблиць, уявлень, індексів та інших логічних елементів бази даних, що дозволяють власне програмування та використання бази даних.
- **Фізична модель бази даних** представляє собор набір бінарних даних у вигляді файлів, структурованих та згрупованих згідно з призначенням.

Проектування БД полягає в опрацюванні архітектури програмної системи, оскільки база даних потрібна для зберігання даних, одержуваних з програмних класів.

Нормальна форма – властивість, яка допомагає уникати надмірності, що потенційно призводить до логічно помилкових результатів вибірки або зміни даних.

Процес перетворення відносин БД до виду, що відповідає нормальним формам, називається нормалізацією. **Нормалізація** призначена для приведення структури БД до виду, що забезпечує мінімальну логічну надмірність, і не має на меті зменшення або збільшення продуктивності

роботи або зменшення або збільшення фізичного обсягу бази даних. Кінцевою метою нормалізації є зменшення потенційної суперечливості інформації, що зберігається в базі даних.

Нормальні форми:

- **1НФ** – кожний атрибут відношення повинен мати лише 1 значення;
- **2НФ** – виконання 1НФ; кожен неключовий атрибут залежить від ключа.
- **3НФ** – виконання 2НФ; відсутні транзитивні функціональні залежності неключових атрибутів від ключових.
- **НФ Бойса-Кодда** – посилена 3НФ; кожна функціональна залежність має певний ключ.

Діаграма прецедентів

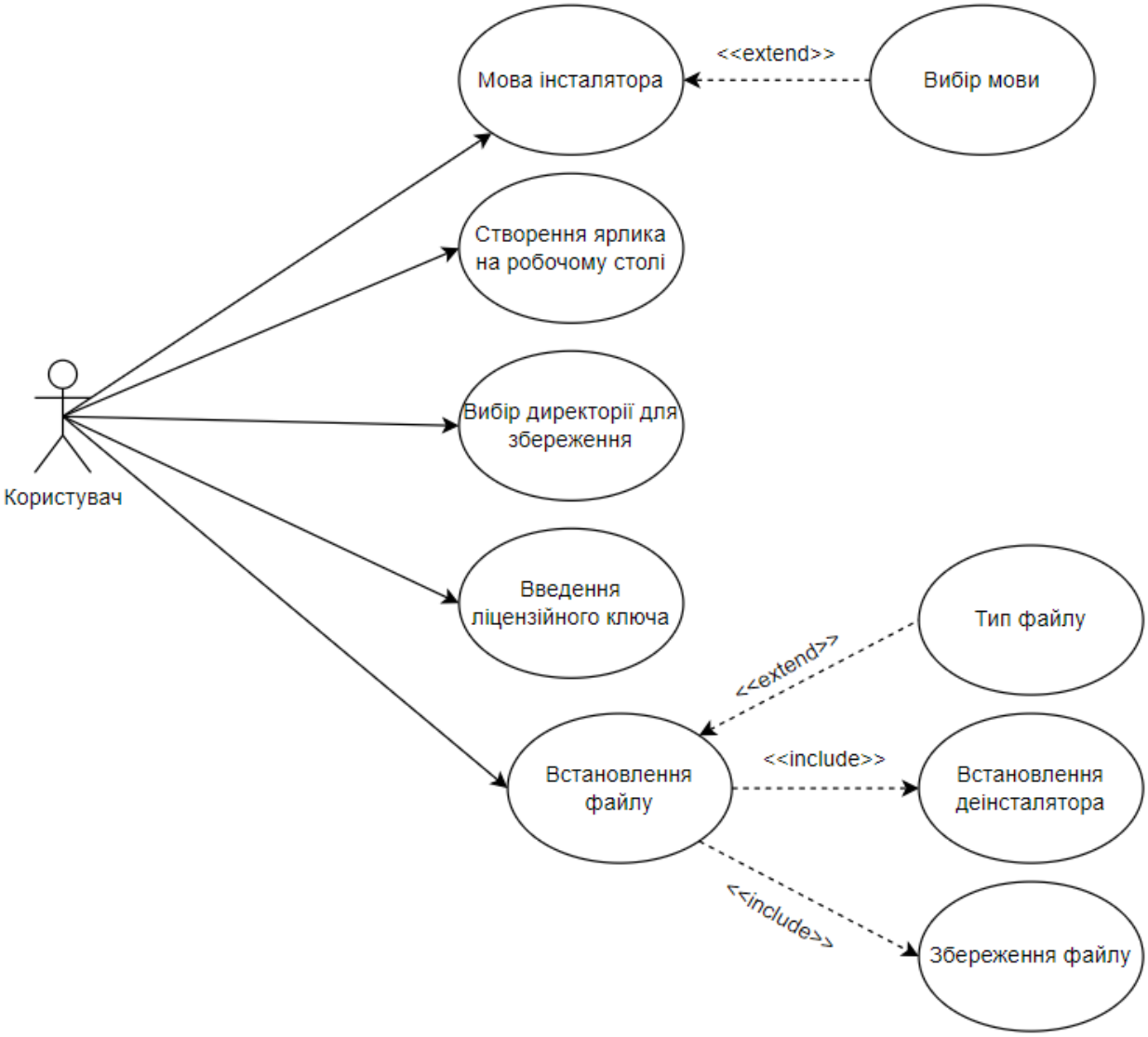

Рис. 1 – Діаграма прецедентів

На рис. 1 зображено діаграму прецедентів, яка показує як користувач може використовувати систему.

Прецеденти:

- **Мова інсталятора:** користувач може вибрати мову інтерфейсу інсталятора (англійську чи українську);
- **Створення ярлика на робочому столі:** можна створити ярлик за бажанням користувача;
- **Вибір директорії для збереження:** створення (або вибір) папки де буде зберігатися файл;
- **Введення ліцензійного ключа:** ключ для встановлення файлу;
- **Встановлення файлу:** користувач вибирає тип файлу, який буде збережено (.exe або .msi); в директорії автоматично створюється деінсталятор; користувач зберігає файл на пристрої натиснувши на «Зберегти».

Сценарії використання

Основні сценарії використання:

Прецедент: Базове встановлення файлу

Передумови: Користувач встановив інсталятор

Післяумови: Файл встановлюється без додаткових налаштувань

Актор: Користувач

Короткий опис: Користувач не вибирає додаткові налаштування, а файл встановлюється з базовими налаштуваннями

Основний сценарій:

1. Користувач запускає інсталятор
2. Користувач не змінює мову інсталятора і наступні вікна та текст в них будуть на базовій мові інсталятора - англійській
3. Користувач вводить ліцензійний ключ. Якщо ключ неправильний, наступні використання інсталятора недоступні
4. Користувач не вибирає «Створити ярлик на робочому столі» і ярлик не створюється
5. Користувач не змінює директорію для файлу і файл створюється в закладеній директорії, наприклад «C:\InstalledFile»

6. Користувач не вибирає тип файлу і файл зберігається з базовим типом - .exe
7. Появляється останнє вікно де демонструються налаштування, в даному випадку лише директорія файлу та тип фалу. Користувач встановлює файл і файл зберігається на пристрої з базовим розширенням .exe, а разом з ним встановлюється деінсталятор

Винятки:

- Якщо директорія для файлу вже існує, наприклад «C:\InstalledFile», то користувачу пропонується вибрати іншу директорію або встановити файли в дану директорію.
- Якщо файл вже існує в директорії, то користувачу пропонується замінити його.

Прецедент: Встановлення файлу з додатковими налаштуваннями

Передумови: Користувач встановив інсталятор

Післяумови: Файл зберігається на пристрої з певними налаштуваннями

Актор: Користувач

Короткий опис: Користувач використовує додаткові налаштування, які надає інсталятор, та зберігає файл з додатковими налаштуваннями

Основний сценарій:

1. Користувач запускає інсталятор
2. Користувач змінює мову інсталятора і наступні вікна та текст в них будуть на вибраній мові, наприклад на українській
3. Користувач вводить ліцензійний ключ. Якщо ключ неправильний, наступні використання інсталятора недоступні
4. Користувач вибирає «Створити ярлик на робочому столі» і коли файл буде збережено на пристрої додатково буде створено ярлик
5. Користувач змінює директорію для збереження файлу, наприклад «C:\NewDirectory» і файл буде збережено в новій директорії
6. Користувач змінює тип файлу – вибирає тип .msi
7. Появляється останнє вікно де демонструються налаштування, в даному випадку «Створення ярлика на робочому столі», директорія файлу та тип файлу - .msi. Користувач встановлює файл і файл зберігається на пристрої з розширенням .msi, а разом з ним встановлюється деінсталятор і на робочому столі з’являється ярлик

Винятки:

- Якщо файл вже існує в директорії, то користувачу пропонується замінити його.

Прецедент: Видалення файлу за допомогою деінсталятора

Передумови: Користувач встановив файл

Післяумови: Файл буде видалено разом з деінсталятором

Актор: Користувач

Короткий опис: Користувач запускає деінсталятор де він підтверджує що хоче видалити файл. Після підтвердження файл, разом з деінсталятором, буде видалено з пристрою

Основний сценарій:

1. Користувач встановив файл
2. Користувач відкриває деінсталятор
3. Деінсталятор потребує підтвердження користувача на видалення файлу
4. Отримавши підтвердження деінсталятор видаляє встановлений файл та сам деінсталятор
5. Після видалення на пристрої користувача не буде встановленого файлу та пов’язаних з ним складових (деінсталятор)

Винятки: Немає

Діаграма класів

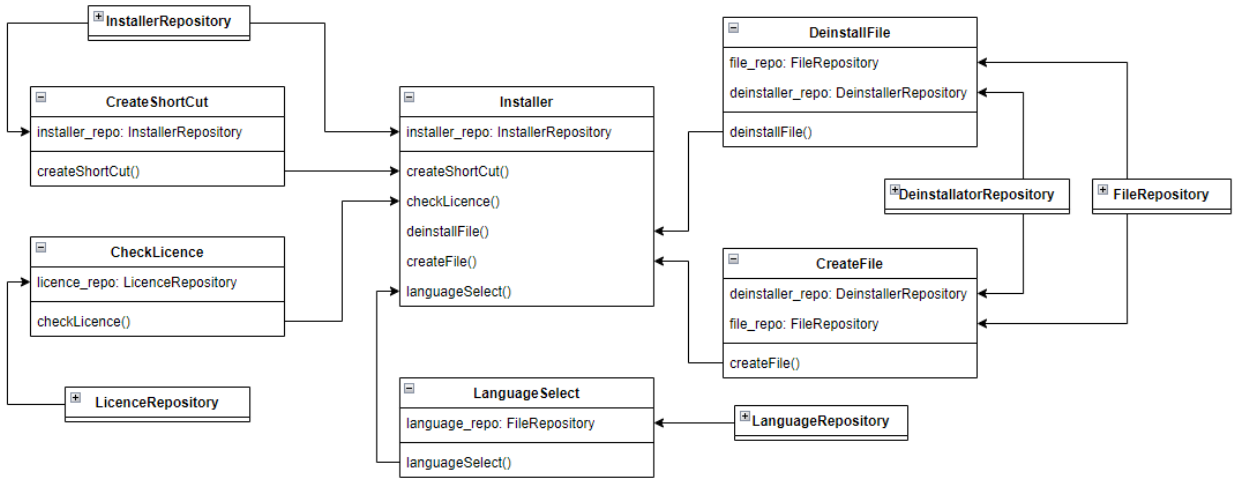

Рис. 2 – Діаграма класів

Діаграма класів має: репозиторії, методи та головний клас «Installer».

Installer:

- **installer_repo** - бере дані з InstallerRepository
- **createShortCut(), checkLicence(), deinstallFile(), createFile(), languageSelect()** - методи, які використовує «Installer»

CreateShortCut:

- **installer_repo** - бере дані з InstallerRepository, а саме isShortcut
- **createShortCut()** - метод для створення ярлика на робочому столі

CheckLicence:

- **licence_repo** - бере дані з LicenceRepository
- **checkLicence()** - метод для перевірки ліцензійного ключа

DeinstallFile:

- **file_repo, deinstaller_repo** - бере дані з FileRepository та DeinstallerRepository відповідно
- **deinstallFile()** - метод для видалення встановленого файлу та деінсталятора

CreateFile:

- **file_repo, deinstaller_repo** - бере дані з FileRepository та DeinstallerRepository відповідно
- **createFile()** - метод для створення файлу та деінсталятора

LanguageSelect:

- **language_repo** - бере дані з LanguageRepository
- **languageSelect()** - метод для вибору мови інсталятора

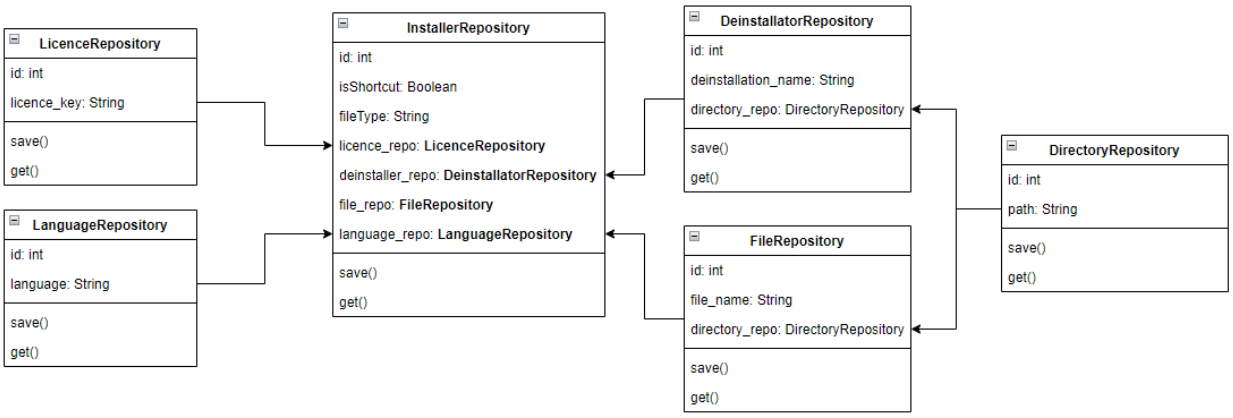

Рис. 3 - Шаблон репозиторію

Репозиторій містить наступні дані:

InstallerRepository:

- **isShortcut** - містить дані про тип файлу, який буде встановлено
- **fileType** - містить дані про створення ярлика
- **deinstaller_repo, file_repo, licence_repo, language_repo** - дані з репозиторіїв DeinstallerRepository, FileRepository, LicenceRepository, LanguageRepository відповідно

LicenceRepository:

- **licence_key:** містить дані про ліцензійний ключ - назву ключа

DirectoryRepository:

- **path:** містить дані про розташування файлу

FileRepository:

- **file_name:** містить дані про назву файла
- **directory_repo:** бере дані з DirectoryRepository

DeinstallerRepository:

- **deinstallation_name:** містить дані про назву деінсталятора
- **directory_repo:** бере дані з DirectoryRepository

LanguageRepository:

- **language:** містить дані про мову

Структура системи баз даних

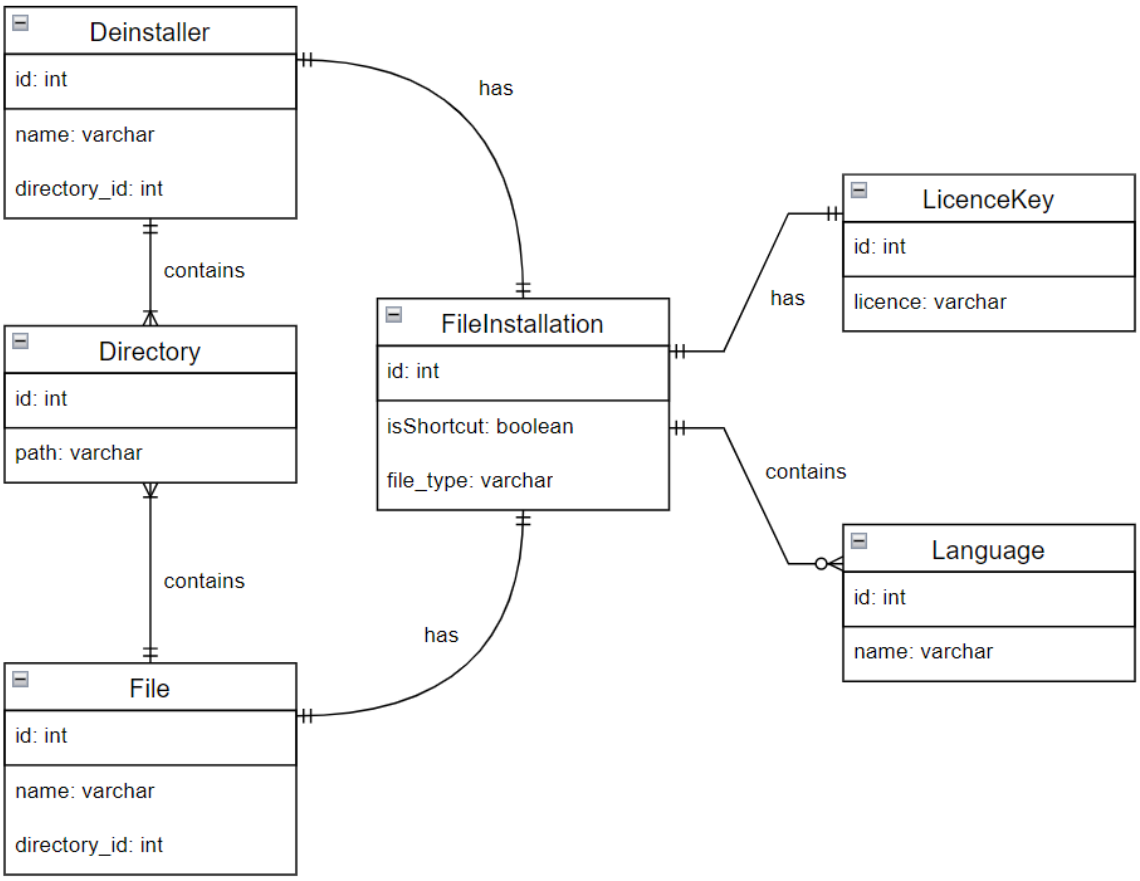

Рис. 4 - Структура бази даних

База даних містить наступні таблиці:

- **FileInstallation:** містить дані про інсталятор; містить дані про тип файлу, який буде встановлено, та дані про створення ярлика; пов’язує в собі всі таблиці.
- **LicenceKey:** містить дані про ліцензійний ключ - назву ключа.
- **Language:** містить дані про мову - яку мову використовує інсталятор.
- **Deinstallator:** містить дані про деінсталятор - назву деінсталятора; потребує дані з таблиці Directory, тобто розташування деінсталятора.
- **File:** містить дані про встановлювальний файл - назву файла; потребує дані з таблиці Directory, тобто розташування файлу.
- **Directory:** містить дані про розташування файлів.

Код можна знайти за посиланням:
https://github.com/FryMondo/TRPZ/tree/master/InstallGenerator

Висновок: на цій лабораторній роботі я створив діаграму прецедентів, яка описує функціональність системи з точки зору взаємодії користувача з

інсталятором; діаграму класів, яка відображає статичну структуру системи; описав сценарій використання, які деталізують взаємодію користувача з інсталятором, демонструючи покроковий процес встановлення файлу; створив структуру бази даних, яка містить таблиці, що забезпечують збереження даних та створив основні класи які реалізовують шаблон Репозиторію для взаємодії з базою даних.