Санкт-Петербургский государственный университет

Программная инженерия

Кафедра системного программирования

Группа 20Б.11-мм

Касимов Владислав Андреевич

Поддержка плагина для языка Vyper в IntelliJ Platform

Отчёт по учебной практике

Научный руководитель:
к.ф.-м.н., доцент кафедры системного программирования СПбГУ Березун Д. А.

Консультант:
Мишин Н.М.

Санкт-Петербург
2022

Оглавление

Введение **3**

1. Постановка задачи **5**

2. Обзор **6**

2.1. Обзор существующего решения 6

2.1.1. Компиляция и исполнение кода 6

2.1.2. Комплексная поддержка синтаксиса 6

2.2. Используемые технологии 7

2.2.1. IntelliJ Platform 7

2.2.2. Язык программирования 7

3. Описание решения **8**

3.1. Навигация по коду 8

3.2. Комментирование кода 8

3.3. Улучшение графического интерфейса окна вывода программы 9

3.4. Find Usages 9

3.5. Публикация плагина в JetBrains Marketplace 10

4. Результаты **11**

Список литературы **12**

Введение

Все чаще в медиапространстве можно услышать о технологии блокчейн и криптовалютах. *Блокчейн(англ. Blockchain)*[12] — выстроенная по определённым правилам цепочка блоков, содержащих информацию. Такая система является децентрализованной, то есть запись транзакций, совершенных в криптовалюте, ведется на нескольких компьютерах, соединенных в *одноранговую сеть(Peer-to-peer)*[7]. Существующая технология блокчейна активно используется не только в сфере криптовалют, но и в банковском секторе, инвестициях и биржах[12]. Каждый пользователь сети имеет доступ к реестру и неизменяемой записи транзакций. Общий реестр реализует однократную запись транзакций, исключая дублирование усилий. Изначально блокчейн ассоциировался исключительно с криптовалютой, но в настоящее время многие блокчейн-платформы имеют поддержку смарт-контрактов.

Смарт-контракт(англ. Smart contract)[3] — алгоритм, формирующий и контролирующий информацию о владении чем-либо. Смарт-контракты являются программируемыми объектами, поэтому существует возможность создания сторонних приложений для расширения области их применения. Разрабатывать смарт-контракты возможно благодаря специальным языкам программирования, в их числе находится Vyper.

Vyper[13] — это контрактный язык программирования, предназначенный для виртуальной машины Ethereum[5] и имеющий *pythonic* синтаксис. Лейтмотивом языка стало не только улучшение безопасности смарт-контрактов, но и упрощение процесса их разработки. Vyper находится в ранней версии, на момент написания работы, поэтому его поддержка не осуществлена в большинстве популярных IDE(Integrated development environment).

Учитывая растущий интерес общества к технологии блокчейна[12] и платформы Ethereum, ожидается, что Vyper станет одним из ведущих языков программирования в сфере разработки смарт-контрактов[11]. Таким образом, актуальной задачей становится создание и поддержка программного обеспечения, позволяющего удобно использовать Vyper в современных IDE.

1. Постановка задачи

Целью учебной практики является поддержка и улучшение существующего плагина[14] для языка Vyper в IDE, основанных на Intellij Platform. Плагин является результатом курсовой работы Тюрина А.В. "Комплексная поддержка синтаксиса языка Vyper в IntelliJ Platform"[16]. Для достижения данной цели были поставлены следующие задачи:

- изучить IntelliJ Platform, а также инструменты для разработки плагина
- расширить функции навигации по коду в плагине
- добавить поддержку функции комментирования и поиска использований в IDE
- опубликовать плагин в JetBrains Marketplace

2. Обзор

В данной главе рассматриваются существующие решения и инструменты плагина для поддержки языка Vyper в IntelliJ Platform. Для каждой задачи приводится обоснование выбора определенного инструмента.

2.1. Обзор существующего решения

В данной главе рассматриваются методы поддержки языка Vyper.

2.1.1. Компиляция и исполнение кода

В данный момент присутствует возможность компиляции и исполнения кода написанного на языке программирования Vyper. Для реализации используется технология Docker [1]. Таким образом, в Docker-контейнере запускается образ[15], содержащий отправленный код, компилятора языка программирования Vyper. Инструмент позволяет исполнить переданный код, а также при необходимости получить байт-код и запустить его вручную.

2.1.2. Комплексная поддержка синтаксиса

Поддержка синтаксиса является неотъемлемой задачей по поддержке языка Vyper в IntelliJ IDEA, она позволяет реализовать навигацию по коду, его подсветку, автодополнение и синтаксический анализ.

Для комплексной поддержки синтаксиса была получена грамматика языка Vyper благодаря парсеру-генератору Grammar Kit[8]. В работу инструмента входит генерация парсера на основе формы Бэкуса-Наура [4]. Таким образом, время разработки синтаксического анализатора значительно снижается, так как сгенерированные анализаторы могут быть модифицированы.

Так как синтаксический анализатор не всегда позволяет выявлять нужные ошибки, в плагине используется статистический анализатор

SmartCheck. Данный инструмент был создан для работы с виртуальной машиной Ethereum (EVM). Анализатор запускается из специального меню во время работы с IntelliJ IDEA. Для работы инструмента используется Docker-образ SmartCheck'a.

2.2. Используемые технологии

В данной главе рассматривается выбор инструментов разработки.

2.2.1. IntelliJ Platform

Intellij Platform — является одной из популярных платформ и имеет более 10 млн. пользователей, а также предоставляет 30 продуктов для разных разработчиков(по данным JetBrains на 2020 год)[1]. Более того, данная платформа предоставляет большой выбор инструментов разработки и имеет открытый исходный код.

Таким образом, плагин может быть использован в любой интегрированной среде разработки, которая основывается на IntelliJ Platform.

2.2.2. Язык программирования

Существующее решение было написано на языке программирования **Kotlin**, поэтому он остается основным для поддержки плагина.

[1] https://www.jetbrains.com/ru-ru/lp/annualreport-2020/(online; accessed: 2021-12-18)

3. Описание решения

В данной главе рассматривается решение задач по поддержке существующего плагина[14].

3.1. Навигация по коду

Быстрый переход от участков кода, где используется какая-либо сущность, к участку, где она была декларирована или с которой она логически связана.

Конечной целью являлась реализация следующих функций:

- go-to-symbol позволяет пользователю выполнить поиск по имени объекта
- go-to-declaration позволяет пользователю перейти к месту объявления объекта и узнать его тип
- go-to-implementation позволяет пользователю отслеживать реализации классов и переопределяющие методы либо с помощью специальных значков в редакторе, либо с помощью соответствующих сочетаний клавиш

Для реализации навигации по коду были определены специальные функции и поля, на которые можно будет ссылаться при поиске элементов. Неотъемлемой частью реализации является PSI(Program Structure Interface)[9], слой IntelliJ Platform, который отвечает за синтаксический анализ файлов, создание синтаксической модели кода и обеспечение многих других функций платформы. Благодаря *PSI References* реализована возможность искать PSI элемент по его имени или типу.

3.2. Комментирование кода

Комментатор(англ. Code Commenter)[2] позволяет пользователю автоматически закомментировать блок кода при наведении курсора. Ос-

новной задачей стало конфигурация комментатора для его использования в IDE.

Реализация *SimpleCommenter* зарегистрирована в файле конфигурации плагина с использованием точки расширения *lang.commenter*. Префикс комментария к строке был указан в объявлении комментатора.

Таким образом, реализовано быстрое преобразования кода в комментарий и обратно.

3.3. Улучшение графического интерфейса окна вывода программы

Так как приложение взаимодействует с окном вывода программы, то была поставлена задача автоматического изменения графического интерфейса в зависимости от текущей темы IDE, установленного шрифта и других параметров.

Языковой плагин предоставляет текстовые атрибуты по умолчанию для связанных схем *Default* и *Darcula*, а также для любой другой схемы, если её имя известно.

Благодаря расширению *com.intellij.additionalTextAttributes*, содержащему имя файла с текстовыми атрибутами, в файле *plugin.xml* удалось настроить конфигурацию графического интерфейса плагина.

Таким образом, плагин автоматически подстраивается под внешний вид IDE пользователя.

3.4. Find Usages

Когда пользователь пишет или редактирует код, он может столкнуться с элементом кода, который хочет изменить или удалить. Прежде чем вносить изменения, рекомендуется посмотреть, где используется элемент кода, и как он влияет работоспособность программы. С помощью действий Find Usages[6] пользователь может выполнять поиск ссылок на элемент кода по всему проекту.

Пользователь может управлять процессом поиска и выполнять его только в одном файле, расширить на весь проект или создать определенную область поиска. Кроме того, есть возможность настроить цвет подсветки найденных вхождений или вовсе её отключить.

Само действие *Find Usages* представляет собой многоступенчатый процесс, каждый шаг которого требует участия языкового плагина. Плагин участвует в процессе поиска способов использования, регистрируя реализацию *FindUsagesProvider* в расширении *findUsagesProvider*, благодаря PSI и интерфейсам *PsiNamedElement*, *PsiReference*. Результат сбора данных отображается пользователю на специальной панели.

3.5. Публикация плагина в JetBrains Marketplace

Для публикации в JetBrains Marketplace[10] в плагин была добавлена информация о разработчиках и их контактном адресе, а также введена история версий с описанием и другая требуемая информация. После загрузки на платформу плагин проходил проверку, где были обнаружены проблемы с совместимостью на некоторых версиях IntelliJ IDEA, о чем на электронный адрес сообщил менеджер платформы. Когда все проблемы были устранены, на сайт была выгружена актуальная, на момент написания работы, версия плагина.

Публикация в магазин была сделана для удобства установки и обновления плагина прямо из IDE, без необходимости делать это вручную с официальной страницы плагина на GitHub. Более того, после публикации появляется возможность продвигать плагин на широкую аудиторию с помощью магазина JetBrains.

4. Результаты

В рамках учебной практики были выполнены следующие задачи:

- Изучена существующая реализации плагина с целью поддержания его работоспособности в IntelliJ Platform.
- Изучены возможности IntelliJ Platform, а также инструменты для разработки плагина.
- Расширена поддержка синтаксиса языка программирования Vyper в плагине.
- Опубликована актуальная версия плагина во внутренний магазин JetBrains.[2]

Код плагина доступен в репозитории[14] на GitHub.

[2] https://plugins.jetbrains.com/plugin/19039-vyper(online; accessed: 2022-05-20)

Список литературы

[1] Anderson Charles. Docker [software engineering] // Ieee Software. — 2015. — Vol. 32, no. 3. — P. 102–c3.

[2] Code Commenter. — URL: https://plugins.jetbrains.com/docs/intellij/commenter.html#register-the-commenter(online; accessed: 2022-05-20).

[3] Daniel Macrinici Cristian Cartofeanu Shang Gao. Smart contract applications within blockchain technology: A systematic mapping study // Telematics and Informatics. — 2018. — Vol. 35, no. 8. — P. 2337–2354.

[4] Deremer Franklin L. Generating parsers for BNF grammars // Proceedings of the May 14-16, 1969, spring joint computer conference. — 1969. — P. 793–799.

[5] Ethereum Yellow Paper. — URL: https://ethereum.github.io/yellowpaper/paper.pdf(online; accessed: 2022-05-20).

[6] Find Usages. — URL: https://plugins.jetbrains.com/docs/intellij/find-usages.html(online; accessed: 2022-05-20).

[7] James Cope. What's a Peer-to-Peer (P2P) Network? — 2002. — URL: https://www.computerworld.com/article/2588287/networking-peer-to-peer-network.html (online; accessed: 2022-05-20).

[8] JetBrains Grammar-Kit. — URL: https://github.com/JetBrains/Grammar-Kit(online; accessed: 2022-05-20).

[9] Program Structure Interface. — URL: https://plugins.jetbrains.com/docs/intellij/psi.html(online; accessed: 2022-05-20).

[10] Publishing a Plugin. — URL: https://plugins.jetbrains.com/docs/intellij/publishing-plugin.html(online; accessed: 2022-05-20).

[11] Smart-contracts: Languages. — URL: https://ethereum.org/en/developers/docs/smart-contracts/languages/(online; accessed: 2022-05-20).

[12] Stephan Leible Steffen Schlager Moritz Schubotz Bela Gipp. A Review on Blockchain Technology and Blockchain Projects Fostering Open Science. — 2019. — URL: https://doi.org/10.3389/fbloc.2019.00016 (online; accessed: 2022-05-20).

[13] Vyper Docs. — URL: https://vyper.readthedocs.io/en/stable/ (online; accessed: 2022-05-20).

[14] Vyper Plugin. — URL: https://github.com/NikitaMishin/vyper-plugin(online; accessed: 2022-05-20).

[15] Vyper-container. — URL: https://hub.docker.com/search?q=vyper(online; accessed: 2022-05-20).

[16] Алексей Тюрин. Комплексная поддержка синтаксиса языка Vyper в IntelliJ Platform. — 2019. — URL: https://oops.math.spbu.ru/SE/YearlyProjects/spring-2019/371/Tyurin-report.pdf(online; accessed: 2022-05-20).