#### **15.9 Итоги пройденных тем. Проверьте себя**

#### Цель практической работы

Закрепить знания и навыки, полученные в модуле:

* добавление Maven и Gradle в проект;
* добавление файлов pom.xml и build.gradle;
* подключение библиотек;
* сборка JAR-файлов;
* загрузка и поднятие контейнеров из Docker-образов.

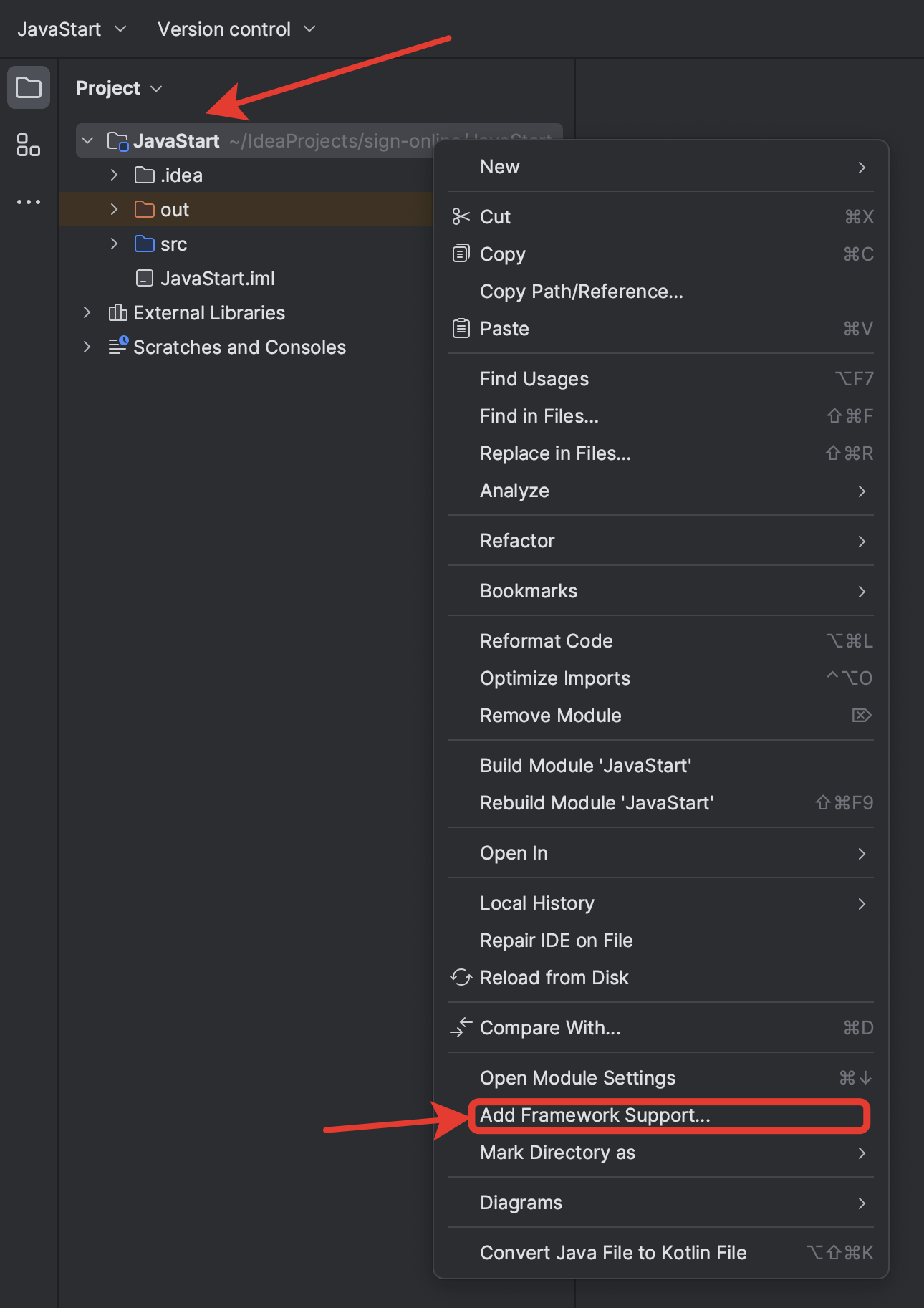
#### Что входит в работу

1. Отработка базовых навыков работы с Maven.
2. Отработка базовых навыков работы с Gradle.
3. Отработка базовых навыков работы с Docker.

### Задание 1. Отработка базовых навыков работы с Maven

##### Шаг 1. Добавление Maven в проект

1. На этом шаге мы добавим Maven в проект.
2. Откройте среду разработки IntelliJ IDEA и проект, в который хотите добавить поддержку Maven. Или вы можете создать пустой проект File->New->Project…
3. Нажмите правой кнопкой мыши на корень вашего проекта.
4. В открывшемся окне выберите строку Add Framework Support.

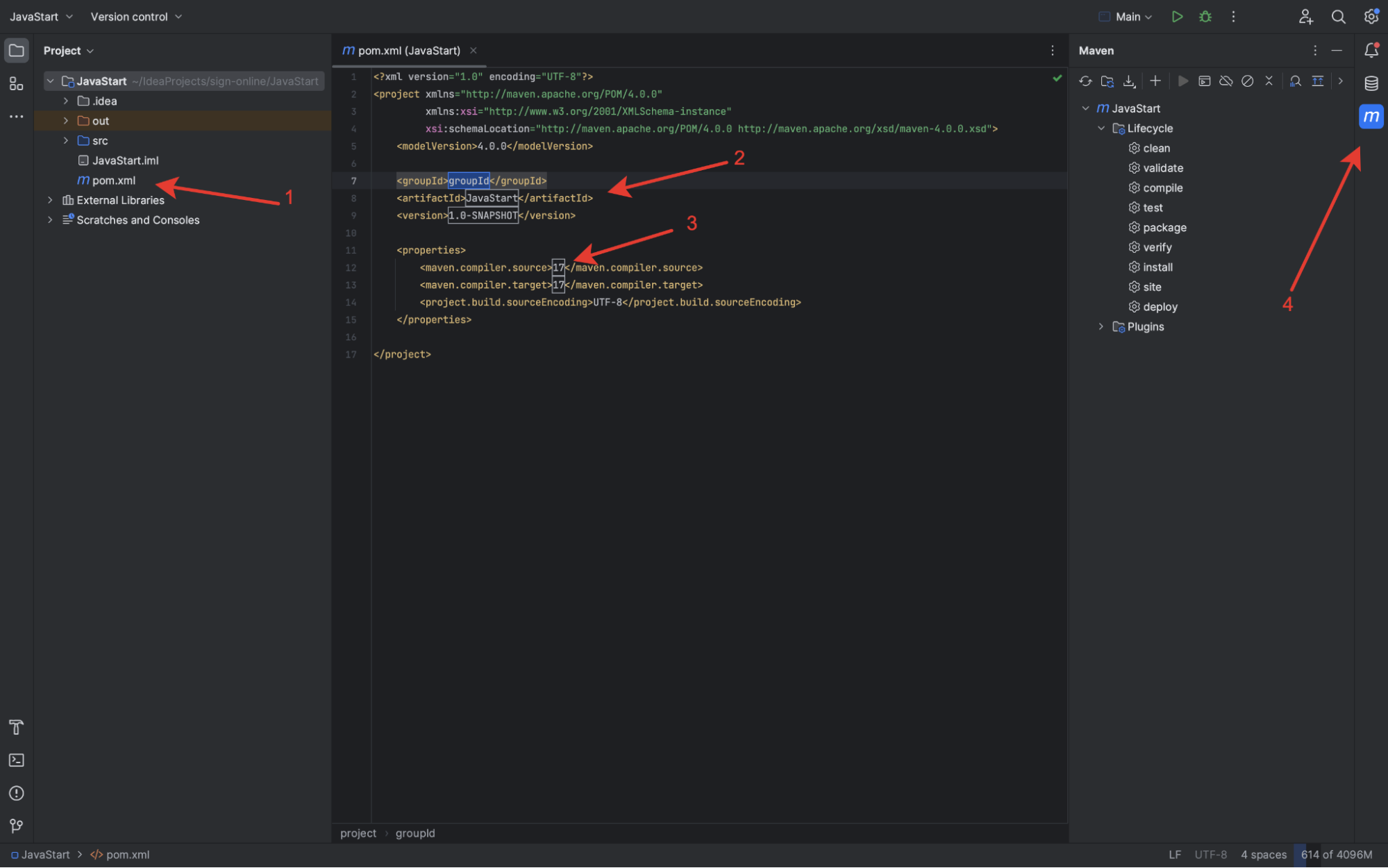


Здесь и далее — изображения Skillbox

1. Отметьте галочкой Maven и нажмите ОК.



В результате вы должны увидеть следующее:



1. Сгенерированный pom.xml-файл.
2. Настройки для сборки вашего JAR-файла.
3. Блок настроек проекта. В данном случае указана версия Java 17. Установите ту, которую используете в проекте.
4. Кнопка “Maven”, открывающая панель с перечнем команд, которые предоставляет Maven. Нажмите на неё.

##### Шаг 2. Подключение библиотеки к проекту

1. На этом шаге мы подключим к проекту библиотеку OkHttp.
2. В файл pom.xml внутри тегов **<project></project>** и после закрывающего тега **</properties>** добавьте блок тегов **<dependencies>**, в которых надо будет указать зависимости проекта:

<dependencies>

</dependencies>

1. Внутри блока тегов **<dependencies>** добавьте описание зависимости — конкретную библиотеку (в данном случае — OkHttp). Готовый код можно скопировать с [официальной страницы этой библиотеки](https://mvnrepository.com/artifact/com.squareup.okhttp3/okhttp/4.10.0) в центральном Maven-репозитории. В итоге блок кода с зависимостями будет выглядеть у вас так:

<dependencies>

<dependency>

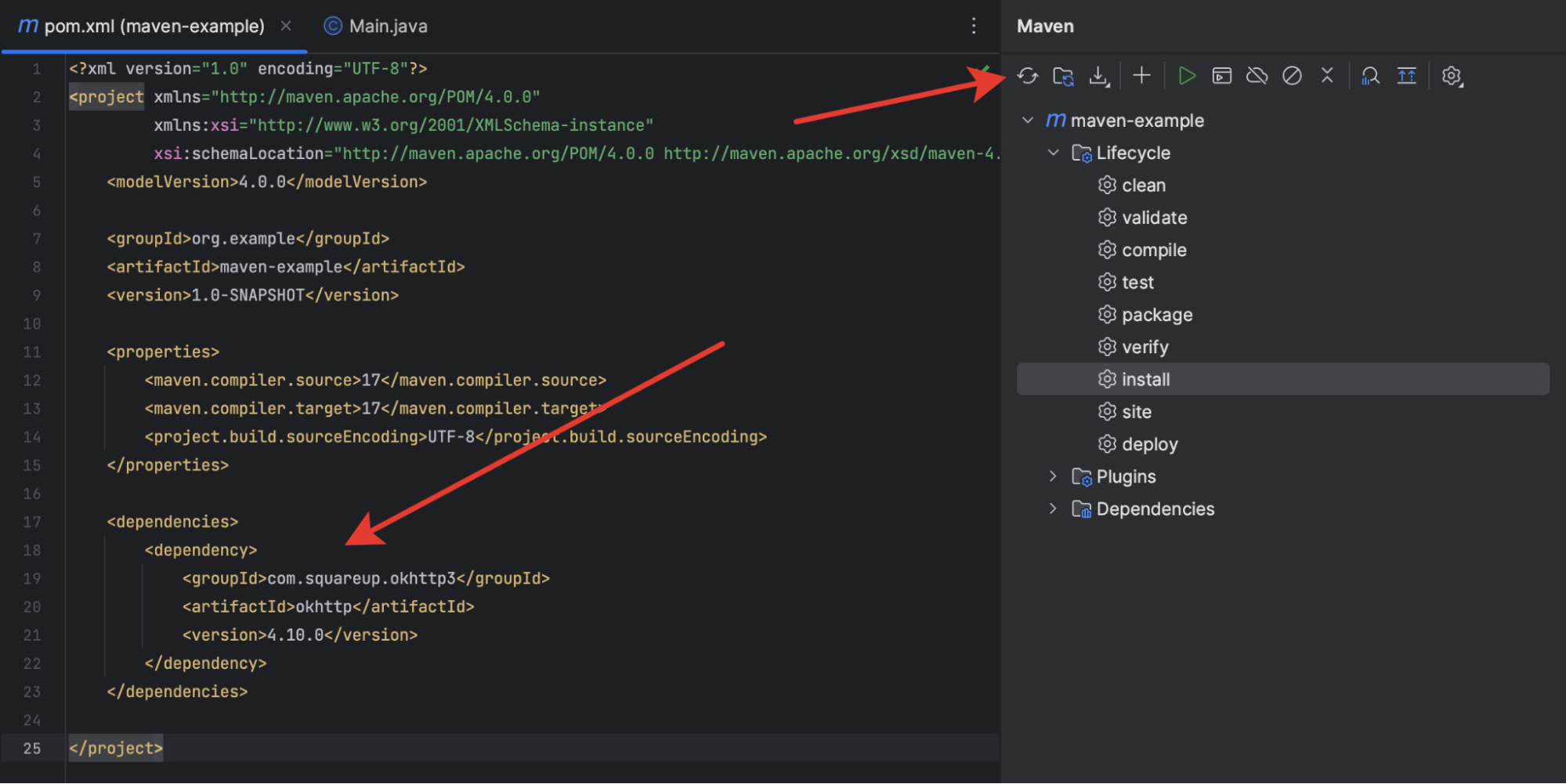
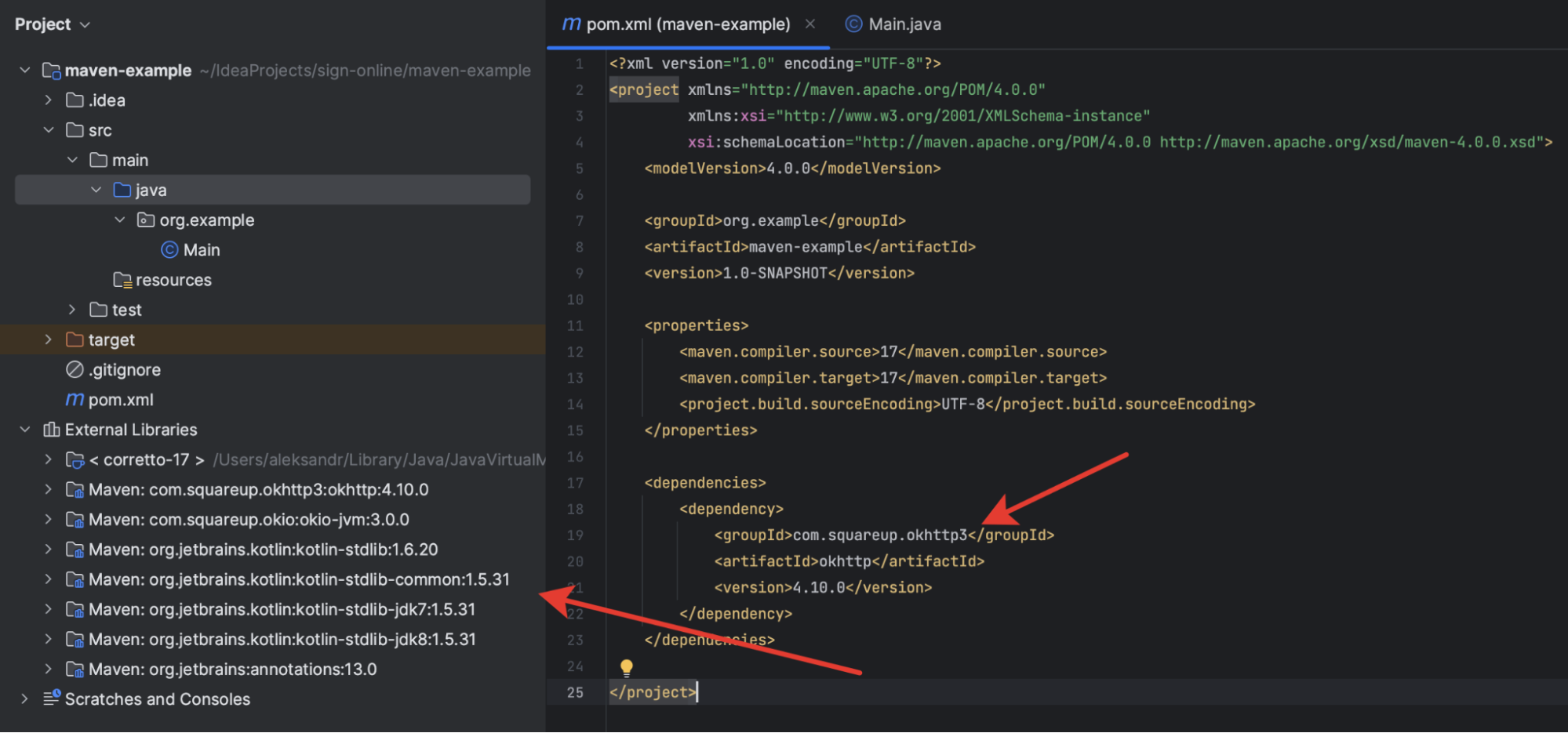
<groupId>com.squareup.okhttp3</groupId>

<artifactId>okhttp</artifactId>

<version>4.10.0</version>

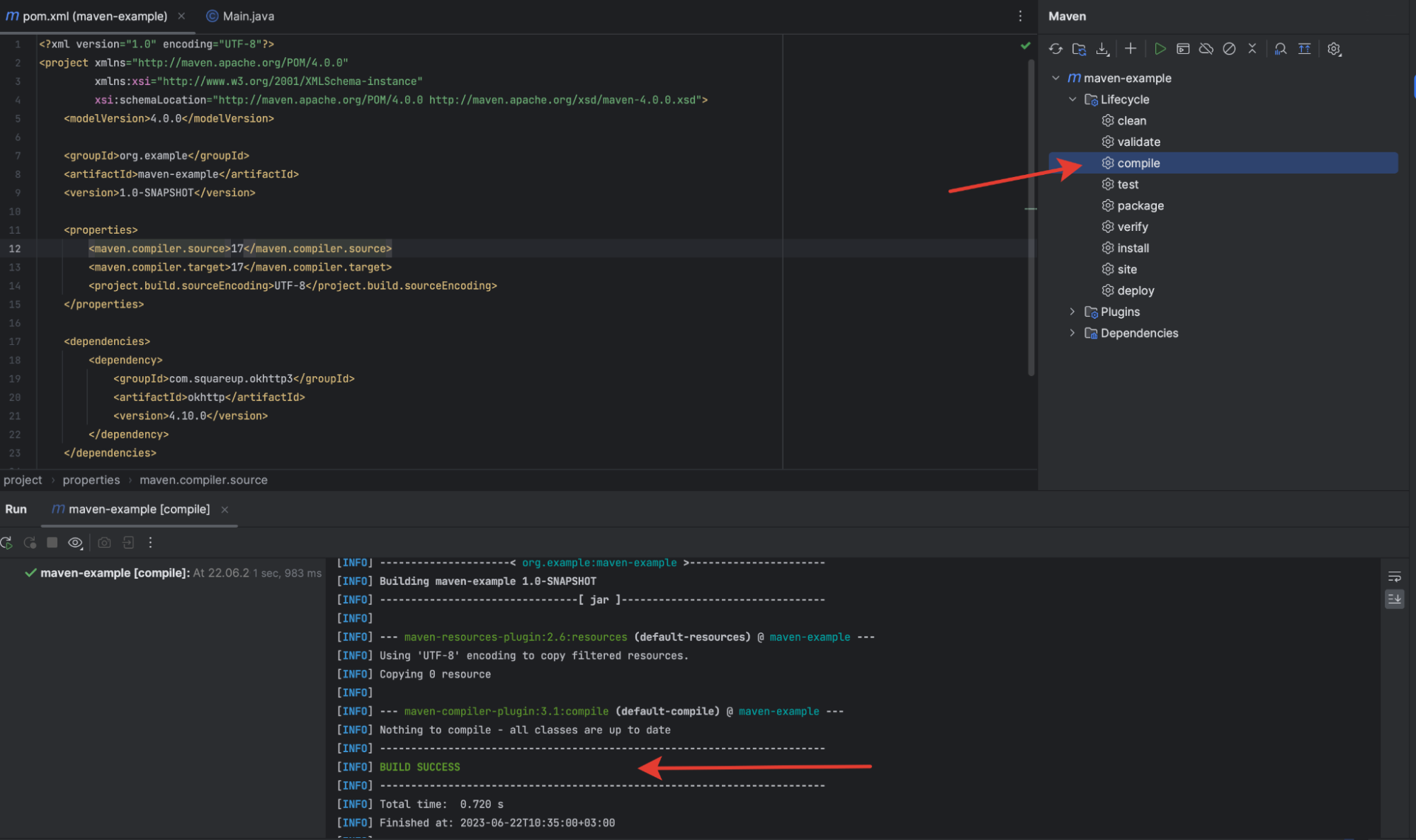
</dependency>

</dependencies>

1. Нажмите кнопку “Reload All Maven Projects”, чтобы Maven обработал изменения, которые вы внесли в pom.xml. В нашем случае Maven скачает библиотеку OkHttp в локальный репозиторий и добавит JAR-библиотеки в проект.  
   
2. После завершения обновлений в pom.xml библиотека окажется в проекте. **Обратите внимание:**в проект импортировалось несколько библиотек, это зависимости библиотеки OkHttp. Мы не описывали их, перечень библиотек находится внутри OkHttp. Maven сам увидел это и импортировал всё, что необходимо.  
   

##### Шаг 3. Компиляция проекта

Проверим, что код компилируется после импорта библиотек. Запустите команду compile, посмотрите на вывод в консоли. Должно вывестись сообщение “Build Success!”



##### Шаг 4. Использование библиотеки

Мы подключили библиотеку OkHttp. Библиотека OkHttp предоставляет простой способ взаимодействия с HTTP-серверами.

Напишем код в Main классе для обращения к странице сайта полученный в аргументах args метода main(). Это нам позволит передавать в приложение адрес сайта при запуске скомпилированного исполняемого файла:

public class Main {

public static void main(String[] args) {

if (args.length != 1) {

System.out.println("Ошибка передачи параметров\nТребуется один параметр");

System.exit(1);

}

String url = args[0];

System.out.println("URL from args:" + url);

OkHttpClient client = new OkHttpClient();

Request request = new Request.Builder()

.url(url)

.build();

try {

Response response = client.newCall(request).execute();

if (response.isSuccessful()) {

String responseBody = response.body().string();

System.out.println("Response: " + responseBody);

} else {

System.out.println("Request failed with HTTP error code: " + response.code());

}

} catch (IOException e) {

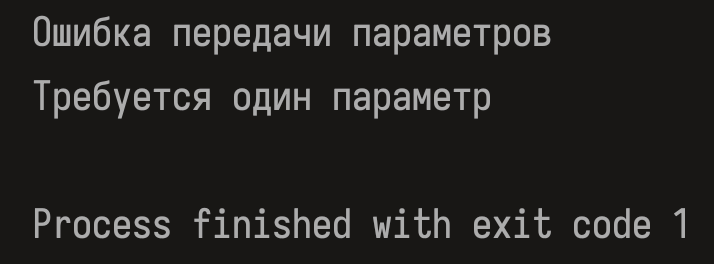
e.printStackTrace();

}

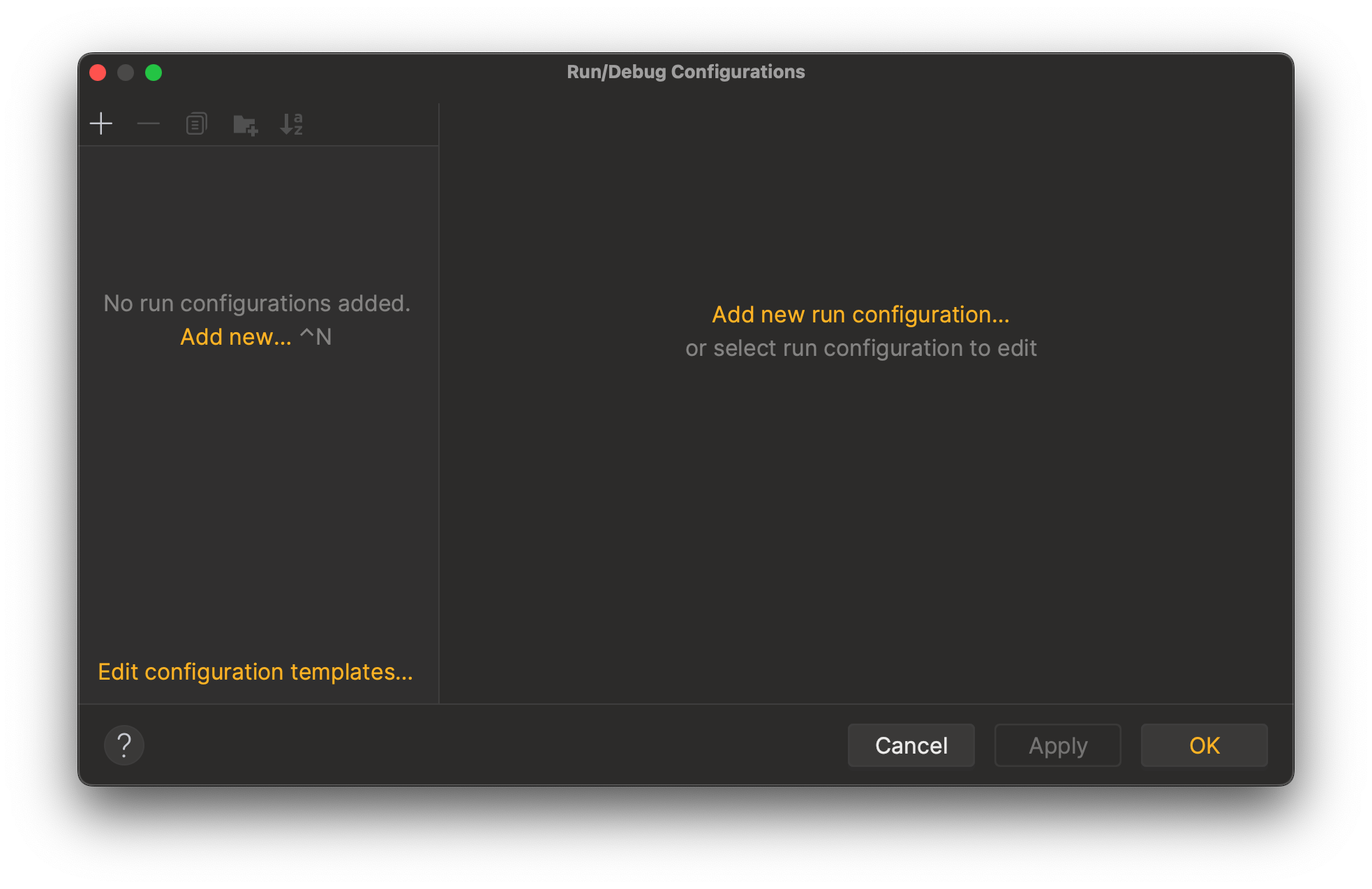
}

}

Если вы запустите приложение из среды разработки, то ожидаемый ответ будет следующим:



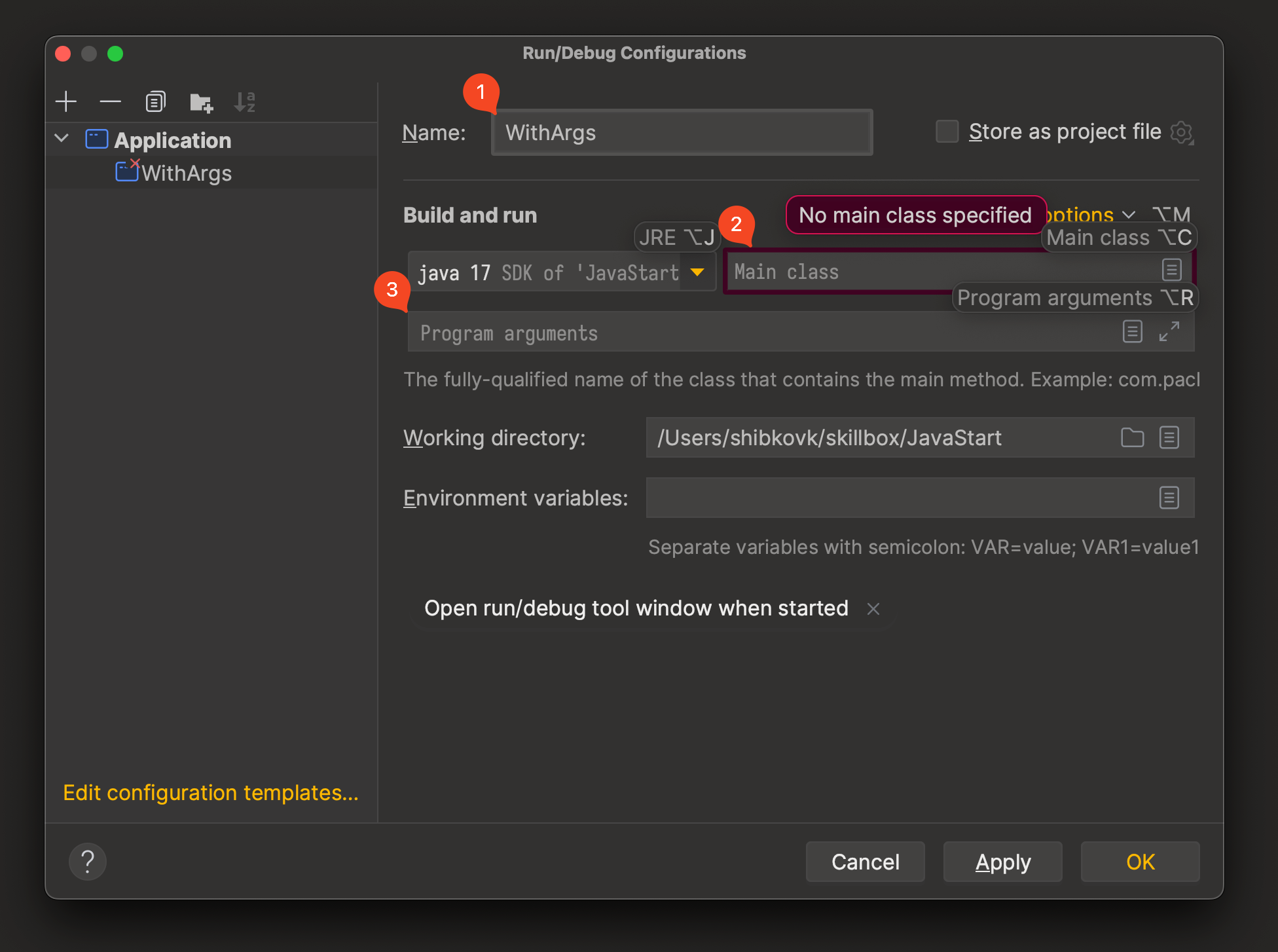
Это происходит потому, что мы не передали ни одного параметра. Чтобы указать параметры приложения при запуске из среды разработки IntelliJ IDEA, откройте параметры запуска через меню Run → Edit Configuration.



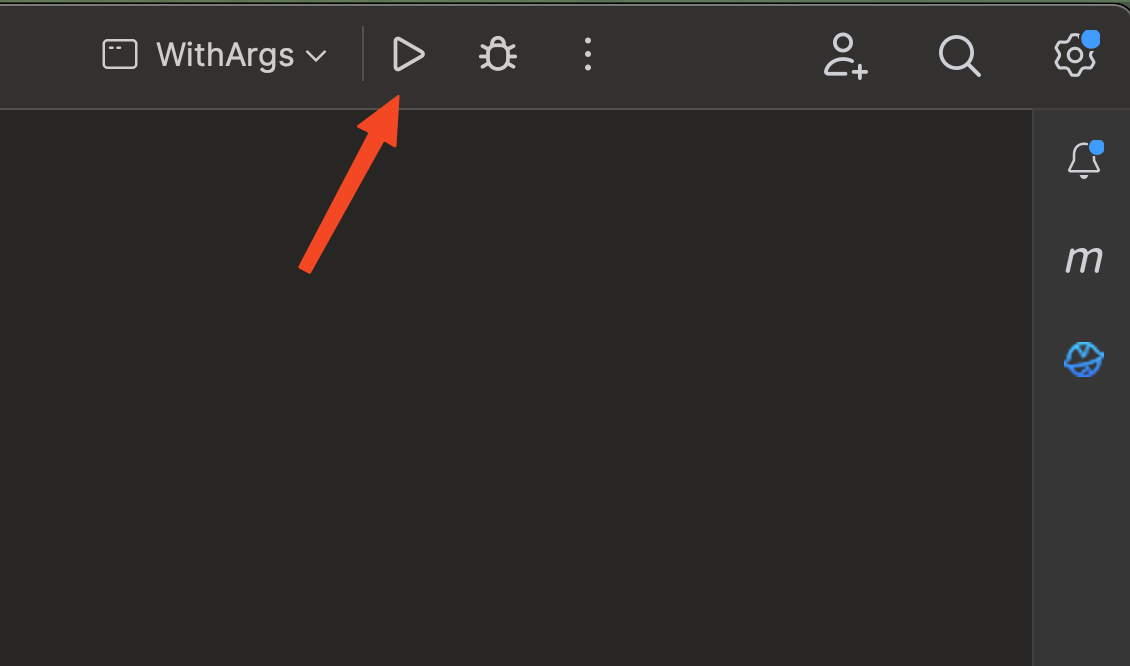
Нажмите «**+**» и выберите Application.

Введите параметры:

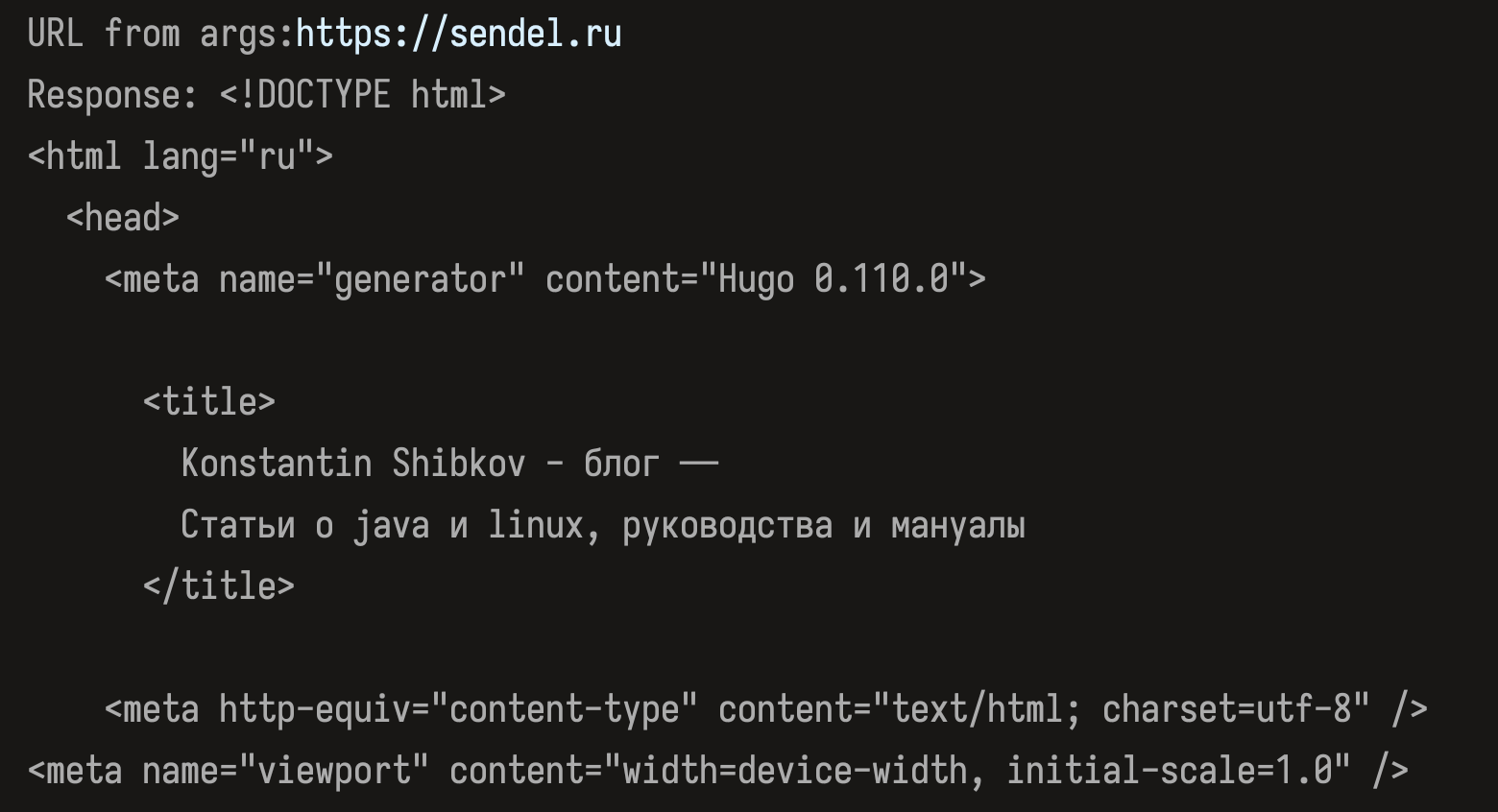
1. Название конфигурации запуска.
2. Выберите ваш Main класс в котором находится метод main() .
3. Укажите адрес страницы сайта, контент которого вы хотите получить. Обязательно должен начинаться с http:// или https://. В примере ниже показано получение контента с сайта https://sendel.ru/, можете использовать его же.
4. Нажмите OK.



Запустите ваше приложение, используя выбранную конфигурацию:



В итоге у вас должен измениться ответ, и если вы указали существующий адрес страницы сайта, то в консоли вы получите HTML-код страницы:



После сборки JAR файла это позволит нам передавать URL страницы сайта при каждом новом запуске приложения.

##### Шаг 5. Сборка JAR-файла

По умолчанию Maven-команда package собирает JAR-файл. Но не включает в него зависимости, используемые в проекте, и не создаёт указатель на класс, в котором находится метод main().

Для создания полноценного JAR-файла с зависимостями необходимо настроить Maven-плагин сборки. Для этого надо в pom.xml добавить:

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-assembly-plugin</artifactId>

<version>3.3.0</version>

<configuration>

<archive>

<manifest>

*<!-- Замените на ваш класс с методом main -->*

<mainClass>Main</mainClass>

</manifest>

</archive>

<descriptorRefs>

<descriptorRef>jar-with-dependencies</descriptorRef>

</descriptorRefs>

</configuration>

<executions>

<execution>

<id>make-assembly</id>

<phase>package</phase>

<goals>

<goal>single</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

**Что в нём происходит?**

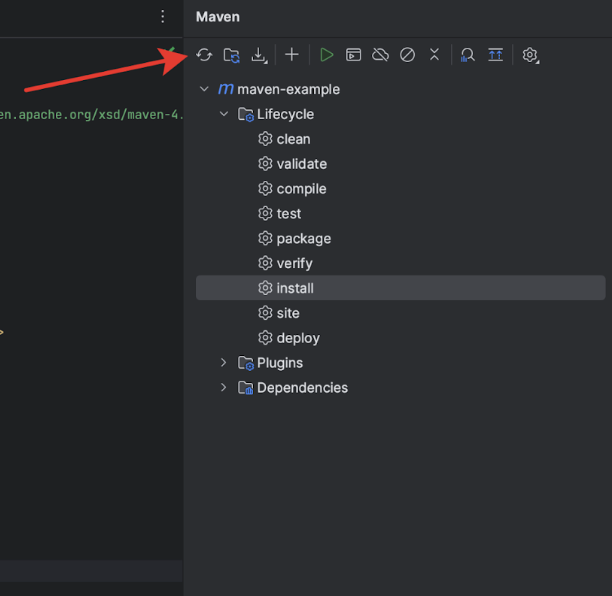
При добавлении плагина обязательно (как и для зависимостей) указывается groupId, artifactId и version. Ведь плагины это тоже JAR-приложения, только они запускаются на различных фазах жизненного цикла команд Maven.

В блоке <configuration> указываются параметры, с которыми выполняется плагин:

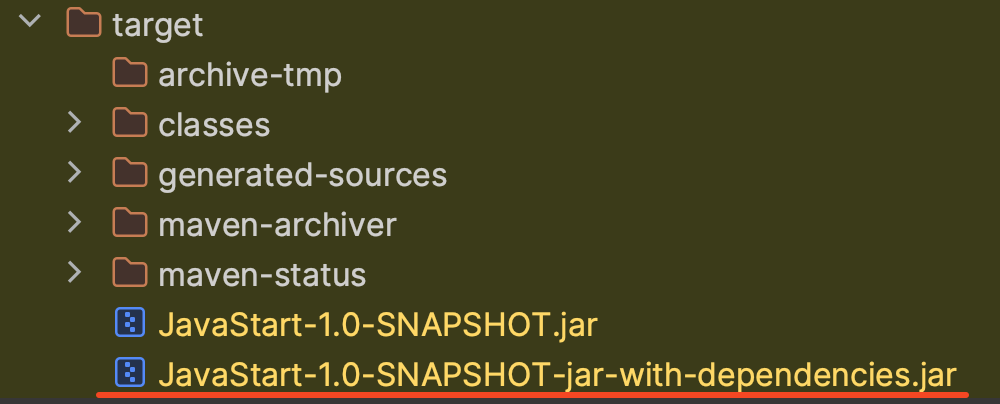
* archive.manifest.mainClass — указываем полный путь до класса Main.
* descriptorRefs — указатель, какой JAR нужно создать.

В блоке executions указываем, на какой фазе будет выполняться данный плагин, в нашем случае это будет package.

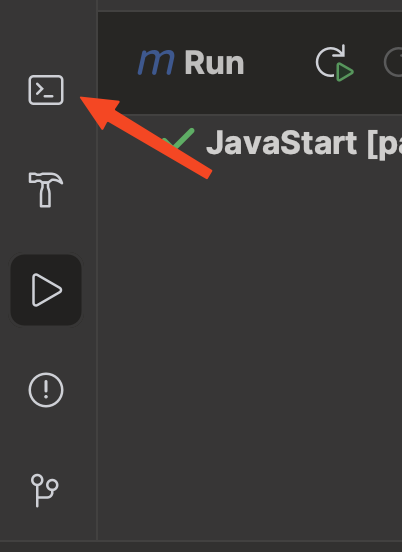
Не забудьте обновить pom.xml.



Запустите поочерёдно фазы clean и package. В итоге в папке target у вас появиться JAR-файл со всеми зависимостями:

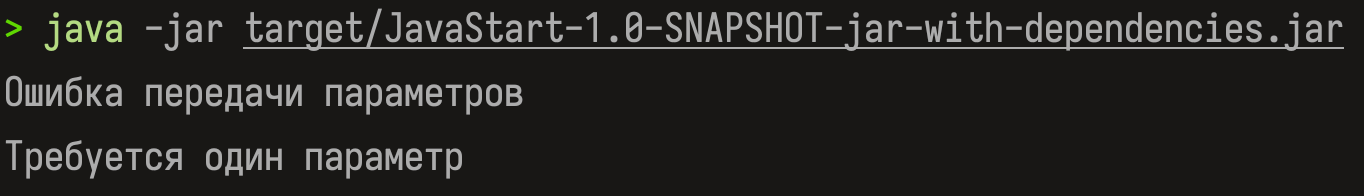


Осталось его запустить. Сделайте это через консоль. Запустите консоль из среды разработки через меню View → Tool Windows → Terminal или нажмите на иконку терминала на панели инструментов:



Откроется окно терминала и рабочая директория будет корнем вашего проекта.

Запустите JAR-файл с помощью следующей командой:



И передайте параметр после пути до JAR-файла:



Таким образом вы поочерёдно выполнили все необходимые шаги и в результате у вас получилось приложение, которое скачивает исходный код с выводом в консоль указанной страницы сайта.

#### Критерии для самопроверки

При запуске JAR-файла с параметром в виде URL-адреса в консоли должен выводиться HTML-код страницы. Если этого не происходит, прочитайте вывод результата в консоль, при необходимости используйте переводчик.

Частые причины ошибки:

* неправильно указан путь до jar файла при запуске, проверьте пути и папку из которой вы запускаете консольные команды
* ошибка в указании параметра, например формат URL до сайта
* если не работает команда Maven, выполните clean и попробуйте снова
* не найдена команда java - у вас не установлена Java 17 в операционной системе, выполните установку.

### Задание 2. Отработка базовых навыков работы с Gradle

##### Шаг 1. Добавление Gradle в проект

1. Откройте среду разработки IntelliJ IDEA и проект, в который хотите добавить поддержку Gradle.
2. Создайте в корне проекта файл с названием build.gradle. Внутри него разместите код:

plugins {

id 'java'

}

group = 'org.example'

version = '1.0-SNAPSHOT'

repositories {

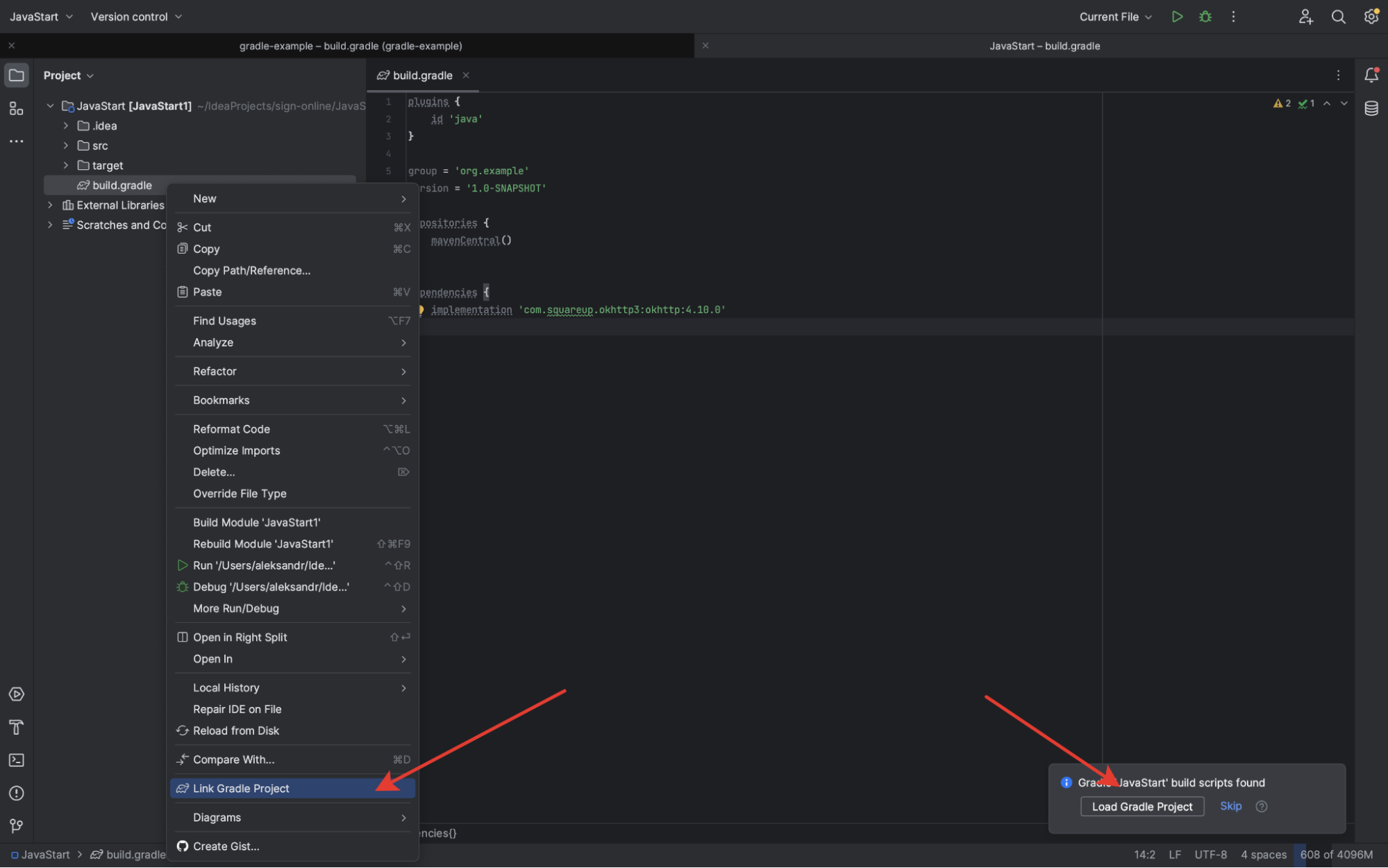
mavenCentral()

}

dependencies {

implementation 'com.squareup.okhttp3:okhttp:4.10.0'

}

1. Среда разработки с помощью уведомления в правом нижнем углу оповестит вас о том, что найден файл build.gradle.  
   
2. Если этого не произошло, нажмите правой кнопкой мыши на файл build.gradle и в открывшемся меню выберите Link Gradle Project.

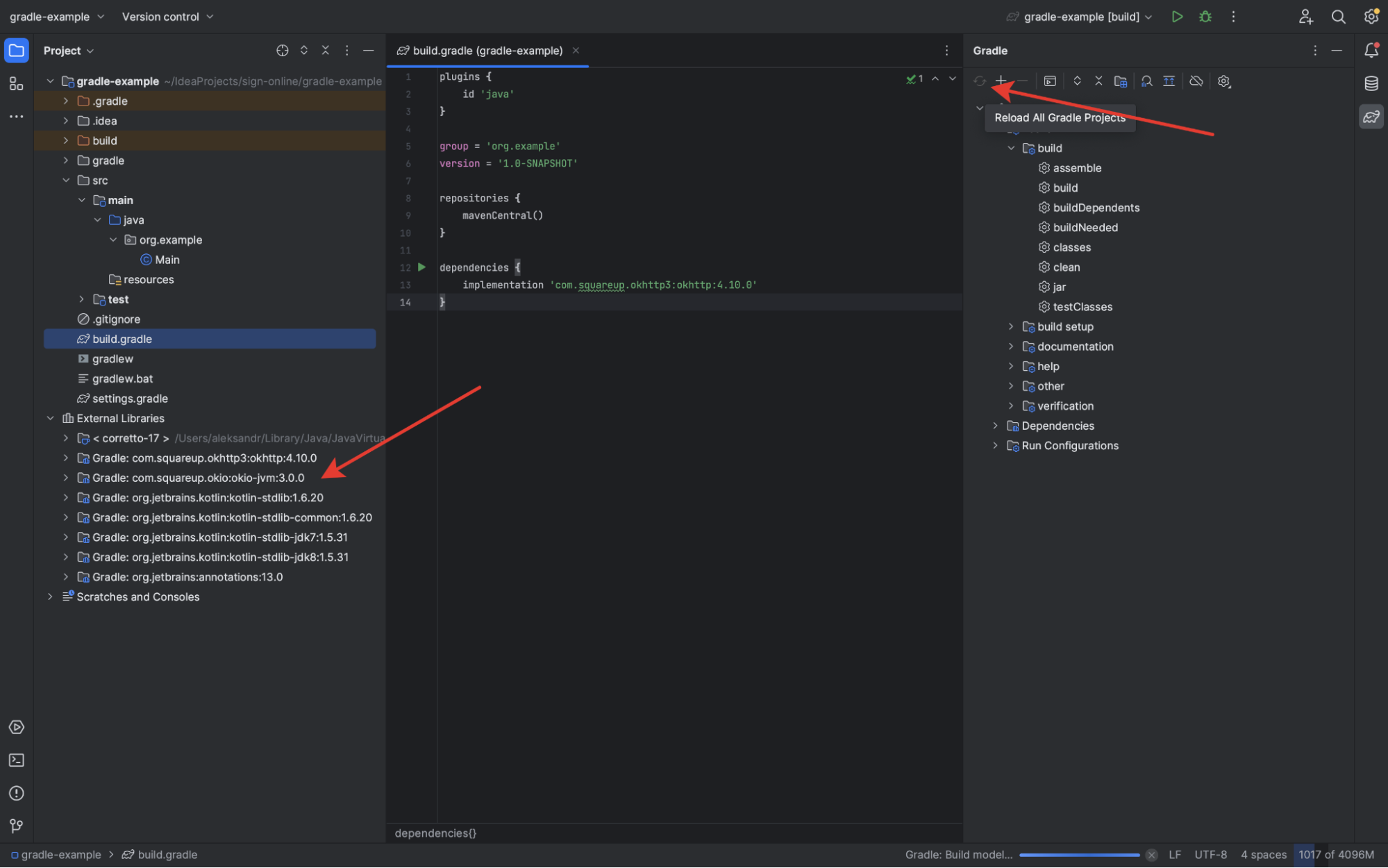
##### Шаг 2. Добавление библиотеки в проект

1. Добавьте зависимость в библиотеку. Для этого необходимо добавить зависимость в build.gradle-файл.

dependencies {

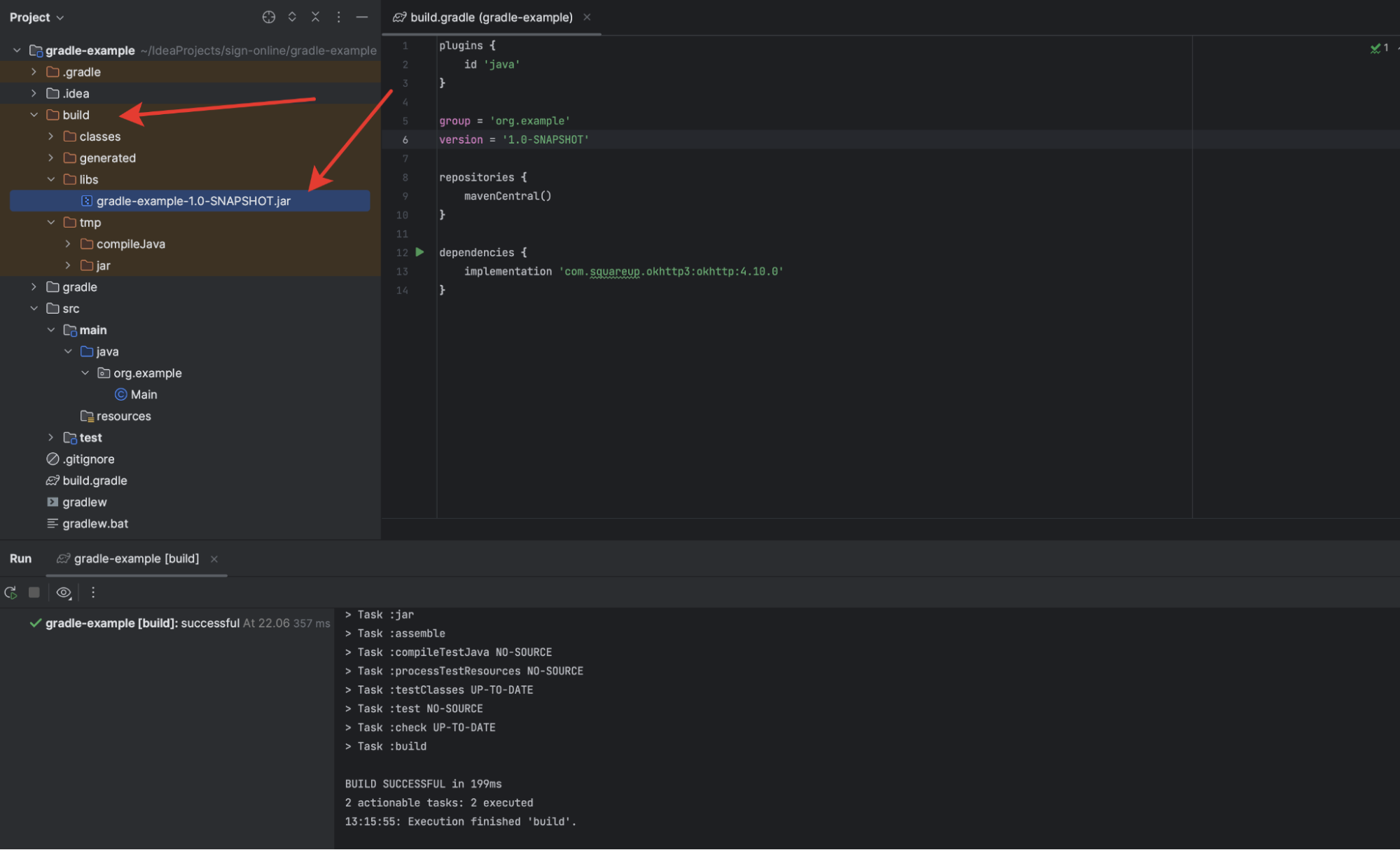
implementation 'com.squareup.okhttp3:okhttp:4.10.0'

}

1. Нажмите на вкладку Reload All Gradle Projects и проверьте, что библиотека успешно импортирована и отображается в External Libraries.  
   

##### Шаг 3. Сборка JAR-файла

Запустите команду build для сборки JAR-файла. Вы получите результат — лог в консоли. Build Success!



##### 

##### Шаг 4. Запуск JAR-файла

1. Запустите JAR-файл, как было показано на шаге 5 задания 1.
2. Убедитесь, что результат соответствует ожидаемому.

##### 

##### Критерии для самопроверки

При запуске JAR-файла с параметром в виде URL-адреса в консоли должен выводиться HTML-код страницы. Если этого не происходит, прочитайте вывод результата в консоль, при необходимости используйте переводчик.

Частые причины ошибки:

* неправильно указан путь к JAR-файлу при запуске — проверьте путь и папку, из которой вы запускаете консольные команды;
* ошибка в указании параметра, например формат URL для сайта;
* не найдена команда java — у вас не установлена Java 17 в операционной системе, выполните установку.

### Задание 3. Отработка базовых навыков работы с Docker

В этом задании необходимо скачать и поднять docker-образ на примере базы данных MySQL. Обращаем ваше внимание, что подробнее про работу с MySQL вы узнаете в следующих модулях. Это задание позволит вам подготовить инфраструктуру для работы с базами данных. А также аналогичным способом вы сможете разворачивать другие базы данных и сервисы с помощью Docker.

Для установки Docker следует обратиться к инструкциям на официальном сайте.

* Windows — [Install Docker Desktop on Windows](https://docs.docker.com/desktop/install/windows-install/).
* Linux — [Install Docker Desktop on Linux](https://docs.docker.com/desktop/install/linux-install/).
* Mac — [Install Docker Desktop on Mac](https://docs.docker.com/desktop/install/mac-install/).

#### Шаг 1. Загрузка docker-образа

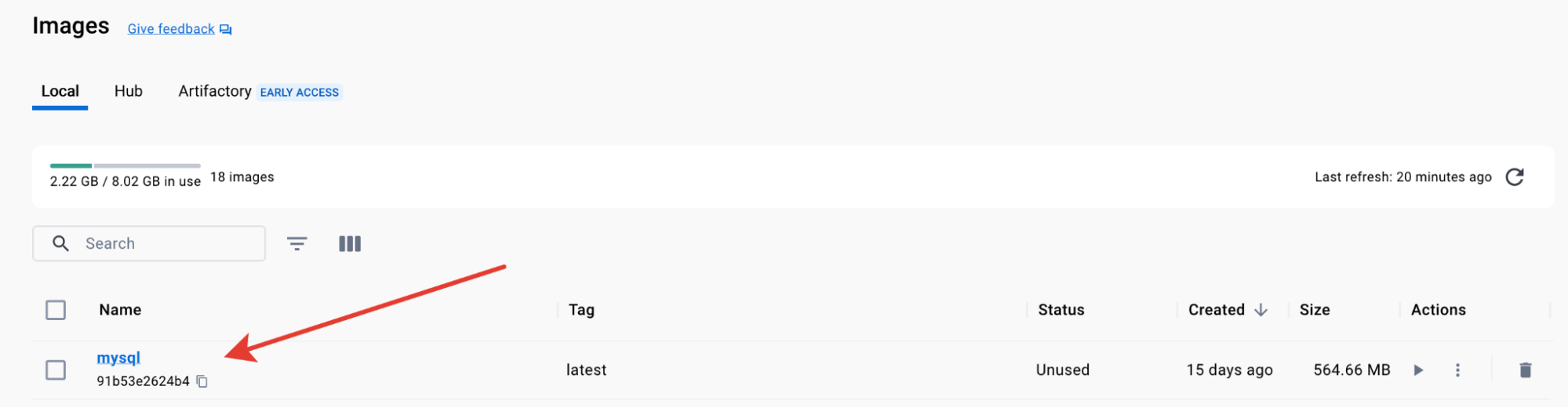
##### Загрузка docker-образа через терминал

1. Откройте терминал и скачайте образ MySQL с Docker Hub (удалённого репозитория для хранения образов) с помощью команды docker pull mysql.  
     
   Начнётся загрузка образа, это займёт некоторое время.
2. Выполните команду docker images и убедитесь, что образ загружен.  
   

##### Загрузка docker-образа через desktop-приложение

1. Откройте Docker и в главном меню Docker Desktop, в строке поиска в верхней панели напишите MySql.
2. Вы увидите ряд образов, нас интересует первый. Выберите его и нажмите Pull. Загрузка займёт некоторое время.  
   

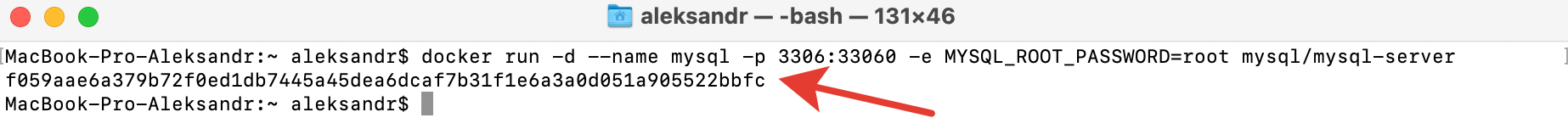
По завершении операции вы увидите следующее:

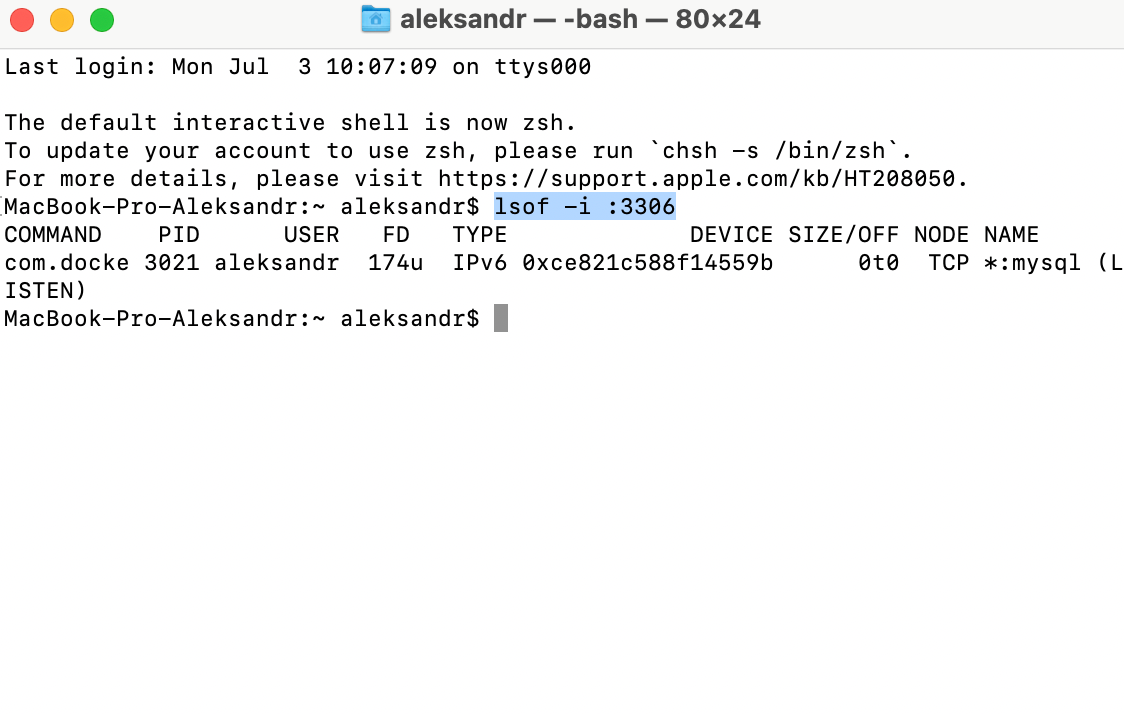


В списке образов теперь есть MySQL-образ.

#### Шаг 2. Поднятие контейнера из docker-образа

##### Поднятие контейнера через терминал

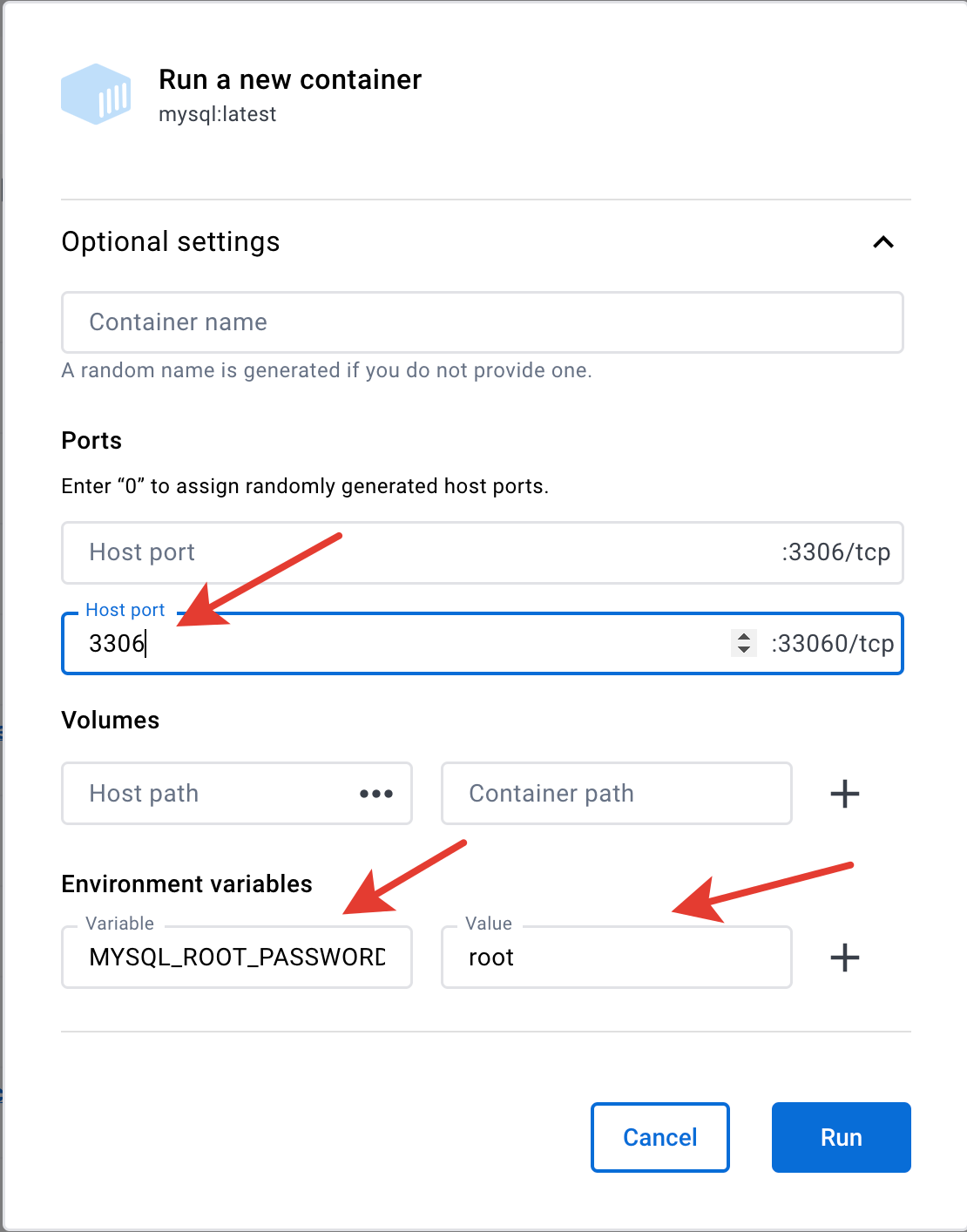
1. Выполните команду для запуска контейнера из образа MySQL с маппингом портов и установкой параметра MYSQL\_ROOT\_PASSWORD:  
   **docker run -d --name mysql -p 3306:33060 -e MYSQL\_ROOT\_PASSWORD=root mysql/mysql-server  
   **
2. В результате вы получите ID запущенного контейнера.
3. Проверьте, занят ли теперь порт 3306. Команда, с помощью которой можно это сделать:
   * Linux/Mac: lsof -i :3306
   * Windows: netstat –a



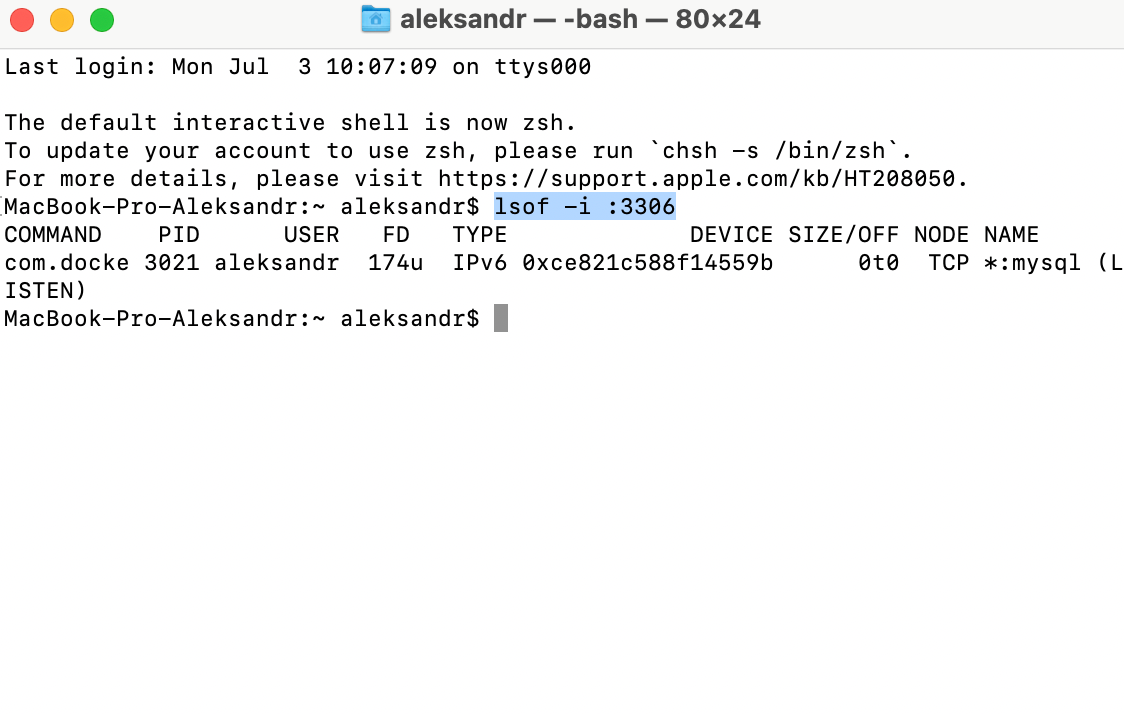
##### Поднятие контейнера через desktop-приложение

1. Чтобы поднять контейнер, нажмите на run справа от нужного образа MySQL.  
   
2. В открывшемся окне задайте ряд настроек:
   * Проброс порта — 3306:33060.
   * MYSQL\_ROOT\_PASSWORD — параметр, который задаёт пароль (последняя версия образа не запустилась без этой настройки) и значение root.

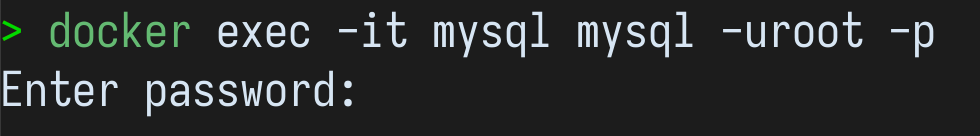
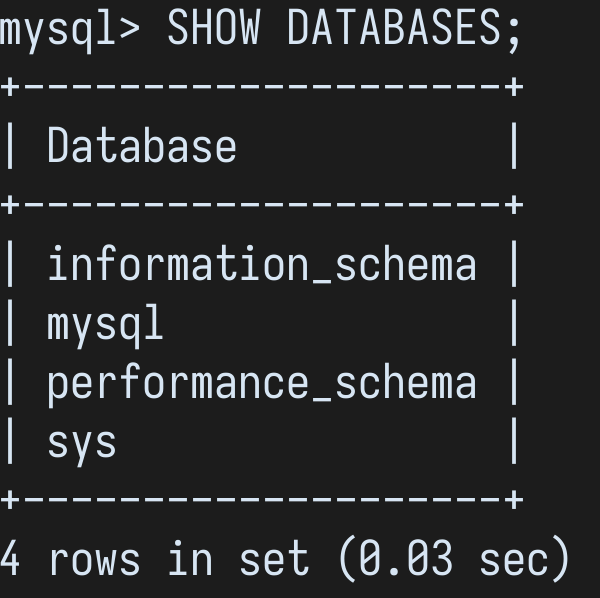
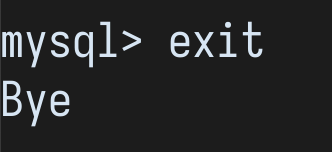
Нажмите Run.



1. Проверьте, занят ли теперь порт 3306. Команда, с помощью которой можно это сделать:
   * Linux/Mac: lsof -i :3306
   * Windows: netstat –a



Зайдите в mysql внутри контейнера и посмотрите список баз данных следующим образом:

1. Выполните docker exec для запуска mysql консоли внутри контейнера:  
   **docker exec -it mysql mysql -uroot -p**Таким образом будет запущена консольная команда mysql с параметрами -uroot -p в контейнере с именем mysql.
2. Введите пароль root. При вводе пароль не будет отображаться, но символы будут вводиться.  
   
3. Введите SQL команду для показа всех баз данных на сервере:  
   
4. По окончании работы, введите команду exit для выхода из консоли MySQL.  
   

##### Критерии для самопроверки

Задание выполнено успешно, если вам удалось зайти в контейнер и получить список имеющихся баз данных (шаг 3, пункт 3).