НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Програмне забезпечення для моніторингу натискання клавіш з автоматичним переключенням мови та корекцією

Виконав студент IV курсу, групи ІТ-94
(шифр групи)

Шульга Сергій Сергійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник ст. викл к.т.н. Стельмах О.П. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації ст.викл. Вітковська І.І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент к.т.н. доцент Сперкач М.О. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Шульга Сергію Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Програмне забезпечення для моніторингу натискання клавіш з автоматичним переключенням мови та корекцією

керівник проєкту Стельмах Олександр Петрович, ст. викл к.т.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «_31_» травня 2023 р. №2101-с_____

2. Термін подання студентом проєкту « 17 » червня 2023 року

3. Вихідні дані до проєкту: технічне завдання_____

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: загальні положення, змістовний опис та аналіз предметної області, аналіз існуючих рішень та відомих програмних продуктів, розробка функціональних та нефункціональних вимог.
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, архітектура програмного забезпечення, конструювання програмного забезпечення та аналіз безпеки даних. _____
3) Аналіз якості та тестування програмного забезпечення: аналіз якості програмного забезпечення, опис процесів тестування, опис контрольного прикладу. _____
4) Впровадження та супровід програмного забезпечення_____
_____

5. Перелік графічного матеріалу

1) Діаграма бізнес процесів _____
2) Схема структурних класів програмного забезпечення _____
3) Схема структурна варіантів використання _____
_____

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» березня 2023 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	10.03.2023	
2	Аналіз існуючих методів розв’язання задачі	15.03.2023	
3	Постановка та формалізація задачі	21.03.2023	
4	Розробка інформаційного забезпечення	02.04.2023	
5	Алгоритмізація задачі	05.04.2023	
6	Обґрунтування вибору використаних технічних засобів	07.04.2023	
7	Розробка програмного забезпечення	09.04.2023	
8	Налагодження програми	17.04.2023	
9	Виконання графічних документів	20.04.2023	
10	Оформлення пояснювальної записки	25.04.2023	
11	Подання ДП на попередній захист	02.06.2023	
12	Подання ДП рецензенту	12.06.2023	
13	Подання ДП на основний захист	17.06.2023	

Студент _____ Сергій ШУЛЬГА
(підпис) (ініціали, прізвище)

Керівник _____ Олександр СТЕЛЬМАХ
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 30 таблиць, 31 рисунок та 9 джерел – загалом 60 сторінок.

Дипломний проєкт присвячений програмному забезпеченню для моніторингу натискання клавіш з автоматичним переключенням мови та корекцією.

Метою дипломного проекту є полегшення роботи з набором тексту для користувачів Macintosh OS.

Об'єкт дослідження даного дипломного проекту є програмне забезпечення для моніторингу натискання клавіш з автоматичним переключенням мови та корекцією.

Предметами дослідження даного дипломного проекту є методи та алгоритми моніторингу натиснутих клавіш, автоматична зміна мови вводу клавіатури, методи та алгоритми корекції слова.

У першому розділі розглянуто аналіз вимог до програмного забезпечення, а саме загальні положення, змістовний опис і аналіз предметної області, аналіз існуючих продуктів та формування вимог до програмного забезпечення.

Другий розділ присвячено моделюванню та конструюванню програмного забезпечення.

Третій розділ присвячено аналізу якості та тестуванню програмного забезпечення.

У четвертову розділі виконано впровадження та супровід програмного забезпечення.

Програмне забезпечення впроваджено на онлайн платформі GitHub.

КЛЮЧОВІ СЛОВА: ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, АВТОМАТИЧНА ЗМІНА МОВИ, ЕМУЛЯЦІЯ, НИЗЬКОРІВНЕВІ ФУНКЦІЇ.

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 30 tables, 31 figures and 9 sources - a total of 60 pages.

The diploma project is dedicated to software for monitoring keystrokes with automatic language switching and correction.

The goal of the thesis project is to make typing easier for Macintosh OS users.

The research object of this diploma project is software for monitoring keystrokes with automatic language switching and correction.

The research subjects of this diploma project are methods and algorithms for monitoring pressed keys, automatic keyboard input language change, methods and algorithms for word correction.

The first chapter deals with the analysis of software requirements, namely general provisions, meaningful description and analysis of the subject area, analysis of existing products and the formation of software requirements.

The second chapter is devoted to software modeling and design.

The third chapter is devoted to quality analysis and software testing.

In the fourth section, the implementation and maintenance of the software was carried out.

The software is implemented on the online platform GitHub

KEY WORDS: SOFTWARE, AUTOMATIC LANGUAGE CHANGE, EMULATION, LOW-LEVEL FUNCTIONS.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІТ-9430.045200.01.91	Технічне завдання	10	
3	А4	КПІ.ІТ-9430.045200.02.81	Пояснювальна записка	60	
4	А4	КПІ.ІТ-9430.045200.03.12	Текст програми	19	
5	А4	КПІ.ІТ-9430.045200.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІТ-9430.045200.05.34	Керівництво користувача	7	
7	А3	КПІ.ІТ-9430.045200.06.99	Діаграма бізнес процесів	1	
8	А3	КПІ.ІТ-9430.045200.06.99	Схема структурних класів	1	
			програмного забезпечення		
9	А3	КПІ.ІТ-9430.045200.06.99	Схема структурна варіантів	1	
			використання		

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІТ-9430.045200.00.90		
Розроб.		Шульга С.С.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Стельмах О.П.					
Н.контр.		Ліщук К.І.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІТ-94	
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2023 р.

Програмне забезпечення для моніторингу натискання клавіш з автоматичним переключенням мови та корекцією

Технічне завдання

КПІ.ІТ-9430.045200.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олександр СТЕЛЬМАХ

Нормоконтроль:

_____ Ірина ВІТКОВСЬКА

Виконавець:

_____ Сергій ШУЛЬГА

Київ – 2023

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: програмне забезпечення для моніторингу натискання клавіш з автоматичним переключенням мови та корекцією.

Галузь застосування: робота з набором тексту на операційній системі Macintosh.

Наведене технічне завдання поширюється на розробку програмного забезпечення для моніторингу натискання клавіш з автоматичним переключенням мови та корекцією КПІ.ІТ-9430.045200.01.91, котра використовується для автоматичної зміни мови вводу клавіатури та корекцією написаного слова та призначена для роботи з набором тексту на операційній системі Macintosh.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки програмного забезпечення для моніторингу натискання клавіш з автоматичним переключенням мови та корекцією є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для автоматичної зміни мови вводу клавіатури на правильну та виправлення неправильно написаного слова.

Метою розробки є полегшення роботи при наборі тексту для користувачів операційної системи Macintosh.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Для користувача:

– аналіз доступних для роботи мов;
– обрання мов для роботи;
– пауза роботи програмного забезпечення;
– відновлення роботи програмного забезпечення;
– запуск програмного забезпечення.

Додаткові вимоги:

– розробки повинна відбуватися на платформі операційної системи Macintosh.

4.2 Вимоги до надійності

– передбачити можливість правильно обрати мови для роботи програмного забезпечення;
– передбачити контроль введення інформації та захист від некоректних дій користувача.

4.3 Умови експлуатації

Умови експлуатації згідно ДСанПІН 3.3.2.007-98.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на платформах Macintosh OS.

Мінімальна конфігурація технічних засобів:

- тип процесору: Intel Core I5;
- об‘єм озп: 4 гб;
- підключення до мережі інтернет зі швидкістю від 20 мегабіт;
- об’єм жорсткого диску: від 512 мб;
- рекомендована конфігурація технічних засобів
- тип процесору: intel core i7;
- об‘єм озп: 8 гб;
- підключення до мережі Інтернет зі швидкістю від 100 мегабіт;
- об’єм жорсткого диску: від 4 гб.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства Mac OS.

Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі: введене користувачем слово у форматі тексту, доступні мови для зміни та поточна мова вводу клавіатури у форматі тексту.

4.5.1 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі: виправлене на правильну мову слово та змінена мова вводу клавіатури.

4.5.2 Вимоги до мови розробки

Розробку виконати на мові програмування Java.

4.5.3 Вимоги до середовища розробки

Розробку виконати за допомогою середовища IntelliJ Idea Ultimate 2023.1.2.

4.5.4 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді набору файлів.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Згенерувати інсталяційну версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

- пояснювальна записка;
- технічне завдання;
- текст програми;
- програма та методика тестування;
- керівництво користувача.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

- схема структурних бізнес процесів;
- схема структурних класів програмного забезпечення;
- схема структурна варіантів використання.

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

	Назва етапів	Строк	Звітність
1	Вивчення рекомендованої літератури	10.03.2023	
2	Аналіз існуючих методів розв’язання задачі	15.03.2023	
3	Постановка та формалізація задачі	21.03.2023	
4	Розробка інформаційного забезпечення	02.04.2023	
5	Алгоритмізація задачі	05.04.2023	
6	Обґрунтування вибору використаних технічних засобів	07.04.2023	
7	Розробка програмного забезпечення	09.04.2023	
8	Налагодження програми	17.04.2023	
9	Виконання графічних документів	20.04.2023	
10	Оформлення пояснювальної записки	25.04.2023	
11	Подання ДП на попередній захист	02.06.2023	
12	Подання ДП рецензенту	12.06.2023	
13	Подання ДП на основний захист	17.06.2023	

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: Програмне забезпечення для моніторингу натиснутих клавіш з автоматичним переключенням мови та корекцією

КПІ.ІТ-9430.045200.02.81

Київ – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
API	–	Application programming interface, прикладний програмний Інтерфейс
SDK	–	Software development kit
IT	–	Інформаційні технології
ОС	–	Операційна система.

ВСТУП

В сучасному інформаційному суспільстві, де використання комп’ютерів і Інтернету стало невід’ємною частиною нашого повсякденного життя, ефективне використання клавіатури стає критично важливою навичкою. Однак, багато користувачів стикаються з проблемою неправильної розкладки клавіатури при наборі тексту, що призводить до помилок, сповільнює робочий процес, збиває з думки та викликає дискомфорт. Для вирішення цієї проблеми потрібен інструмент, здатний автоматично перемикати розкладку клавіатури та корегувати введені символи.

Світові тенденції свідчать про зростаючу потребу в інтелектуальних системах, здатних адаптуватися до потреб користувачів, і забезпечувати оптимальні умови для роботи. Декілька компаній вже займаються дослідженнями та розробкою в області автоматичної корекції неправильної розкладки клавіатури при наборі тексту. Однак, більшість існуючих рішень мають певні недоліки, обмежені можливості або потребують додаткових дій від користувача.

Метою даного дипломного проекту є розробка програмного забезпечення для моніторингу натискання клавіш з автоматичним переключенням мови та корекцією. Програма буде здатна визначити, коли користувач пише слова на неправильній розкладці клавіатури і автоматично перемикати розкладку на вірну.

Дане програмне забезпечення має потенціал для застосування в різних сферах. Воно може бути використане в професійному середовищі, де швидкий і точний друк тексту відіграє важливу роль, наприклад при роботі с текстовими редакторами, електронними таблицями або програмуванні. Крім того, дане програмне забезпечення може бути корисним для вивчення іноземних мов, де правильна розкладка клавіатури має особливо важливе значення.

В даному дипломному проекті ми будемо докладно розглядати алгоритми і методи, необхідні для реалізації програмного забезпечення моніторингу натискання клавіш з автоматичним перемиканням мови і корекцією. Також, проведемо експерименти і оцінимо ефективність програми.

Очікується, що результати цього дослідження і розробки приведуть до створення ефективного інструмента, який буде здатен значно полегшити роботу користувачів і забезпечити більш точний і швидкий друк тексту на комп’ютері.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Програмне забезпечення для моніторингу натискання клавіш з автоматичним переключенням мови – певний набір бібліотек, API, класів, методів та сторонніх мов сценаріїв, які виконують низькорівневі функції для роботи з клавіатурою та мовою вводу клавіатури даного персонального комп’ютера [1].

Низькорівневі функції – це функції, які працюють на більш низькому рівні абстракції, і надають більш прямий доступ до апаратного забезпечення або до операційної системи комп’ютера. Саме до таких функцій відносяться глобальні слухачі клавіатури та методи переключення мови вводу клавіатури.

Бібліотеки в Java представляють собою набір попередньо написаних класів, інтерфейсів та методів, які надають готові рішення для певних задач.

API (Application Programming Interface) – набір певних правил і протоколів, які дозволяють програмним компонентам взаємодіяти один з одним. API забезпечують абстракцію між різними компонентами програмного забезпечення, дозволяючи їм взаємодіяти без необхідності знати подробиці реалізації один одного. Фактично, API – сторонній інтерфейс, який дозволяє основній програмі взаємодіяти з іншими сервісами та використовувати їх функціонал без необхідності повної реалізації цього функціоналу у основній програмі.

Сторонні мови сценаріїв призначені для створення сценаріїв в операційних системах. Тобто, з їх допомогою можна створювати певні скрипти або макроси, які автоматизують задачі на рівні операційної системи.

Макроси та скрипти представляють собою набір інструкцій або команд, які дозволяють користувачам автоматизувати задачі та покращити ефективність і продуктивність роботи на комп’ютері. При запуску вони повторюють свої дії в точності так, як вони були записані.

1.2 Змістовний опис і аналіз предметної області

Основною задачею даного дипломного проекту є розробка програмного забезпечення для моніторингу натискання клавіш та автоматичного перемикання мови.

Наразі у відкритому доступі є декілька продуктів, які частково реалізують цю задачу. Але вони мають певні недоліки, які будуть виправлені в програмному забезпеченні даного дипломного проекту. Насамперед, це несумісність програмного забезпечення з операційною системою Macintosh OS. На думку автора дипломного проекту, це пов’язано з тим, що дана операційна система є закритою, і в ній доволі складно змінювати або відстежувати певні дані та функції на рівні операційної системи. Також більшість програм працюють на базі яндексу, або інших російських компаній, використання яких є морально недопустимим на території України. А європейська та американська частина подібних програм як правило, не підтримує українську мову. Крім того, майже все програмне забезпечення є платним. Автор даного дипломного проекту вважає, що такий функціонал як автоматичне змінення мови розкладки клавіатури є дуже корисним для усіх користувачів, і прагне зробити його доступним для всіх.

На даний момент часу існують декілька нових розробок, які значно покращать ефективність роботи програми, і зроблять її більш простою як для подальшого вдосконалення, так і для повсякденного використання.

Використання даного програмного забезпечення надасть можливість усім користувачам Macintosh OS значно покращити швидкість та ефективність роботи за комп’ютером.

1.3 Аналіз існуючих рішень та успішних IT-проектів

Проаналізуємо відоме на сьогодні програмне забезпечення у даній області та технічні рішення, що допоможуть у реалізації програмного

забезпечення для моніторингу натискання клавіш та автоматичного перемикання мови. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

1.3.1 Аналіз відомих алгоритмічних та технічних рішень

Оскільки готові програми, які можна завантажити, не надають доступу до коду, було прийнято рішення писати оригінальне програмне забезпечення з нуля, використовуючи різні інструменти для вирішення окремих задач, та об’єднання їх в єдину програму.

Структура програмного забезпечення буде представлена у класичному для мови програмування Java форматі: наслідування класів. Це є однією з головних переваг даної мови програмування. При створенні програмного забезпечення кожен клас буде наслідувати один чи декілька інших, і викликатися у потрібний момент часу виконання програми.

Перевагою даного рішення можна визначити його ефективність: певний метод буде виконуватися у необхідний момент часу, тим самим підвищуючи швидкодію програми та її ефективність.

1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

Допоміжними засобами розробки програмного забезпечення для моніторингу натискання клавіш та автоматичного перемикання мови є стороннє API, декілька мов сценаріїв та бібліотек. Розглянемо їх докладніше.

Для більш структурованої розробки даного програмного забезпечення, розділимо допоміжні програмні засоби у порядку їх використання.

Для початку роботи оберемо мову програмування та IDE. У якості мови програмування була обрана Java, тому що вона з самого початку підтримує велику кількість стандартних бібліотек та методів, і крім того, для неї було розроблену велику кількість сторонніх інструментів, які підтримуються і

зараз. Також Java підтримує усі свої попередні версії, що допоможе в майбутньому модифікувати та підтримувати дане програмне забезпечення без необхідності повного змінення коду. Слід зазначити, що Java є кросплатформеною мовою програмування. Тому для подальшої реалізації даного програмного забезпечення на інших операційних системах (Windows OS, Linux OS) буде необхідно змінити декілька методів, які відповідають за низькорівневі функції окремої операційної системи. Тобто, повністю переписувати код не є необхідним. Окрім того, Java є дуже популярною мовою програмування, що надає можливість в майбутньому розвивати дане програмне забезпечення у команді зацікавлених розробників.

Середовищем розробки була обрана IntelliJ IDEA. Вона має увесь необхідний інструментарій для роботи з даним проектом на основі мови програмування Java. Також, IntelliJ Idea пропонує розумний кодовий аналіз, що допоможе швидко виявляти вразливі місця коду, і виправляти їх для більш швидкої роботи. Слід зазначити, що IntelliJ Idea є простою у використанні: це середовище розробки має інтуітивно зрозумілий інтерфейс, що значно полегшить процес розробки програмного забезпечення.

Першою задачею для реалізації програмного забезпечення даного дипломного проекту є створення глобального слухача клавіатури, тобто методу, який буде відстежувати усі дії з клавіатурою по всій операційний системі. Для його реалізації можна використати декілька інструментів: перший з них – бібліотека Java Native Access (JNA).[2] А саме клас Carbon, який дає можливість працювати з низькорівневими функціями в Macintosh OS. Але один з його методів – HotKeyHandlerProc не підтримується в останніх версіях операційної системи Macintosh OS. Тому замість бібліотеки Java Native Access можна використовувати бібліотеку Java Native Hook (JNH). [3] Java Native Hook підтримується на даний момент часу, і надає змогу відстежувати всі дії методів вводу, таких як клавіатура та миша.

Другою задачею є обробка слова та аналіз його правильності відносно поточної мови розкладки клавіатури. В цій задачі з допоміжних програмних

засобів можна визначити API Google Translator.[4] З його допомогою ми зможемо відправляти поточне слово і отримувати відповідь: чи наявне воно в словнику відповідної мови. Також частковим рішенням цієї задачі є імпорт повного словника необхідних мов, і пошук у ньому потрібного слова. Але такий спосіб вирішення задачі має декілька недоліків. Перший - в такому випадку, програма буде менш ефективною і буде займати набагато більше місця. Другий - програмне забезпечення буде обмежено тим словниковим запасом, який в нього імпортує розробник. Google Translator API має декілька переваг. По-перше, воно підтримує велику кількість мов, що робить програмне забезпечення більш універсальним. По-друге, Google Translator API постійно навчається, отримуючи нові слова з усього світу, і запам'ятовуючи їх.

Третьою задачею є перемикання мови розкладки клавіатури. Для реалізації цього методу, доступно декілька рішень: використання бібліотеки JNI (Java Native Interface). [5] Але дана бібліотека не надає нам необхідного доступу до рівня операційної системи, а саме до мови вводу та її зміни. Також ми можемо розглянути емуляцію натискання на клавіши для зміни мови, але таке рішення є сумнівним, тому що комбінація клавіш для перемикання мови у всіх різна, і Macintosh OS не надає можливість таким чином змінювати поточну мови розкладки клавіатури. Ще одним рішенням даної задачі є використання спеціально розробленої для Macintosh OS мови сценаріїв – Apple Script. [6] З його допомогою можливо створити певні скрипти (сценарії), які будуть перемикати поточну мову розкладки клавіатури. Таким чином, можна створити єдиний скрипт, який буде спроможний на рівні операційної системи перемикати мову вводу клавіатури.

Четвертою задачею для реалізації програмного забезпечення даного дипломного проекту є автоматичний друк поточного слова на правильній розкладці клавіатури. Для цього можна використати бібліотеку Robot, яка надає змогу реалізувати емуляцію натискання клавіш клавіатури. Для того, щоб програмне забезпечення розуміло, яке слово необхідно емулювати,

можемо зберігати кей-коди неправильно написаного слова, та використовувати їх для классу Robot.

1.3.3 Аналіз відомих програмних продуктів

Розглянемо декілька готових програмних продуктів, які частково або повністю реалізують функціонал даного дипломного проекту.

– Punto Switcher. Логотип Punto Switcher зображений на рисунку 1.1.

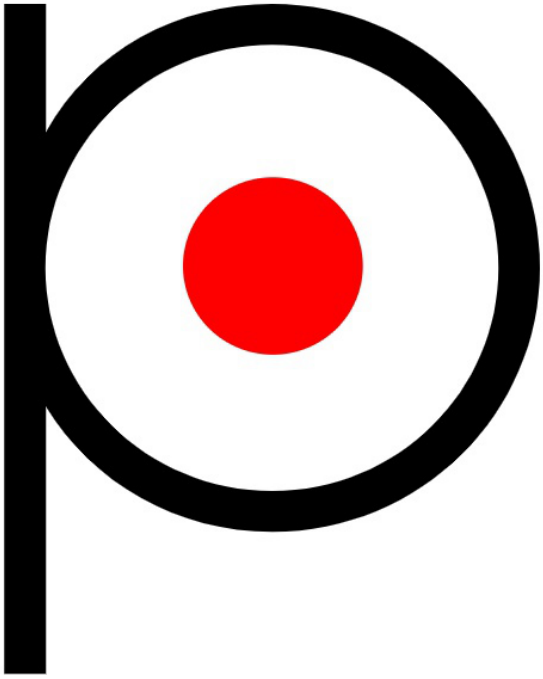


Рисунок 1.1 – Логотип Punto Switcher’a

Punto Switcher – широко розповсюджений інструмент для автоматичної зміни мови клавіатури. Він був розроблений для некомерційного використання Сергієм Москальовим. На даний момент програма належить до російської компанії Яндекс.

Працюючи у фоновому режимі, Punto Switcher робить статичний аналіз послідовності введених символів. У разі, якщо поєднання літер є нетопічним для поточної мови вводу, дане програмне забезпечення

перемикає мову розкладки клавіатури, видаляє неправильно написане слово за допомогою емуляції клавіши Backspace, і замінює слово на вірне.

Останній реліз даного програмного забезпечення був доданий 19 березня 2019 року. На даний момент часу це програмне забезпечення не підтримуєтсья.

Punto Switcher є безкоштовним у використанні. У інтерфейсі можна обрати одну з двох мов: англійську, або російську [7].

– Caramba Switcher. Логотип Caramba Switcher зображений на рисунку 1.2.

Рисунок 1.2 – Логотип Caramba Switcher’a

Caramba Switcher була розроблена Сергієм Москальовим.

Дана програма, як і Punto Switcher працює у фоновому режимі, та робить статичний аналіз послідовності введених символів, і у випадку їх нетипічності для поточної мови вводу - видаляє останнє слово шляхом емуляції натискання

клавіші Backspace, та за допомогою емуляції натискання клавіш змінює його на правильне.

Caramba Switcher підтримує три мови: російську, англійську та німецьку.

Дане програмне забезпечення для операційної системи Macintosh OS є платним [8].

– RuSwitcher.

RuSwitcher була розроблена Олексієм Проскуряковим.

Це програмне забезпечення автоматично визначає відповідність набраних символів до поточної мови розкладки клавіатури, та у випадку невідповідності змінює її на правильну, видаляє неправильно набране слово, та змінює його на правильне відповідно до поточної мови вводу системи.

RuSwitcher підтримує дві мови: англійську та російську.

Ця програма є безкоштовною для використання на Macintosh OS [9].

Ми ознайомились з відомими програмними продуктами, які частково чи повністю реалізують функціонал, відповідний до даного дипломного проекту. Скористаємось таблицею 1.1 для порівняння аналогів з програмним забезпеченням даного дипломного проекту.

Таблиця 1.1 - Порівняння дипломного проекту з аналогами

Функціонал	Програмне забезпечення для моніторингу натискання клавіш з	Punto Switcher	Caramba Switcher	RuSwitcher	Пояснення

Продовження таблиці 1.1.

	автоматичним перемиканням мови та корекцією				
Моніторинг та аналіз введеного слова користувачем відповідно до поточної мови розкладки клавіатури.	+	+	+	+	Програма відстежує натиснуті клавіші та аналізує, чи правильно введене слово відповідно до поточної мови.
Автоматична зміна мови у випадку невідповідності введеного слово до поточної мови вводу.	+	+	+	+	Програма автоматично змінює мову вводу операційної системи, якщо було набрано слово на

Продовження таблиці 1.1.

					неправильній розкладці.
Підтримка Macintosh OS.	+	+	+	+	Програма працює на основі операційної системи Macintosh OS.
Підтримка української мови.	+	-	-	-	Програма здатна працювати з українською мовою.
Наявність безкоштовної версії	+	+	-	+	Програма є безкоштовною для використання
Підтримка на даний момент часу	+	-	+	-	Програмне забезпечення не отримує новий оновлень та релізів.

1.4 Аналіз вимог до програмного забезпечення

Головними функціями програмного забезпечення є аналіз введеного слова, перевірка його на правильність відповідно до поточної мови розкладки клавіатури, перемикання мови на правильну, у випадку неправильно написаного слова, стирання неправильно написаного слова та емуляція того самого слова на правильній розкладці клавіатури. Загальна модель вимог зображена на рисунку 1.3.

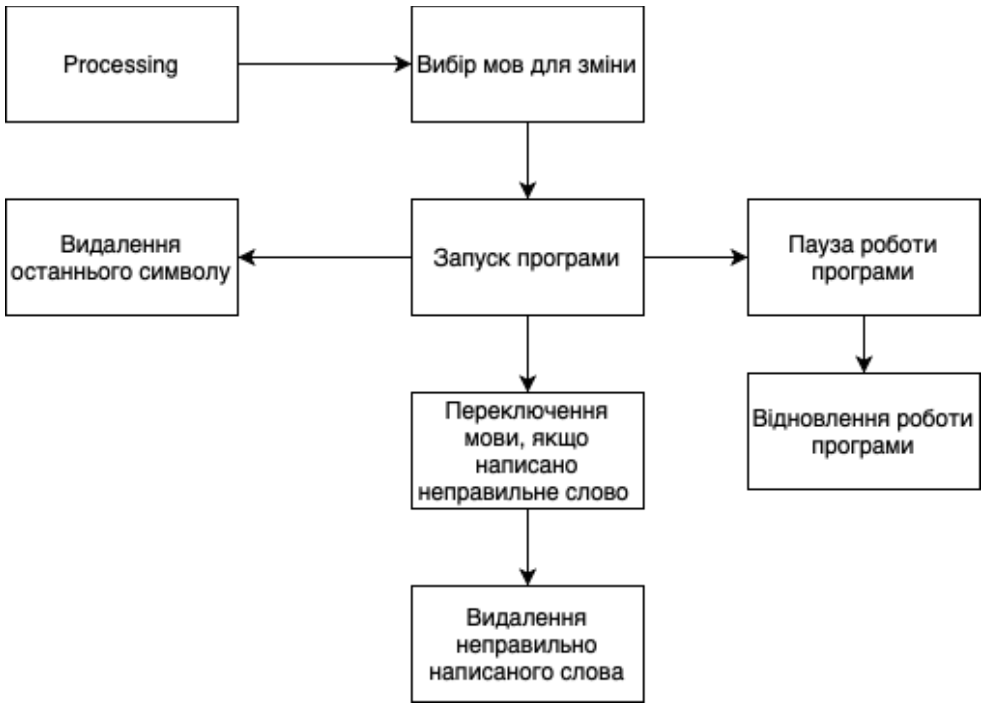


Рисунок 1.3 - Загальна модель вимог.

Стурктурну схему варіантів використання можна побачити у графічному матеріалі 3.

В таблицях 1.2 - 1.7 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 - Варіант використання UC-1

Use case name	Автоматичне переключення мови вводу
Use case ID	UC-01
Goals	Автоматично замінити мову розкладки клавіатури, у випадку, якщо слово неправильно написано відповідно до поточної мови вводу.
Actors	Користувач
Trigger	Користувач набирає текст
Pre-conditions	-
Flow of Events	Користувач вводить текст натискаючи послідовність літер на клавіатурі. Програмне забезпечення аналізує кожне введене слово, і у випадку, коли цього слова немає в словнику поточної мови розкладки клавіатури перемикає мову вводу на потрібно.
Extension	У випадку, якщо користувач натискає хот-кеї (наприклад Shift, Escape) або символи, які не є буквами (наприклад «+», «=») програмне забезпечення ігнорує їх.
Post-Condition	Перемикання мови розкладки клавіатури на необхідну.

Таблиця 1.3 - Варіант використання UC-2

Use case name	Об’єднання останніх написаних символів у одну змінну - слово.
Use case ID	UC-02
Goals	Поєднати надруковані символи у слово.
Actors	Користувач
Trigger	Натискання клавіші пробіл.
Pre-conditions	-
Flow of Events	Користувач натискає на пробіл, і програма автоматично поєднує останні надруковані букви у слово.
Extension	-
Post-Condition	Отримання надрукованого слова у змінну.

Таблиця 1.4 - Варіант використання UC-3

Use case name	Видалення останнього надрукованого символу.
Use case ID	UC-03
Goals	Видалити останній надрукований символ.
Actors	Користувач.
Trigger	Натискання клавіші Backspace.
Pre-conditions	-
Flow of Events	Користувач натискає на клавішу Backspace, програма видаляє стільки останніх надрукованих символів, скільки разів користувач натисне клавішу Backspace.
Extension	-
Post-Condition	Видалення останнього надрукованого символу.

Таблиця 1.5 - Варіант використання UC-4

Use case name	Пауза програми
Use case ID	UC-04
Goals	Зупинити виконання програми, з можливістю подальшого продовження роботи.
Actors	Користувач
Trigger	Натискання на відповідну кнопку у Tray-меню.
Pre-conditions	-
Flow of Events	При натисканні на відповідну кнопку програма зупиняє свою роботу, і не аналізує введені слова.
Extension	-
Post-Condition	Паузи роботи программи, її ігнорування подій клавіатури до подальшого відновлення роботи.

Таблиця 1.6 - Варіант використання UC-5

Use case name	Відновлення роботи програми після паузи.
Use case ID	UC-05
Goals	Відновити роботу програми після паузи.
Actors	Користувач
Trigger	Натискання на відповідну кнопку у Tray-меню.
Pre-conditions	-

Продовження таблиці 1.6.

Flow of Events	При натисканні на відповідну кнопку програма відновлює свою роботу.
Extension	-
Post-Condition	Відновлення роботи програми після натискання на паузу.

Таблиця 1.7 - Варіант використання UC-6

Use case name	Вибір мов для зміни.
Use case ID	UC-06
Goals	Обрати мови, між якими буде працювати програмне забезпечення.
Actors	Користувач
Trigger	Вибір мов у відповідному Trey вікні.
Pre-conditions	-
Flow of Events	Програмне забезпечення запам'ятовує обрані мови для подальшого аналізу слов з цими мовами.
Extension	-
Post-Condition	Отримання мов для аналізу.

1.4.1 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. В таблицях 1.8 - 1.14 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 1.4.

Таблиця 1.8 - Функціональна вимога FR-1

Назва	Реалізація глобального слухача клавіатури
Номер	FR-1
Опис	Програма відстежує всі дії користувача з клавіатурою по території всієї операційної системи, і записує їх.

Таблиця 1.9 - Функціональна вимога FR-2

Назва	Запис кожного введеного слова
Номер	FR-2
Опис	Поєднання натиснутих клавіш клавіатури у окремі слова.

Таблиця 1.10 - Функціональна вимога FR-3

Назва	Перевірка слова у словнику за допомогою API
Номер	FR-3
Опис	Програмне забезпечення перевіряє, чи наявне останнє введене слово у словнику відповідної до поточної розкладки мови.

Таблиця 1.11 - Функціональна вимога FR-4

Назва	Перевірка правильності введеного слово відповідно до поточної мови розкладки клавіатури
Номер	FR-4
Опис	Програма перевіряє, чи правильна обрана мова вводу відповідно до останнього введеного користувачем слова.

Таблиця 1.12 - Функціональна вимога FR-5

Назва	Перемикання мови розкладки клавіатури.
Номер	FR-5
Опис	Програма перемикає поточну мову розкладки клавіатури на правильну у випадку, якщо на даний момент час вона є неправильною для останнього введеного слова.

Таблиця 1.13 - Функціональна вимога FR-6

Назва	Видалення останнього слова, написаного на неправильній мові розкладки клавіатури
Номер	FR-6
Опис	У випадку, коли останнє слово не було знайдено у словнику поточної мови вводу, програма видаляє останнє написане слово

Таблиця 1.14 - Функціональна вимога FR-7

Назва	Автоматичне введення поточного слова на правильній розкладці клавіатури.
Номер	FR-7
Опис	У випадку, коли останнє неправильно написане слово було видалено, програмне забезпечення автоматично набирає дане слово на правильній розкладці клавіатури.

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7
UC-01	+	+	+	+	+		
UC-02	+	+					
UC-03	+	+				+	
UC-04	+						
UC-05	+						
UC-06							+

Рисунок 1.4 - Матриця трасування вимог

1.4.2 Розроблення нефункціональних вимог

Під час розробки програмного забезпечення для моніторингу натиснутих клавіш клавіатури та автоматичного перемикання мови розкладки клавіатури, необхідно буде імплементувати наступні нефункціональні вимоги:

– система повинна бути захищенною від нестандартних дій неавторизованих користувачів;

– система повинна забезпечувавати ефективне та швидке вирішення її задачі у режимі реального часу;

– програмне забезпечення повинно працювати у фоновому режимі, та не завдавати користувачам зайвих незручностей;

– програмне забезпечення повинно мати зрозумілий для користувача інтерфейс;

– програмне забезпечення повинно бути спроможним працювати з всіма мовами вводу, які підтримуються Macintosh OS, тобто бути універсальним.

1.5 Постановка задачі

Основною задачею при написанні даного дипломного проекту є розробка програмного забезпечення на мові програмування Java, що буде виконувати моніторинг натискань клавіш на клавіатурі персонального комп’ютера, автоматично перемикати мову поточного вводу і замінювати неправильно написане слово на те саме слово, але на правильній розкладці клавіатури. Програмне забезпечення повинне бути простим і зрозумілим в реалізації для подальшої модифікації.

Програмне забезпечення має містити декілька класів, кожен з яких має містити певний набір методів, за допомогою яких буде відбуватися моніторинг натиснутих клавіш, автоматичне перемикання мови, стирання неправильно написаного слова та емуляція того самого слова на правильній мові вводу

клавіатури. Кожен клас має виконувати певну функцію, і передавати потрібні дані у наступний клас.

Основною ідеєю для розробки даного програмного забезпечення є створення інструменту для усіх користувачів Macintosh OS, яке значно полегшить роботу з друком тексту.

Висновки до розділу

У першому розділі було висвітлено загальні терміни, які будуть використовуватися для написання дипломного проекту (бібліотеки, API, класи, методи, сторонні мови сценаріїв).

Було описано предметну область, яку буде покривати програмне забезпечення.

Також, було описано шість варіантів використання, сім функціональних вимог та чотири нефункціональні вимоги. На основі цих даних було створено матрицю трасування.

Кінцевим етапом першого розділу був опис постановки задачі, де було висвітлено основну ідею та головну мету проекту.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для більш наочного опису бізнес-процесів проекту використаємо BPMN-модель. Вона зображена на рисунку 2.1.

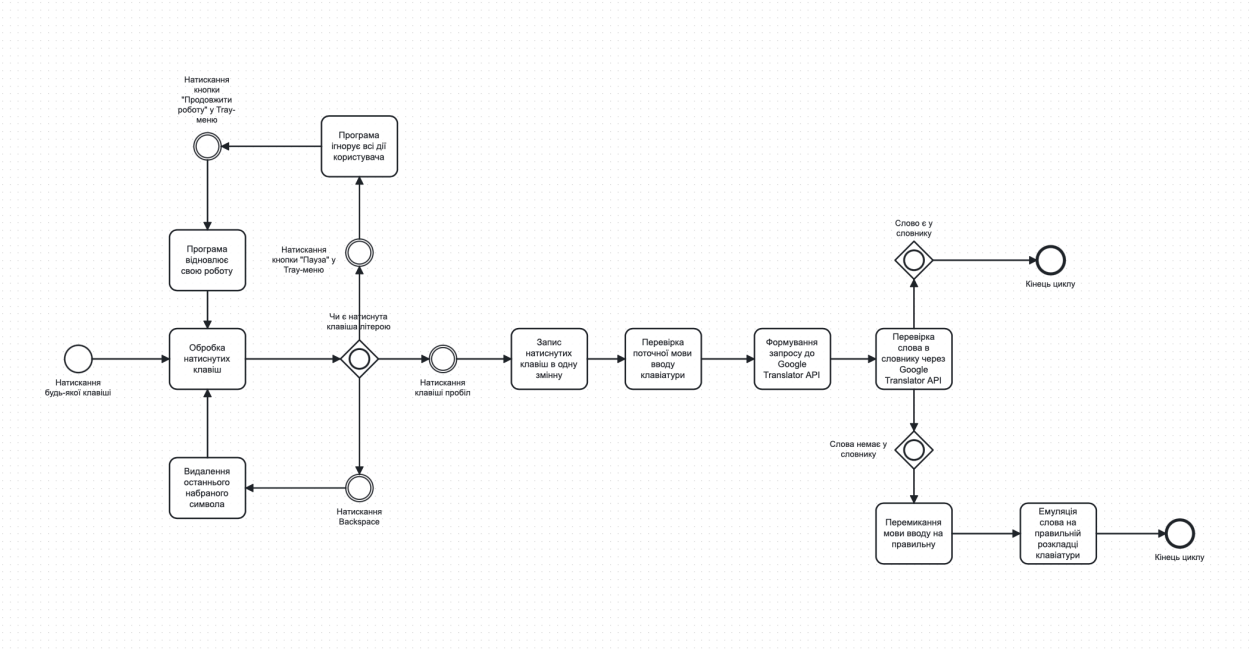


Рисунок 2.1 - BPMN модель програмного забезпечення.

Загальну модель використання програмного забезпечення для моніторингу натиснутих клавіш та автоматичного перемикання мови можна представити у наступній послідовності:

– користувач натискає будь-яку клавішу клавіатури;

– програма аналізує чи є ця клавіша літерою;

– якщо була натиснута клавіша backspace, програма видаляє останній символ;

– користувач натискає клавішу пробіл;

– програма записує натиснуті клавіші в одне слово, перевіряє поточну

мову вводу клавіатури та за допомогою api google translator перевіряє: чи є дане слово у відповідному словнику;

– якщо слова немає у відповідному словнику, програма перемикає мову на правильну;

– програмне забезпечення стирає неправильно написане слово;

– програма за допомогою емуляції вводить неправильно написане слово на правильній мові розкладки клавіатури;

– якщо натиснута кнопка «пауза» у tray-меню, програма ігнорує всі дії користувача з клавіатурою;

– якщо натиснута кнопка «продовжити роботу», програма повністю відновлює свою роботу.

2.2 Архітектура програмного забезпечення

Для розробки програмного забезпечення даного дипломного проекту використовуватиметься мова програмування Java стандарту 2023 року. Програмне забезпечення буде містити сім класів, кожен з яких буде мати певний набір методів.

На рисунку 2.2 зображена діаграма класів даного програмного забезпечення. Стуктурну схему класів програмного забезпечення можна побачити у графічному матеріалі 2.

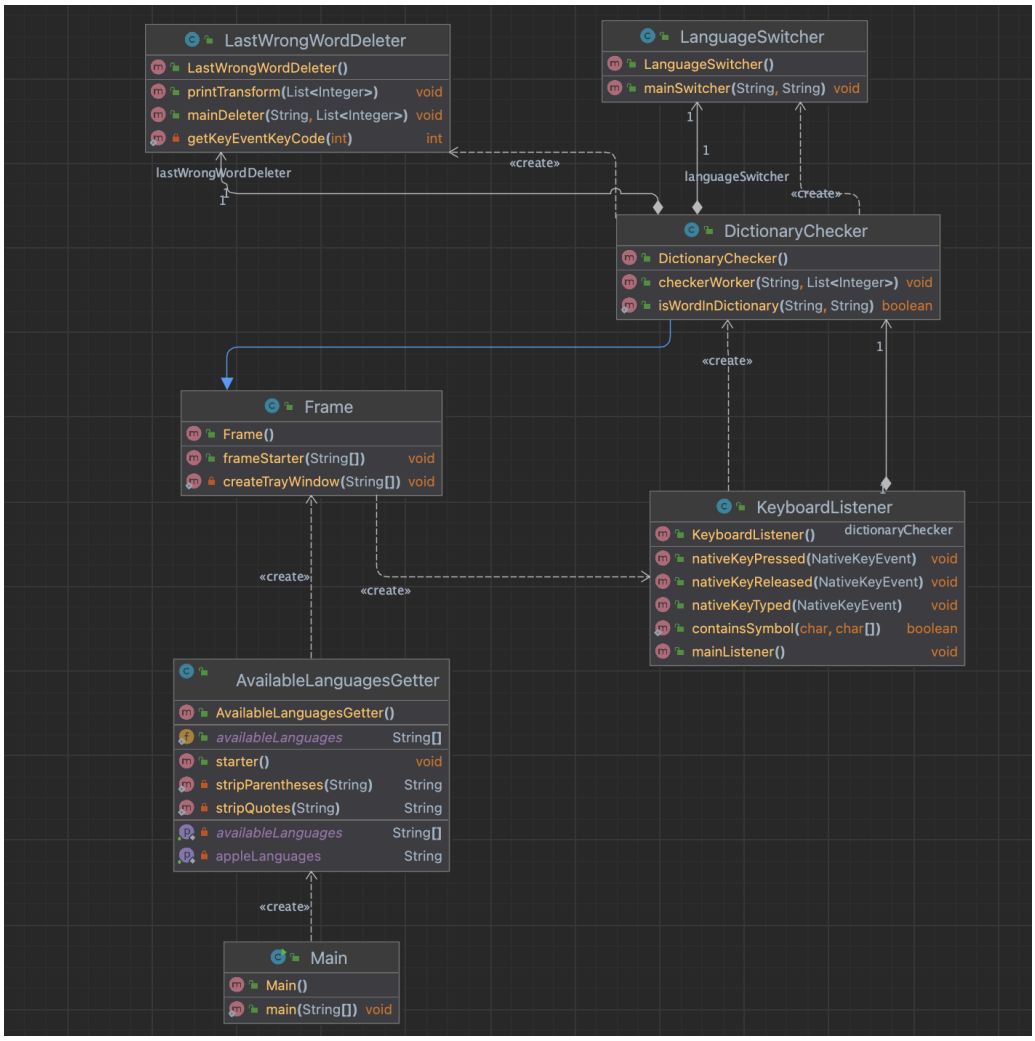


Рисунок 2.2 - Діаграма класів програмного забезпечення

В таблиці 2.1 наведені класи, які буде реалізовано, та основа інформація про використання цих класів.

Таблиця 2.1 - Класи програмного забезпечення та їх короткий опис

Назва класу	Опис класу
AvailableLanguagesGetter	Аналіз мов, які може обрати користувач у даний момент часу (мови, які наразі доступні у швидкому доступі.
KeyboardListener	Реалізація глобального слухача клавіатури по всій території операційної

Продовження таблиці 2.1.

	системи. Поєднання надрукованих літер у слова і запис їх у змінну.
DictionaryChecker	Перевірка отриманого слова від класу KeyboardListener та перевірка його наявності у словнику відповідної мови за допомогою API Google Translator
LanguageSwitcher	Перемикання мови розкладки клавіатури на правильну, якщо останнє написане слово не знайдено у словнику відповідної мови.
LastWrongWordDeleter	Видалення останнього неправильно набраного слова, друк за допомогою емуляції того самого слова на правильній розкладці клавіатури.
Frame	Інтерфейс програмного забезпечення для взаємодії користувача з програмним забезпеченням за допомогою Tray-вікна.
Main	Стартовий клас, викликає клас AvailableLanguagesGetter.

Основну інформацію про методи, які будуть використовуватися у кожному класі можна побачити у таблицях 2.2 - 2.6.

Таблиця 2.2 - Опис методів класу AvailableLanguagesGetter.

Назва методу	Опис
getAvailableLanguages	Отримання мов розкладки клавіатури, які є у швидкому доступі користувача.
getAppleLanguages	Отримання списку мов у форматі Apple.

Продовження таблиці 2.2.

starter	Збирання доступних мов у масив, виклик класу Frame.

Таблиця 2.3 - Опис методів класу KeyboardListener.

Назва методу	Опис
mainListener	Викликає екземпляр класу GlobalScreen.
nativeKeyPressed	Викликається при натискання на клавішу клавіатури.
nativeKeyReleased	Викликається при відпусканні клавіші клавіатури.
nativeKeyTyped	Викликається при наборі символу на клавіатури.

Таблиця 2.4 - Опис методів класу DictionaryChecker.

Назва методу	Опис
isWordInDictionary	Робить запит до API Google Translator з словом, визначеним у класі KeyboardListener.
checkerWorker	Перевіряє у відповіді API, чи наявне поточне слово у словнику відповідної мови клавіатури. Визначає поточно встановлено мову вводу.

Таблиця 2.5 - Опис методів класу LanguageSwitcher

Назва методу	Опис
mainSwitcher	Перемикає поточну мову розкладки клавіатури, якщо останнє введене слово відсутнє у відповідному словнику.

Таблиця 2.6 - Опис методів класу LastWrongWordDeleter

Назва методу	Опис
mainDeleter	Видаляє останнє неправильно написане слово.
getKeyEventCode	Трансформує кей-коди отримані від бібліотеки Java Native Hook у кей-коди для класу Robot.
printTransform	Друкує поточне слово на правильній мові розкладки клавіатури за допомогою емуляції натискання клавіш.

Таблиця 2.7 - Опис методів класу Frame

Назва методу	Опис
createTrayWindow	Створює Tray-вікно.
frameStarter	Виклакає метод createTrayWindow.

2.3 Конструювання програмного забезпечення

Розглянемо більш детально функціонал кожного окремого класу, та інструменти, які будуть в них використовуватися.

Клас Main є стартовим у даному програмному забезпеченні. Він викликає клас AvailableLanguagesGetter.

Клас AvailableLanguagesGetter аналізує мови, які може обрати користувач на даний момент часу. Тобто мови, які знаходяться у швидкому доступі. Це реалізовано за допомогою обходу всіх доступних мов у форматі Apple, та виявлення тих, які обрані користувачем. Фрагмент коду з реалізацією методу, який виконує пошук усіх доступних мов у форматі Apple можна побачити на рисунку 2.3.

```
private static String getAppleLanguages() {
    StringBuilder appleLanguages = new StringBuilder();
    try {
        Process process = Runtime.getRuntime().exec( command: "defaults read -g AppleLanguages");
        java.io.BufferedReader reader = new java.io.BufferedReader(new java.io.InputStreamReader(process.getInputStream()));

        String line;
        while ((line = reader.readLine()) != null) {
            appleLanguages.append(line);
        }
        reader.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return appleLanguages.toString();
}
```

Рисунок 2.3 - Лістинг методу getAppleLanguages.

Клас KeyboardListener реалізує глобального слухача подій клавіатури по всій операційній системі. Реалізувати слухача можна за допомогою стандартного класу Scanner, який вже імпліментований у базову бібліотеку Java. Але він не надає можливості відстежувати натиснуті символи по всій операційній системі і відстежувати натиснуті Hot-Key'ї: Escape, CapsLock, Shift та інші. Тому було застосовано бібліотеку Java Native Hook (JNH), яка дає можливість відстежувати усі дії з інструментами вводу персонального комп'ютера: клавіатури та миші. Для запобігання реєстрацій дій миші (її пересування по екрану згідно координатам - пікселям, натискання ЛКМ, ПКМ або СКМ), було застосовано метод Logger бібліотеки Java Util. З його допомогою можна повністю ігнорувати усі події миші. З бібліотеки Java Native Hook було використано три методи: nativeKeyPressed, nativeKeyReleased та nativeKeyTyped. Їх лістинг зображений на рисунку 2.4.

```java
public void nativeKeyPressed(NativeKeyEvent e) {

    if (e.getKeyCode() == NativeKeyEvent.VC_BACKSPACE) {
        if (currentWord.length() > 0) {
            currentWord = currentWord.substring(0, currentWord.length() - 1);
        }
    }
    //System.out.println("Key Pressed: " + NativeKeyEvent.getKeyText(e.getKeyCode()));
}

public void nativeKeyReleased(NativeKeyEvent e) {
    //System.out.println("Key Released: " + NativeKeyEvent.getKeyText(e.getKeyCode()));
}

public void nativeKeyTyped(NativeKeyEvent e) {
    char actualSymbolTyped = e.getKeyChar();

    if (actualSymbolTyped == ' ') {
        System.out.println("Actual word typed -> " + currentWord);
        dictionaryChecker.checkerWorker(currentWord);
        currentWord = "";
    } else if (Character.isLetter(actualSymbolTyped)) {
        currentWord += actualSymbolTyped;
    }
}
```

Рисунок 2.4 - Лістинг методів nativeKeyPressed, nativeKeyReleased, nativeKeyTyped.

У методі nativeKeyTyped програма відстежує всі натиснуті клавіші, і якщо вони є буквами, записує їх у єдине слово. Тригером для відокремлення слів слугує натискання клавіші Пробіл. При її натискання програма поєднує набрані літери у одне слово - змінну currentWord. Це слово передається наступним класам для подальшого аналізу та обробки, і після подальших дій клас KeyboardListener очищує змінну currentWord, і знову збирає набрані букви до натискання на клавішу пробіл. Метод nativeKeyPressed забезпечує стирання останньої натиснутої букви при натискання на клавішу Backspace. Реалізацію даного рішення також можна побачити на рисунку 2.4. Це реалізовано для того, щоб користувач мав змогу самостійно змінити останнє слово, якщо він допустив помилку у його написанні.

Клас DictionaryChecker реалізує перевірку наданого слова за допомогою API Google Translator. Метод isWordInDictionary створює запит до Google

Translator API за допомогою стандартної бібліотеки Java - .net. Його реалізація зображена на рисунку 2.5.

```java
public static boolean isWordInDictionary(String word, String apiKey) throws IOException {
    String apiUrl = "https://translation.googleapis.com/language/translate/v2/detect?key=" + apiKey + "&q=";
    String urlEncodedWord = URLEncoder.encode(word, StandardCharsets.UTF_8);

    URL url = new URL( spec: apiUrl + urlEncodedWord);
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    connection.setRequestMethod("GET");

    int responseCode = connection.getResponseCode();

    if (responseCode == HttpURLConnection.HTTP_OK) {
        InputStream inputStream = connection.getInputStream();
        Scanner scanner = new Scanner(inputStream, StandardCharsets.UTF_8);
        String response = scanner.useDelimiter( pattern: "\\A").next();
        scanner.close();

        return response.contains("\"language\": \"" + currentLanguage + "\"");

    } else {
        throw new IOException("Не удалось выполнить запрос к API, код ошибки: " + responseCode);
    }
}
```

Рисунок 2.5 - Реалізація методу isWordInDictionary

Мова словнику, у якому буде відбуватися пошук поточного слова виставляється за допомогою змінної currentLanguage, яка визначає поточно обрану мову вводу клавіатури. Таке рішення є дуже гнучким, адже нам не обов'язково створювати окремий метод для кожної мови. Можна вводити слово на будь-якій мові, програма автоматично визначить, на якій саме наразі друкує користувач, і буде первіряти поточне слово на мові, на якій користувач надрукував поточне слово. Поточно обрана мова визначається за допомогою методу .getInstance() класу InputContext. Даний метод звертається безпосередньо до пристрою вводу. У нашому випадку - до клавіатури та її поточної мови. Слід зазначити, що даний метод не грунтується на надрукованих символах: він відрізняє розкладки з однаковими літерами, що свідчить про його пряме звернення до мови вводу. Метод getInstance() повертає нам код отриманої мови розкладки клавіатури, а метод String.valueOf() робить з коду отриманої мови вводу знайомі для нас «en», «uk»

та інші. Метод checkerWorker отримує змінну currentWord з класу KeyboardListener і відправляє її до API Google Translator. Його реалізацію можна побачити на рисунку 2.6.

```
    public void checkerWorker(String currentWord) {
        InputContext inputContext = InputContext.getInstance();

        if ("_US_UserDefined_252".equals(String.valueOf(inputContext.getLocale()))) {
            currentLanguage = "en";
        } else {
            currentLanguage = String.valueOf(inputContext.getLocale());
        }
        String apiKey = "AIzaSyDISFdioPgs9MR2DXVuQT9sscjpTYYVjr8";

        try {

            if (isWordInDictionary(currentWord, apiKey)) {
                System.out.println("Слово '" + currentWord + "' найдено в словаре");
            } else {
                System.out.println("Слово '" + currentWord + "' не найдено в словаре");

                if (Objects.equals(currentLanguage, b: "en")) {
                    correctlanguage = "uk";
                } else {
                    correctlanguage = "en";
                }
            }
            System.out.println("Actual language chosen ---> " + currentLanguage);
            System.out.println("----------------------------------------");

        } catch (IOException e) {
            System.err.println("Ошибка при выполнении запроса к API: " + e.getMessage());
        }
    }
}
```

Рисунок 2.6 - Реалізація методу checkerWorker

Google API Translator - сервіс, який перевіряє наявність поточного слова у відповіді API. Інтерфейс Google Translator API зображений на рисунку 2.7.

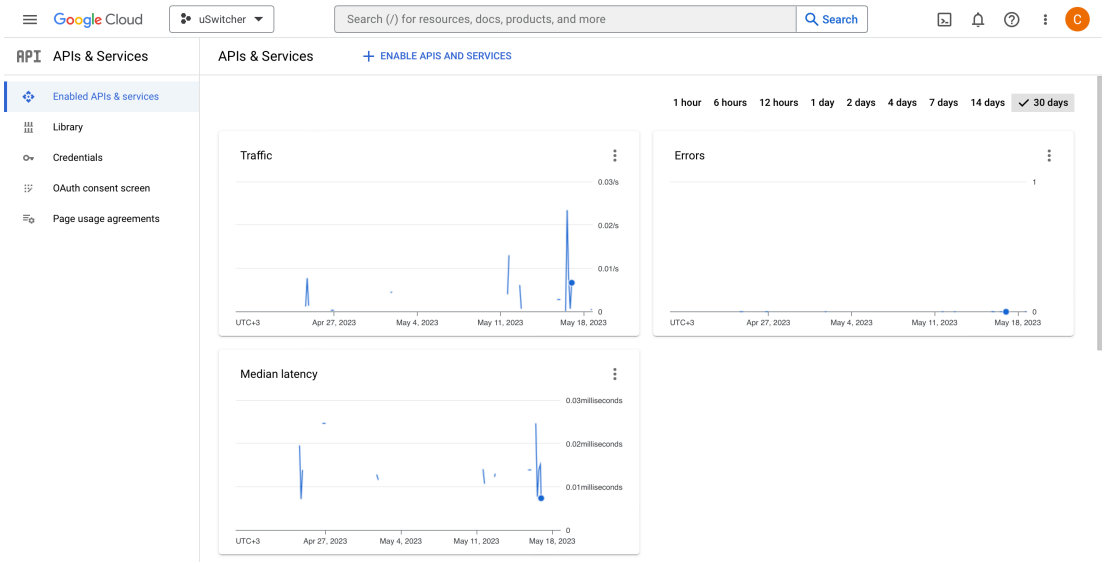


Рисунок 2.7 - Інтерфейс Google Translator API

У випадку, якщо слово є в словнику відповідної мови, програма нічого не змінює, і готова повертатися до виконання класу KeyboardListener. У випадку, якщо введеного слова немає в словнику відповідної мови, програмне забезпечення запускає метод mainSwitcher класу LanguageSwitcher. Його реалізацію можна побачити на рисунку 2.8.

```
try {
    String script = "tell application \"System Events\"\n" +
            "    key code 49 using control down\n" +
            "    key code 49\n" +
            "end tell";

    String[] cmd = {"osascript", "-e", script};
    Runtime.getRuntime().exec(cmd);

    System.out.println("Язык ввода был изменен");
} catch (    IOException e) {
    e.printStackTrace();
}

Thread.sleep( millis: 50);
InputContext inputContext = InputContext.getInstance();
System.out.println(String.valueOf(inputContext.getLocale()));
```

Рисунок 2.8 - Реалізація методу mainSwitcher

Даний метод використовує мову сценаріїв Apple Script. Apple Script - спеціально розроблена Apple мова сценаріїв для операційних систем Macintosh OS та iOS. Цей інструмент надає можливість створити певні скрипти, які будуть виконуватись у точності так, як вони були описані. У випадку даного дипломного проекту, за допомогою команди osascript виконується код Apple Script, який перемикає мову вводу на наступну до тих пір, доки вона не буде співпадати з необхідною.

Клас DictionaryChecker у разі виявлення слова, написаного на неправильній мові, також викликає клас LastWrongWordDeleter. Цей клас виконує дві головні функції: видалення неправильно написаного слова, та друк слова на правильній мові вводу за допомогою емуляції натискання клавіш клавіатури. Метод mainDeleter можна побачити на рисунку 2.9.

```
public void mainDeleter(String currentWord, List<Integer> currentKeyCodes) throws AWTException {

    int numberOfBackspaces = currentWord.length() + 1;

    try {
        StringBuilder scriptBuilder = new StringBuilder();
        scriptBuilder.append("tell application \"System Events\"\n");
        for (int i = 0; i < numberOfBackspaces; i++) {
            scriptBuilder.append("    key code 51\n");
        }
        scriptBuilder.append("end tell");

        String[] cmd = {"osascript", "-e", scriptBuilder.toString()};
        Runtime.getRuntime().exec(cmd);
    } catch (IOException e) {
        e.printStackTrace();
    }

    printTransform(currentKeyCodes);
}
```

Рисунок 2.9 - Реалізація методу mainDeleter

У класі LastWrongWordDeleter метод getKeyEventKeyCode трансформує кей-коди, отримані з бібліотеки Java Native Hook у кей-коди для класу Robot. А метод printTransform виконує емуляцію друку поточного слова на

правильній розкладці клавіатури. Їх реалізацію можна побачити на рисунку 2.10.

```java
public void printTransform(List<Integer> currentKeyCodes) {

    try {
        Robot robot = new Robot();
        robot.delay( ms: 215);
        for (int keyCode : currentKeyCodes) {
            int vkCode = getKeyEventKeyCode(keyCode);
            robot.keyPress(vkCode);
            robot.keyRelease(vkCode);
        }

        int spaceVkCode = KeyEvent.VK_SPACE;
        robot.keyPress(spaceVkCode);
        robot.keyRelease(spaceVkCode);
    } catch (AWTException e) {
        e.printStackTrace();
    }
}

1 usage
private static int getKeyEventKeyCode(int keyCode) {
    int vkCode = -1;
    int[][] keycodeMapping = LastWrongWordDeleter.KEYCODES;

    for (int[] mapping : keycodeMapping) {
        if (mapping[1] == keyCode) {
            vkCode = mapping[0];
            break;
        }
    }
    return vkCode;
}
```

Рисунок 2.10 - Реалізація методів getKeyEventKeyCode та printTransform.

Клас Frame створює Tray-вікно, за допомогою якого користувач матиме змогу взаємодіяти з програмним забезпеченням. Tray-вікно надаватиме користувачу наступні дії:

- обрання мов для роботи;
- запуск або вихід з програми;

- пауза або відновлення роботи програми;
- відкриття вікна з інструкціями до використання;
- зв’язок с розробник програмного забезпечення.

Фрагмент реалізації класу Frame можна побачити на рисунку 2.11.

```
MenuItem button1Item = new MenuItem( label: "Обрати мови для зміни");
button1Item.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {

        JFrame frame = new JFrame( title: "Доступні мови");
        frame.setSize( width: 300,  height: 200);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setLocationRelativeTo(null);
        frame.setResizable(false);

        JPanel panel = new JPanel();
        panel.setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.anchor = GridBagConstraints.CENTER;
        gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);

        Font font = new Font( name: "Tahoma", Font.PLAIN,  size: 14);

        final int[] selectedCount = {0};

        for (String language : availableLanguages) {
            JCheckBox checkBox = new JCheckBox(language);
            checkBox.setFont(font);
            gbc.gridy++;
            panel.add(checkBox, gbc);

            checkBox.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    if (checkBox.isSelected()) {
                        if (selectedCount[0] == 0) {
                            availableLanguage1 = checkBox.getText();
                        } else if (selectedCount[0] == 1) {
```

Рисунок 2.11 - Фрагмент реалізації класу Frame

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці програмного забезпечення даного дипломного проекту наведено у таблиці 2.7.

Таблиця 2.8 - Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	IntelliJ IDEA	Головне середовище розробки програмного забезпечення дипломного проекту.
2	API Google Translator	API, що використовується для перевірки правильності написання слова. Програма виконує запит до API для отримання словника з поточної мови вводу. У випадку, якщо поточне слово наявне у відповіді API (словнику), програма робить висновок, що слово написане на правильній мови клавіатури. У випадку, якщо слова немає у словнику відповідної мови програма робить висновок, що слово написане на неправильній мові вводу.
3	Java Native Hook (JNH)	Бібліотека, що надає доступ програмного забезпеченню відстежувати усі дії клавіатури персонального комп’ютера по всій території операційної системи.
4	Java Util	Бібліотека, що надає можливість контролювати глобального слухача подій інструментів вводу (клавіатури та миші). У випадку даного дипломного проекту надає можливість ігнорувати усі події миші.
5	Apple Script	Мова сценаріїв, яка надає доступ до низькорівневих функцій Macintosh OS. А саме до зміни поточної мови розкладки

Продовження таблиці 2.8.

5		клавіатури, чого не може запропонувати стандартна бібліотека мови програмування Java.
6	Robot	Бібліотека, що надає можливість реалізувати емуляцію натискання клавіш клавіатури.
7	Swing	Бібліотека, що надає можливість створити User-Interface. У нашому випадку - Tray-меню.

2.4 Аналіз безпеки даних

Безпека програмного забезпечення для моніторингу натискання клавіш та автоматичного перемикання мови вводу буде досягнута за рахунок декількох пунктів:

– інкапсуляція методів та об’єктів. Завдяки цьому, змінні не будуть вразливими до зовнішнього чи нестандартного використання;

– безпека мови програмування Java. Байт-код Java виконує Java Virtual Machine (JVM), що дозволяє контролювати виконання програм та забезпечувати ізоляцію між різними процесами. Також java має вбудовану систему безпеки Java Security, яка включає в себе політики безпеки, керування дозволами та інші механізми, які допомогають контролювати доступ до ресурсів та обмежувати можливість виконання певних операцій.

Висновки до розділу

В другому розділі були визначені технічні деталі для реалізації програмного забезпечення для моніторингу натискання клавіш та автоматичного перемикання мови розкладки клавіатури.

Було висвітлено загальну архітектуру програмного забезпечення, наведено та розглянуто основні інструменти, які будуть використовуватися в програмі.

Також, були розглянуті основні компоненти коду, які складають програмне забезпечення продукту, класи та основні методи і функції, що забезпечуватимуть роботу програми.

В завершенні другого розділу було розглянуто фактори, що забезпечуватимуть безпеку даних в програмному забезпеченні.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості програмного забезпечення

Тестування є важливою частиною розробки програмного забезпечення. Воно відіграє вирішальну роль в забезпеченні якості та надійності програми, а також у виявленні та усуненні помилок та дефектів програмного забезпечення. Перед публікацією програмного забезпечення у відкритий доступ, необхідно перевірити його на наявність помилок, багів та дефектів. В цьому розділі ми застосуємо основні принципи та методики тестування, які допоможуть нам перевірити якість та надійність даного програмного забезпечення.

Проведемо загальний аналіз коду за допомогою сервіса “Embold”. Embold - онлайн застосунок, який перевіряє код програмного забезпечення за такими параметрами:

– аналіз та пошук можливих помилок в коді;
– пошук повторюваних місць коду;
– аналіз використання антипаттернів.

Протестуємо наше програмне забезпечення у цьому сервісі. Результат його виконання можна побачити на рисунку 3.1.

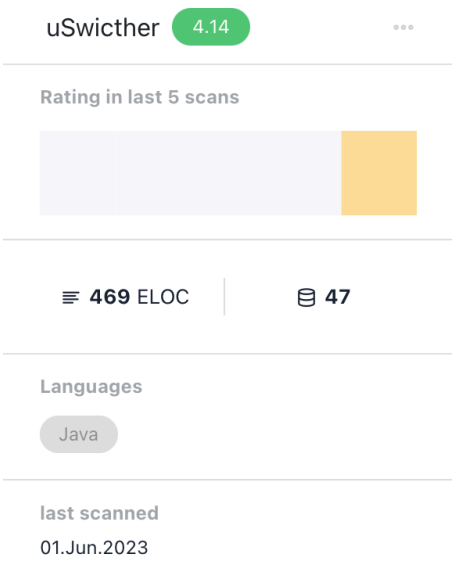


Рисунок 3.1 - Результат тестування програми у сервісі «Embald»

Як бачимо, програмне забезпечення даного дипломного проекту отримало оцінку 4.14 з 5 можливих балів, що свідчить про гарний стан коду.

3.2 Опис процесів тестування

В даній дипломній роботі було виконане мануальне тестування програмного забезпечення «Моніторинг натискання клавіш з автоматичним перемиканням мови клавіатури та корекцією». У таблицях 3.1 - 3.8 наведені описи тестів.

Таблиця 3.1 – Тест 1

Тест	Обрання мов для роботи
Модуль	Обрання мов для роботи
Номер тесту	1
Початковий стан системи	Користувач запустив програмне забезпечення
Вхідні данні	Доступні мови вводу клавіатури, які знаходяться у швидкому доступі операційної системи на даний момент часу.
Опис проведення тесту	При натисканні на кнопку «Обрати мови» у Tray-меню, з’являється список мов, які може обрати користувач. Доступні лише ті мови, які на даний момент часу встановлені у швидкому доступі (між якими перемикається користувач). Користувач обирає мови для зміни, та натискає на кнопку «Обрати».
Очікуваний результат	Список доступних мов складається лише з мов, які є у швидкому доступі. Вибір мов проходить успішно. Обрані мови записуються у відповідні змінні.

Продовження таблиці 3.1.

Фактичний результат	Список доступних мов складається лише з мов, які є у швидкому доступі. Вибір мов проходить успішно. Обрані мови записуються у відповідні змінні.

Таблиця 3.2 – Тест 2

Тест	Відображення помилки у випадку неправильного вибору мов вводу клавіатури для зміни.
Модуль	Обрання мов для роботи
Номер тесту	2
Початковий стан системи	Користувач запустив програмне забезпечення
Вхідні данні	Доступні мови вводу клавіатури, які знаходяться у швидкому доступі операційної системи на даний момент часу.
Опис проведення тесту	При натисканні на кнопку «Обрати мови» у Tray-меню, з’являється список мов, які може обрати користувач. Доступні лише ті мови, які на даний момент часу встановлені у швидкому доступі (між якими перемикається користувач). Користувач обирає лише одну мову для зміни, після цього натискає на кнопку «Обрати».
Очікуваний результат	Список доступних мов складається лише з мов, які є у швидкому доступі. Вибір мов не проходить успішно. Обрані мови не записуються у відповідні змінні.
Фактичний результат	Список доступних мов складається лише з мов, які є у швидкому доступі. Вибір мов не проходить успішно. Обрані мови не записуються у відповідні змінні.

Таблиця 3.3 – Тест 3

Тест	Визначення поточної мови вводу клавіатури.
Модуль	Визначення поточної мови вводу клавіатури.
Номер тесту	3
Початковий стан системи	Користувач запустив програмне забезпечення.
Вхідні данні	Доступні мови вводу клавіатури.
Опис проведення тесту	При роботі програмного забезпечення, програма аналізує, яка на даний момент часу встановлена мови вводу клавіатури.
Очікуваний результат	Програмне забезпечення правильно аналізує поточну мову вводу клавіатури.
Фактичний результат	Програмне забезпечення правильно аналізує поточну мову вводу клавіатури.

Таблиця 3.4 – Тест 1.4

Тест	Запис надрукованого слова у змінну
Модуль	Запис надрукованого слова у змінну
Номер тесту	4
Початковий стан системи	Користувач обрав мови для зміни і почав друкувати слово. Після друку слова натиснув клавішу «пробіл».
Вхідні данні	Масив надрукованих символів до натискання на клавішу «пробіл».
Опис проведення тесту	При натисканні на клавішу «пробіл» програмне забезпечення поєднує надруковані символи у слово, зберігає його у відповідну змінну та передає подальшим класам та методам для подальшої обробки та аналізу.
Очікуваний результат	Програмне забезпечення правильно записує у змінну останнє надруковане слово при натисканні на клавішу

Продовження таблиці 3.4.

	«пробіл». До складу символів слова не входять «хот-кеї» (Shift, Escape), символи («=», «+») та цифри.
Фактичний результат	Програмне забезпечення правильно записує у змінну останнє надруковане слово при натисканні на клавішу «пробіл». До складу символів слова не входять «хот-кеї» (Shift, Escape), символи («=», «+») та цифри.

Таблиця 3.5 – Тест 5

Тест	Перевірка поточного слова у словнику відповідної мови за допомогою Google Translator API.
Модуль	Перевірка поточного слова у словнику відповідної мови.
Номер тесту	5
Початковий стан системи	Користувач натиснув клавішу «пробіл», програмне забезпечення записує поточне слово у змінну.
Вхідні данні	Останнє введене слово.
Опис проведення тесту	При натисканні на клавішу «пробіл» програмне забезпечення відправляє запит у Google Translator API та повертає відповідь: чи є поточне слово у словнику відповідної мови чи ні.
Очікуваний результат	Програмне забезпечення правильно аналізує чи на правильній мові вводу надруковане поточне слово.
Фактичний результат	Програмне забезпечення правильно аналізує чи на правильній мові вводу надруковане поточне слово.

Таблиця 3.6 – Тест 6

Тест	Зміна мови розкладки на правильну у випадку, коли поточне слово надруковане на неправильній мові вводу клавіатури.

Продовження таблиці 3.6.

Модуль	Зміна мови розкладки на правильну.
Номер тесту	6
Початковий стан системи	Програмне забезпечення робить висновок, що поточне слово надруковане на неправильній мові вводу клавіатури.
Вхідні данні	Останнє введене слово, доступні мови вводу клавіатури.
Опис проведення тесту	Якщо програмне забезпечення розуміє, що поточне слово було надруковане на неправильній мові вводу клавіатури, воно повинне змінити поточну мову вводу клавіатури на правильну.
Очікуваний результат	Програмне забезпечення правильно змінює поточну мову вводу клавіатури на правильну у разі потреби.
Фактичний результат	Програмне забезпечення правильно змінює поточну мову вводу клавіатури на правильну у разі потреби.

Таблиця 3.7 – Тест 7

Тест	Видалення слова, надрукованого на неправильній мові вводу клавіатури.
Модуль	Видалення слова, надрукованого на неправильній мові вводу клавіатури.
Номер тесту	7
Початковий стан системи	Програмне забезпечення робить висновок, що поточне слово надруковане на неправильній мові вводу клавіатури.
Вхідні данні	Останнє введене слово.
Опис проведення тесту	Якщо програмне забезпечення розуміє, що поточне слово було надруковане на неправильній мові вводу клавіатури, воно видаляє його з того місця, де воно було введено.

Продовження таблиці 3.7.

Очікуваний результат	Програмне забезпечення правильно видаляє поточне слово, якщо воно було надруковано на неправильній мові вводу клавіатури.
Фактичний результат	Програмне забезпечення правильно видаляє поточне слово, якщо воно було надруковано на неправильній мові вводу клавіатури.

Таблиця 3.8 – Тест 8

Тест	Друк поточного слова на правильній мові вводу клавіатури.
Модуль	Друк поточного слова на правильній мові вводу клавіатури.
Номер тесту	8
Початковий стан системи	Програмне забезпечення видалило останнє написане слово, якщо воно було надруковане на неправильній мові вводу клавіатури.
Вхідні данні	Останнє введене слово.
Опис проведення тесту	Після видалення слова, яке було надруковане на неправильній мові вводу клавіатури, воно за допомогою емуляції друкує те саме слово, але на правильній для цього слова мові вводу клавіатури.
Очікуваний результат	Програмне забезпечення друкує правильно написане слово.
Фактичний результат	Програмне забезпечення друкує правильно написане слово.

3.3 Опис контрольного прикладу

Проведемо контрольний приклад тестування програмне забезпечення даного дипломного проекту. У цьому розділі будуть описані кроки тестування програмного забезпечення, наведені ілюстрації до них. Розділимо тестування на пункти, для більш лекого сприйняття загальної роботи застосунку.

– Запускаємо програмне забезпечення, бачимо нове Tray-вікно з інтерфейсом. (рисунок 3.2).

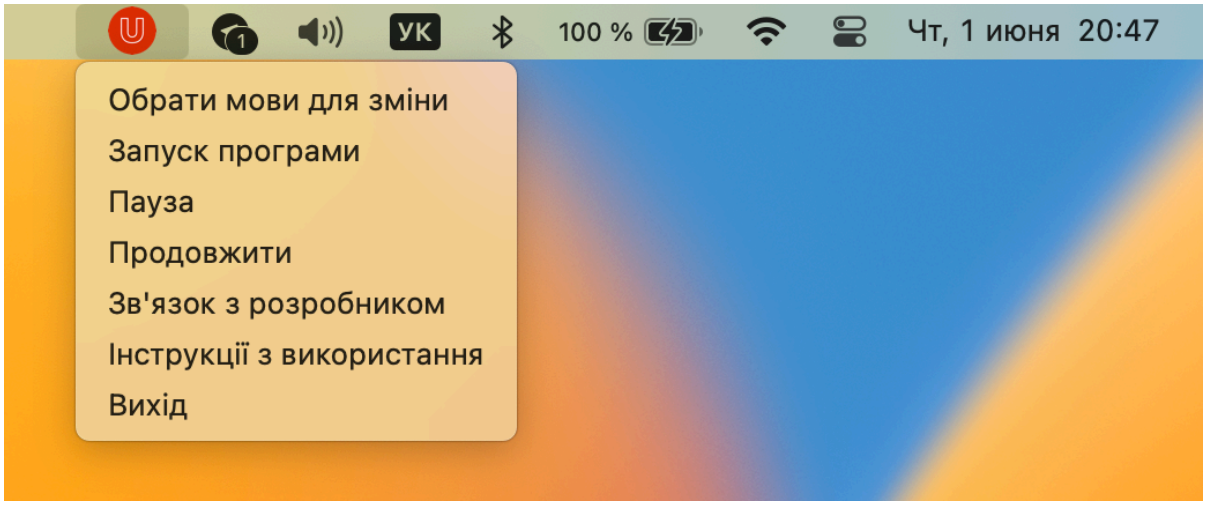


Рисунок 3.2 - Інтерфейс програмного забезпечення.

– Для початку, оберемо лише одну мову для зміни (рисунок 3.3).

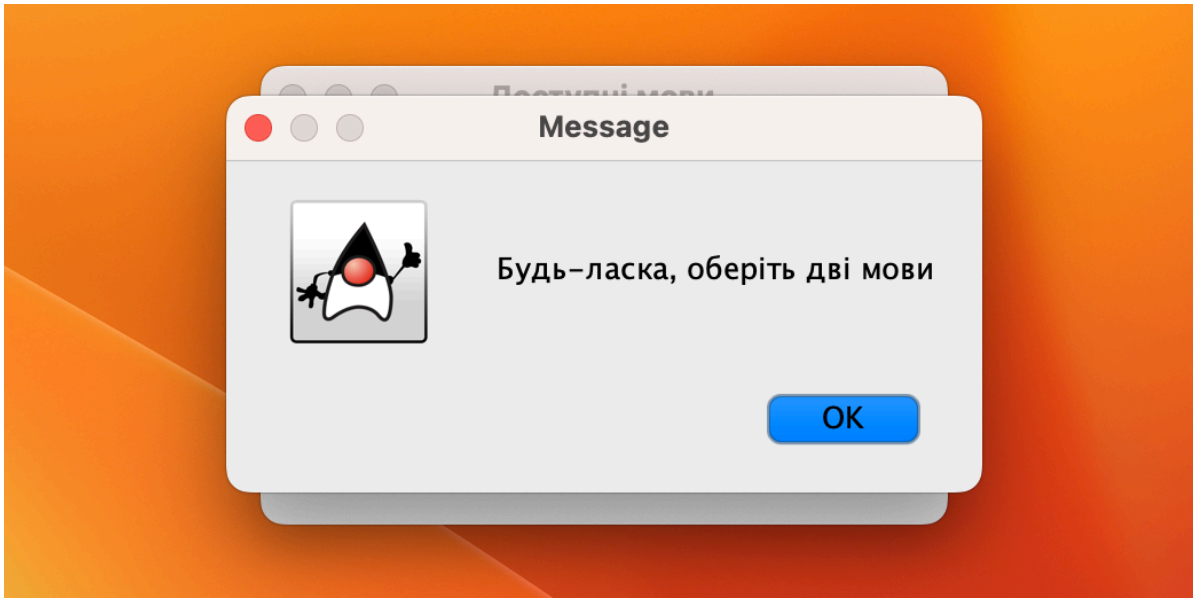


Рисунок 3.3 - Помилка при виборі лише одної мови для зміни.

Як бачимо на рисунку 3.3, програмне забезпечення показує повідомлення, яке прохає обрати дві мови для зміни.

– Оберемо дві мови для зміни (рисунок 3.4).

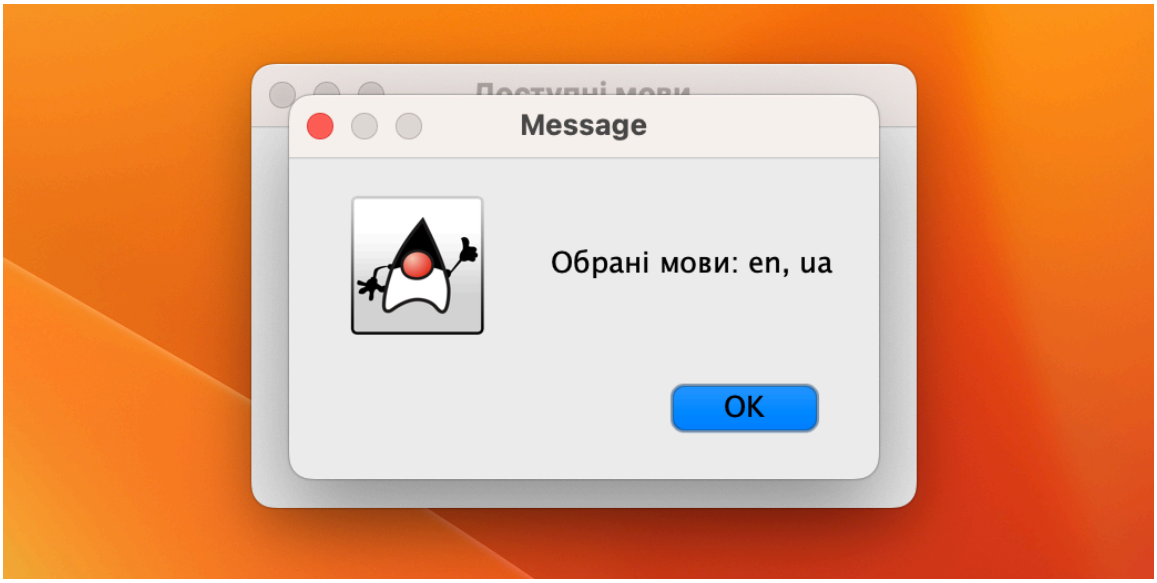


Рисунок 3.4 - Успішне обирання мов для зміни.

Оскільки дане програмне забезпечення процює у фоновому режимі, для перевірки тестів будемо використовувати логи у консолі середовища розробки. На рисунку 3.5 можемо побачити наступні результати тестів: програмне забезпечення коректно показує доступні для обрання мови, та при виборі двох з них записує їх у відповідні змінні.

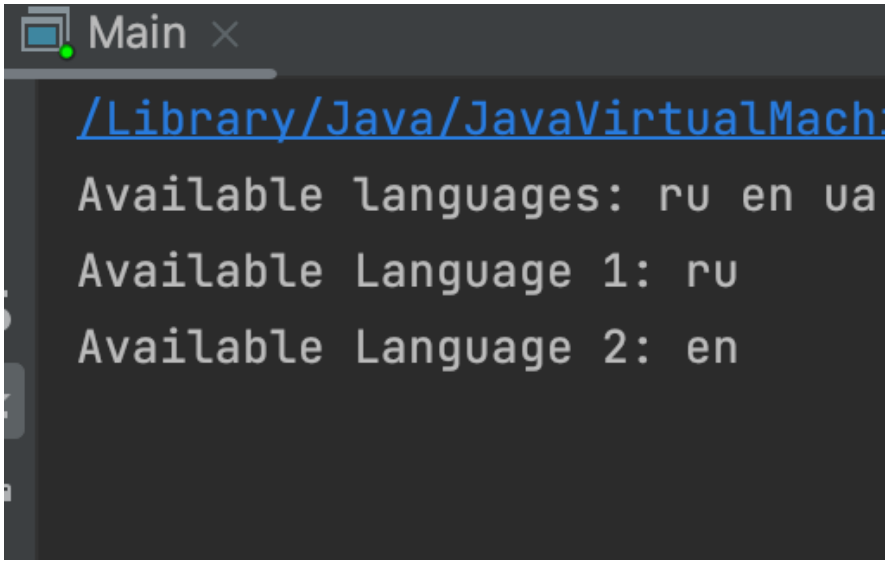


Рисунок 3.5 - Результати доступних мов, та обраних для роботи

– Додамо у набір мов швидкого доступу ще одну мову. Наприклад, албанську (у шифрі мов Apple вона позначається як «sq»), та проведемо запуск програмного забезпечення ще раз. Результат виконання цього тесту зображений на рисунку 3.6.

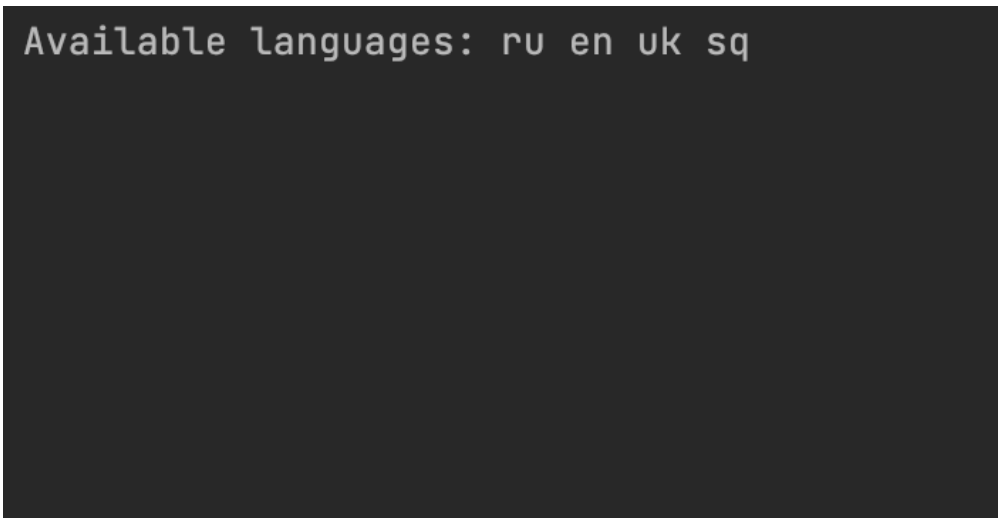


Рисунок 3.6 - Результат доступних мов після додавання ще одної.

Як бачимо, програма побачила нову мову, і додала її у список доступних мов. З цього можемо зробити висновок, що програмне забезпечення підтримує мови, які доступні на Macintosh OS, тобто є універсальним рішенням.

– Введемо будь-яке слово та перевіримо, чи правильно воно зберігається при натисканні на клавішу «пробіл». Результати можна побачити на рисунку 3.7.

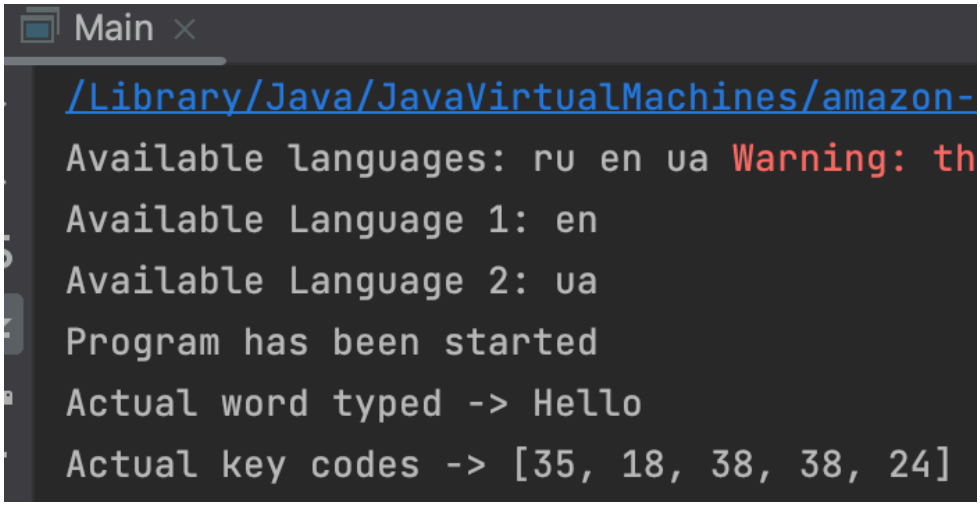


Рисунок 3.7 - Результат зчитування поточного слова.

Як бачимо на рисунку 3.7, програма правильно записує поточно введене слово. Також програмне забезпечення зберігає кей-коди введеного слова. Це потрібно для емуляції натискання клавіш, у випадку якщо поточне слово введено неправильно, і його потрібно виправити.

– Перевіримо роботу методу, який перевіряє, чи правильно введено слово за допомогою Google Translator API. Для початку, введемо правильно написане слово. Результат обробки правильно написаного слова можна побачити на рисунку 3.8.

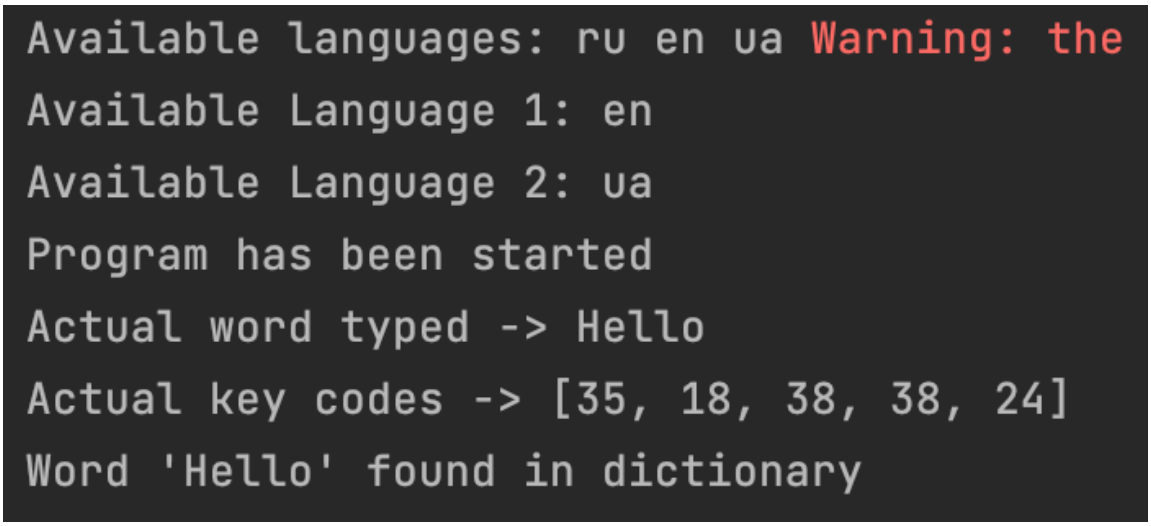


Рисунок 3.8 - Результат обробки правильно написаного слова

Судячи з рисунку 3.8, програма правильно перевірила слово, написане на коректній мові вводу клавіатури.

– Тепер введемо слово на неправильній розкладці клавіатури. Результат цього тесту можна побачити на рисунку 3.9.

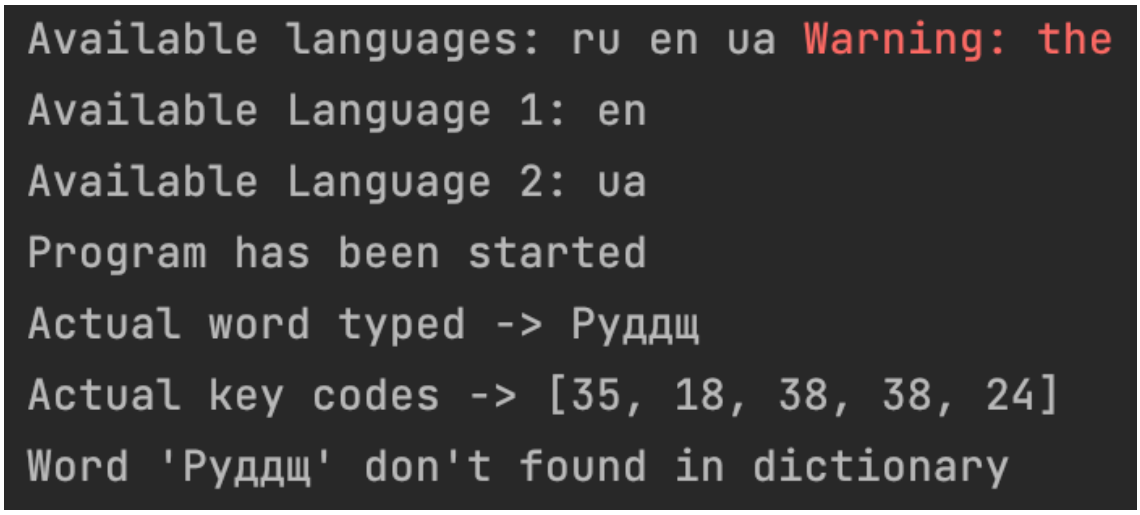


Рисунок 3.9 - Результат обробки неправильно написаного слова.

Як бачимо, програмне забезпечення показує, що поточне слово не знайдено в словнику відповідної мови (словнику української мови, оскільки слово було введено саме на даній мові).

– Введемо слово на неправильній мові вводу клавіатури, для того, щоб перевірити: чи змінюється мова вводу клавіатури на правильну, при друку неправильно написаного слова. У якості тестової пари мов оберемо англійську та українську. Результат цього тесту можна побачити на рисунку 3.10.

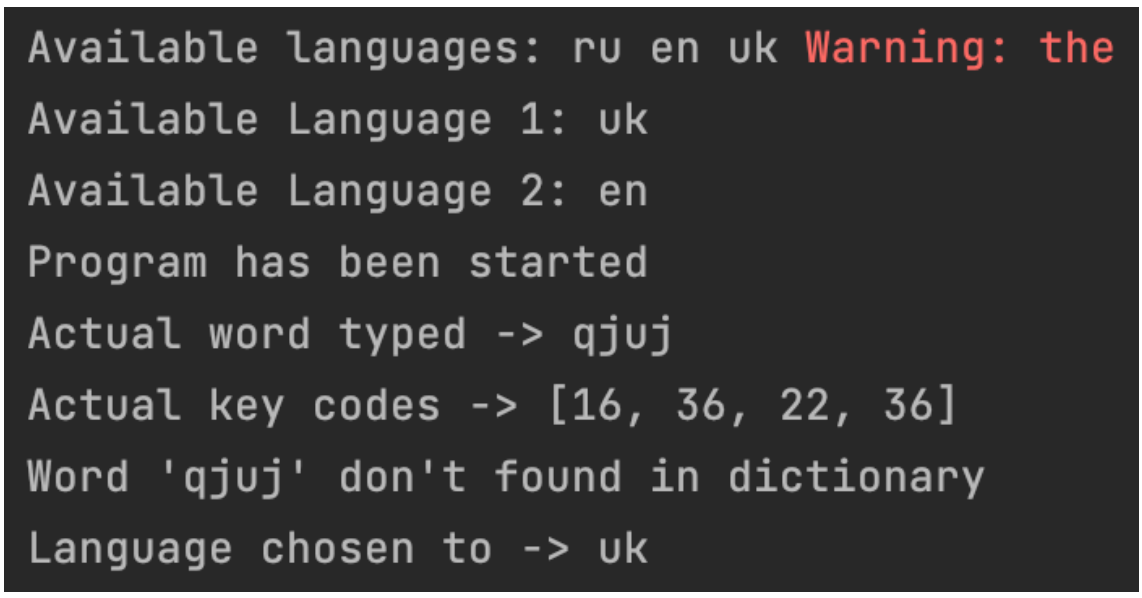


Рисунок 3.10 - Результат зміни поточної мови.

Як бачимо на рисунку, спочатку було обрана англійська мова вводу клавіатури. Програма проаналізувала, що такого слова, як «qjuj», немає у словнику англійської мови, після чого змінила поточну мову вводу на українську (uk).

– Тепер проведемо тестування видалення неправильно написаного слова. Очікується, що при написанні слова на неправильній мові вводу, програма автоматично видалить дане слово з того місця, де воно було введено. Результат цього тестування зображений на рисунку 3.11.

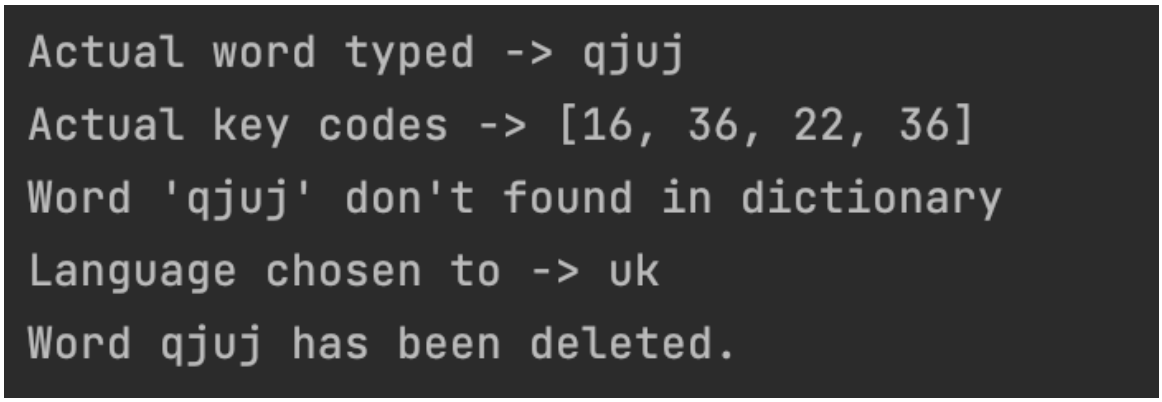


Рисунок 3.11 - Видалення неправильно написаного слова.

Судячи з рисунку 3.11 бачимо, що слово «qjuj», яке відсутнє у словнику відповідної мови було видалено.

– Автоматичне написання поточного слова на правильній мові вводу клавіатури. Введемо неправильне слово на англійській мові вводу, наприклад «,fnmrj», що на українській мові вводу означає «батьки». Очікується, що програмне забезпечення автоматично видалить слово «,fnmrj», і замість нього надрукує слово батьки». Результат цього тестування зображений на рисунках 3.12 та 3.13.

Рисунок 3.12 - Неправильно введене слово.

Рисунок 3.13 - Результат виправлення слова.

Як бачимо на рисунках, програмне забезпечення правильно виконало емуляцію натискання клавіш клавіатури, у повній відповідності до заданого слова. Результатом тестування є повністю правильно виправлене слово.

Висновки до розділу

У процесі аналізу якості та тестування програмного забезпечення даного дипломного проекту, було проведено загальний аналіз програмного забезпечення за допомогою сервіса «Embald». За його результатами визначили, що загальний рівень коду є гарним (4.11/5 балів). Також провели тестування всіх основних методів та функцій програмного забезпечення. Кожний тест було детально описано та сформовано у окремих таблицях. Після виконання окремих тестів програмного забезпечення, було проведено

контрольний приклад, у якому повністю оцінили роботу програмного забезпечення. Після проведення всіх тестів визначили, що програмне забезпечення коректно виконує свої задачі та є універсальним для усіх мов.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Розроблений у рамках даного дипломного проекту застосунок буде розгорнуто на платформу GitHub. Існує декілька вагомих причин, чому GitHub є оптимальним вибором для публікації, зберігання та розповсюдження даного програмного забезпечення.

По-перше, GitHub являє собою одну з найпопулярніших платформ для розробки програмного забезпечення. Завдяки своїй популярності, GitHub надає можливість легко ділитися своїм кодом з широкою аудиторією розробників та спільнотою користувачів. Це дозволить отримувати відгуки, співпрацювати з іншими розробниками та покращити якість даного програмного забезпечення.

По-друге, GitHub пропонує надійні інструменти для контролю версій та керування проектами. Система контролю версій Git, на котрій заснован GitHub, дозволить відстежувати зміни в коді, вносити зміни та версіонувати програмне забезпечення. Крім того, GitHub забезпечує зручні функції роботи з гілками, запросами на злиття (pull requests) та можливістю ведення обговорень, що робить процес розробки ще більш організованим та ефективним.

Третьою причиною є те, що GitHub надає можливість розміщення документації та інструкцій по використанню програмного забезпечення у вигляді файлів README.md. Це дозволить детально описати функціональність, особливості та застосування програми, що полегшить розуміння і використання даної програми іншими розробниками.

Також слід зазначити, що GitHub забезпечує інтеграцію з іншими інструментами розробки, такими як системи безперервної інтеграції та доставки (CI/CD), автоматичною перевіркою коду, керування помилками та

інше. Це значно полегшить процес подальшої розробки, тестування та розповсюдження програмного забезпечення.

Окрім того, GitHub є відкритою платформою, де будь-який бажаючий розробник має можливість вільно переглядати даний проект. Це надасть можливість залучити більше людей до розвитку даного програмного забезпечення, і також сприяє широкому розповсюдженню і використанню даного програмного забезпечення.

Для того, що розгорнути програмне забезпечення даного дипломного проекту було використано плагін GitHub для середовища розробки IntelliJ Idea. З його допомогою можна доволі легко завантажити проект на GitHub. Процес розгортання програмного забезпечення на GitHub можна побачити на рисунках 4.1 та 4.2.

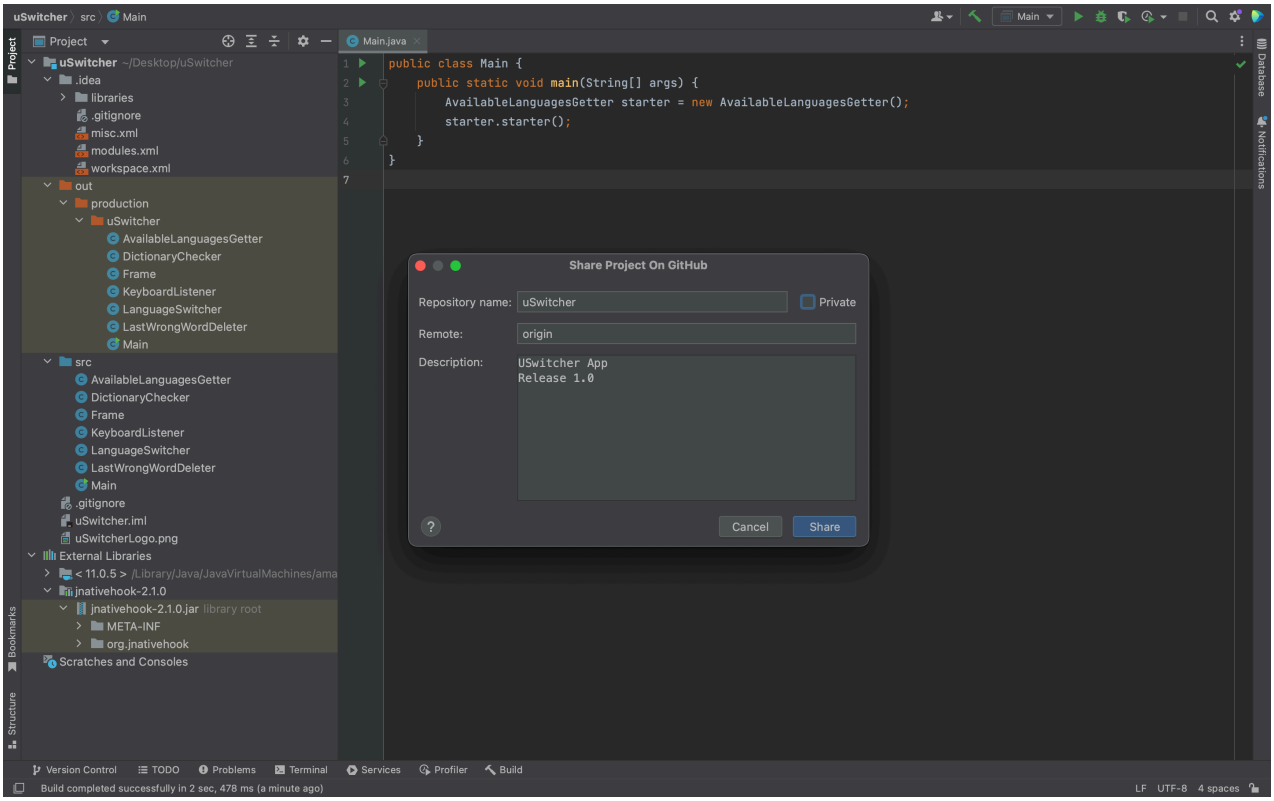


Рисунок 4.1 - Розгортання програмного забезпечення на GitHub.

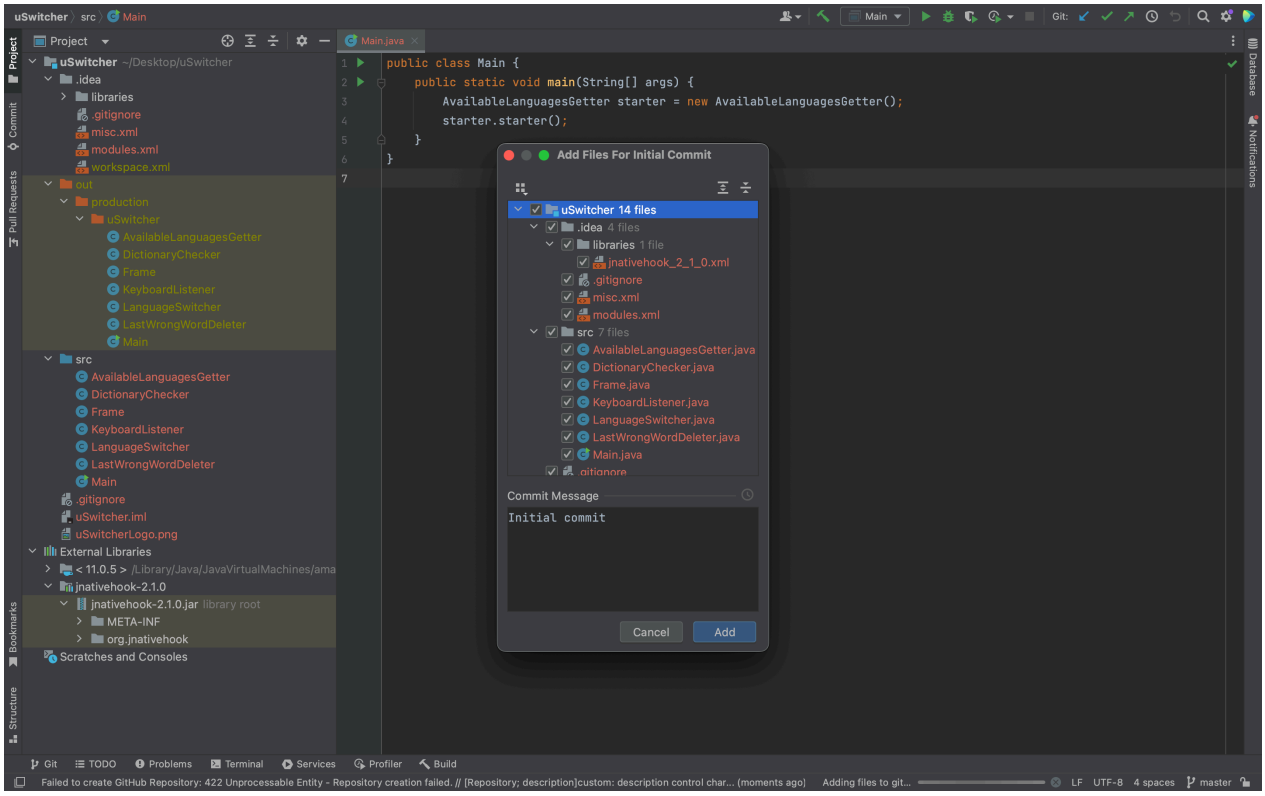


Рисунок 4.2 - Розгортання програмного забезпечення на GitHub.

Результат розгортання програмного забезпечення на GitHub можна побачити на рисунку 4.3.

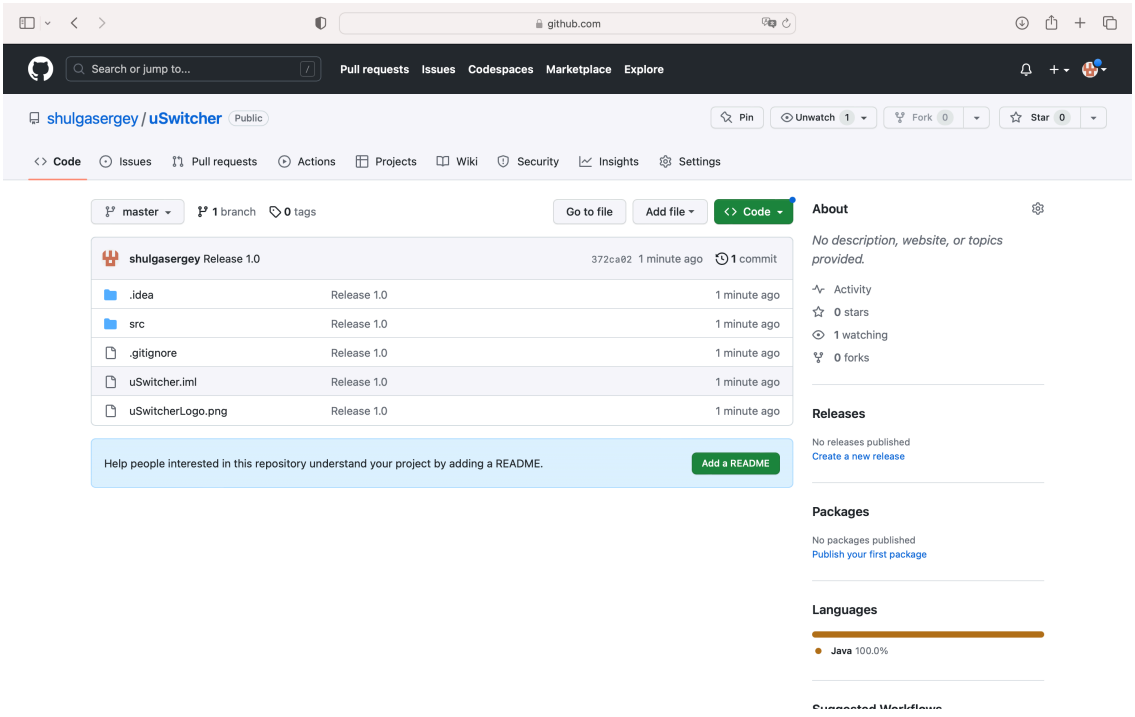


Рисунок 4.3 - Репозиторій програмного забезпечення на GitHub.

4.2 Підтримка програмного забезпечення

Підтримка застосунку також відбуватиметься через GitHub. Процес оновлення схожий з процесом першого розгортання, і також відбувається черзе плагін GitHub у середовищі розробки IntelliJ Idea.

Такий спосіб підтримки застосунку надає можливість контролювати версіїї, бачити усі зміни в коді програмного забезпечення та швидко впроваджувати зміни у застосунку.

Інтерйфейс Git у IntelliJ Idea для подальшого оновлення програмного забезпечення можна побачити на рисунку 4.4.

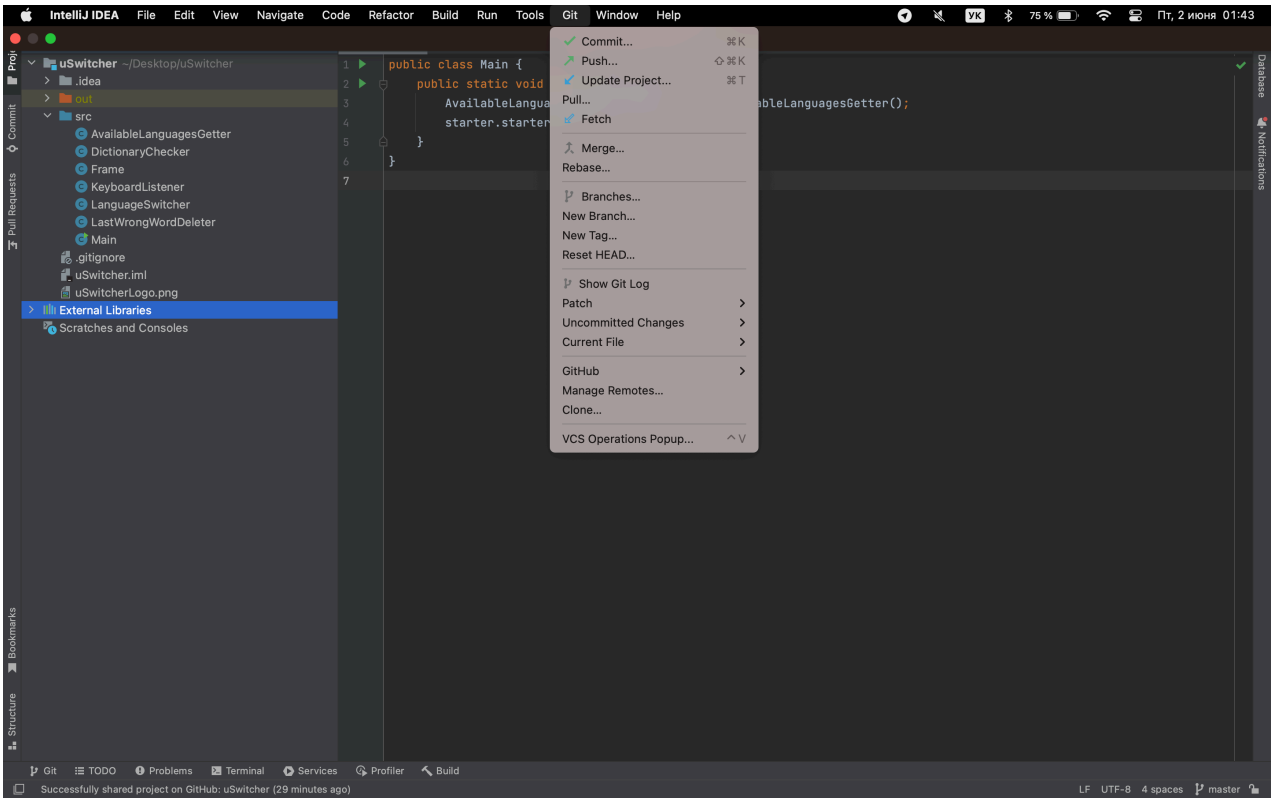


Рисунок 4.4 - Інтерфейс Git у IntelliJ Idea.

Висновки до розділу

У даному розділі дипломного проекту було обрано спосіб розгортання програмного забезпечення на GitHub, проаналізовано усі переваги даного рішення. Після цього, проект був безпосередньо доданий на GitHub у

відповідний репозиторій. Також було висвітлено процес подальшої підтримки та оновлення даного програмного забезпечення.

ВИСНОВКИ

В процесі написання даного дипломного проекту було виконано, досліджено та проаналізовано наступні аспекти:

– проаналізовано загальні положення досліджуваної області, наведено опис та загальні положення;

– було досліджено відомі алгоритми та засоби розробки, які можуть бути корисними в процесі написання даного дипломного проекту;

– розроблено користувацькі, функціональні та нефункціональні вимоги до програмного забезпечення, крім того було сформовано постановку задачі;

– створено бізнес-модель проекту, створено та описано загальну архітектуру проекту, проаналізовано безпеку даних;

– був проведений загальний аналіз програмного забезпечення, також тестування головних функцій та методів. Крім того, був проведений контрольний приклад програмного забезпечення;

– описано процес розгортання програмного забезпечення у відкритий доступ, та процес подальшої підтримки та оновлення;

– в якості середовища розробки було обрано IntelliJ Idea версії Ultimate 2023.1.2. В якості платформи для розгортання програмного забезпечення було обрано GitHub.

Для написання даного програмного забезпечення були використані останні версії інструментів, які знадобились для реалізації даного проекту.

Результатом дипломного проектування є застосунок для Macintosh OS, який здатний реалізувати моніторинг натисискання клавіш з автоматичною зміною мови та корекцією. Дане програмне забезпечення є унівпрсальним рішенням для користувачів Macintosh OS.

Програмне забезпечення було детально протестовано. Були змінені вхідні дані для впевненості в універсальності даного програмного забезпечення. Розроблений проект є повністю готовим для використання іншими користувачами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Java Documentation [Електронний ресурс] — Режим доступу: https://docs.oracle.com/en/java/;
2. https://www.baeldung.com/java-jna-dynamic-libraries;
3. Using JNA to Access Native Dynamic Libraries [Електронний ресурс] — Режим доступу: https://orientdb.com/docs/2.2.x/Java-Hooks.html;
4. USwithcer [Електронний ресурс] — Режим доступу: https://console.cloud.google.com/apis/dashboard?project=uswitcher;
5. Guide to JNI (Java Native Interface) [Електронний ресурс] — Режим доступу: https://www.baeldung.com/jni;
6. Introduction to AppleScript Language Guide [Електронний ресурс] — Режим доступу: https://developer.apple.com/library/archive/documentation/AppleScript/Conceptual/AppleScriptLangGuide/introduction/ASLR_intro.html;
7. Punto Switcher [Електронний ресурс] — Режим доступу: [no enemy's links]
8. Caramba Switcher [Електронний ресурс] — Режим доступу: [no enemy's links]
9. RuSwitcher [Електронний ресурс] — Режим доступу: [no enemy's links]

ДОДАТОК А ЗВІТ ПОДІБНОСТІ

UNICHECK

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1015400349

Дата перевірки:
02.06.2023 18:42:37 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
02.06.2023 18:45:25 EEST

ID користувача:
76913

Назва документа: **IT-94_Шульга_ПЗ**

Кількість сторінок: **61** Кількість слів: **8028** Кількість символів: **64290** Розмір файлу: **6.79 MB** ID файлу: **1015064235**

9.32% Схожість

Найбільша схожість: **4.2%** з джерелом з Бібліотеки (ID файлу: **1015058461**)

3.26% Джерела з Інтернету	131	Сторінка 63
9.18% Джерела з Бібліотеки	192	Сторінка 65

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0% Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	6

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

Програмне забезпечення для моніторингу натискання клавіш з автоматичним переключенням мови та корекцією

Текст програми

КПІ.ІТ-9430.045200.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олександр СТЕЛЬМАХ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Сергій ШУЛЬГА

Київ – 2023

Файл Main.java

```
public class Main {
    public static void main(String[] args) {
        AvailableLanguagesGetter starter = new AvailableLanguagesGetter();
        starter.starter();
    }
}
```

Файл AvailableLanguagesGetter.java

```
import java.util.ArrayList;
import java.util.List;


public class AvailableLanguagesGetter {
    public static String[] availableLanguages;

    public void starter() {
        availableLanguages = getAvailableLanguages();

        System.out.print("Available languages: ");

        if (availableLanguages.length > 0) {
            String firstLanguage = availableLanguages[0];
            if (firstLanguage.length() > 4) {
                availableLanguages[0] = firstLanguage.substring(4);
            }
        }

        availableLanguages = removeLanguage(availableLanguages, "ua");

        for (String language : availableLanguages) {
            System.out.print(language + " ");
        }

        Frame frame = new Frame();
        frame.frameStarter(availableLanguages);
    }

    private static String[] getAvailableLanguages() {
        List<String> languages = new ArrayList<>();

        String appleLanguages = getAppleLanguages();

        String[] languageCodes = appleLanguages.split(",");

        for (int i = 0; i < languageCodes.length; i++) {
            String code = languageCodes[i].trim();
            if (!code.isEmpty()) {
                String language = code.split("-")[0].toLowerCase();
```

```
                if (i == languageCodes.length - 1) {
                    String lastLanguage = code.split("-")[1].toLowerCase();
                    languages.add(stripQuotes(stripParentheses(language)));
                    languages.add(stripQuotes(stripParentheses(lastLanguage)));
                } else {
                    languages.add(stripQuotes(stripParentheses(language)));
                }
            }
        }

        return languages.toArray(new String[0]);
    }

    private static String getAppleLanguages() {
        StringBuilder appleLanguages = new StringBuilder();
        try {
            Process process = Runtime.getRuntime().exec("defaults read -g AppleLanguages");
            java.io.BufferedReader reader = new java.io.BufferedReader(new
java.io.InputStreamReader(process.getInputStream()));

            String line;
            while ((line = reader.readLine()) != null) {
                appleLanguages.append(line);
            }
            reader.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return appleLanguages.toString();
    }

    private static String stripQuotes(String str) {
        return str.replace("\"", "");
    }

    private static String stripParentheses(String str) {
        return str.replace("(", "").replace(")", "");
    }

    public static String[] removeLanguage(String[] languages, String languageToRemove) {
        int index = -1;
```

```
        for (int i = 0; i < languages.length; i++) {
            if (languages[i].equals(languageToRemove)) {
                index = i;
                break;
            }
        }

        if (index != -1) {
            String[] updatedLanguages = new String[languages.length - 1];
            System.arraycopy(languages, 0, updatedLanguages, 0, index);
            System.arraycopy(languages, index + 1, updatedLanguages, index, languages.length - index - 1);
            return updatedLanguages;
        }

        return languages;
    }
}
```

Файл KeyboardListener.java

```
import java.util.*;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import org.jnativehook.GlobalScreen;
import org.jnativehook.NativeHookException;
import org.jnativehook.keyboard.NativeKeyEvent;
import org.jnativehook.keyboard.NativeKeyListener;

public class KeyboardListener implements NativeKeyListener {
    DictionaryChecker dictionaryChecker = new DictionaryChecker();
    public String currentWord = ""; // Переменная текущего слова
    List<Integer> currentKeyCodes = new ArrayList<>();
    List<Integer> previousKeyCodes = new ArrayList<>();
    int keyCode;
    char[] extrasymbols = {'[', ']', ';', '\'', '\\', ',', '.'};


    public void nativeKeyPressed(NativeKeyEvent e) {
        keyCode = e.getKeyCode();

        if (e.getKeyCode() == NativeKeyEvent.VC_BACKSPACE) {
            if (currentWord.length() > 0) {
                currentWord = currentWord.substring(0, currentWord.length() - 1);
                currentKeyCodes.remove(currentKeyCodes.size() - 1);
            }
        }
        //System.out.println("Key Pressed: " + NativeKeyEvent.getKeyText(e.getKeyCode()));
    }

    public void nativeKeyReleased(NativeKeyEvent e) {

        //System.out.println("Key Released: " + NativeKeyEvent.getKeyText(e.getKeyCode()));
    }

    public void nativeKeyTyped(NativeKeyEvent e) {
        if (Frame.runningProgramm) {
            char actualSymbolTyped = e.getKeyChar();
```

```
        if (actualSymbolTyped == ' ') {
            if (!currentKeyCodes.equals(previousKeyCodes)) {
                if (!currentWord.isEmpty()) {

                    System.out.println("Actual word typed -> " + currentWord);
                    System.out.println("Actual key codes -> " + currentKeyCodes);

                    dictionaryChecker.checkerWorker(currentWord, currentKeyCodes);

                    currentWord = "";
                    previousKeyCodes.clear();
                    previousKeyCodes.addAll(currentKeyCodes);
                    currentKeyCodes.clear();

                    System.out.println("------------------------------");
                    System.out.println(" ");
                }
            } else {
                System.out.println("Word has been processed already");
                currentWord = "";
                currentKeyCodes.clear();
                System.out.println("------------------------------");
            }
        } else if (Character.isLetter(actualSymbolTyped) || containsSymbol(actualSymbolTyped, extrasymbols)) {
            currentWord += actualSymbolTyped;
            currentKeyCodes.add(keyCode);
        }
    }
}

public static boolean containsSymbol(char symbol, char[] symbols) {
    for (char s : symbols) {
        if (s == symbol) {
            return true;
        }
    }
    return false;
}

public void mainListener() {
```

```
        Logger logger = Logger.getLogger(GlobalScreen.class.getPackage().getName());
        logger.setLevel(Level.OFF);

        try {
            GlobalScreen.registerNativeHook();
        } catch (NativeHookException ex) {
            System.err.println("There was a problem registering the native hook.");
            System.err.println(ex.getMessage());
            System.exit(1);
        }
        GlobalScreen.addNativeKeyListener(new KeyboardListener());
    }
}
```

Файл DictionaryChecker.java

```
import java.awt.*;
import java.awt.im.InputContext;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;
import java.util.*;
import java.util.List;

public class DictionaryChecker extends Frame{

    LanguageSwitcher languageSwitcher = new LanguageSwitcher();
    LastWrongWordDeleter lastWrongWordDeleter = new LastWrongWordDeleter();

    static String currentLanguage = "";

    String correctlanguage = " ";

    public static boolean isWordInDictionary(String word, String apiKey) throws IOException {
        String apiUrl = "https://translation.googleapis.com/language/translate/v2/detect?key=" + apiKey + "&q=";
        String urlEncodedWord = URLEncoder.encode(word, StandardCharsets.UTF_8);

        URL url = new URL(apiUrl + urlEncodedWord);
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        connection.setRequestMethod("GET");

        int responseCode = connection.getResponseCode();

        if (responseCode == HttpURLConnection.HTTP_OK) {
            InputStream inputStream = connection.getInputStream();
            Scanner scanner = new Scanner(inputStream, StandardCharsets.UTF_8);
            String response = scanner.useDelimiter("\\A").next();
            scanner.close();

            return response.contains("\"language\": \"" + currentLanguage + "\"");

        } else {
            throw new IOException("Не удалось выполнить запрос к API, код ошибки: " + responseCode);
```

```
        }
    }

    public void checkerWorker(String currentWord, List<Integer> currentKeyCodes) {

        InputContext inputContext = InputContext.getInstance();
        if ("_US_UserDefined_252".equals(String.valueOf(inputContext.getLocale()))) {
            currentLanguage = "en";
        } else {
            currentLanguage = String.valueOf(inputContext.getLocale());
        }

        String apiKey = "AIzaSyDISFdioPgs9MR2DXVuQT9sscjpTYYVjr8";

        try {

            if (isWordInDictionary(currentWord, apiKey)) {
                System.out.println("Word '" + currentWord + "' found in dictionary");
            } else {

                System.out.println("Word '" + currentWord + "' don't found in dictionary");

                if (Objects.equals(currentLanguage, availableLanguage1)) {
                    correctlanguage = availableLanguage2;
                } else {
                    correctlanguage = availableLanguage1;
                }

                languageSwitcher.mainSwitcher(correctlanguage, currentLanguage);
                lastWrongWordDeleter.mainDeleter(currentWord, currentKeyCodes);
            }
        } catch (IOException e) {
            System.err.println("Ошибка при выполнении запроса к API: " + e.getMessage());
        } catch (InterruptedException | AWTException e) {
            throw new RuntimeException(e);
        }
    }
}
```

Файл LanguageSwitcher.java

```
import java.awt.im.InputContext;
import java.io.IOException;

public class LanguageSwitcher {
    public void mainSwitcher(String correctlanguage, String currentLanguage) throws InterruptedException {

        while (!correctlanguage.equals(currentLanguage)) {
            try {
                String script = "tell application \"System Events\"\n" +
                        "   keystroke tab using {control down}\n" +
                        "end tell";

                String[] cmd = {"osascript", "-e", script};

                Process process = Runtime.getRuntime().exec(cmd);
                process.waitFor();
                Thread.sleep(70);
            } catch (IOException | InterruptedException e) {
                e.printStackTrace();
            }

            InputContext inputContext = InputContext.getInstance();
            if ("_US_UserDefined_252".equals(String.valueOf(inputContext.getLocale()))) {
                currentLanguage = "en";
            } else {
                currentLanguage = String.valueOf(inputContext.getLocale());
            }

        }
        System.out.println("Language chosen to -> " + currentLanguage);
        Thread.sleep(10);
    }
}
```

Файл LastWrongWordDeleter.java

```
import java.awt.*;
import java.awt.event.KeyEvent;
import java.io.IOException;
import java.util.List;

public class LastWrongWordDeleter {

    public void mainDeleter(String currentWord, List<Integer> currentKeyCodes) throws AWTException {

        int numberOfBackspaces = currentWord.length() + 1;

        try {
            StringBuilder scriptBuilder = new StringBuilder();
            scriptBuilder.append("tell application \"System Events\"\n");
            for (int i = 0; i < numberOfBackspaces; i++) {
                scriptBuilder.append("    key code 51\n");
            }
            scriptBuilder.append("end tell");

            String[] cmd = {"osascript", "-e", scriptBuilder.toString()};
            Runtime.getRuntime().exec(cmd);
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.println("Word " + currentWord + " has been deleted.");
        printTransform(currentKeyCodes);
    }

    public void printTransform(List<Integer> currentKeyCodes) {

        try {
            Robot robot = new Robot();
            robot.delay(215);
            for (int keyCode : currentKeyCodes) {
                int vkCode = getKeyEventKeyCode(keyCode);
                robot.keyPress(vkCode);
                robot.keyRelease(vkCode);
            }

            int spaceVkCode = KeyEvent.VK_SPACE;
```

```
            robot.keyPress(spaceVkCode);
            robot.keyRelease(spaceVkCode);
        } catch (AWTException e) {
            e.printStackTrace();
        }
    }

    private static int getKeyEventKeyCode(int keyCode) {
        int vkCode = -1;
        int[][] keycodeMapping = LastWrongWordDeleter.KEYCODES;

        for (int[] mapping : keycodeMapping) {
            if (mapping[1] == keyCode) {
                vkCode = mapping[0];
                break;
            }
        }
        return vkCode;
    }

    public static final int[][] KEYCODES = {
            {KeyEvent.VK_Q, 16},
            {KeyEvent.VK_W, 17},
            {KeyEvent.VK_E, 18},
            {KeyEvent.VK_R, 19},
            {KeyEvent.VK_T, 20},
            {KeyEvent.VK_Y, 21},
            {KeyEvent.VK_U, 22},
            {KeyEvent.VK_I, 23},
            {KeyEvent.VK_O, 24},
            {KeyEvent.VK_P, 25},
            {KeyEvent.VK_OPEN_BRACKET, 27},
            {KeyEvent.VK_CLOSE_BRACKET, 28},
            {KeyEvent.VK_A, 30},
            {KeyEvent.VK_S, 31},
            {KeyEvent.VK_D, 32},
            {KeyEvent.VK_F, 33},
            {KeyEvent.VK_G, 34},
            {KeyEvent.VK_H, 35},
            {KeyEvent.VK_J, 36},
            {KeyEvent.VK_K, 37},
            {KeyEvent.VK_L, 38},
```

```
            {KeyEvent.VK_SEMICOLON, 39},
            {KeyEvent.VK_QUOTE, 40},
            {KeyEvent.VK_BACK_SLASH, 43},
            {KeyEvent.VK_Z, 44},
            {KeyEvent.VK_X, 45},
            {KeyEvent.VK_C, 46},
            {KeyEvent.VK_V, 47},
            {KeyEvent.VK_B, 48},
            {KeyEvent.VK_N, 49},
            {KeyEvent.VK_M, 50},
            {KeyEvent.VK_COMMA, 51},
            {KeyEvent.VK_PERIOD, 52}
    };
}
```

Файл Frame.java

```
import java.awt.*;
import java.awt.event.*;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import javax.swing.*;
import javax.swing.border.EmptyBorder;

public class Frame {

    public static String availableLanguage1 = null;
    public static String availableLanguage2 = null;
    private static TrayIcon trayIconObj;
    public static boolean runningProgramm = true;

    public void frameStarter(String[] availableLanguages) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createTrayWindow(availableLanguages);
            }
        });
    }

    private static void createTrayWindow(String[] availableLanguages) {
        if (!SystemTray.isSupported()) {
            System.out.println("SystemTray is not supported on this platform.");
            return;
        }

        PopupMenu popup = new PopupMenu();

        MenuItem button1Item = new MenuItem("Обрати мови для зміни");
        button1Item.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {

                JFrame frame = new JFrame("Доступні мови");
                frame.setSize(300, 200);
                frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                frame.setLocationRelativeTo(null);
                frame.setResizable(false);
```

```
JPanel panel = new JPanel();
panel.setLayout(new GridBagLayout());
GridBagConstraints gbc = new GridBagConstraints();
gbc.anchor = GridBagConstraints.CENTER;
gbc.insets = new Insets(5, 5, 5, 5);

Font font = new Font("Tahoma", Font.PLAIN, 14);

final int[] selectedCount = {0};

for (String language : availableLanguages) {
    JCheckBox checkBox = new JCheckBox(language);
    checkBox.setFont(font);
    gbc.gridy++;
    panel.add(checkBox, gbc);

    checkBox.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            if (checkBox.isSelected()) {
                if (selectedCount[0] == 0) {
                    availableLanguage1 = checkBox.getText();
                } else if (selectedCount[0] == 1) {
                    availableLanguage2 = checkBox.getText();
                }
                selectedCount[0]++;
            } else {
                if (selectedCount[0] == 1 && checkBox.getText().equals(availableLanguage1)) {
                    availableLanguage1 = null;
                } else if (selectedCount[0] == 1 && checkBox.getText().equals(availableLanguage2)) {
                    availableLanguage2 = null;
                } else if (selectedCount[0] == 2 && checkBox.getText().equals(availableLanguage1)) {
                    availableLanguage1 = availableLanguage2;
                    availableLanguage2 = null;
                }
                selectedCount[0]--;
            }
        }
    });
}

JButton okButton = new JButton("Зберегти");
```

```
            okButton.setFont(font);
            okButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    if (selectedCount[0] != 2) {
                        JOptionPane.showMessageDialog(null, "Будь-ласка, оберіть дві мови");
                    } else {
                        JOptionPane.showMessageDialog(null, "Обрані мови: " + availableLanguage1 + ", " +
availableLanguage2);
                        System.out.println("Available Language 1: " + availableLanguage1);
                        System.out.println("Available Language 2: " + availableLanguage2);
                    }
                    frame.dispose();
                }
            });

            gbc.gridy++;
            panel.add(okButton, gbc);

            frame.add(panel);
            frame.setVisible(true);
        }
    });
    popup.add(button1Item);

    MenuItem button2Item = new MenuItem("Запуск програми");
    button2Item.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            System.out.println("Program has been started");
            KeyboardListener keyboardListener = new KeyboardListener();
            keyboardListener.mainListener();

        }
    });
    popup.add(button2Item);

    MenuItem pauseItem = new MenuItem("Пауза");
    pauseItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            System.out.println("Program paused");
            runningProgramm = false;
        }
    });
```

```
popup.add(pauseItem);

MenuItem resumeItem = new MenuItem("Продовжити");
resumeItem.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        System.out.println("Program resumed");
        runningProgramm = true;
    }
});
popup.add(resumeItem);

MenuItem button3Item = new MenuItem("Зв'язок з розробником");
button3Item.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            Desktop.getDesktop().browse(new URI("https://t.me/userfromworld"));
        } catch (IOException | URISyntaxException ex) {
            ex.printStackTrace();
        }
    }
});
popup.add(button3Item);

MenuItem button4Item = new MenuItem("Інструкції з використання");
button4Item.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {

        JFrame instructionsFrame = new JFrame("Інструкції");
        instructionsFrame.setSize(400, 300);
        instructionsFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        instructionsFrame.setLocationRelativeTo(null);
        instructionsFrame.setResizable(false);

        JTextArea instructionsText = new JTextArea();
        instructionsText.setFont(new Font("Verdana", Font.PLAIN, 14));
        instructionsText.setEditable(false);
        instructionsText.setLineWrap(true);
        instructionsText.setWrapStyleWord(true);
        instructionsText.setText("Для запуску програми натисніть кнопку \"Запуск\"\n\n"
                + "Якщо Вам потрібно на певний час зупинити програму, натисніть кнопку \"Пауза\"\n\n"
                + "Якщо Вам потрібно відновити роботу програми, натисніть кнопку \"Продовжити\"\n\n"
                + "Для виходу з програми натисніть кнопку \"Вихід\"\n\n"
```

```
                + "Дякую за використання uSwitcher. Все буде Україна!");

            instructionsText.setBorder(new EmptyBorder(10, 10, 10, 10));

            instructionsFrame.add(instructionsText);
            instructionsFrame.setVisible(true);
        }
    });
    popup.add(button4Item);

    MenuItem exitItem = new MenuItem("Вихід");
    exitItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            SystemTray.getSystemTray().remove(trayIconObj);
            System.exit(0);
        }
    });
    popup.add(exitItem);

    Image trayIcon = Toolkit.getDefaultToolkit().getImage("uSwitcherLogo.png");

    SystemTray tray = SystemTray.getSystemTray();
    trayIconObj = new TrayIcon(trayIcon, "uSwitcher", popup);

    try {
        tray.add(trayIconObj);
    } catch (AWTException e) {
        System.out.println("TrayIcon could not be added.");
    }
  }
}
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

Програмне забезпечення для моніторингу натискання клавіш з автоматичним переключенням мови та корекцією

Програма та методика тестування

КПІ.9430.045200.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олександр СТЕЛЬМАХ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Сергій ШУЛЬГА

Київ – 2023

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є програмне забезпечення для моніторингу натискання клавіш з автоматичним переключенням мови та корекцією. Програмне забезпечення працює на платформі операційної системи Macintosh OS.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

- перевірка правильності считування доступних мов вводу клавіатури, які доступні у швидкому доступі операційної системи;
- перевірка правильності считування поточно втсановленої мови вводу клавіатури;
- перевірка глобального слухача подій клавіатури - чи правильно програмне забезпечення аналізує введені слова;
- перевірка роботи API, яке перевіряє правильність написання поточного слова;
- перевірка зміни мови вводу клавіатури у разі неправильного вводу слова;
- перевірка стирання неправильно написаного слова;
- перевірка правильності емуляції виправленого слова.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуватиметсья метод тестування «білої скриньки» – об’єктом тестування є внутрішня поведінка програми. Такий спосіб тестування програмного забезпечення надає нам змогу перевірити виконання всіх функціональних вимог, окремих методів та їх взаємодію один з одним. Процес тестування проводився мануально.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування програмного забезпечення відбувається за допомогою середовища розробки IntelliJ Idea, з використанням інструментів даного середовища розробки, таких як Breakpoints, логування певних даних у консоль. IntelliJ Idea надає змогу спостерігати за змінами певних значень, методів, їх взаємодії один з одним та роботу програмного забезпечення в цілому.

Виходячи з цього, є змога повністю контролювати та аналізувати роботу всього програмного забезпечення.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

Програмне забезпечення для моніторингу натискання клавіш з автоматичним переключенням мови та корекцією

Керівництво користувача

КПІ.9430.045200.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олександр СТЕЛЬМАХ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Сергій ШУЛЬГА

Київ – 2023

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

“uSwitcher” - це програмне забезпечення для моніторингу натиснутих клавіш з автоматичним переключенням мови та корекцією - інструмент, мета якого - спростити роботу з друком тексту на операційній системі Macintosh. Дане програмне забезпечення працює у фоновому режимі, аналізує правильність набраних слів відносно поточної мови вводу клавіатури. Якщо поточне слово було набрано на неправильній мові вводу клавіатури, програмне забезпечення автоматично змінить мову вводу на коректну та виправить слово на правильне.

Кінцева збірка програмного забезпечення є повністю готовою до використання користувачами. Програмне забезпечення є універсальним рішенням, та здатно працювати з усіма мовами, які підтримуються операційною системою Macintosh.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного програмного забезпечення необхідне виконання наступних вимог:

– наявність комп’ютера з операційною системою на базі Macintosh з версією 13;

– наявність доступу до Інтернету;

– для встановлення програмного забезпечення на комп’ютері повинно бути не менше 200 МБ вільної пам’яті.

2.2 Завантаження застосунку

На даний момент застосунок можна встановити власноруч, використовуючи відповідний відкритий програмний код у відповідному репозиторії на платформі GitHub.. Для цього спершу необхідно завантажити його на комп’ютер, після чого виконати наступну вимогу:

Надати програмному забезпеченню доступ до керування комп’ютером. Це необхідно зробити, тому що програмне забезпечення працює з низькорівневими функціями, доступ до яких не надається автоматично. Ці дії необхідно зробити лише один раз.

1. Відкрити застосунок «Налаштування». Він позначений логотипом шестерні;
2. У меню обрати пункт «Безпека та конфіденційність»;
3. Обрати пункт «Універсальний доступ»;
4. Обрати Ваше середовище розробки, і переключити повзунок у активний стан.

Зображення цих дій можна побачити на рисунку 2.1.

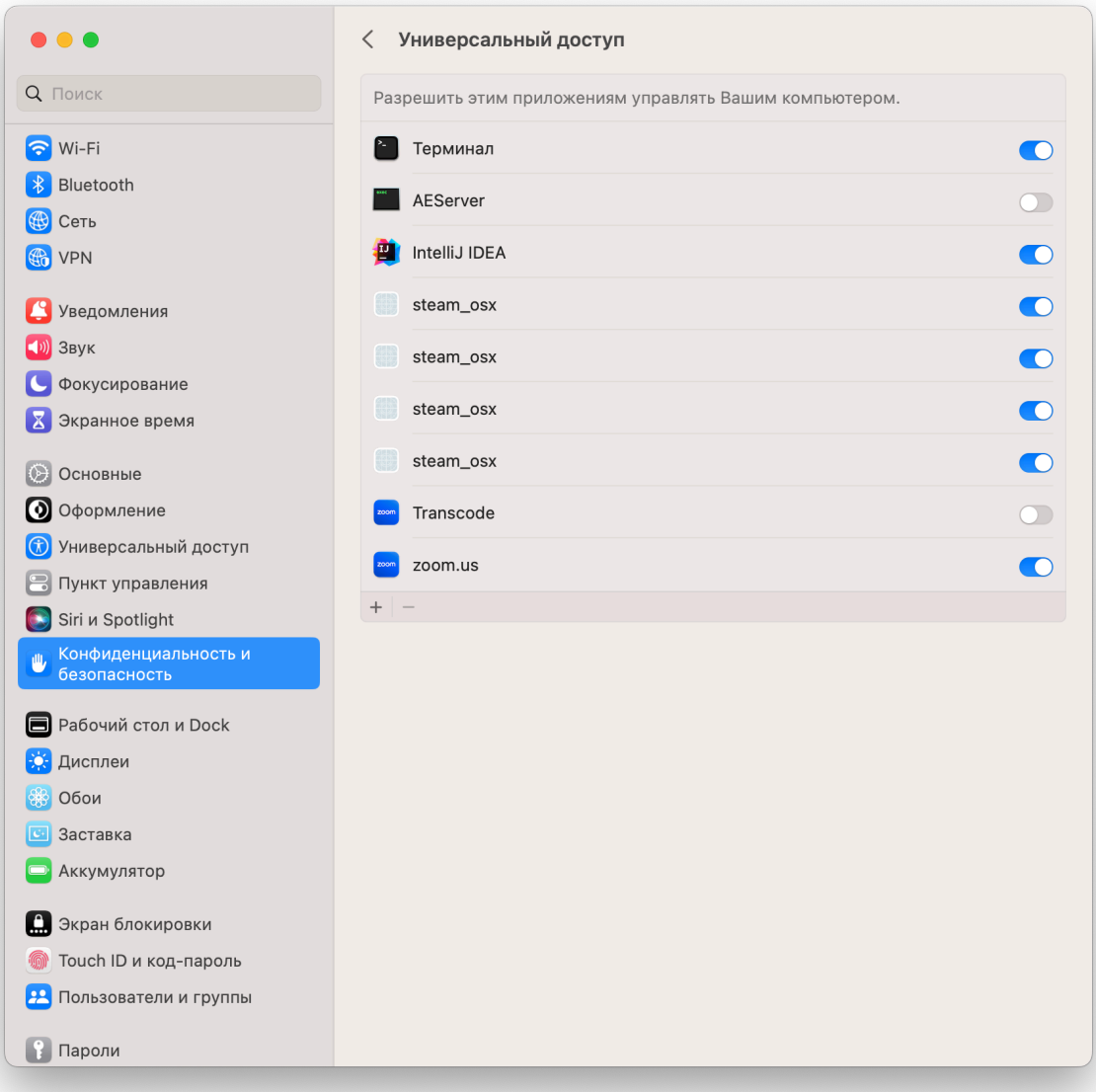


Рисунок 2.1 - Надання доступу для програмного забезпечення.

Після виконання цих дій, використовуючи будь-яке середовище розробки, можна відкрити програмне забезпечення і запустити його за допомогою кнопки «Run», яка зображена у вигляді зеленого трикутника.

2.3 Перевірка коректної роботи

По завершенню завантаження і запуску програмного забезпечення у меню-барі (Tray-меню) повинна відобразитись іконка даного програмного забезпечення. Після натискання на дану іконку, повинно відобразитись меню,

за допомогою якого користувач має можливість керувати програмним забезпеченням.

3 ВИКОНАННЯ ПРОГРАМИ

Після натискання на іконку програмного забезпеченням у меню-барі, користувач має можливість виконати декілька дій. Повний список доступних дій зображений на рисунку 3.1.

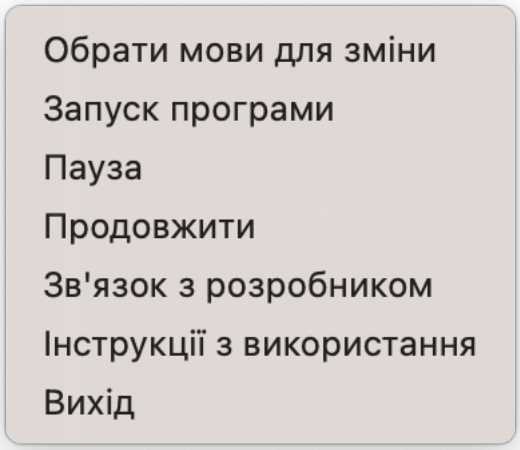


Рисунок 3.1 - Функціонал інтерфейсу користувача.

Для початку роботи необхідно обрати пару мов, між якими буде працювати програмне забезпечення. Список доступних мов складається з мов, які доступні у швидкому доступні. Вибір мов зображений на рисунку 3.2.

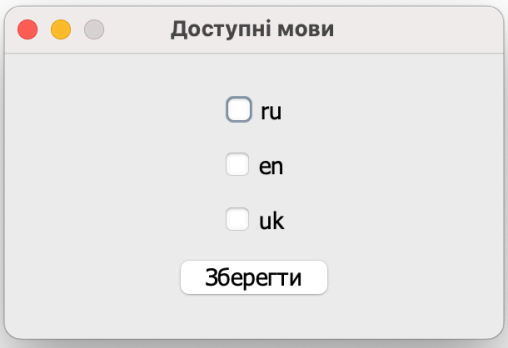


Рисунок 3.2 - Обрання мов для роботи.

Після успішного обирання мов для роботи, слід натиснути на кнопку «Зберегти». Користувач побачить повідомлення про успішне обрання мов. Його можна побачити на рисунку 3.3.

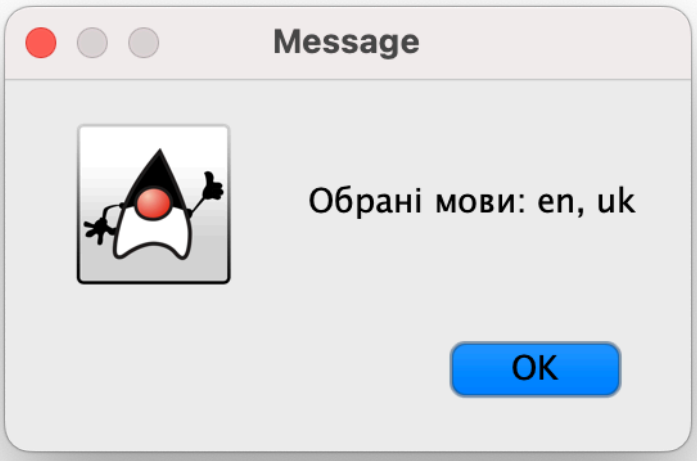


Рисунок 3.3 - Повідомлення про успішне обрання мов.

Наступним кроком для роботи програмного забезпечення є безпосередньо запуск програми. Для цього користувач повинен натиснути кнопку «Запуск програми» у головному трей-меню, яке можна побачити на рисунку 3.1.

Після виконання цих дій програмне забезпечення почне свою роботу в автоматичному режимі.

Для завершення роботи програмного забезпечення необхідно натиснути кнопку «Вихід», яка розташована у головному трей-меню програми. Її можна побачити на рисунку 3.1.

Програмне забезпечення пропонує декілька додаткових функцій.

Для паузи, або відновлення роботи програми, необхідно натиснути на відповідні кнопки «Пауза» або «Продовжити» у головному трей-меню

програми. Ці кнопки можна побачити на рисунку 3.1. Програмне забезпечення ігнорує все дії користувача, якщо натиснута кнопка «Пауза». Це може знадобитись, наприклад, для вводу пароля.

Також, користувач має змогу прочитати інструкції з використання програмного забезпечення, натиснувши кнопку «Інструкціїї з використання» у головному трей-меню. Дана кнопка також зображена на рисунку 3.1.

Користувач здатен зв’язатися з розробником програмного забезпечення, натиснувши на кнопку «Зв’язатись з розробником» у головному трей-меню програми. Ця кнопка також зображена на рисунку 3.1.

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2023 р.

Програмне забезпечення для моніторингу натискання клавіш з автоматичним переключенням мови та корекцією

Графічний матеріал

КПІ.9430.045200.06.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олександр СТЕЛЬМАХ

Нормоконтроль:

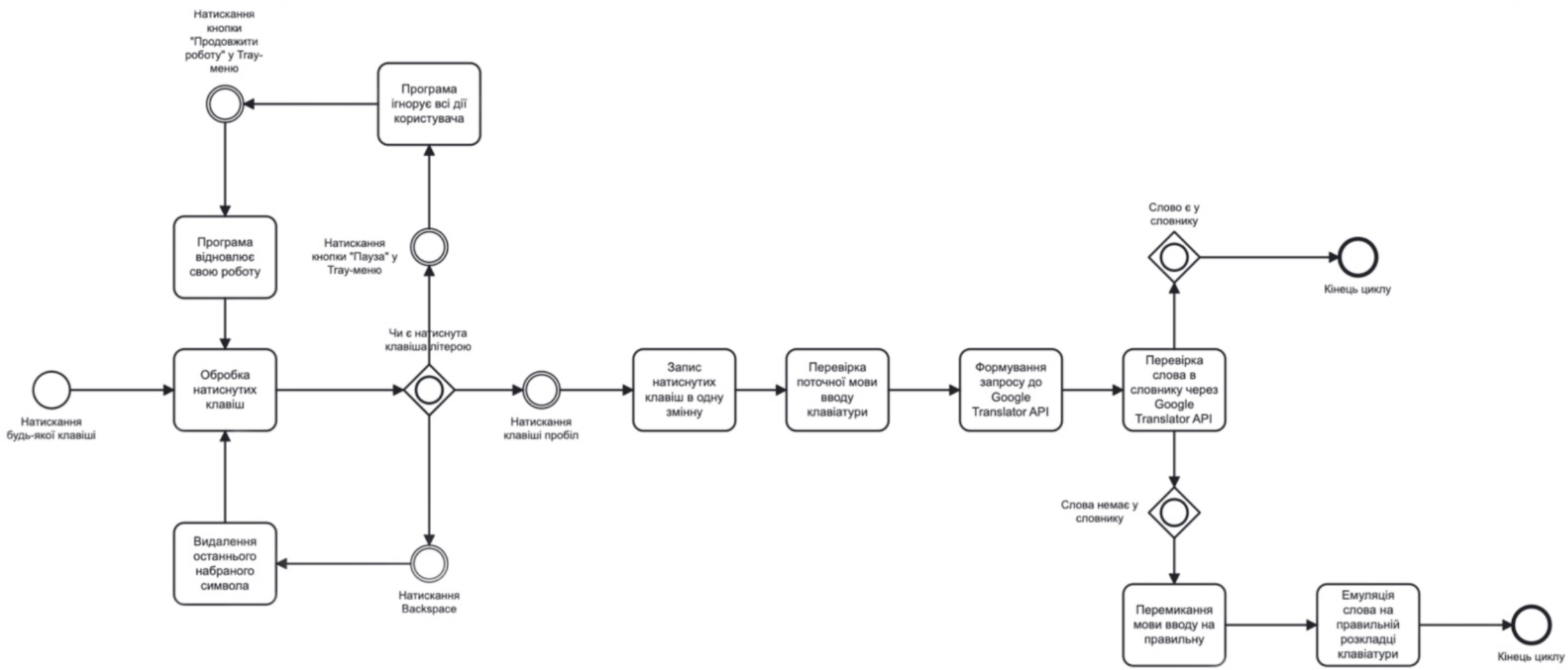
_____ Ірина ВІТКОВСЬКА

Виконавець:

_____ Сергій ШУЛЬГА

Київ – 2023

Діаграма бізнес-процесів



Демонстраційний плакат до дипломного проекту

Програмне забезпечення для моніторингу натискання клавіш з автоматичним переключенням мови та корекцією

Виконав студент гр. ІТ-94 Шульга С.С.

Керівник Стельмах О.П.

КПІ.9430.045200.06.99.СС

					КПІ.9430.045200.06.99.СС	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Шульга С.С.				
Перевірив		Стельмах О.П.				Аркуш / Аркушів
Т. кон.						
Н. кон.		Вітковська І.			Програмне забезпечення для моніторингу натискання клавіш з автоматичним переключенням мови та корекцією	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТТ-94
Затвердив		Стельмах О.П.				

КПІ.9430.045200.06.99.СС

					КПІ.9430.045200.06.99.СС		
						Літера	Маса Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна варіантів використання		
Розробив		Шульга С.С.				Аркуш	Аркушів
Перевірив		Стельмах О.П.					
Т. кон.							
Н. кон.		Вітковська І.			Програмне забезпечення для моніторингу натискання клавіш з автоматичним переключенням мови та корекцією	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТТ-94	
Затвердив		Стельмах О.П.					