**Министр науки и высшего образования Российской̆ Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«Национальный исследовательский университет ИТМО»**

Факультет информационных технологий и программирования

Лабораторная работа № 5

*Название работы: Разработка плагина.*

**Выполнил студент группы № М3117**

Жовнир Артём Владимирович

Санкт-Петербург

2022

**IDE**

Интегрированная среда разработки (IDE) – это программное приложение, которое помогает программистам эффективно разрабатывать программный код. Оно повышает производительность разработчиков, объединяя такие возможности, как редактирование, создание, тестирование и упаковка программного обеспечения в простом для использования приложении. Так же как писатели используют текстовые редакторы, а бухгалтеры – электронные таблицы, разработчики программного обеспечения применяют IDE, чтобы упростить свою работу.

Для написания кода можно использовать любой текстовый редактор. Однако большинство интегрированных сред разработки (IDE) включают в себя функции, выходящие за рамки редактирования текста. Они предоставляют центральный интерфейс для общих инструментов разработчика, делая процесс разработки программного обеспечения гораздо более эффективным. Разработчики могут быстро приступить к программированию новых приложений вместо того, чтобы вручную интегрировать и настраивать различное программное обеспечение.

Интегрированные среды разработки (IDE) можно разделить на несколько различных категорий в зависимости от того, какую разработку приложений они поддерживают и как работают. Однако многие программные приложения IDE можно отнести к нескольким категориям. Ниже представлены некоторые типы IDE.

**Виды IDE:**

**Локальные IDE**

Разработчики устанавливают и запускают локальные IDE непосредственно на своих локальных машинах. Им также приходится загружать и устанавливать различные дополнительные библиотеки в зависимости от их предпочтений в кодировании, требований проекта и языка разработки. Хотя локальные IDE являются настраиваемыми и не требуют подключения к Интернету после установки, они сопряжены с несколькими проблемами.

* Их установка может отнимать много времени и быть сложной.
* Они потребляют локальные ресурсы машины и могут значительно замедлить ее работу.
* Различия в конфигурации между локальной машиной и производственной средой могут привести к ошибкам в программном обеспечении.

**Облачные IDE**

Разработчики используют облачные IDE для написания, редактирования и компиляции кода непосредственно в браузере, что избавляет их от необходимости загружать программное обеспечение на локальные машины. Облачные IDE имеют ряд преимуществ перед традиционными. Некоторые из них указаны ниже.

**Стандартизированная среда разработки**

Команды разработчиков программного обеспечения могут централизованно настроить облачную IDE для создания стандартной среды разработки. Этот метод помогает им избежать ошибок, которые могут возникнуть из-за различий в конфигурации локальной машины.

**Независимость от платформы**

Облачные IDE работают в браузере и не зависят от локальных сред разработки. Это означает, что они подключаются непосредственно к облачной платформе поставщика, поэтому разработчики могут использовать их с любой машины.

**Улучшенная производительность**

Создание и компиляция функций в IDE требует много памяти и может замедлить работу компьютера разработчика. Облачная IDE использует вычислительные ресурсы из облака и освобождает ресурсы локальной машины.

**Intellij Idea**

IntelliJ – одна из самых мощных и популярных интегрированных сред разработки (IDE) для Java. Он разработан и поддерживается **JetBrains** и доступен как окончательная версия для сообщества. Эта многофункциональная IDE обеспечивает быструю разработку и помогает улучшить качество кода.

IntelliJ IDEA обладает некоторыми наиболее эффективными функциями завершения кода Java. Его алгоритм прогнозирования может точно предполагать, что кодер пытается набрать, и завершает его для него, даже если он не знает точного имени определенного класса, члена или любого другого ресурса.



**Особенности IntelliJ IDEA и ее возможности**

IntelliJ IDEA действительно понимает и **глубоко понимает** ваш код, а также контекст кодера, что делает его таким уникальным среди других Java IDE.

**Цепное завершение кода** – это расширенная функция завершения кода, которая перечисляет соответствующие символы, доступные через методы или методы получения в текущем контексте.

**Завершение статического члена** – позволяет использовать статические методы или константы и автоматически добавляет необходимые операторы импорта, чтобы избежать ошибки компиляции.

**Обнаружение дубликатов** – Он обнаруживает фрагменты дублированного кода на лету и дает уведомление / предложение об этом пользователю.

**Инспекции и быстрые исправления.** Всякий раз, когда IntelliJ обнаруживает, что вы собираетесь совершить ошибку, в одной строке появляется небольшое ламповое уведомление. Нажав на нее, вы увидите список предложений.

Чтобы помочь разработчикам организовать рабочий процесс, IntelliJ IDEA предлагает им удивительный набор инструментов, который включает в себя декомпилятор, поддержку Docker, средство просмотра байт-кода, FTP и многие другие инструменты:

**Контроль версий** – IntelliJ поддерживает большинство популярных систем контроля версий, таких как Git, Subversion, Mercurial, CVS, Perforce и TFS.

**Инструменты сборки** – IntelliJ поддерживает Java и другие инструменты сборки, такие как Maven, Gradle, Ant, Gant, SBT, NPM, Webpack, Grunt и Gulp.

**Тестовый прогон и покрытие кода** – IntelliJ IDEA позволяет с легкостью выполнять модульное тестирование. Среда IDE включает в себя тестовые прогоны и инструменты покрытия для основных сред тестирования, включая JUnit, TestNG, Spock, Cucumber, ScalaTest, spec2 и Karma.

**Декомпилятор** – IntelliJ поставляется со встроенным декомпилятором для классов Java. Если вы хотите заглянуть внутрь библиотеки, для которой у вас нет исходного кода, вы можете сделать это без использования сторонних плагинов.

**Терминал** – IntelliJ предоставляет встроенный терминал. В зависимости от вашей платформы вы можете работать с командной строкой, например PowerShell или Bash.

**Инструменты базы данных** – IntelliJ предоставляет инструменты базы данных, которые позволяют подключаться к действующим базам данных; выполнять запросы; просматривать и обновлять данные; и даже управлять своими схемами в визуальном интерфейсе из самой IDE.

**Сервер приложений** – IntelliJ поддерживает основные серверы приложений: Tomcat, JBoss, WebSphere, WebLogic, Glassfish и многие другие. Вы можете развернуть свои артефакты на серверах приложений и отладить развернутые приложения в самой IDE.

**Поддержка Docker.** Через отдельный плагин IntelliJ предоставляет специальное окно инструментов, которое позволяет подключаться к локально работающим компьютерам Docker.

### Инструменты для совместной и удаленной работы

Новые версии среды разработки предлагают несколько удобных сервисов для командной работы над проектами.

* **Режим Code With Me.** Дает возможность в реальном времени работать над проектом, открытым в IDE, вместе с коллегами. При этом можно общаться с помощью голосовых и видеозвонков.
* **Интеграция с платформой JetBrain Space.** Позволяет организовать командную работу.
* **Режим удаленной разработки.** Помогает подключиться к мощному удаленному серверу, на котором работает среда IntelliJ IDEA, и пользоваться IDE, как если бы она была установлена на собственном компьютере программиста.

**Недостатки IntelliJ IDEA**

**Требовательность к системным ресурсам**

Обратная сторона впечатляющей функциональности — ощутимая нагрузка на оперативную память и процессор. Для работы с IDEA нужен достаточно мощный компьютер, имеющий не менее 8 Гб (лучше больше) оперативной памяти и процессор Intel не ниже 10-11 поколения.

**Урезанная функциональность бесплатной версии**

Проблема решается приобретением лицензии на использование Ultimate-варианта.

**Отсутствие перевода интерфейса на русский язык**

Компания JetBrains не русифицирует свои продукты, и английский язык интерфейса может поставить в тупик начинающих программистов. Однако среди плагинов на официальном портале компании есть два решения для проверки русской орфографии.

**Java**

Java – это язык программирования общего назначения. То есть язык, который применяется в разработке различных программных продуктов, без четкой специализации в конкретной сфере. Он во многом похож на Python, JavaScript и другие языки того же уровня, что и Java. Кроме того, Java заимствует массу синтаксических конструкций из C и C++.

Это объектно-ориентированный язык. Вся структура строится вокруг объектов, классов, инстансов и прочих формальных сущностей, принятых в сообществе программистов за стандарт разработки ООП. Это как проектирование домов с использованием отдельных строительных блоков, которые потом можно задействовать при постройке различных типов зданий. Программы, написанные на Java, состоят из подобных «блоков», что сокращает время разработки, а также делает код более читаемым и удобным для модификации.

А еще Java выступает в роли платформы. Код, написанный на этом языке, запускается в виртуальной машине JVM и без проблем инициализируются в любой системе, где поддерживается соответствующая виртуальная машина.

### История появления Java

Язык Java был разработан командой инженеров Sun Microsystems в 1995 году. Позднее компания вместе с Java была поглощена корпорацией Oracle.

Ключевой задачей при разработке Java было создать язык программирования, который мог бы запускаться не только на компьютерах. То есть был достаточно универсальным, чтобы работать даже на операционной системе холодильника. На момент разработки такое решение казалось чересчур дальновидным, но оно привнесло вместе с собой несколько важных архитектурных изменений, ставших критически важными для Java и сделавших этот язык популярным.

**Gradle**

Gradle — система для автоматизации сборки приложений и сбора статистики об использовании программных библиотек, применяющая языки Groovy, Java, JavaScript, Kotlin и т. д., а также решения из [фреймворков](https://blog.skillfactory.ru/glossary/framework/) Apache Ant и Apache Maven.

Gradle написана на языках Java, Groovy и [Kotlin](https://blog.skillfactory.ru/glossary/kotlin/), также поддерживает плагины на [JavaScript](https://blog.skillfactory.ru/glossary/javascript/), C++, Swift, Scala. Система распространяется как программное обеспечение с открытым исходным кодом по лицензии Apache License 2.0.



## Назначение Gradle

Ручная сборка приложения для запуска на устройстве требует много времени и ресурсов. Она затягивает процесс разработки и релиза.

Чтобы ускорить его, программисты создают скрипты, автоматизирующие выполнение типовых задач. Это сокращает время сборки, но усложняет работу над командным проектом другим разработчикам.

Проблему решили системы автоматической сборки приложений. Первая — Ant — появилась в 2000 году. В 2004 году вышла Maven. Для описания сценариев сборки системы использовали язык разметки [XML](https://blog.skillfactory.ru/glossary/xml/). Его недостаток — неконтролируемое разрастание при попытках собрать крупный многомодульный проект.

В 2006 году появилась первая версия Gradle. Она позволила решить основные проблемы:

* реализацию больших многомодульных сборок путем последовательного и параллельного выполнения основных и добавочных модулей;
* ускорение сборки за счет кеширования компонентов через общую сеть с использованием Gradle Build Cache;
* управление зависимостями, то есть подключаемыми к проекту фреймворками, библиотеками и другими компонентами;
* упрощение рабочего процесса за счет автоматизации типовых задач и веб-визуализации сборки.

Gradle использует многие решения, которые применяются в Ant и Maven. Три системы широко интегрированы друг с другом: проект, разработанный для одной из них, можно собрать в другой. Например, основным источником подключаемых внешних компонентов в Gradle является хранилище Maven.

**Идея плагина**

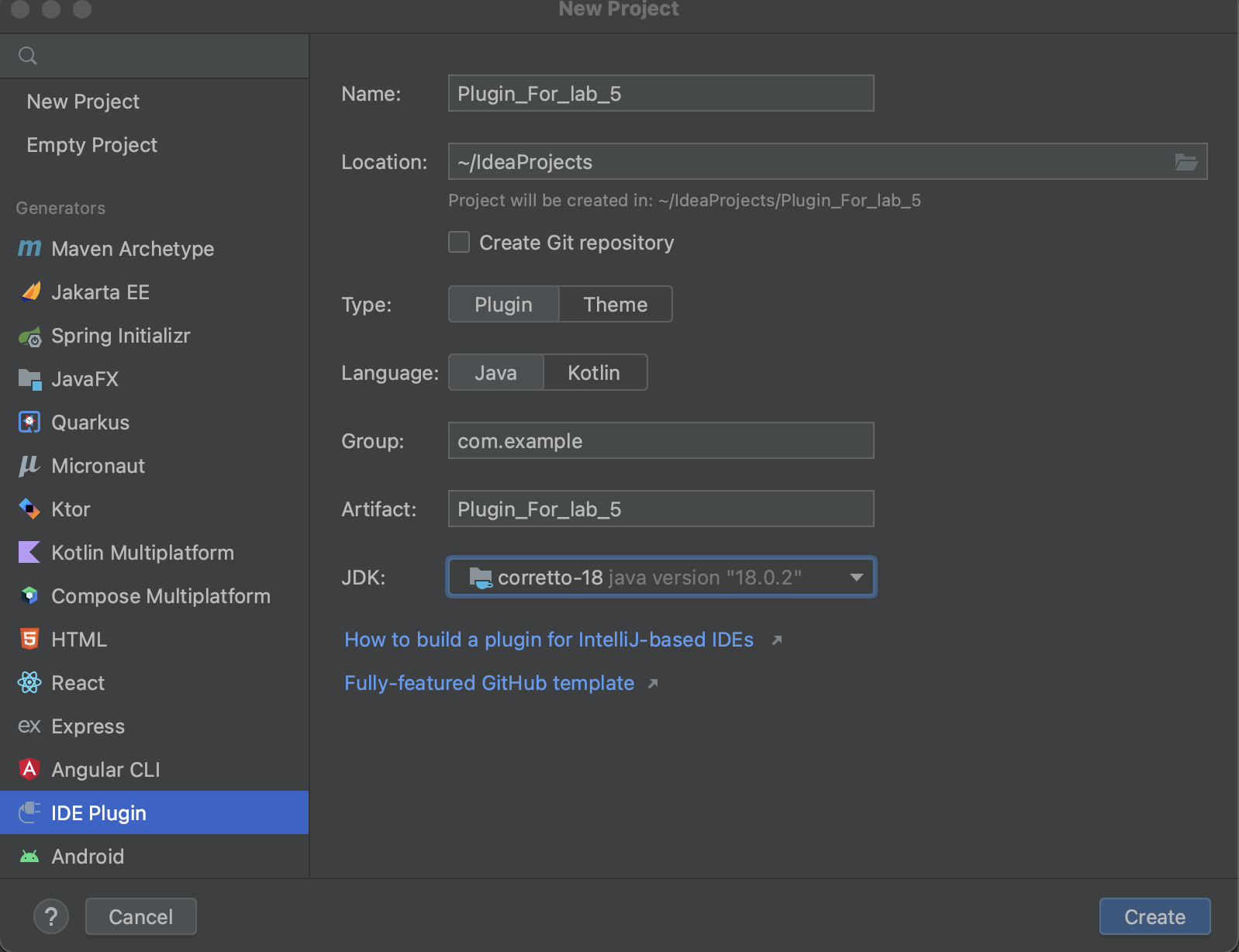
Для написания плагина также использовалась IntelliJ. Идея плагина состояла в том, чтобы пользователь в любое время мог посмотреть в интернете как работает та или иная функция, или что какие параметры она принимает.

Работать это будет так:

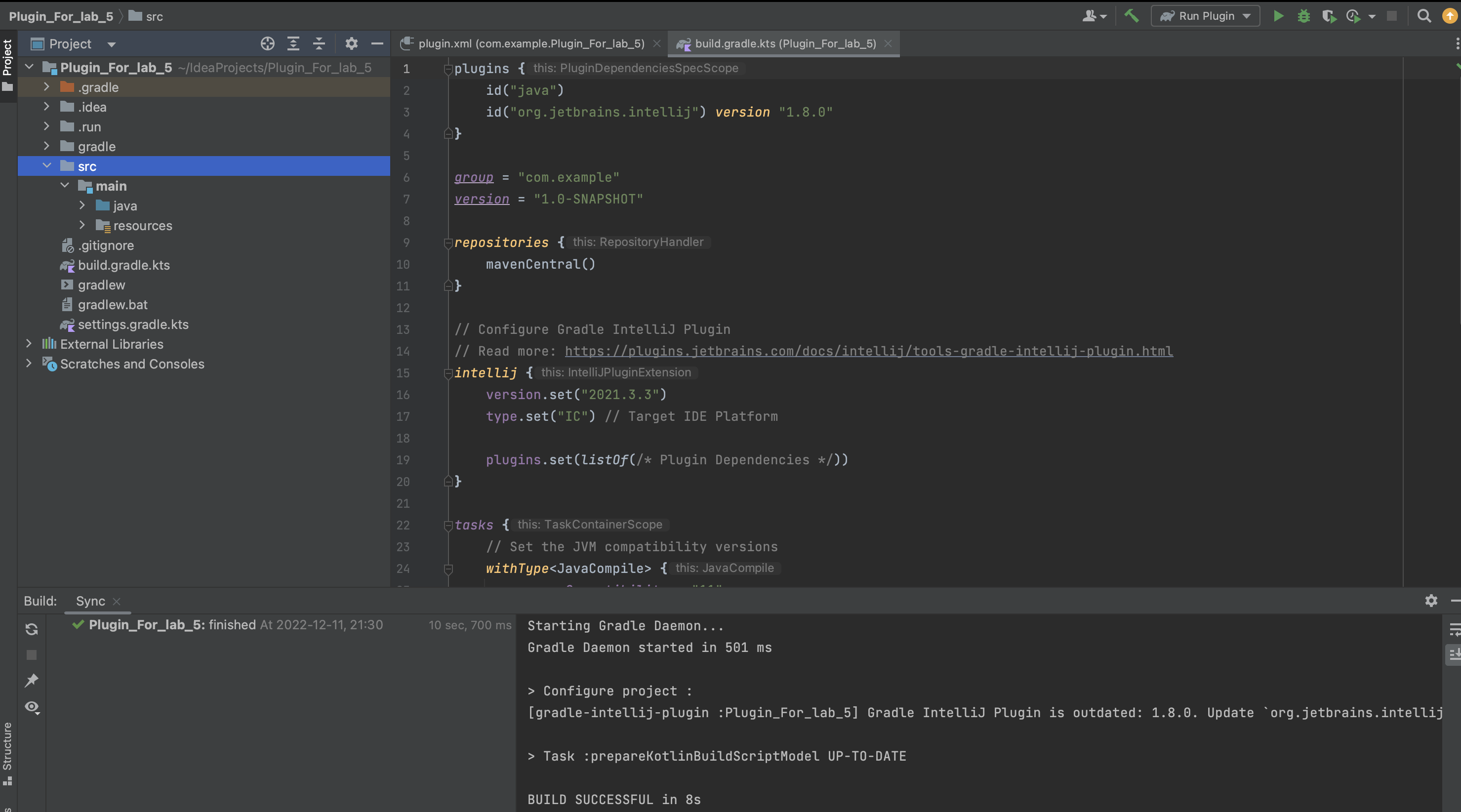
1. Пользователь, работая в IntelliJ, выделяет любое слово или слова.
2. Нажимает комбинацию клавиш (option + p).
3. Далее плагин автоматически перекидывает нас в интернет, в котором сразу же открывается вкладка с нужной нам информацией.

**Написание плагина**

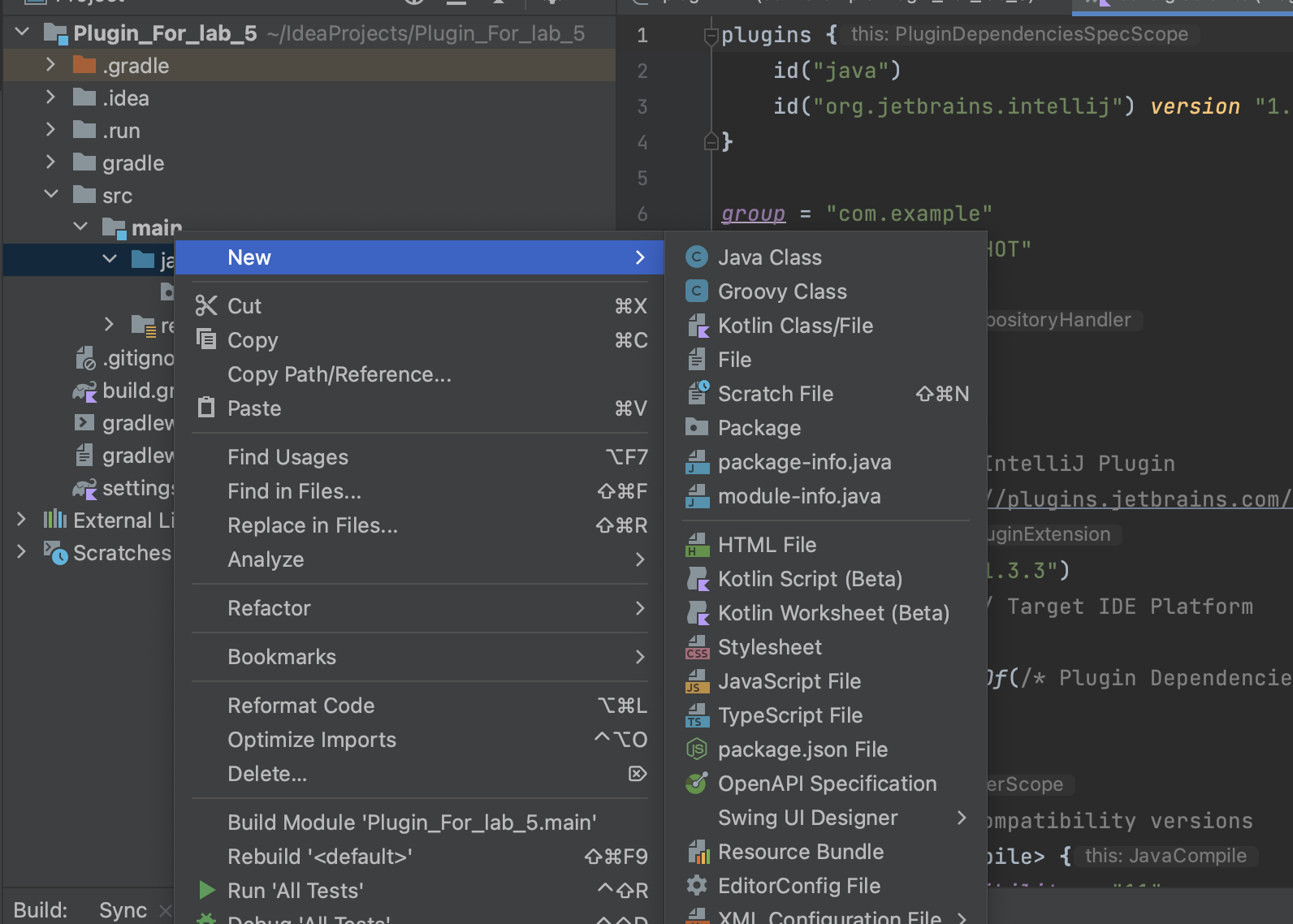
1. Создадим новый проект.



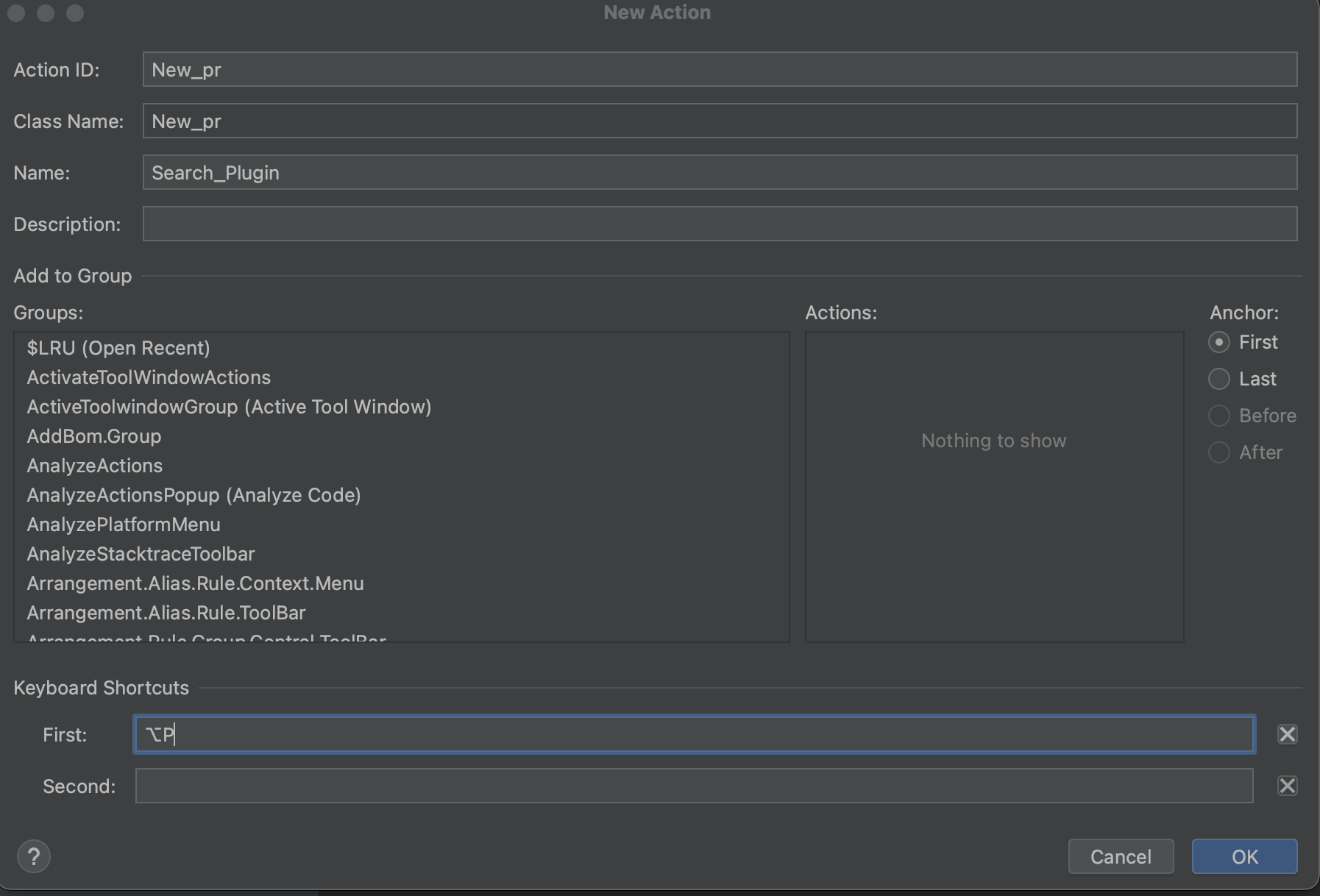
В качестве языка, на котором будем писать плагин выберем Java.



1. Теперь создадим новый класс java, в котором будет находиться основной функционал нашей программы.



1. Регистрируем плагин (Action) и задаем комбинацию клавиш.



1. Пишем код, который выполняет нужные нам действия.



**AnAction** используется для отделения функциональности и состояния от компонента.

**getData** - возвращает буфер данных.

**getSelectionModel** () - предоставляет API, с помощью которого можно выбрать один или несколько элементов в ListView, а также проверить, какие элементы были выбраны пользователем.

**getSelectedText** () - Возвращает выделенный текст, содержащийся в этом TextComponent

**URLEncoder** - Служебный класс для кодирования форм HTML. Этот класс содержит статические методы для преобразования строки в формат application/x-www-form-urlencodedMIME. Для получения дополнительной информации о кодировании форм HTML обратитесь к спецификации HTML.

**UnsupportedEncodingException** - сообщает, что Кодировка символов не поддерживается.

**printStackTrace** - сообщает где произошел сбой в программе.

**format** - это формат общедоступного абстрактного класса расширяет объект.

Format это абстрактный базовый класс для форматирования информации, зависящей от локали, такой как даты, сообщения и числа.

Format определяет программный интерфейс для форматирования объектов, зависящих от локали, в Strings (formatметод) и для синтаксического Stringанализа s обратно в объекты (parseObjectметод).

Как правило, parseObject метод format должен иметь возможность анализировать любую строку, отформатированную его formatметодом. Однако могут быть исключительные случаи, когда это невозможно. Например, formatметод может создать два смежных целых числа без разделителя между ними, и в этом случае parseObjectневозможно определить, какие цифры принадлежат какому числу.

**snowMessageDialog** - сообщает пользователю, что произошло.

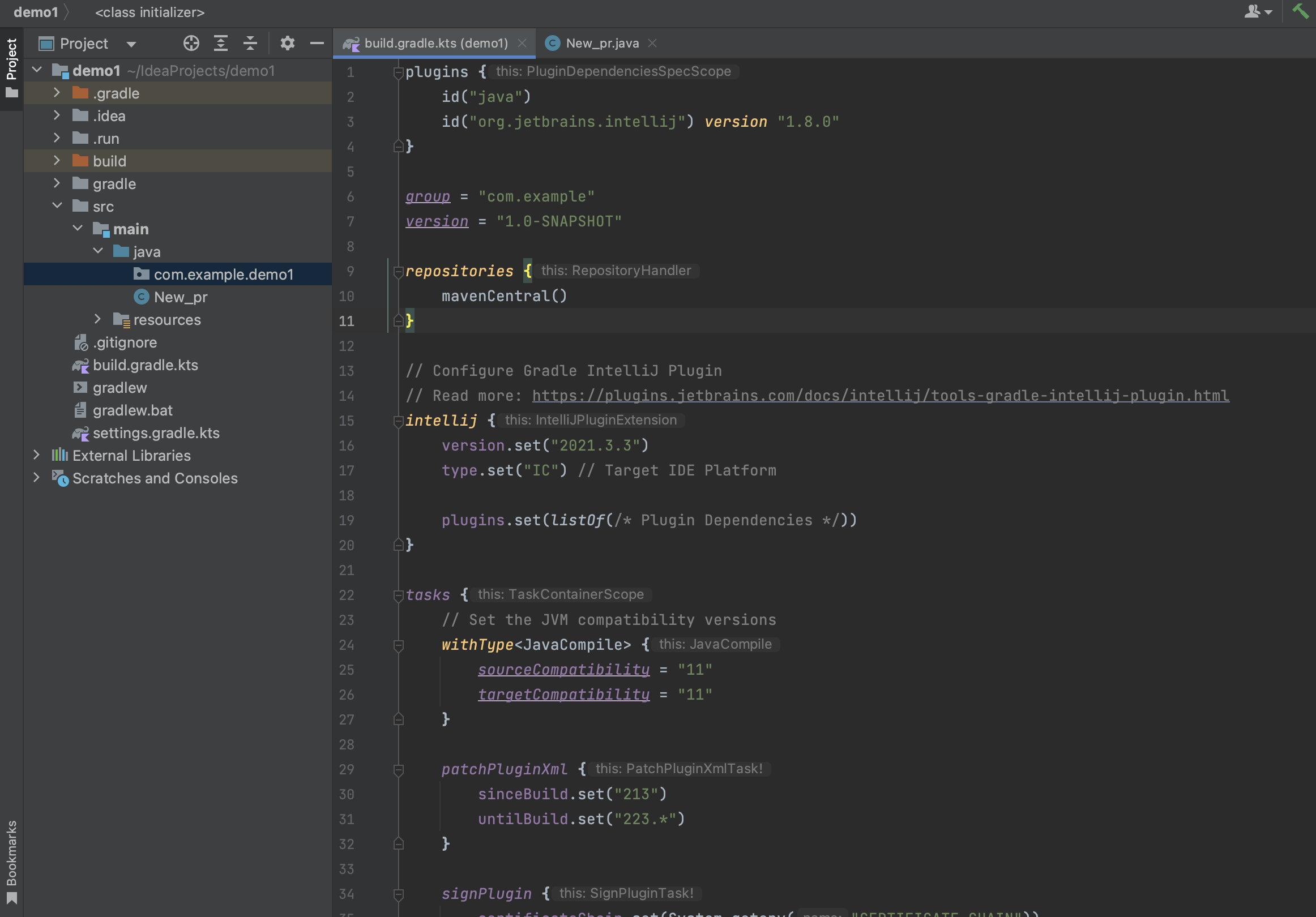
@**Override** - Указывает, что объявление метода предназначено для переопределения объявления метода в супертипе. Если метод аннотируется с помощью этого типа аннотации, компиляторы должны генерировать сообщение об ошибке, если не выполняется хотя бы одно из следующих условий:

* Метод переопределяет или реализует метод, объявленный в супертипе.
* Метод имеет сигнатуру, эквивалентную сигнатуре любого открытого метода, объявленного в Object.

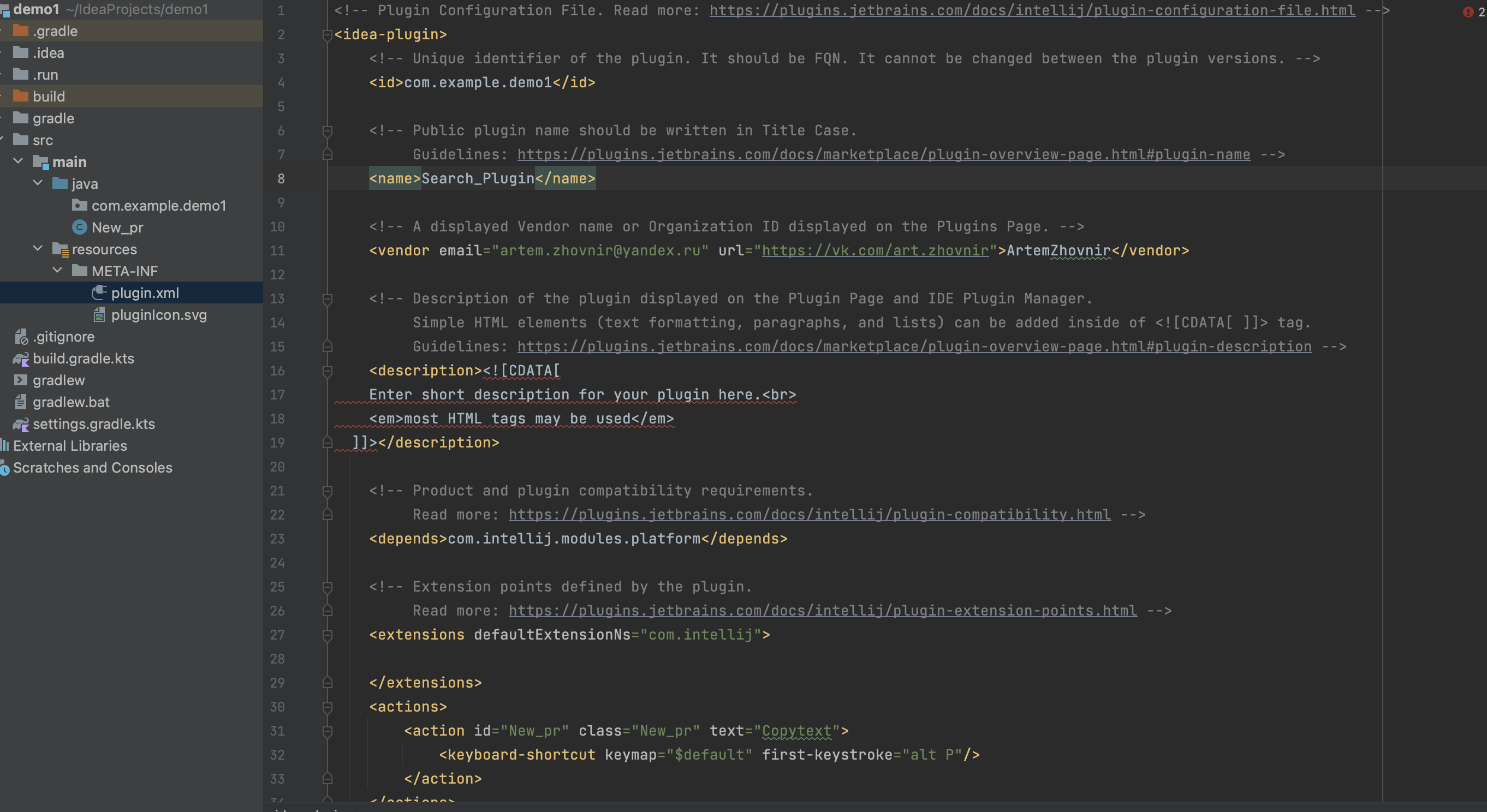
String url = String.*format*("https://yandex.ru/search/?text=%s&search\_source=dzen\_desktop\_safe&lr=2", Highlighted);

Данная строка говорит нам о том, что поиск выделенного текста будет производиться именно в поисковой системе Яндекс.

1. Настраиваем сборку Gradle.



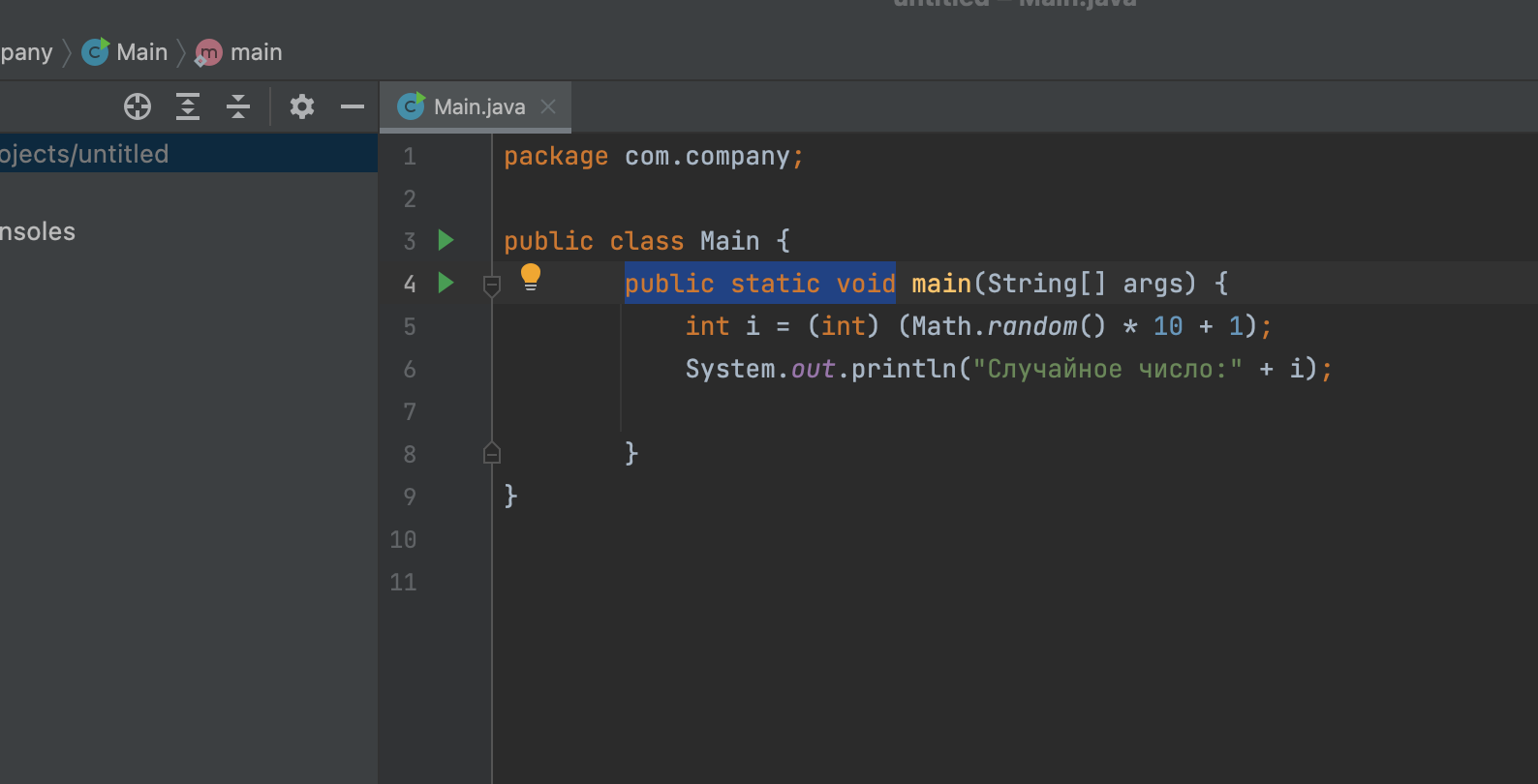
1. Заполняем информацию о плагине.



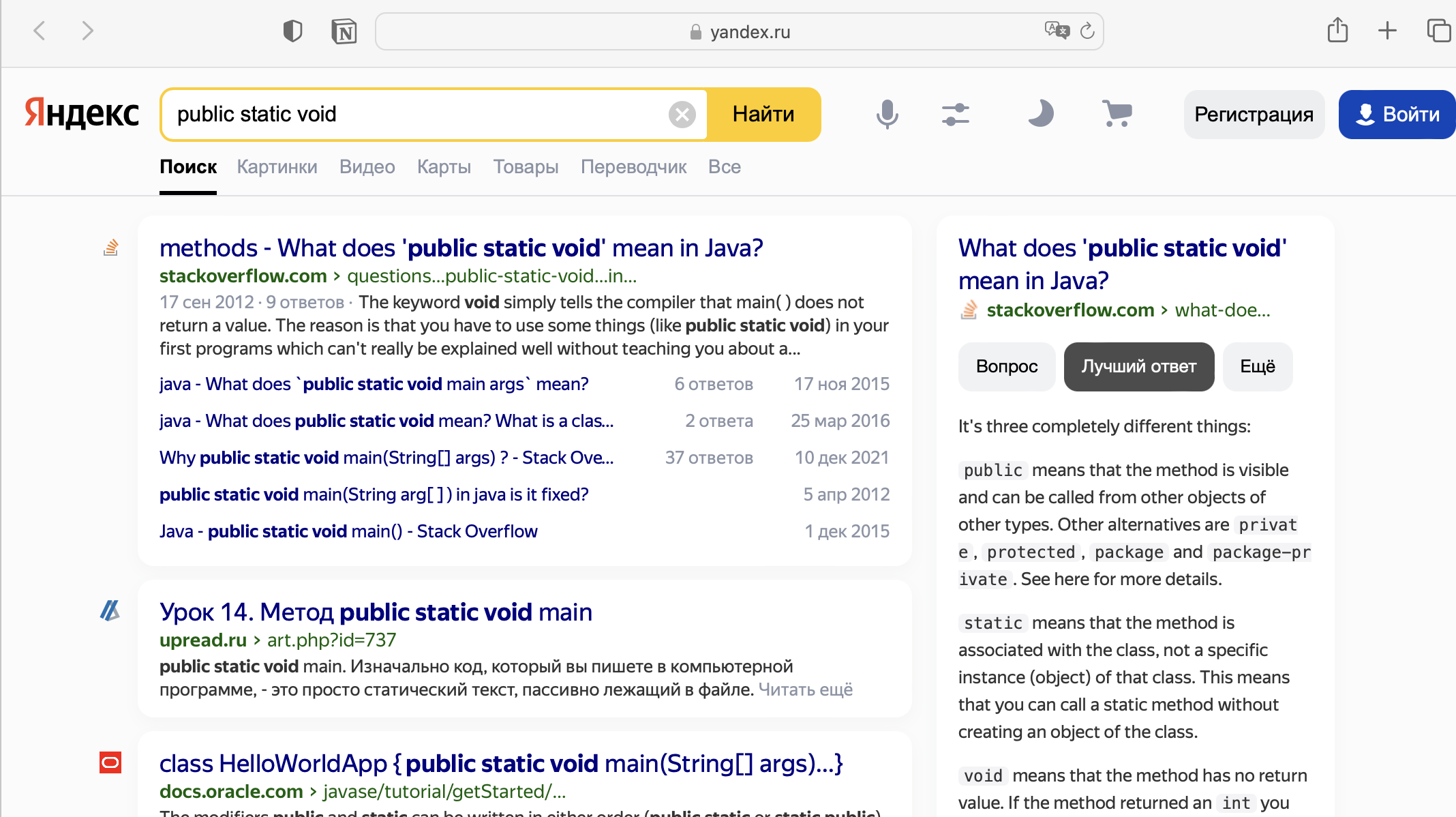
1. Напишем тестовую программу на Java:

package com.company;  
  
public class Main {  
 public static void main(String[] args) {  
 int i = (int) (Math.*random*() \* 10 + 1);  
 System.*out*.println("Случайное число:" + i);  
  
 }  
}

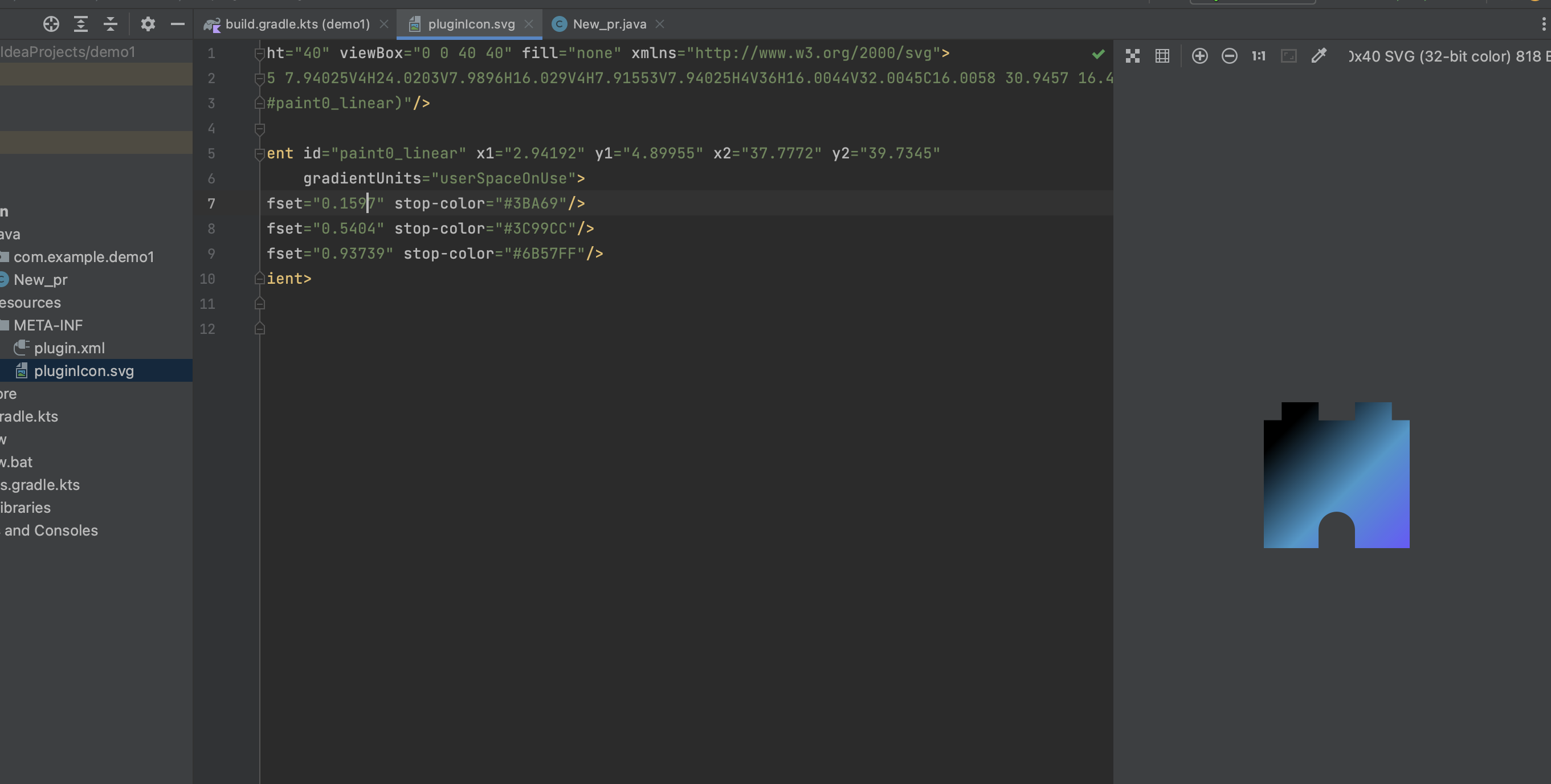
1. Протестируем работу плагина.



Нажимаем: **option + p**.



1. Оформил иконку нашего плагина.



**Итоги**

В результате выполнения данной лабораторной работы был создан плагин для среды разработки IntelliJ, который ищет выделенный в IDE текст прямо в интернете.

GitHub: <https://github.com/Artemiksss/DevTools_Lab5_Plugin.git>