

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

РОЗРОБКА ІНТЕРПРЕТАТОРА НАЙПРОСТІШОЇ МОВИ ПРОГРАМУВАННЯ

**Текстова частина до курсової роботи
за спеціальністю „Інженерія програмного забезпечення” 121**

Керівник курсової роботи
к.т.н., доц. Франчук О. В.

(підпис)
“__” _____ 2020 р.

Виконала студентка
Яременко С. О.
“__” _____ 2020 р.

Київ 2020

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЇВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

ЗАТВЕРДЖУЮ

Зав. Кафедри мережних технологій,

проф., д.ф.-м.н.

_____ Г.І. Малашонок

(підпис)

“ _____ ” _____ 2019 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Яременко Софії Олесівні факультету інформатики 4 курсу

ТЕМА: Розробка інтерпретатора найпростішої мови програмування

Вихідні дані: Інтерпретатор мови BASIC

Зміст ТЧ до курсової роботи:

Вступ

Розділ 1: Теоретичні відомості. Постановка завдання

Розділ 2: Опис реалізації програмного продукту

Висновки

Список літератури

Додатки

Дата видачі “ _____ ” _____ 2019 р. Керівник _____

(підпис)

Завдання отримала _____

(підпис)

Календарний план виконання курсової роботи

Тема: Розробка інтерпретатора найпростішої мови програмування

№ п/п	Назва етапу курсового проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	листопад 2019 р.	
2.	Огляд літератури за темою роботи	січень 2020 р.	
3.	Створення практичної частини роботи	лютий 2020 р.	
4.	Написання текстової частини роботи	березень 2019 р.	
5.	Створення презентації для доповіді	квітень 2019 р.	
6.	Надання роботи керівнику для перевірки	квітень 2019 р.	
7.	Остаточне оформлення текстової частини та слайдів	квітень 2019 р.	
8.	Подання роботи для перевірки на плагіат	19 квітня 2020 р.	

Студентка Яременко С. О.

Керівник Франчук О. В.

“ ” _____ р.

АНОТАЦІЯ

У данній курсовій роботі розроблена програма, яка є інтерпретатором мови BASIC. Інтерпретатор працює з базовими командами вхідної мови програмування.

В текстовій частині обґрунтовано здійснений вибір засобів реалізації застосунку. Також проведений аналіз програми, розроблена її структура та користувацький інтерфейс, наведені вимоги для роботи програми і дії, необхідні для її запуску і функціонування. Окрім цього було проведено тестування та виявлено працездатність розробленого застосунку. Програмний код написаний на мові Java.

ЗМІСТ

АНОТАЦІЯ.....	4
ВСТУП.....	6
РОЗДІЛ 1: Теоретичні відомості. Визначення завдання курсової роботи	7
1.1 Розподіл мов програмування за способом реалізації	7
1.2 Інтерпретатори.....	9
1.3 Мова програмування BASIC.....	10
1.4 Опис основних програмних конструкцій та команд BASIC	12
1.5 Визначення завдання.....	15
РОЗДІЛ 2: Опис реалізації програмного продукту	16
2.1 Аналіз технічного завдання	16
2.2 Визначення граматики для інтерпретатора.....	17
2.3 Обґрунтування вибору засобів розробки	17
2.4 Опис програми та процесу розробки.....	19
2.6 Опис інтерфейсу програми	21
2.8 Тестування програми і результати її виконання.....	24
ВИСНОВКИ	26
Список літератури	27
Додаток А	28

ВСТУП

Мови програмування класифікують за багатьма критеріями. Один з них – це спосіб реалізації, за ним мови програмування поділяють на компільовані та інтерпретовані. Для цієї курсової роботи важливий саме цей розподіл, адже ми детальніше заглибемося в мови програмування, що інтерпретуються, а саме в мову BASIC. Розглянемо мету створення цієї мови, її основні команди та можливості.

Метою курсової роботи є розробка інтерпретатора для мови програмування BASIC.

Для досягнення мети необхідно виконати такі завдання:

- Дослідити інформацію про інтерпретуємі мови програмування.
- Розглянути доступні засоби розробки програми інтерпретатора.
- Розробити алгоритм розробки програми.
- Розробити програму обраною мовою програмування.
- Провести тестування працездатності реалізованої програми.

Об'єктом дослідження є найпростіші мови програмування та інтерпретатори.

Предметом дослідження є розробка інтерпретатора для певної мови програмування використовуючи Java.

Використане програмне забезпечення - IntelliJ IDEA.

Логіка дослідження зумовила таку структуру курсової роботи: вступ, два розділи, висновок, список використаних джерел та додатки.

Перший розділ присвячено теоретичній частині.

В другому розділі описаний аналіз технічного завдання, з визначенням основних функціональних можливостей, якими повинен бути наділений застосунок, процес розробки інтерпретатора. Окрім цього, описані вибрані способи та засоби реалізації застосунку. Також описані результати тестування розробленого інтерпретатора.

РОЗДІЛ 1: Теоретичні відомості. Визначення завдання курсової роботи

1.1 Розподіл мов програмування за способом реалізації

Мови програмування класифікують за такими критеріями:

- рівень абстракції (низькорівневі мови програмування та мови програмування високого рівня);
- область застосування (універсальні чи спеціалізовані);
- підтримувані парадигми програмування;
- спосіб реалізації мови (компільовані та інтерпретовані мови програмування).

В цій курсовій роботі нам цікавий саме останній критерій – спосіб реалізації мови. За цим критерієм вони поділяються на компільовані та інтерпретовані. Обидва типи мов мають свої сильні та слабкі сторони. Зазвичай рішення про використання інтерпретованої мови ґрунтується на часових обмеженнях розробки або для полегшення майбутніх змін програми. Терміни компільованих та інтерпретованих мов не надто чітко визначені, оскільки, теоретично, будь-яку мову програмування можна інтерпретувати або компілювати. У реалізаціях сучасних мов програмування все більш популярним стає підтримка обох варіантів. Але при цьому, для багатьох мов існує відмінність у продуктивності між компільованою та інтерпретованою реалізацією.

Інтерпретована мова - це тип мови програмування, для якої більшість її реалізації виконують інструкції напряму та вільно, без попереднього компілювання програми в інструкції машинною мовою. Інтерпретатор буде проходити через програму рядок за рядком і виконувати кожну команду. Приклади деяких поширених мов, що інтерпретуються, включають BASIC, Ruby, JavaScript та Python.

Інтерпретатори приймають в якості вхідних даних абстрактне представлення вхідної програми, та оцінюють стосовно додаткових вхідних даних. Вивід інтерпретатора - це результат програми, яка виконується.

Переваги інтерпретованих мов програмування:

- прості у вивченні та використанні;
- потрібно мінімум знань та досвіду у програмуванні;
- дозволяють складні завдання виконувати у відносно невелике число кроків;
- просте створення та редагування програми в різних текстових редакторах;
- дозволяють додавати динамічні та інтерактивні дії на веб-сторінках;
- швидкий запуск та редагування коду.

Недоліки інтерпретованих мов програмування:

- як правило, працюють досить повільно;
- обмежені команди для виконання детальних операцій.

Компілятор - це програма, яка перетворює читабельний для людини код в інструкції які розуміє та зчитує комп'ютер – цей процес, відбувається лише один раз за час виконання цього коду. Спочатку це займає більше часу, тому що компілятору потрібно спочатку переставити, оптимізувати або "скомпілювати" об'єктний код. Приклади чисто компільованих мов включають C, C++, Erlang, Haskell, Rust та Go.

Переваги компільованих мов програмування:

- швидке виконання;
- оптимізовані для цільового обладнання.

Недоліки компільованих мов програмування:

- вимагають застосування компілятора;
- редагування та деплой коду відбувається набагато повільніше, ніж у інтерпретованих мов програмування.

1.2 Інтерпретатори

Інтерпретатор - це комп'ютерна програма, яка безпосередньо виконує інструкції або сценарії, написані певною мовою програмування, не вимагаючи того, щоб вони попередньо були переведені в машинну мову.

Інтерпретатор, як правило, використовує одну з наступних стратегій для виконання програми:

- Розбиття вхідного коду і його безпосереднє виконання;
- Переклад вхідного коду у певне ефективне проміжне подання та його виконання;
- Виконання попередньо збереженого попередньо скомпільованого коду, створеного компілятором, який є частиною системи інтерпретатора.

Ранні версії таких мов програмування як Lisp та Dartmouth BASIC будуть прикладами першого типу. Perl, Python, MATLAB та Ruby - приклади другого, тоді як UCSD Pascal - приклад третього типу. Деякі системи, такі як Smalltalk та сучасні версії BASIC та Java, можуть також поєднувати два або три типи. Інтерпретатори різних типів також були побудовані для багатьох мов, які традиційно пов'язують зі компіляцією, наприклад, таких як Algol, Fortran, Cobol, C і C ++.

Хоча інтерпретація та компіляція є двома основними засобами, за допомогою яких реалізуються мови програмування, вони не є взаємовиключними, оскільки більшість інтерпретаційних систем також виконують певну роботу з «перекладу», як і компілятори. Мова програмування високого рівня в ідеалі є абстракцією, незалежною від конкретних реалізацій.

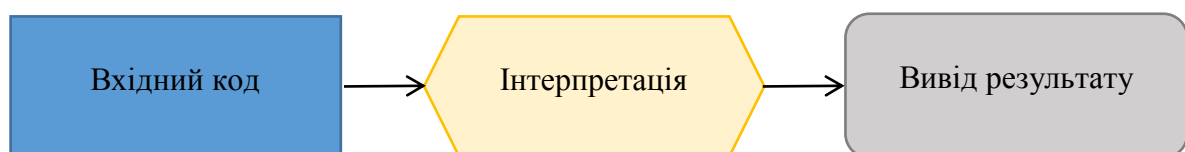


Рисунок 1.1 Схема роботи інтерпретатора

Різновиди інтерпретаторів:

1. Інтерпретатори байт-коду
2. Інтерпретатори потокового коду
3. Інтерпретатори абстрактних синтаксичних дерев
4. JIT-компіляція
5. Само-інтерпретатор (наприклад, інтерпретатор BASIC написаний на BASIC)

1.3 Мова програмування BASIC

Мова програмування BASIC була розроблена в 1963 році в Дартмутському коледжі в Ганновері, Нью-Гемпшир, як мову для навчання. Акронім BASIC (Beginner's All-purpose Symbolic Instruction Code) - це універсальний код символічних інструкцій для початківців. Автори - Джон Г. Кемені та Томас Е. Курц.

Передумовою до появи BASIC стало підвищення доступності комп'ютерів в 1960-х роках. До комп'ютерів отримали доступ учні та фахівці, які не були підготовленими програмістами, але потребували вирішення на комп'ютерах своїх завдань.

При проектуванні мови використовувалися наступні вісім принципів. Нова мова повинна була:

- бути простою у використанні для початківців;
- бути мовою програмування загального призначення;
- надавати можливість розширення функціональності, доступну досвідченим програмістам;
- бути інтерактивним;
- надавати ясні повідомлення про помилки;
- швидко працювати на невеликих програмах;
- не вимагати розуміння роботи апаратного забезпечення;
- захищати користувача від операційної системи.

Синтаксис мови нагадує Фортран і багато елементів - явно запозиченні з нього. Мова замислювалася для навчання, тому її конструкції максимально прості. Як і в інших мовах програмування, ключові слова взяті з англійської мови. Основних типів даних всього два: рядки та числа. З появою версії Visual Basic, а також різних його модифікацій (таких як VBA), в мові з'явилося багато інших типів даних і доповнень, типових для сучасних мов програмування. Оголошення змінних не вимагає спеціальної секції (на відміну від Паскаля). І власне оголошення змінної - це перше її використання.

Зовнішній вигляд програм на ранніх версіях Бейсика багато в чому визначався тим, що він призначався для середовища програмування зі рядковим редактором тексту. В такому редакторі користувач не мав можливості відображати весь текст на екрані (у вікні), переміщатися по ньому в будь-яких напрямках за допомогою клавіатури і / або миші.

В BASIC кожен рядок починається з номера. Щоб створити новий рядок потрібно було дописати рядок з номером, що знаходяться в діапазоні між номерами двох інших рядків. Для спрощення оперативного редагування програми склалася практика нумерувати рядки з кроком 10 - це дозволяло практично у всіх випадках вставляти нові рядки без зміни номерів раніше введених. Наприклад, щоб додати третій рядок між рядками 10 і 20, потрібно було написати рядок команди з номером в діапазоні від 11 до 19.

```

10 LET a = 2
20 LET b = 10
30 LET c = 5
40 LET d = b*b-4*a*c
50 IF d>0 THEN GOTO 100
60 IF d=0 THEN GOTO 200
70 PRINT "NO ROOTS"
80 GOTO 300
100 LET x1 = (-b + SQR(d))/2/a
110 LET x2 = (-b - SQR(d))/2/a
120 PRINT "x1=";
130 PRINT x1
140 PRINT "x2=";
150 PRINT x2
160 GOTO 300
200 PRINT "x=";
210 PRINT -b/2/a
300 PRINT "PROGRAM FINISHED"

```

Рисунок 1.2 Приклад програми мовою програмування BASIC

1.4 Опис основних програмних конструкцій та команд BASIC

Вирази BASIC схожі з більшістю інших процедурних мов програмування, але в перших варіантах - їх набір був досить бідним. У самих ранніх варіантах кількість ключових слів не досягала навіть 20.

А таким був набір основних програмних конструкцій і інтерактивних команд, які були реалізовані практично в будь-якій BASIC-системі кінця 1970-х років:

- LIST - Вивід на монітор тексту програми в правильній послідовності рядків.
- RUN - Запуск поточної програми з рядка оператор якого має найменше значення.
- REM - Коментар, текст, який перебував після цього ключового слова і до кінця рядка, на виконання програми не впливав.

- INPUT "Запрошення: ", Змінна - Вивід на монітор тексту "Запрошення:" і очікування, поки користувач не введе значення змінної і не натисне Enter. Після цього в змінну запишеться введене значення.

- PRINT - Вивід на монітор сказаного об'єкту (тексту або значення змінних).

- CLS - Очищення екрану терміналу.

- LET - Присвоєння (у форматі «LET змінна = значення»).

- DIM - Опис масиву.

- GOTO номер рядка - Команда переходу на рядок з сказаним номером.

- IF ... THEN GOTO ... - Оператор розгалуження. Після IF розміщується логічна умова, після THEN GOTO – оператор рядка на який повинен бути виконаний перехід при істинності цієї умови.

- FOR ЗміннаЦиклу=Початкове значення TO Кінцеве значення STEP крок ітерації - Ініціює цикл, в якому змінна послідовно проходить ряд значень від початкового значення до кінцевого значення з певним кроком. Тіло циклу обмежується заголовком і ключовим словом NEXT. Коли змінна циклу виходить за вказані межі, відбувається перехід за відповідний оператор NEXT.

- NEXT - Завершення тіла циклу FOR.

- WHILE умова - Заголовок циклу з передумовою. Цикл завершується, якщо умова виявляється помилковою. Після цього відбувається перехід на відповідний заголовку оператор WEND.

- WEND – Завершення тіла циклу WHILE.

- GOSUB мітка – Перехід до підпрограми, яка починається з вказаної мітки. Повернення відбувається при досягненні оператора RETURN.

- RETURN - Оператор повернення з підпрограми. Виконується перехід на оператор який є наступним після команди GOSUB.

- END - Завершення виконання програми. Система виходить в інтерпретатор і користувачеві видається запрошення на введення нових команд.

- CHAIN - Завантаження діапазону рядків програми з зовнішнього носія.

- OPEN – Відкриття файлу даних на зовнішньому носії.
- CLOSE - Закриття файлу даних на зовнішньому носії.
- GET - Послідовне читання значень зазначених змінних з файлу, з переміщенням файлового покажчика за останній зчитаний символ.
- PUT - Послідовний запис значень зазначених змінних в файл з переміщенням файлового покажчика за останній записаний символ.

Команди інтерактивного режиму:

- DELETE – видалення рядка з зазначеним номером.
- SAVE - збереження поточної програми в файлі на зовнішньому носії.
- LOAD - завантаження в пам'ять програми з файлу на зовнішньому носії.

Як правило, за замовчуванням, це призводило до знищення вже існуючих даних та програми.

- RENUM - перенумерація всіх, або заданого діапазону, рядків програми, починаючи з заданого числа та з вказаним кроком (за замовчуванням – крок 10).

Математичні функції, розрахунки яких підтримує BASIC:

- ABS - Абсолютне значення.
- ATN – Арктангенс.
- COS – Косинус.
- EXP - Піднесення до степеня.
- INT - Ціла частина.
- LOG - Натуральний логарифм.
- RND - Генерація випадкового числа.
- SIN - Синус.
- SQR - Квадратний корінь.
- TAN - Тангенс.

1.5 Визначення завдання

Для досягнення мети курсової роботи необхідно розробити інтерпретатор для мови програмування BASIC, виконавши такі завдання:

- проаналізувати доступну в Інтернеті інформацію щодо команд BASIC задля визначення базових функціональних можливостей, якими має володіти інтерпретатор;
- дослідити існуючі засоби розробки інтерпретатора;
- розробити структуру класів та інтерфейс застосунку;
- розробити застосунок, який інтерпретує та виконує програму написану на мові програмування BASIC;
- провести тестування інтерпретатора.

РОЗДІЛ 2: Опис реалізації програмного продукту

2.1 Аналіз технічного завдання

Основне завдання – розробити інтерпретатор для найпростішої мови програмування, а саме мови програмування BASIC.

У розробленій програмі мають бути реалізовані такі функції:

- Інтерпретація основних команд мови BASIC, а саме:
 - RUN;
 - LIST;
 - LET;
 - GOTO;
 - IF THEN GOTO;
 - FOR TO STEP;
 - PRINT;
 - REM.
- Виконання основних математичних обчислень, а саме: додавання, віднімання, ділення, множення, повернення абсолютного значення, обрахунок синусу, косинусу, тангенсу, арктангенсу, квадратного кореня, піднесення е до степеня, повернення цілої частини змінної, обрахунок натурального логарифма.
- Порядок виконання математичних операцій може регулюватися дужками ().
- Виконання функції генерації випадкового числа.
- Вивід результату роботи програми для користувача.
- Для спрощення роботи – реалізувати функцію введення програми з файлу та збереження програми в файл.
- Реалізувати функцію вставки основних команд мови програмування BASIC.

- Виводити помилку користувачеві, якщо певні команди були введені не коректно за синтаксисом.

З огляду на визначені функції та вимоги програми можна скласти орієнтовний план роботи:

- Розробка структури, визначення основних функцій майбутньої програми.
- Програмування користувацького інтерфейсу для інтерпретатора.
- Розробка інтерпретатора.
- Тестування працездатності та коректного виконання програми.

2.2 Визначення граматики для інтерпретатора

Перед розробкою самої програми нам потрібно визначити список допустимих лексем для інтерпретатора. Це спростить процес розробки, оскільки ми матимемо чіткий список, що ми маємо реалізувати.

Допустимими є:

- зарезервовані слова (команди) <зарезервовані слова> = < RUN, LIST, LET, GOTO, IF, THEN, FOR, TO, STEP, NEXT, PRINT, REM, ATN, COS, EXP, INT, LOG, RND, SIN, SQR, TAN>

- ім'я змінної <ім'я змінної> = l <c>; (l – маленька буква, C - цифра)

- використовувані символи <символи> = < '+', '-', '/', '*', '(', ')', '!'>

- цифри

Інструкції в тексті програми розділяються переходом на новий рядок. Кожна інструкція починається з номеру стрічки, тобто порядку виконання цього рядка в програмі.

2.3 Обґрунтування вибору засобів розробки

Редагування коду можна здійснити без будь-яких спеціальних інструментів. Тобто, достатньо простого текстового редактора. Але наявність

відповідних інструментів розробки, не тільки полегшує роботу, а й може підвищити її якісний рівень. Сьогодні, програм, призначених для написання і редагування коду веб-застосунків, є величезна кількість. Умовно, їх можна поділити на три категорії:

- Редактори коду
- Багатофункціональні інтегровані середовища розробки (IDE)
- Хмарні IDE (функціональність десктопних середовищ для веб-розробки в вигляді веб-сервісу)

Для реалізації застосунку було обрано IntelliJ IDEA. Оскільки її застосування значно спрощує написання коду та його відлагодження.

Для розробки програми було обрано мову програмування – Java.

Переваги Java:

- Об'єктивно-орієнтована мова: у Java все є об'єктом.
- Платформонезалежна: на відміну від багатьох інших мов, включаючи C і C ++, Java, компілюється не у платформі конкретної машини, а в незалежний байт-код.
- Проста: процеси вивчення і введення в мову програмування Java залишається простими.
- Безпечна: методи перевірки правдивості засновані на шифруванні з відкритим ключем.
- Архітектурно-нейтральна: компілятор генерує архітектурно-нейтральні об'єкти формату файлу, що робить скомпільований код доступним для виконання на багатьох процесорах, при наявності систем Java Runtime.
- Багатопотокова: функції багатопоточності, можна писати програми, які можуть виконувати безліч завдань одночасно.
- Інтерпретована: Java байт-код переводить на льоту в машинні інструкції і ніде не зберігається.
- Високопродуктивна: введений Just-In-Time компілятор, що забезпечує високу продуктивність.

- Розподілена.
- Динамічна: програмування на Java вважається більш динамічним, ніж на C або C++, оскільки вона призначена для адаптації до змін умов.

2.4 Опис програми та процесу розробки

Виходячи з аналізу технічного завдання та визначення граматики - я визначила класи та методи які необхідно реалізувати – створила каркас майбутньої програми - рисунок 2.1.

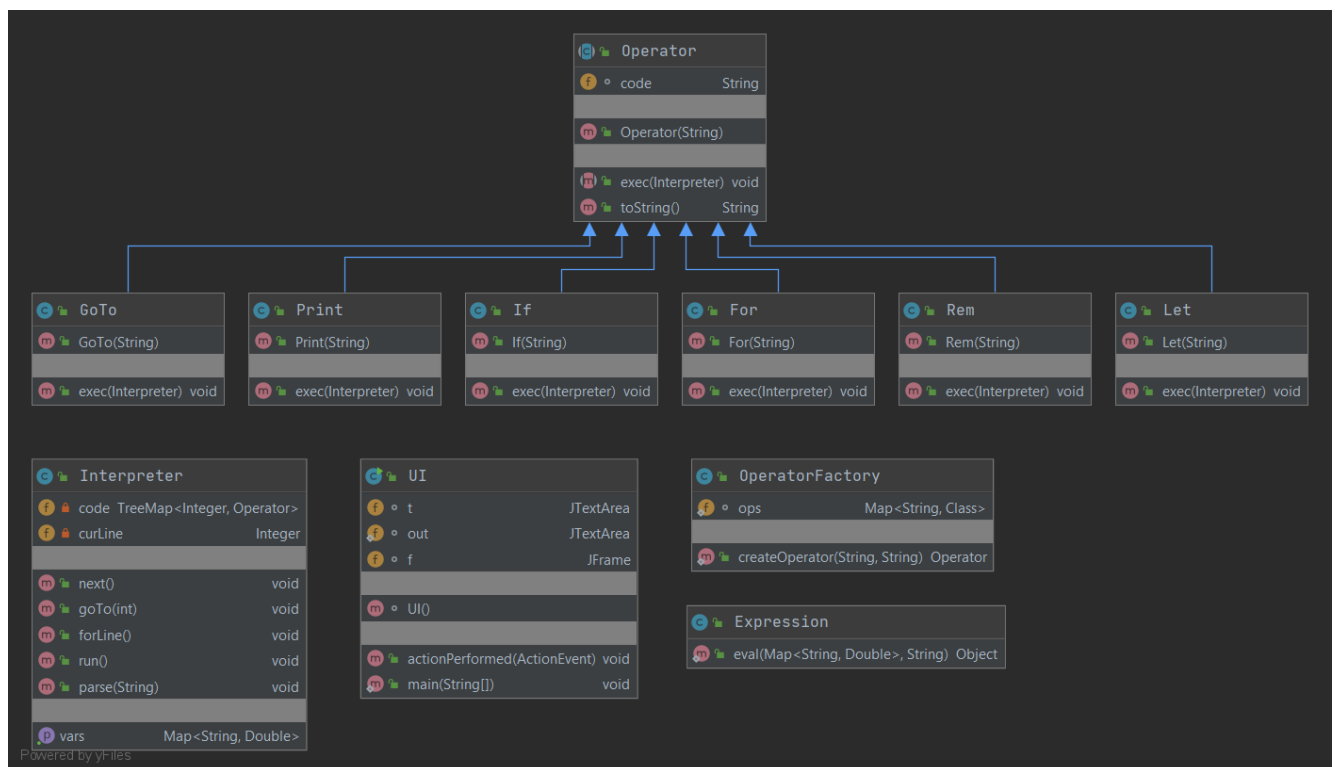


Рисунок 2.1 Блок-схема класів програми

Для операторів мови BASIC було створено основний клас Operator, його наслідниками будуть класи кожного оператора. Класи операторів були названі за назвою самої команди, це спрощує розуміння, читабельність та розробку програми. В середині класів була реалізована поведінка цих операторів в мові BASIC.

Клас UI містить основний метод та реалізацію інтерфейсу користувача. Для реалізації інтерфейсу використано Java Swing.

Клас `Interpreter` відповідає за створення самого інтерпретатора, збереження вхідної програми, розбиття її на лексеми та виконання. Також в ньому реалізовані деякі базові оператори вхідної мови програмування.

Для збереження даних використано `TreeMap` та `HashMap`.

В `TreeMap` після розпарсування записуємо пару `<Integer, Operator>` - де `Integer` – номер рядка програми, а `Operator` – власне оператор з данимим на виконання. `TreeMap` було обрано оскільки нам важливий порядок, а ця структура це забезпечує та не має бути повторів однакових номер рядків.

Проаналізувавши структуру застосунку, а також алгоритми створення інтерпретаторів весь процес розробки основної програми можна поділити на:

- Розробка структури класів.
- Розробка користувацького інтерфейсу для взаємодії користувача з програмою не через командний рядок.
- Розробка основи інтерпретатора.
- Покрокова реалізація всіх можливих команд вхідної мови.
- З'єднання користувацького інтерфейсу з вже працюючим інтерпретатором.

2.6 Опис інтерфейсу програми

Інтерфейс програми складається з форми, також її можна вважати діалоговим вікном між користувачем і внутрішньою програмою, яка виконує інтерпретацію введеної користувачем програми мовою BASIC. На формі розташовані Меню та дві текстові області – одна з них доступна тільки на читання, адже вона призначення для виводу результату виконання інтерпретованої програми.

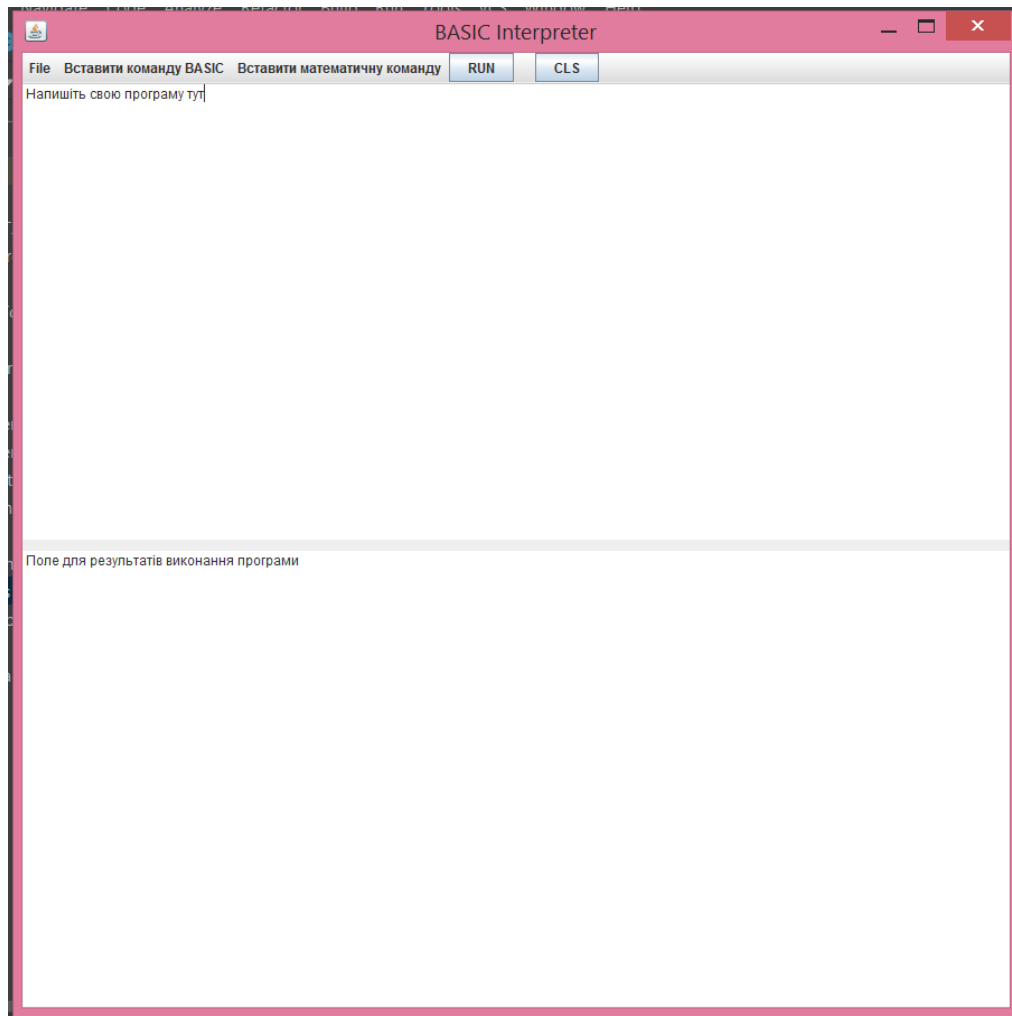


Рисунок 2.2 Вигляд користувацького інтерфейсу при старті програми

Основне меню складається з таких пунктів:

- File
- Вставити команду BASIC
- Вставити математичну команду

- RUN
- CLS

Меню «File» дозволяє працювати з файлами – завантажувати програму з файлу та в файл.

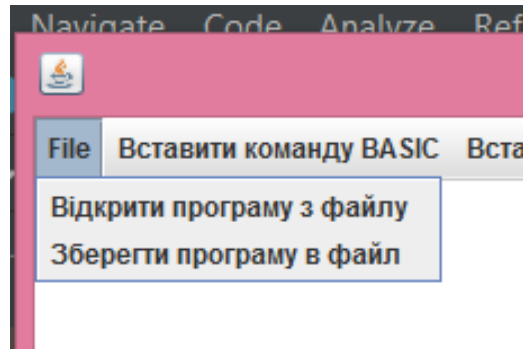


Рисунок 2.3 Меню File

При виборі файлу для зчитування або запису з’являється діалогове вікно вибору файлу.

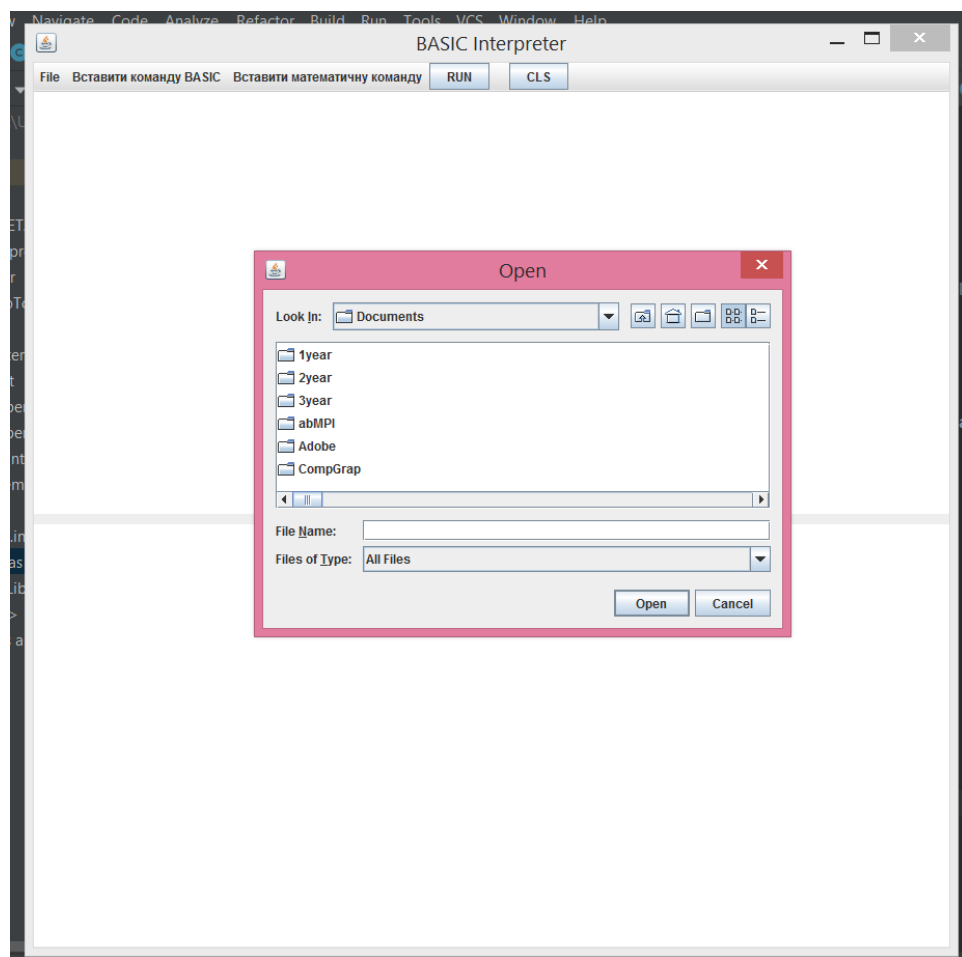


Рисунок 2.4 Діалогове вікно вибору файлу

Меню «Вставити команду BASIC» дозволяє вставляти в текстову область базові команди BASIC.

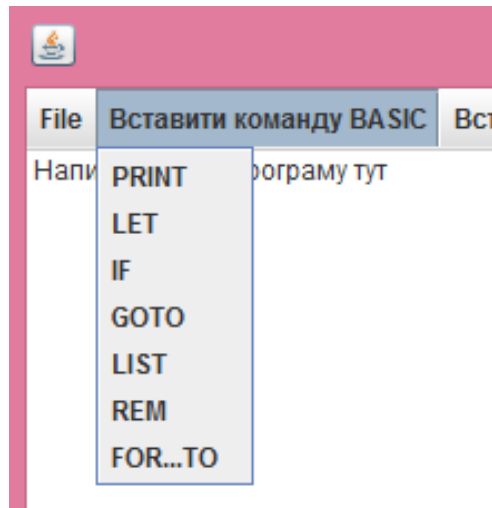


Рисунок 2.5 Меню Вставити команду BASIC

Меню «Вставити математичну команду» дозволяє вставляти в текстову область базові математичні команди, які доступні для виконання в BASIC.

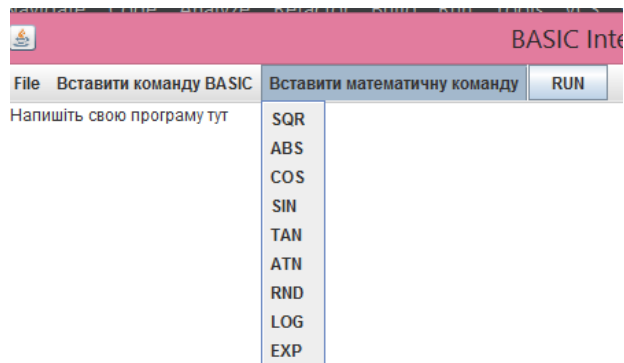


Рисунок 2.6 Меню Вставити математичну команду

Ці два меню спрощують кінцевому користувачу написання програми на BASIC для інтерпретатора.

Кнопка RUN запускає виконання інтерпретації програми написаної користувачем в першій текстовій області. Кнопка CLS очищує обидві текстові області.

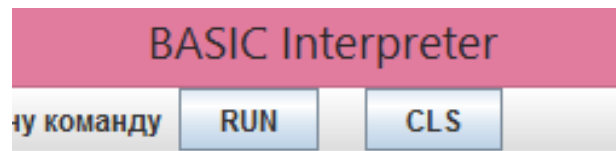


Рисунок 2.7 Кнопки меню RUN та CLS

Результатом обробки тексту програми є вивід інтерпретатором результату виконання або повідомлення про помилку в другу текстову область.

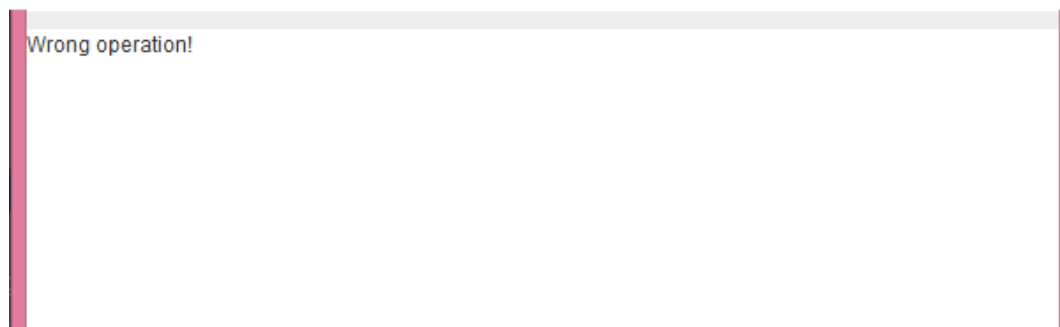


Рисунок 2.8 Приклад виводу помилки користувачу про неправильний оператор

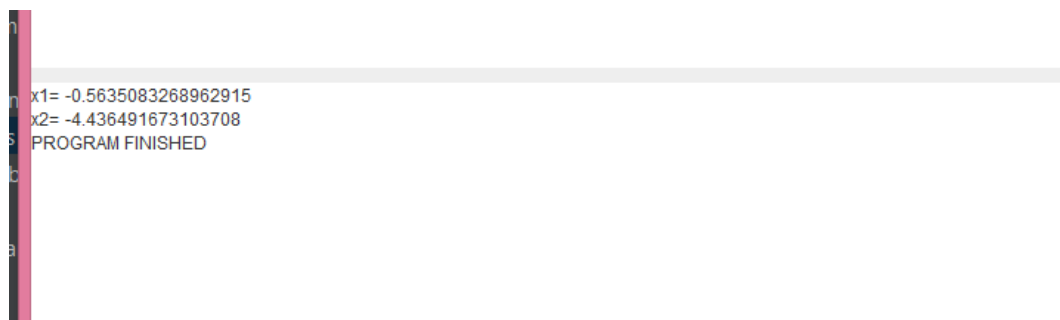


Рисунок 2.9 Приклад виводу результату успішно інтерпретованої програми

2.8 Тестування програми і результати її виконання

Отже, після розробки програми було проведення її тестування. Відбулася перевірка працездатності інтерпретатора. В ході тестування було виявлено і виправлено певні помилки, наприклад, було виявлена некоректна реалізація функції вставлення назв команд з меню. Спочатку

команда завжди дозаписувалася в кінець програми. Після виправлення цієї помилки – команди можна додавати в будь-яке місце програми, де розташований курсор. Можна стверджувати, що програма працює коректно, зрозуміла у користуванні. Швидкість інтерпретації команд висока та результати розрахунків правильні.

Отримані результати свідчать про те, що програма готова до користування.

На Рисунку 2.10 зображено приклад введеної користувачем програми в першу текстову область та результат її виконання в другій текстовій області після натискання на кнопку RUN в меню згори.

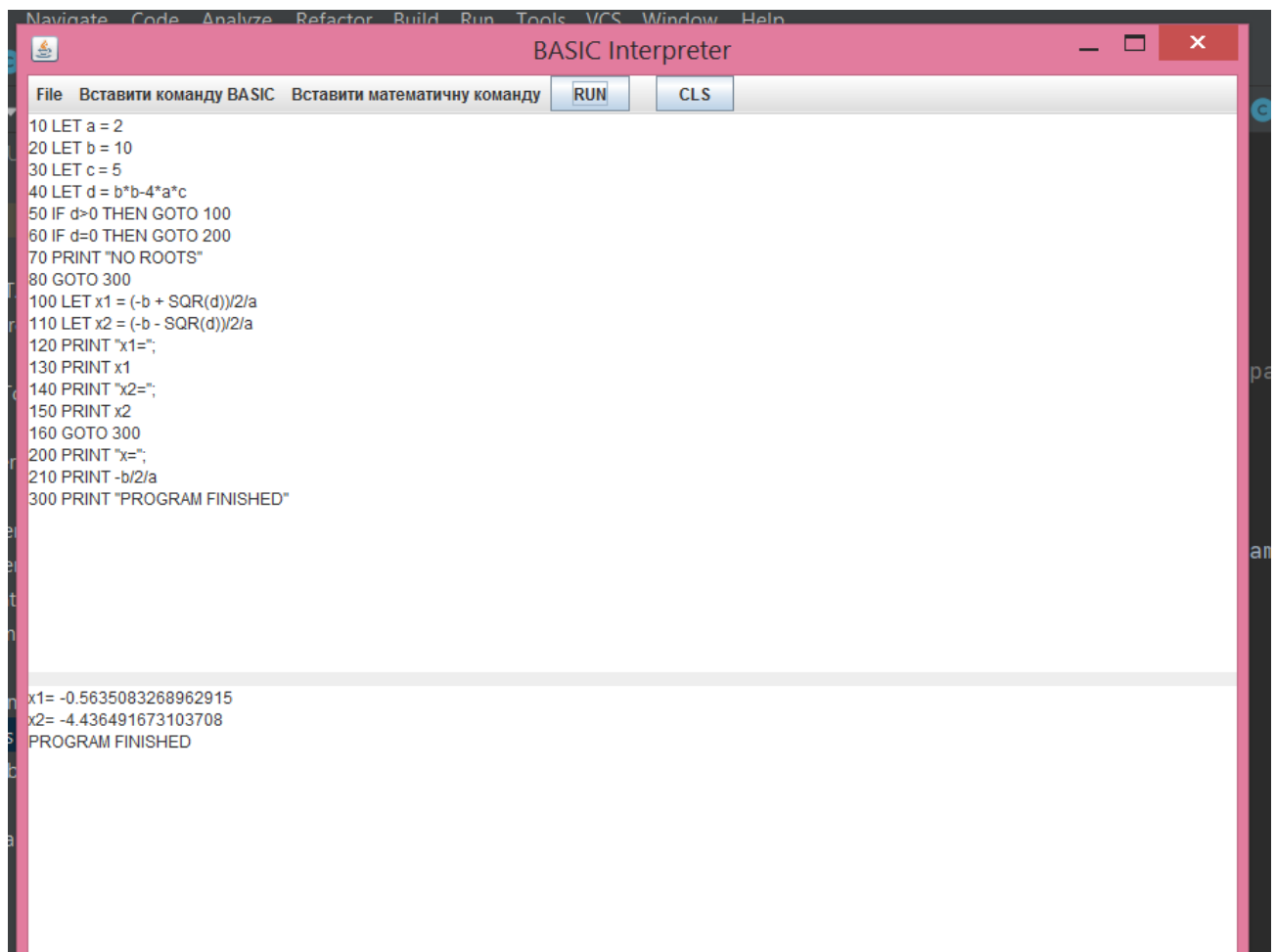


Рисунок 2.10 Приклад роботи програми

ВИСНОВКИ

В результаті написання курсової роботи було досліджено розподіл мов програмування за способом реалізації, як реалізують інтерпретатори для різноманітних мов програмування. Окрім цього було вивчено мову програмування BASIC, оскільки ці знання були потрібні для реалізації інтерпретатора.

Під час розробки інтерпретатора найпростішої мови програмування, а саме для BASIC, було розроблено алгоритм програми, враховані обмеження вхідної мови програмування, реалізовані додаткові функції для спрощення користування програмою зі сторони користувача. Для реалізації була використана мова Java. Також проведено тестування та виявлено повну працездатність розробленого застосунку.

Список літератури

1. Вікіпедія – Мови програмування [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Мови_програмування
2. Programming Languages: Classification, Execution Model, and Errors [Електронний ресурс] – Режим доступу: http://www.uobabylon.edu.iq/eprints/publication_11_23917_6270.pdf
3. Вікіпедія – Інтерпретована мова програмування [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Інтерпретована_мова_програмування
4. Free Code Camp - Compiled Versus Interpreted Languages [Електронний ресурс] – Режим доступу: <https://guide.freecodecamp.org/computer-science/compiled-versus-interpreted-languages/>
5. Вікіпедія – BASIC Programming [Електронний ресурс] – Режим доступу: https://en.wikibooks.org/wiki/BASIC_Programming
6. Менеджеры расположения Layout [Електронний ресурс] – Режим доступу: <http://java-online.ru/swing-layout.shtml>
7. [Електронний ресурс] – Режим доступу: https://www.ntu.edu.sg/home/ehchua/programming/java/J4a_GUI.html
8. Java Programming Tutorial - Programming Graphical User Interface [Електронний ресурс] – Режим доступу: <https://www.guru99.com/difference-compiler-vs-interpreter.html>
9. JAVAWORLD - How to build an interpreter in Java, Part 1: The BASICs [Електронний ресурс] – Режим доступу: <https://www.javaworld.com/article/2076921/how-to-build-an-interpreter-in-java--part-1--the-basics.html>
10. BASIC Commands - [Електронний ресурс] – Режим доступу: <http://www.picaxe.com/BASIC-Commands/Program-Flow-Control/>

Додаток А

Програмний код інтерпретатора

```

import javax.swing.*;
import java.awt.*;
import java.io.*;
import java.awt.event.*;
import javax.swing.plaf.metal.*;
import javax.swing.text.*;

public class UI extends JFrame implements ActionListener {

    JTextArea t;
    static JTextArea out;
    JFrame f;

    // Конструктор
    UI()
    {
        // Створюємо frame
        f = new JFrame("BASIC Interpreter");
        try {
            // Встановити зовнішній вигляд metl
            UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");
            // Задаємо тему океану
            MetalLookAndFeel.setCurrentTheme(new OceanTheme());
        }
        catch (Exception e) {
        }
        // Text component
        t = new JTextArea(" Напишіть свою програму тут");
        out = new JTextArea(" Поле для результатів виконання програми");
        // Створюємо menubar
        JMenuBar mb = new JMenuBar();
        // Створюємо підменю для меню
        JMenu m1 = new JMenu("File");

        // Створюємо елементи меню
        JMenuItem mi1 = new JMenuItem("Відкрити програму з файлу");
        JMenuItem mi2 = new JMenuItem("Зберегти програму в файл");

        // Додаємо слухачі подій
        mi1.addActionListener(this);
        mi2.addActionListener(this);

        m1.add(mi1);
        m1.add(mi2);

        // Створюємо підменю для меню
        JMenu m2 = new JMenu("Вставити команду BASIC");

        // Create menu items
        JMenuItem mi3 = new JMenuItem("PRINT");
        JMenuItem mi4 = new JMenuItem("LET");
        JMenuItem mi5 = new JMenuItem("IF");
    }
}

```

```

JMenuItem mi6 = new JMenuItem("GOTO");
JMenuItem mi7 = new JMenuItem("LIST");
JMenuItem mi8 = new JMenuItem("REM");
JMenuItem mi9 = new JMenuItem("FOR...TO");

// Add action listener
mi3.addActionListener(this);
mi4.addActionListener(this);
mi5.addActionListener(this);
mi6.addActionListener(this);
mi7.addActionListener(this);
mi8.addActionListener(this);
mi9.addActionListener(this);

m2.add(mi3);
m2.add(mi4);
m2.add(mi5);
m2.add(mi6);
m2.add(mi7);
m2.add(mi8);
m2.add(mi9);
// Створюємо підменю для меню
JMenu m3 = new JMenu("Вставити математичну команду");

// Create menu items
JMenuItem ma1 = new JMenuItem("SQR");
JMenuItem ma2 = new JMenuItem("ABS");
JMenuItem ma3 = new JMenuItem("COS");
JMenuItem ma4 = new JMenuItem("SIN");
JMenuItem ma5 = new JMenuItem("TAN");
JMenuItem ma6 = new JMenuItem("ATN");
JMenuItem ma7 = new JMenuItem("RND");
JMenuItem ma8 = new JMenuItem("LOG");
JMenuItem ma9 = new JMenuItem("EXP");

// Add action listener
ma1.addActionListener(this);
ma2.addActionListener(this);
ma3.addActionListener(this);
ma4.addActionListener(this);
ma5.addActionListener(this);
ma6.addActionListener(this);
ma7.addActionListener(this);
ma8.addActionListener(this);
ma9.addActionListener(this);

m3.add(ma1);
m3.add(ma2);
m3.add(ma3);
m3.add(ma4);
m3.add(ma5);
m3.add(ma6);
m3.add(ma7);
m3.add(ma8);
m3.add(ma9);

```

```

JButton mc = new JButton("CLS");
mc.addActionListener(this);

JButton pr = new JButton("RUN");
pr.addActionListener(this);

JMenu madd1 = new JMenu(" ");
madd1.setEnabled(false);

mb.add(m1);
mb.add(m2);
mb.add(m3);
mb.add(pr);
mb.add(madd1);
mb.add(mc);
f.setJMenuBar(mb);

JLabel jl = new JLabel("Результат виконання");
GridLayout experimentLayout = new GridLayout(0,1,10,10);
f.setLayout(experimentLayout);
f.add(t);
out.setEditable(false);
f.add(out);
f.setSize(900, 900);
f.show();
}

// If a button is pressed
public void actionPerformed(ActionEvent e)
{
    String s = e.getActionCommand();

    if (s.equals("PRINT"))
        t.insert("PRINT", t.getCaretPosition());
    else if (s.equals("LET"))
        t.insert("LET", t.getCaretPosition());
    else if (s.equals("IF"))
        t.insert("IF", t.getCaretPosition());
    else if (s.equals("GOTO"))
        t.insert("GOTO", t.getCaretPosition());
    else if (s.equals("LIST"))
        t.insert("LIST", t.getCaretPosition());
    else if (s.equals("REM"))
        t.insert("REM", t.getCaretPosition());
    else if (s.equals("FOR...TO"))
        t.insert("FOR TO STEP ", t.getCaretPosition());
    else if (s.equals("SQR"))
        t.insert("SQR(_)", t.getCaretPosition());
    else if (s.equals("ABS"))
        t.insert("ABS(_)", t.getCaretPosition());
    else if (s.equals("COS"))
        t.insert("COS(_)", t.getCaretPosition());
    else if (s.equals("SIN"))
        t.insert("SIN(_)", t.getCaretPosition());
    else if (s.equals("TAN"))
        t.insert("TAN(_)", t.getCaretPosition());
}

```

```

else if (s.equals("ATN"))
    t.insert("ATN(_)", t.getCaretPosition());
else if (s.equals("RND"))
    t.insert("RND", t.getCaretPosition());
else if (s.equals("LOG"))
    t.insert("LOG(_)", t.getCaretPosition());
else if (s.equals("EXP"))
    t.insert("EXP(_)", t.getCaretPosition());
else if (s.equals("RUN")) {
    out.setText("");
    String[] st = t.getText().split("\\n");
    Interpreter interpreter = new Interpreter();
    while(true) {
        int i = 0;
        for(i = 0; i < st.length; i++)
            //передаємо стрічку на розбиття
            interpreter.parse(st[i]);
        if(i == st.length) break;
    }
    interpreter.run();
}
else if (s.equals("Зберегти програму в файл")) {

    JFileChooser j = new JFileChooser("f:");
    int r = j.showSaveDialog(null);

    if (r == JFileChooser.APPROVE_OPTION) {
        // Set the label to the path of the selected directory
        File fi = new File(j.getSelectedFile().getAbsolutePath());
        try {
            FileWriter wr = new FileWriter(fi, false);
            BufferedWriter w = new BufferedWriter(wr);
            w.write(t.getText());
            w.flush();
            w.close();
        }
        catch (Exception evt) {
            JOptionPane.showMessageDialog(f, evt.getMessage());
        }
    }
    // If the user cancelled the operation
    else
        JOptionPane.showMessageDialog(f, "Користувач скасував операцію");
}
else if (s.equals("Відкрити програму з файлу")) {

    JFileChooser j = new JFileChooser("f:");
    int r = j.showOpenDialog(null);
    // If the user selects a file
    if (r == JFileChooser.APPROVE_OPTION) {
        // Set the label to the path of the selected directory
        File fi = new File(j.getSelectedFile().getAbsolutePath());
        try {
            String s1 = "", sl = "";
            FileReader fr = new FileReader(fi);
            BufferedReader br = new BufferedReader(fr);

```

```

        sl = br.readLine();
        // Take the input from the file
        while ((s1 = br.readLine()) != null) {
            sl = sl + "\n" + s1;
        }
        t.setText(sl);
        out.setText("");
    }
    catch (Exception evt) {
        JOptionPane.showMessageDialog(f, evt.getMessage());
    }
}
// If the user cancelled the operation
else
    JOptionPane.showMessageDialog(f, "Користувач скасував операцію");
}
else if (s.equals("CLS")) {
    t.setText("");
    out.setText("");
}
}

public static void main(String args[]) throws IOException {
    UI ui = new UI();
}
}

```

Лістинг коду класу UI.java

```

import java.util.HashMap;
import java.util.Map;
import java.util.TreeMap;

public class Interpreter {
    //тут зберігається програма на basic <номер рядка, рядок>
    private TreeMap<Integer, Operator> code = new TreeMap<Integer, Operator>();
    private Map<String, Double> vars = new HashMap<String, Double>();
    private Integer curLine;

    //перехі на наступний рядок програми на basic
    public void next(){
        curLine = code.higherKey(curLine);
    }

    //перехід до заданого рядка програми на basic
    public void goTo(int line){
        curLine = line;
    }

    public void forLine() {
        Operator operator = code.get(curLine);
        operator.exec(this);
    }
}

```

```

//метод який виконує програму
public void run (){
    curLine = code.firstKey();
    while(true){
        Operator operator = code.get(curLine);
        operator.exec(this);
        if(curLine==null) break;
    }
}

//Метод який розпарсує рядок програми на basic
//Приклад рядка: 20 LET x=3
public void parse(String line){
    //якщо в інтерпретатор зайшла стрічки програми з командою запуску (RUN)
    //викликаємо метод виконання нашого інтерпретатора
    if(line.equalsIgnoreCase("RUN")){
        this.run();
        return;
    }

    //якщо в інтерпретатор зайшла стрічка з командою LIST (вивід на монітор
    //тексту програми з правильною послідовністю стрічок)
    if(line.equalsIgnoreCase("LIST")){
        for(int l:code.keySet()) {
            UI.out.append(l + " " + code.get(l));
            UI.out.append("\n");
            System.out.println(l + " " + code.get(l));
        }
        return;
    }
    try {
        //розділяємо рядок програми по пробілах
        String parts[] = line.split(" ");
        //перший елемент розпаршеного рядка - номер рядка програми
        int lineNumber = Integer.parseInt(parts[0]);
        // другий елемент розпаршеного рядка - назва оператора
        String opName = parts[1];
        //записуємо залишок стрічки
        Operator operator = OperatorFactory.createOperator(opName,line.substring(parts[0].length() +
parts[1].length() + 2));
        code.put(lineNumber, operator);
    } catch (RuntimeException e){
        UI.out.append("Wrong operation!");
        UI.out.append("\n");
        System.err.println("Wrong operation!");
    }

}

public Map<String, Double> getVars() {
    return vars;
}
}

```

Лістинг коду класу Interpreter.java