

# Домашнее задание Ивлева Алексея по курсу «Java Core»

## Семинар № 1 «Компиляция и интерпретация кода»

### Задание

Разработать приложение с лекции все-таки придется.

Создать проект из трёх классов (основной с точкой входа и два класса в другом пакете), которые вместе должны составлять одну программу, позволяющую производить четыре основных математических действия и осуществлять форматированный вывод результатов пользователю (ИЛИ ЛЮБОЕ ДРУГОЕ ПРИЛОЖЕНИЕ НА ВАШ ВЫБОР, которое просто демонстрирует работу некоторого механизма).

Пример моего приложения я прикрепил к материалам урока.

Необходимо установить Docker Desktop.

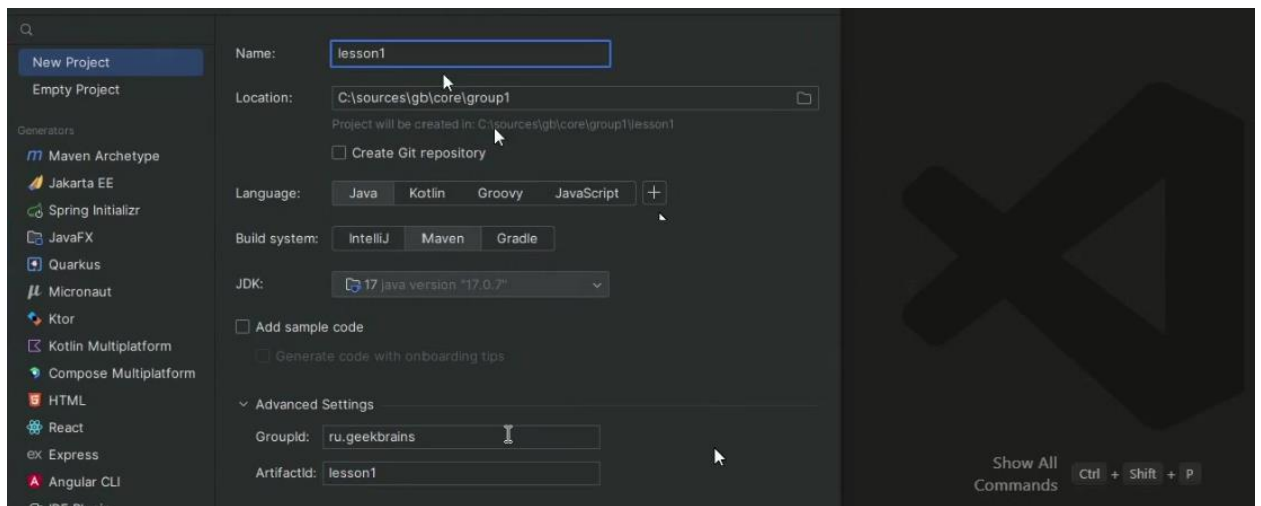
Создать Dockerfile, позволяющий откопировать исходный код вашего приложения в образ для демонстрации работы вашего приложения при создании соответствующего контейнера.

Подобную процедуру мы с вами проделали на уроке, теперь необходимо проделать данную процедуру самостоятельно.

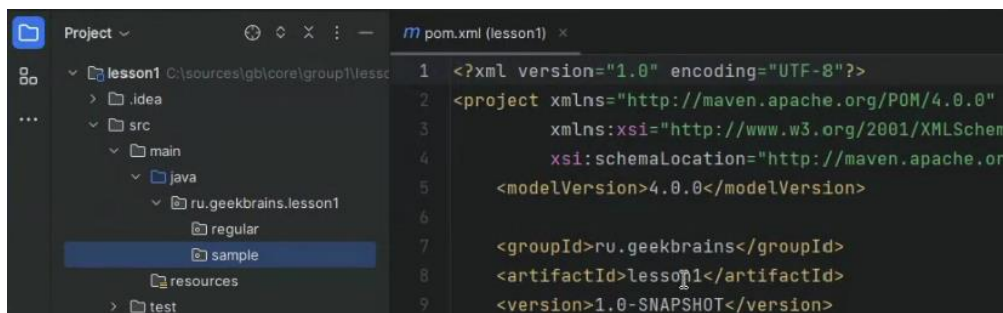
### ЧАСТЬ I: РАБОТА В INTELLIJ IDEA

IntelliJ IDEA - это интегрированная среда разработки (IDE) для языков программирования Java, Kotlin, Groovy, Scala и других. Она предоставляет широкий спектр инструментов для разработки программного обеспечения, включая автодополнение кода, отладку, анализ кода и многое другое. IntelliJ IDEA также имеет встроенную поддержку версий управления, таких как Git и Subversion, что упрощает совместную работу над проектами.

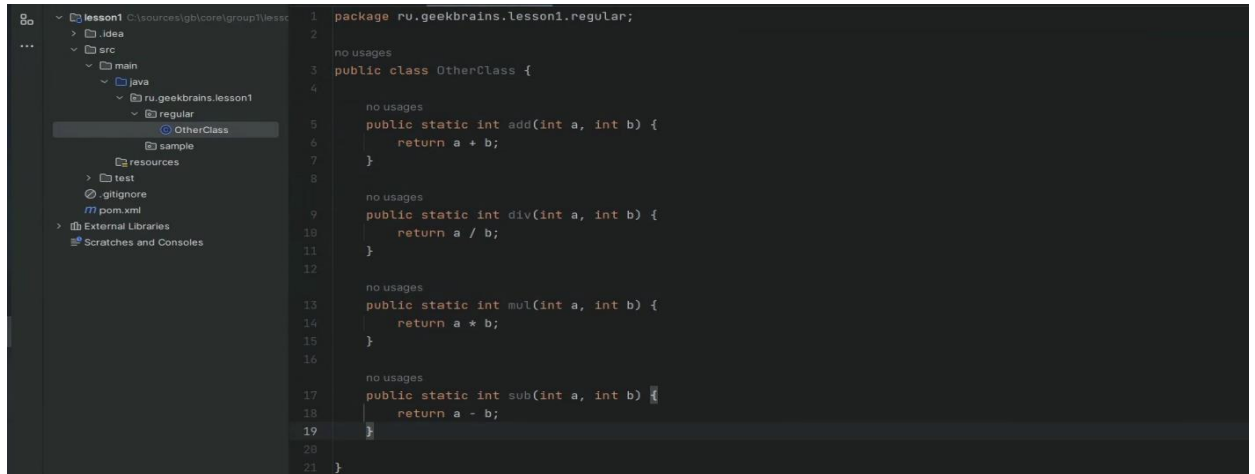
#### Шаг № 1 Создаем приложение в IntelliJ IDEA



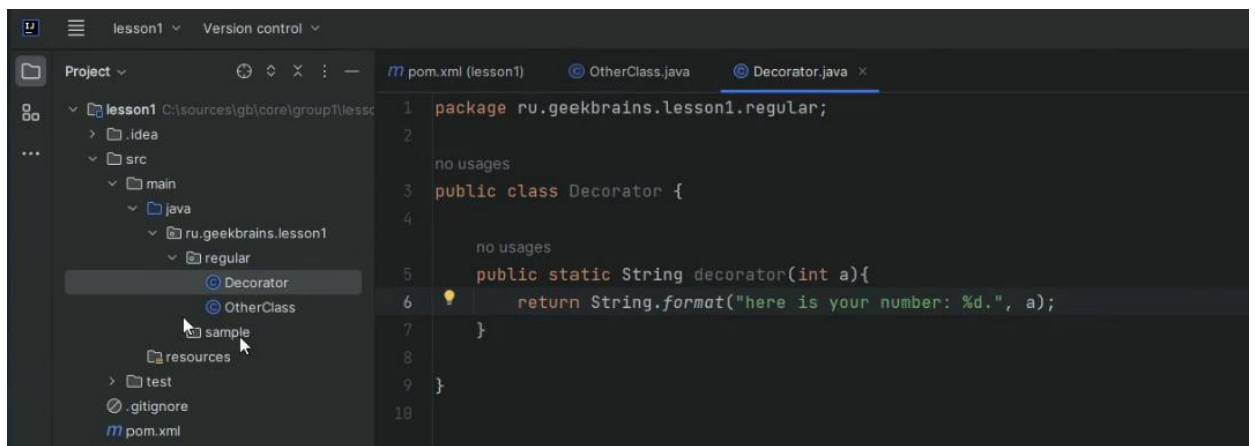
#### Шаг № 2 Создаем пакеты



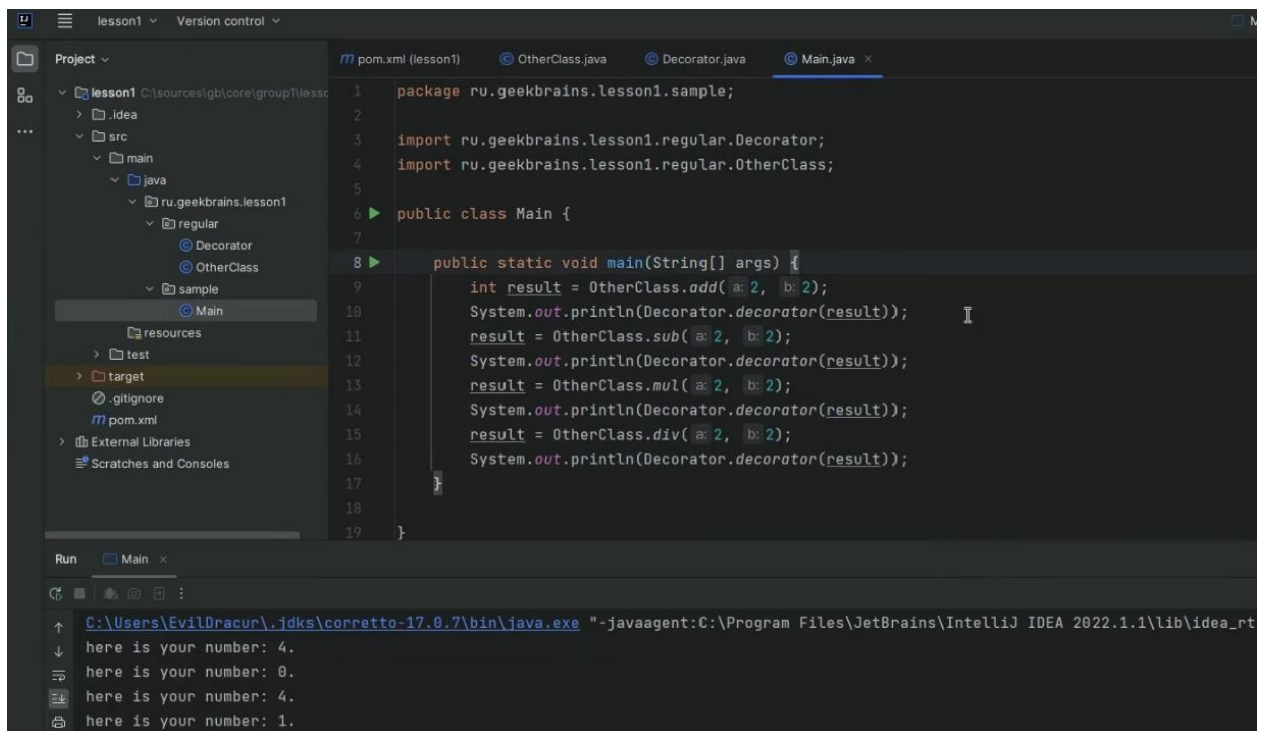
### Шаг № 3 Создаем класс для математических вычислений с двумя цифрами



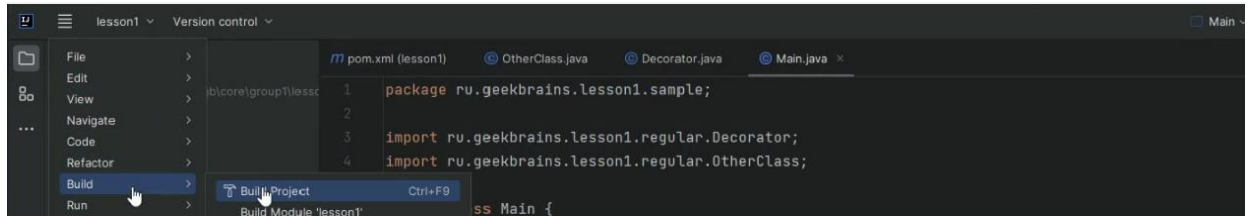
### Шаг № 4 Создадим класс (паттерн) Декоратор, который динамически добавляет объекту новые обязанности.



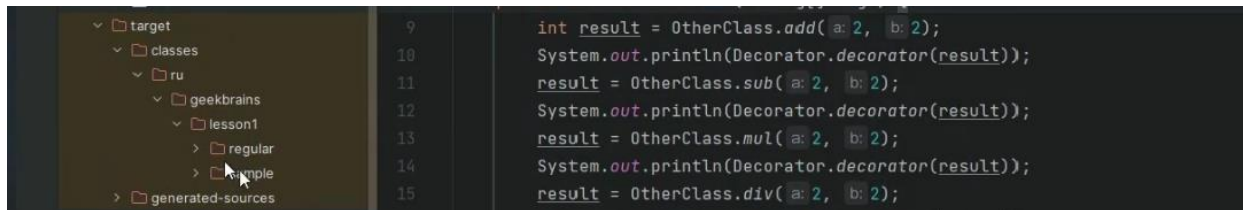
### Шаг № 5 Создадим класс Main (точка входа) в папке sample



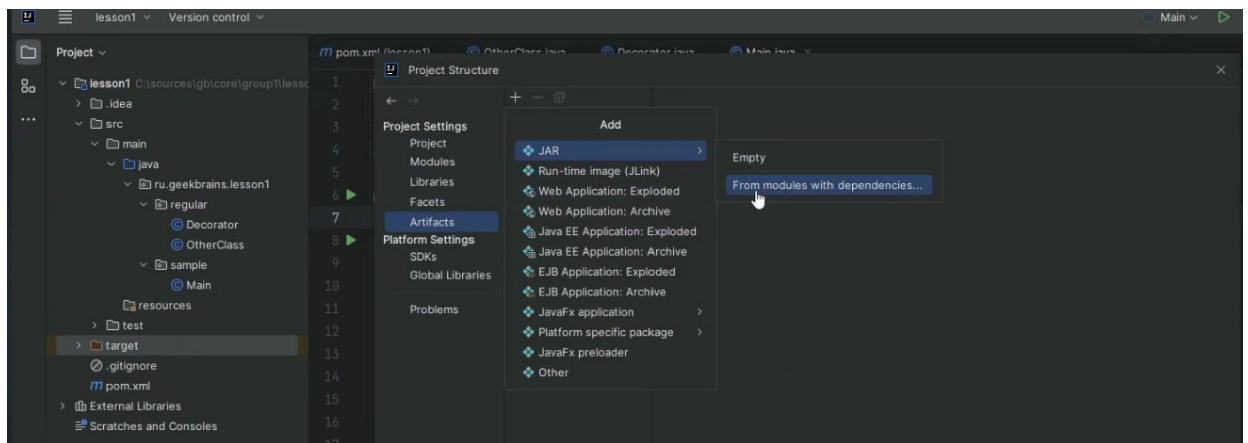
## Шаг № 6 Собрать проект с помощью команды Build Project



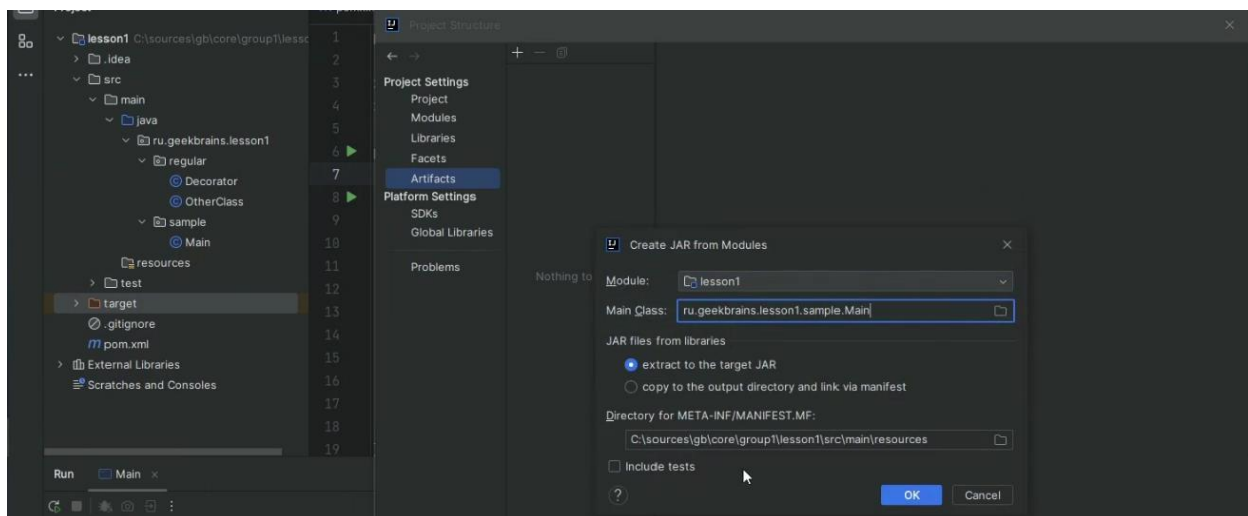
## Проверить формирование директории проекта



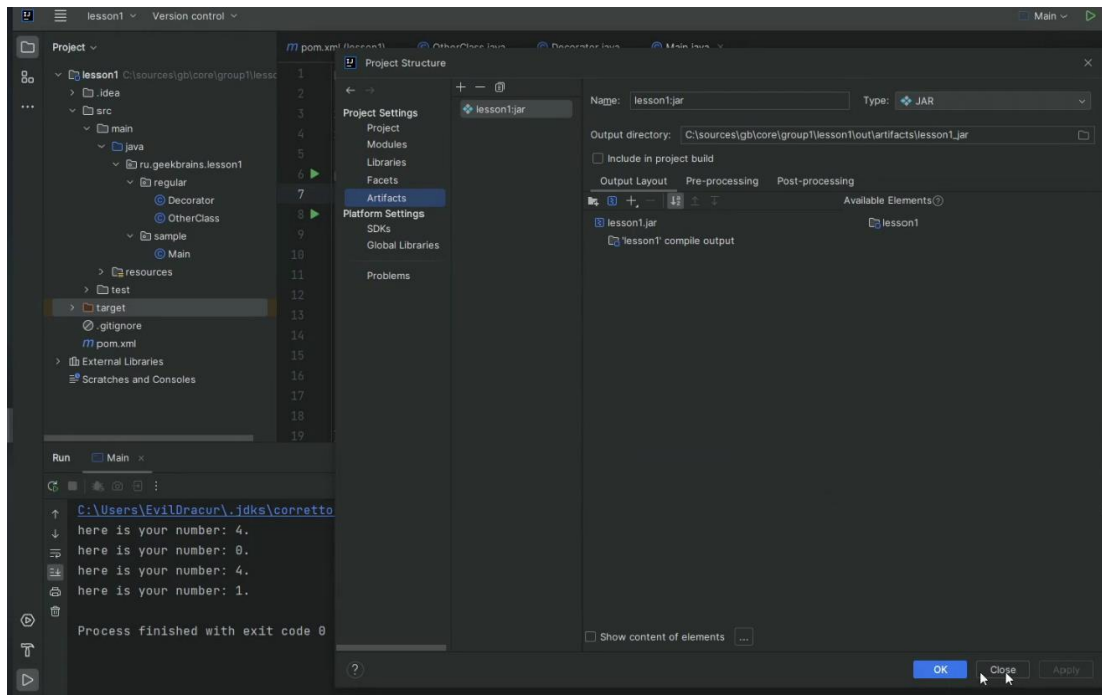
## Шаг № 7 заходим в структуру проекта и переходим в артефакты



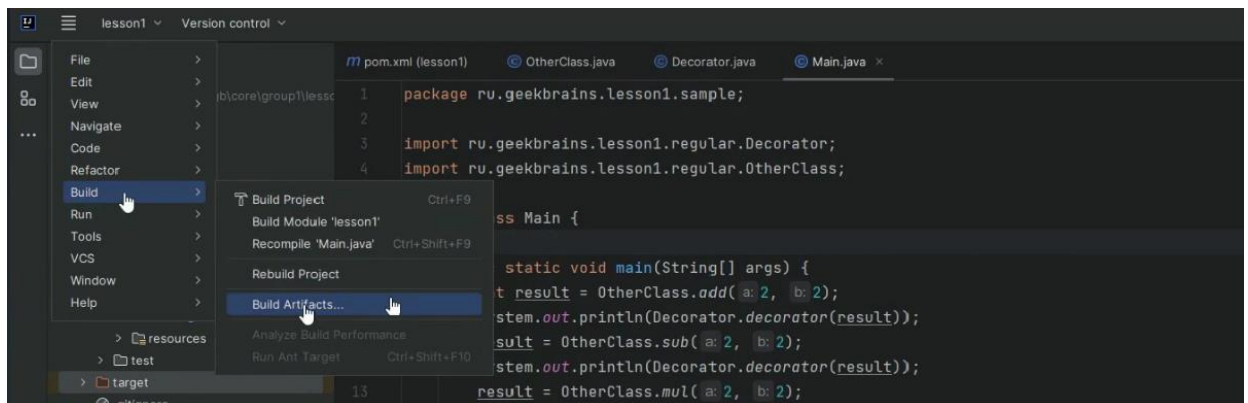
## Шаг № 8 Указываем в зависимостях точку входа, нажимаем Ок.



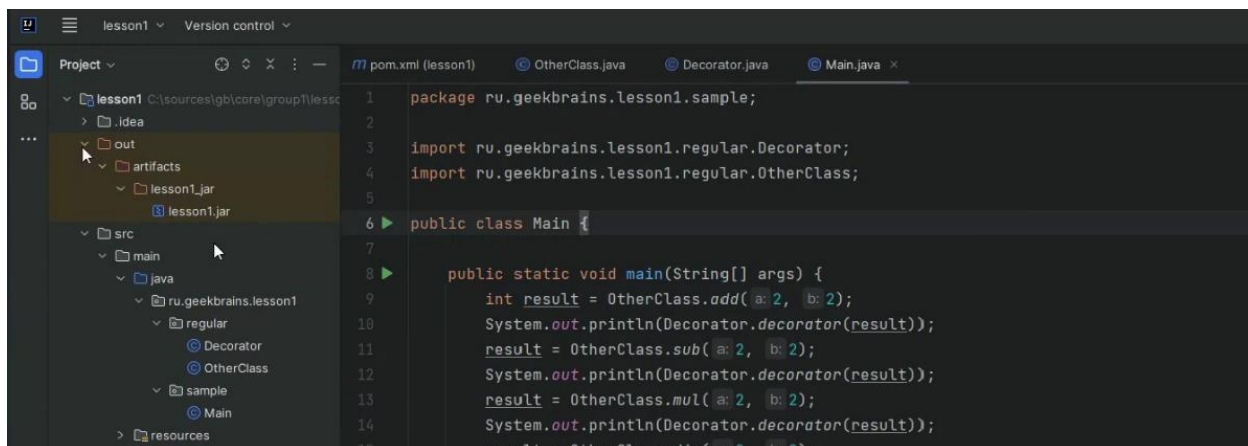
## Шаг № 9 Нажимаем apply и Ок



## Шаг № 10 В пункте меню на закладке билд появляется Build Artifacts, нажимаем ее и контролируем появление директории out.



## Шаг № 11 контролируем появление дистрибутива jar-архив, который на флешке можно передать заказчику



## ЧАСТЬ II: РАБОТА В ТЕРМИНАЛЕ

**Шаг № 1** Запустим терминал в VSK, скопируем путь к архивному файлу и перейдем в директорию, в которой он расположен

Microsoft Windows [Version 10.0.19044.3086]

(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

```
C:\Users\USER1> cd C:\Users\USER1\IdeaProjects\seminar1\out\artifacts\seminar1_jar
```

```
C:\Users\USER1\IdeaProjects\seminar1\out\artifacts\seminar1_jar>java -jar seminar1.jar
```

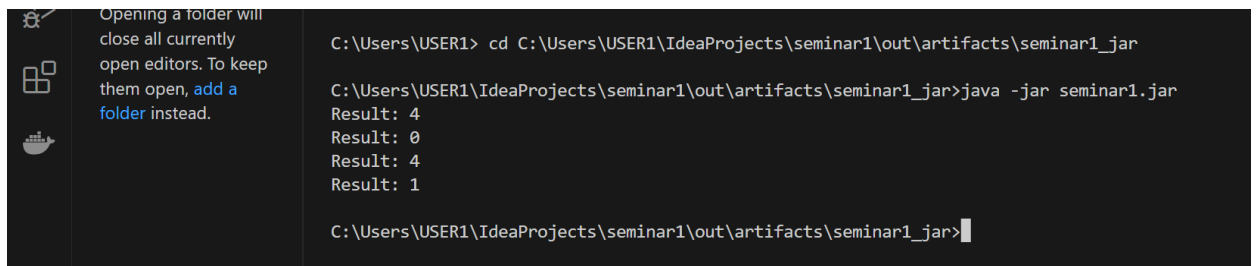
Result: 4

Result: 0

Result: 4

Result: 1

```
C:\Users\USER1\IdeaProjects\seminar1\out\artifacts\seminar1_jar>
```



```

C:\Users\USER1> cd C:\Users\USER1\IdeaProjects\seminar1\out\artifacts\seminar1_jar

C:\Users\USER1\IdeaProjects\seminar1\out\artifacts\seminar1_jar>java -jar seminar1.jar
Result: 4
Result: 0
Result: 4
Result: 1

C:\Users\USER1\IdeaProjects\seminar1\out\artifacts\seminar1_jar>

```

**Шаг № 2** Скомпилируем приложение при помощи утилиты `javac`

```
C:\Users\USER1\IdeaProjects\seminar1\out\artifacts\seminar1_jar>javac
```

Usage: javac <options> <source files>

where possible options include:

```

@<filename>          Read options and filenames from file
-Akey[=value]         Options to pass to annotation processors
--add-modules <module>(<module>)*
    Root modules to resolve in addition to the initial modules, or all modules
    on the module path if <module> is ALL-MODULE-PATH.
--boot-class-path <path>, -bootclasspath <path>
    Override location of bootstrap class files
--class-path <path>, -classpath <path>, -cp <path>
    Specify where to find user class files and annotation processors
-d <directory>        Specify where to place generated class files
-deprecation
    Output source locations where deprecated APIs are used
--enable-preview
    Enable preview language features. To be used in conjunction with either -source
or --release.
-encoding <encoding>  Specify character encoding used by source files
-endorseddirs <dirs>  Override location of endorsed standards path
-extdirs <dirs>       Override location of installed extensions
-g                   Generate all debugging info
-g:{lines,vars,source} Generate only some debugging info
-g:none             Generate no debugging info
-h <directory>
    Specify where to place generated native header files
--help, -help, -?     Print this help message
--help-extra, -X      Print help on extra options
-implicit:{none,class}
    Specify whether or not to generate class files for implicitly referenced files
-J<flag>              Pass <flag> directly to the runtime system
--limit-modules <module>(<module>)*
    Limit the universe of observable modules
--module <module>(<module>)*, -m <module>(<module>)*
    Compile only the specified module(s), check timestamps

```



```

--module-path <path>, -p <path>
    Specify where to find application modules
--module-source-path <module-source-path>
    Specify where to find input source files for multiple modules
--module-version <version>
    Specify version of modules that are being compiled
--nowarn
    Generate no warnings
--parameters
    Generate metadata for reflection on method parameters
--proc: {none,only}
    Control whether annotation processing and/or compilation is done.
--processor <class1>[,<class2>,<class3>...]
    Names of the annotation processors to run; bypasses default discovery process
--processor-module-path <path>
    Specify a module path where to find annotation processors
--processor-path <path>, -processorpath <path>
    Specify where to find annotation processors
--profile <profile>
    Check that API used is available in the specified profile
--release <release>
    Compile for the specified Java SE release. Supported releases: 7, 8, 9, 10, 11,
12, 13, 14, 15, 16, 17
-s <directory>
    Specify where to place generated source files
--source <release>, -source <release>
    Provide source compatibility with the specified Java SE release. Supported releases: 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17
--source-path <path>, -sourcepath <path>
    Specify where to find input source files
--system <jdk>|none
    Override location of system modules
--target <release>, -target <release>
    Generate class files suitable for the specified Java SE release. Supported releases: 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17
--upgrade-module-path <path>
    Override location of upgradeable modules
-verbose
    Output messages about what the compiler is doing
--version, -version
    Version information
-Werror
    Terminate compilation if warnings occur
C:\Users\USER1\IdeaProjects\seminar1\out\artifacts\seminar1_jar>

```

### Шаг № 3 перейдем в директорию main

cd C:\Users\USER1\IdeaProjects\seminar1\src\main

Укажем источник файлов: javac -sourcepath ./java

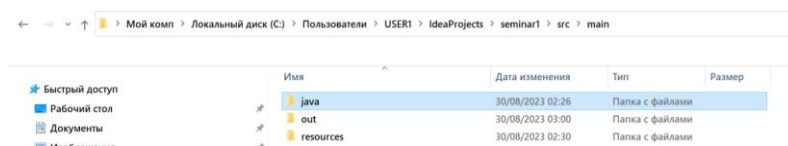
Укажем директорию, в которую поместим скомпилированные файлы: -d out

Укажем точку входа: java\ru\geekbrains\sample\Main.java

C:\Users\USER1\IdeaProjects\seminar1\src\main>javac -sourcepath ./java -d out  
java\ru\geekbrains\sample\Main.java

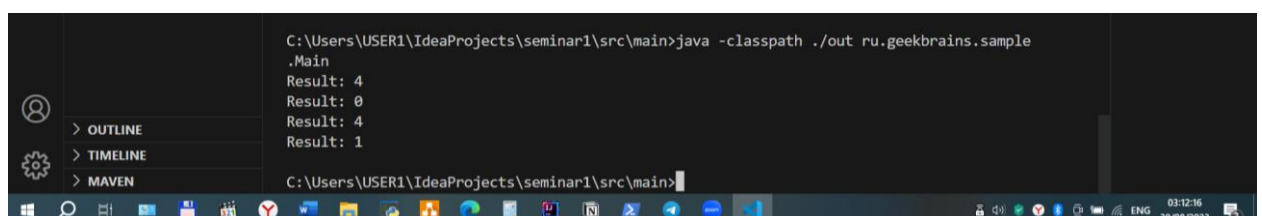
C:\Users\USER1\IdeaProjects\seminar1\src\main>

Контроль: появилась директория out



### Шаг № 4 Запустим программу из созданной директории

java -classpath ./out ru.geekbrains.sample.Main



## Шаг № 5 Создадим документацию к программе

`javadoc -encoding utf8 -d docs -sourcepath ./java -cp ./out -subpackages ru`

```

open editors. To keep them open, add a folder instead.
C:\Users\USER1\IdeaProjects\seminar1\src\main> javadoc -d docs -sourcepath ./java -cp ./out -subpackages ru
Loading source files for package ru...
Constructing Javadoc information...
Creating destination directory: "docs\"
Building index for all the packages and classes...
Standard Doclet version 17.0.8.1+1
Building tree for all the packages and classes...
Generating docs\ru\geekbrains\regular\Calculator.html...
Generating docs\ru\geekbrains\regular\Decorator.html...
Generating docs\ru\geekbrains\sample\Main.html...
Generating docs\ru\geekbrains\regular\package-summary.html...
Generating docs\ru\geekbrains\regular\package-tree.html...
Generating docs\ru\geekbrains\sample\package-summary.html...
Generating docs\ru\geekbrains\sample\package-tree.html...
Generating docs\overview-tree.html...
Generating docs\index.html...
Building index for all classes...

```

## Шаг № 6 Контролируем результат создания документации

- появилась папка docs

← → ▾ ▴ > Мой комп > Локальный диск (C:) > Пользователи > USER1 > IdeaProjects > seminar1 > src > main

Имя	Дата изменения	Тип	Размер
docs	30/08/2023 12:54	Папка с файлами	
java	30/08/2023 02:26	Папка с файлами	
out	30/08/2023 03:00	Папка с файлами	
resources	30/08/2023 02:30	Папка с файлами	

Имя	Дата изменения	Тип	Размер
legal	30/08/2023 12:54	Папка с файлами	
resources	30/08/2023 12:54	Папка с файлами	
ru	30/08/2023 12:54	Папка с файлами	
script-dir	30/08/2023 12:54	Папка с файлами	
allclasses-index.html	30/08/2023 12:54	Yandex Browser ...	5 КБ
allpackages-index.html	30/08/2023 12:54	Yandex Browser ...	3 КБ
element-list	30/08/2023 12:54	Файл	1 КБ
help-doc.html	30/08/2023 12:54	Yandex Browser ...	9 КБ
index.html	30/08/2023 12:54	Yandex Browser ...	3 КБ
index-all.html	30/08/2023 12:54	Yandex Browser ...	9 КБ
jquery-ui.overrides.css	30/08/2023 12:54	CSS-документ	2 КБ
member-search-index.js	30/08/2023 12:54	файл JavaScript	1 КБ
module-search-index.js	30/08/2023 12:54	файл JavaScript	1 КБ
overview-summary.html	30/08/2023 12:54	Yandex Browser ...	1 КБ
overview-tree.html	30/08/2023 12:54	Yandex Browser ...	4 КБ
package-search-index.js	30/08/2023 12:54	файл JavaScript	1 КБ
script.js	30/08/2023 12:54	файл JavaScript	6 КБ
search.js	30/08/2023 12:54	файл JavaScript	14 КБ
stylesheet.css	30/08/2023 12:54	CSS-документ	20 КБ
tag-search-index.js	30/08/2023 12:54	файл JavaScript	1 КБ
type-search-index.js	30/08/2023 12:54	файл JavaScript	1 КБ

- открываем файл index.html

file:///C:/Users/USER1/IdeaProjects/seminar1/src/main/docs/index.html

OVERVIEW PACKAGE CLASS TREE INDEX HELP

SEARCH: Search

Package	Description
ru.geekbrains.regular	
ru.geekbrains.sample	

Package ru.geekbrains.regular

## Class Calculator

java.lang.Object<sup>®</sup>  
ru.geekbrains.regular.Calculator

```
public class Calculator
extends Object®
```

Модуль реализует класс Calculator, предназначенный для осуществления математических операций с целыми числами. Содержит четыре метода, позволяющие выполнять умножения, деления, вычитания и сложения двух целых чисел.

### Constructor Summary

Constructor	Description
Calculator()	

### Method Summary

Modifier and Type	Method	Description
static int	add(int a, int b)	add - метод, позволяющий сложить два целых числа.
static int	div(int a, int b)	div - метод деления двух целых чисел.



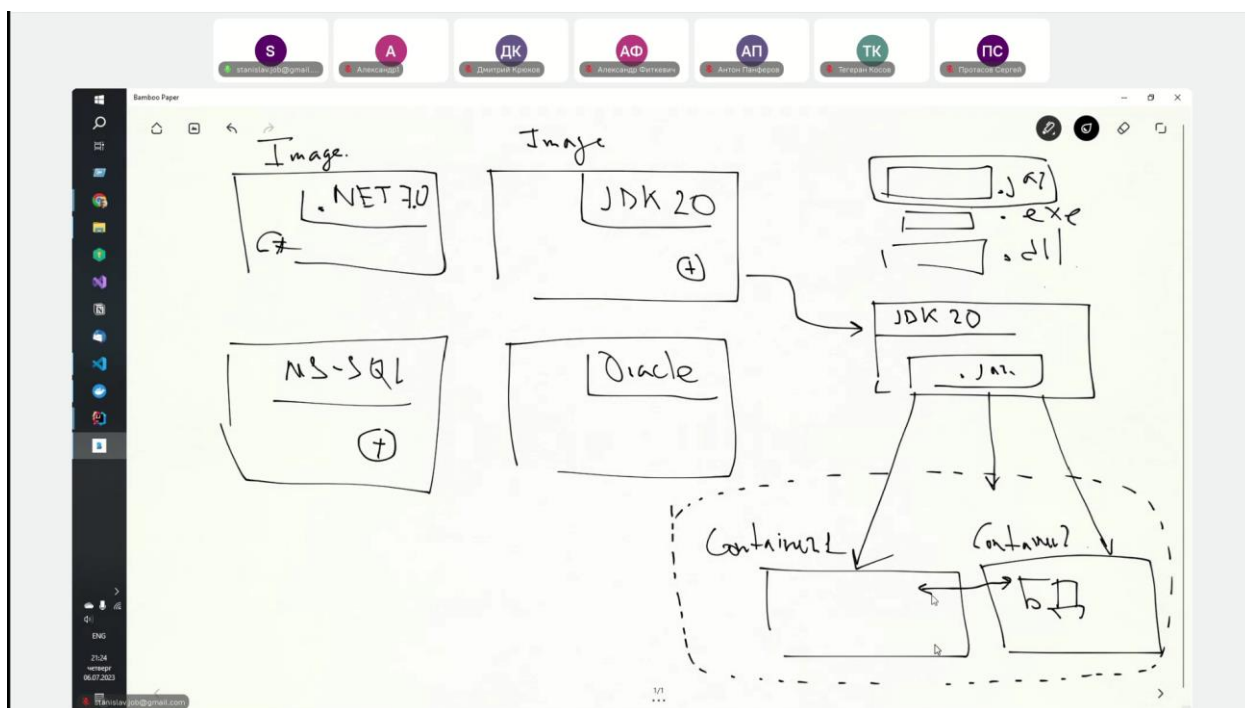
## ЧАСТЬ III: РАБОТА В DOCKER

**Docker** – это программное обеспечение для автоматизации развертывания, доставки и управления приложениями. Он позволяет создавать и запускать приложения в контейнерах, которые изолированы от других приложений и операционных систем.

**Docker решает множество задач, включая:**

1. **Изоляцию приложений:** Docker создает контейнеры, которые являются изолированными средами для запуска приложений. Это позволяет избежать проблем с конфликтами между приложениями и операционными системами.
2. **Развертывание приложений:** Docker позволяет быстро и легко развертывать приложения на различных платформах. Контейнеры могут быть запущены на любой машине с поддержкой Docker, что упрощает процесс развертывания.
3. **Управление зависимостями:** Docker позволяет управлять зависимостями приложений, что упрощает обновление и поддержание приложений.
4. **Масштабируемость:** Docker обеспечивает масштабируемость приложений, что позволяет легко увеличивать производительность и емкость приложений.
5. **Безопасность:** Docker обеспечивает безопасность приложений, так как контейнеры изолированы друг от друга и от операционной системы.
6. **Микросервисная архитектура:** Docker может использоваться для создания микросервисной архитектуры, которая позволяет разделить приложения на мелкие компоненты и управлять ими более эффективно.

<http://docs.microsoft.com/windows/wsl/wsl2-kernel>



## Шаг № 1 запустить докер десктоп и узнать его версию

docker version

```
C:\Users\USER1>docker version
Client:
 Cloud integration: v1.0.35-desktop+001
 Version: 24.0.5
 API version: 1.43
 Go version: go1.20.6
 Git commit: ced0996
 Built: Fri Jul 21 20:36:24 2023
 OS/Arch: windows/amd64
 Context: default

Server: Docker Desktop 4.22.1 (118664)
Engine:
 Version: 24.0.5
 API version: 1.43 (minimum version 1.12)
 Go version: go1.20.6
 Git commit: a61e2b4
 Built: Fri Jul 21 20:35:45 2023
 OS/Arch: linux/amd64
 Experimental: false
 containerd:
```

## Шаг № 2 Загрузим последнюю версию jdk

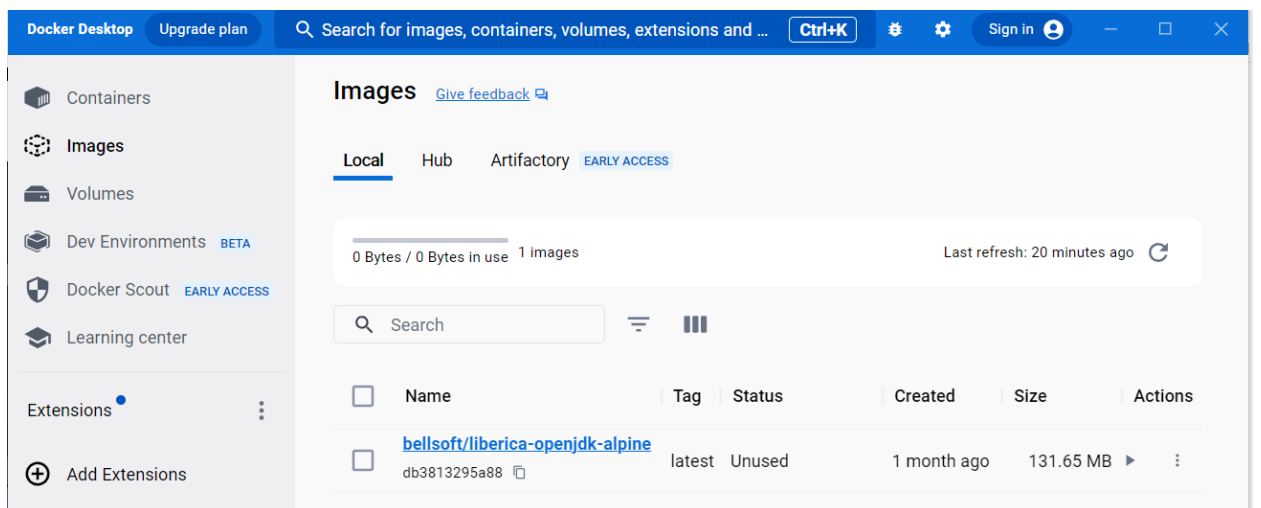
docker pull bellsoft/liberica-openjdk-alpine

```
C:\Users\USER1>docker pull bellsoft/liberica-openjdk-alpine
Using default tag: latest
latest: Pulling from bellsoft/liberica-openjdk-alpine
31e352740f53: Pull complete
c7a74e07fe28: Pull complete
e54bc035902a: Pull complete
Digest: sha256:adb50f2263dfdee07191ffdb64b78e6749049c1e261155c8454928bb0d6e6931
Status: Downloaded newer image for bellsoft/liberica-openjdk-alpine:latest
docker.io/bellsoft/liberica-openjdk-alpine:latest

What's Next?
View summary of image vulnerabilities and recommendations → docker scout quickview bellsoft/liberica-openjdk-alpine

C:\Users\USER1>docker pull bellsoft/liberica-openjdk-alpine
```

Проверим выполнение команды в докер



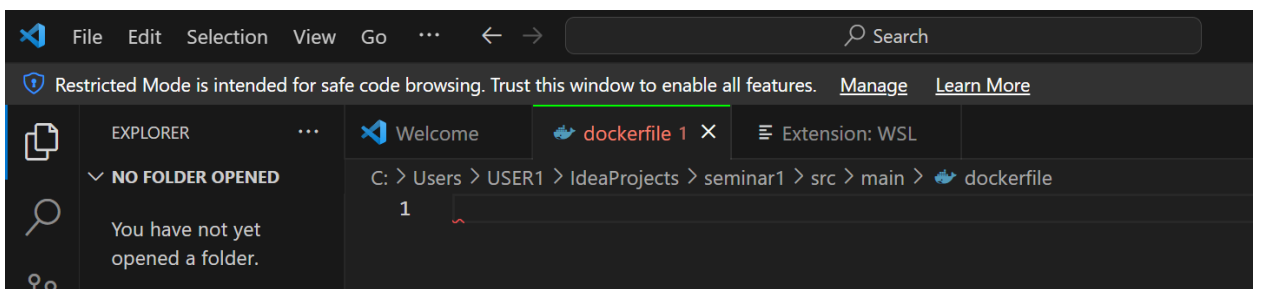
В разделе «образы» появился bellsoft/liberica-openjdk-alpine

## Шаг № 3 На основе скопированного образа jdk создадим собственный образ

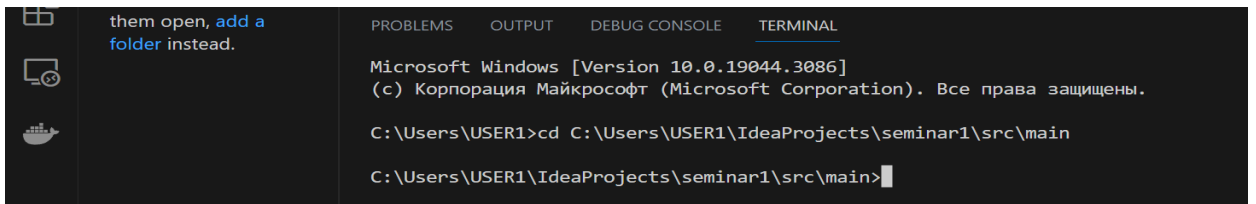
### 3.1. В IntelliJ Idea в папке main создадим dockerfile



### 3.2. Откроем данный файл через редактор VSC



### 3.3. Удостовериться, что мы находимся в директории main, где находится созданный ранее файл dockerfile



### 3.4. Задаем условие копирования нужной версии jdk

```
FROM bellsoft/liberica-openjdk-alpine:11.0.10
```

### 3.5. Указываем директорию, в которой лежат исходники, и путь, куда их нужно скопировать

```
COPY ./java ./src
```

### 3.6. Создадим директорию out

```
RUN mkdir ./out
```

### 3.7. Компилируем наше приложение

```
RUN javac -sourcepath ./src -d out src/ru/geekbrains/sample/Main.java
```

### 3.8. Запустим приложение

```
RUN javac -sourcepath ./src -d out src\ru\geekbrains\sample\Main.java
```

**Итого:**

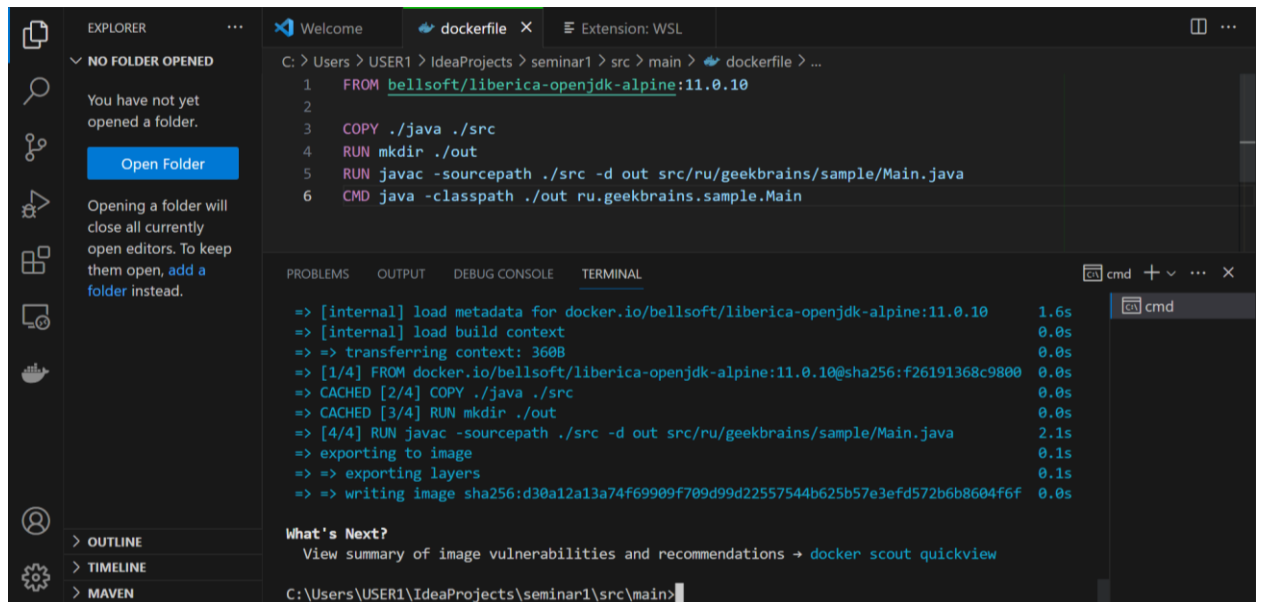
```
FROM bellsoft/liberica-openjdk-alpine:11.0.10
```

```
COPY ./java ./src
```

```
RUN mkdir ./out
```

```
RUN javac -sourcepath ./src -d out src/ru/geekbrains/sample/Main.java
```

```
CMD java -classpath ./out ru.geekbrains.sample.Main
```



### 3.9. Соберем образ и присвоим ему наименование calculator версии № 1

```
C:\Users\USER1\IdeaProjects\seminar1\src\main>docker build . -t calculator:v1
```

### 3.10. Проверим создание образа

```
C:\Users\USER1\IdeaProjects\seminar1\src\main>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
calculator	latest	d30a12a13a74	18 minutes ago	131MB
calculator	v1	d30a12a13a74	18 minutes ago	131MB
bellsoft/liberica-openjdk-alpine	latest	db3813295a88	5 weeks ago	132MB

Name	Tag	Status	Created	Size	Actions
<a href="#">calculator</a>					
d30a12a13a74	latest	Unused	18 minutes ag	131.01 MB	▶ ⋮
<a href="#">calculator</a>					
d30a12a13a74	v1	Unused	18 minutes ag	131.01 MB	▶ ⋮
<a href="#">bellsoft/liberica-openjdk-alpine</a>					
db3813295a88	latest	Unused	1 month ago	131.65 MB	▶ ⋮

## Шаг № 4 Запустим программу из созданного образа (вход, выполнение программы и выход из нее)

C:\Users\USER1\IdeaProjects\seminar1\src\main>**docker run calculator:v1**

```

C:\Users\USER1\IdeaProjects\seminar1\src\main>docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
calculator           latest      d30a12a13a74  25 minutes ago 131MB
calculator           v1         d30a12a13a74  25 minutes ago 131MB
bellsoft/liberica-openjdk-alpine latest      db3813295a88  5 weeks ago   132MB

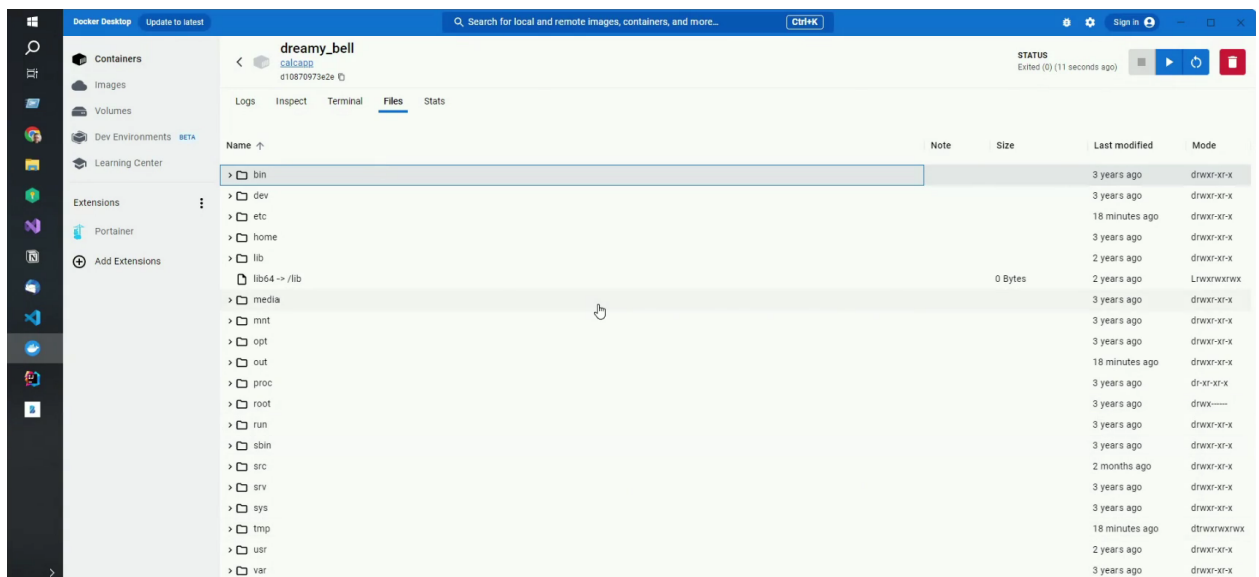
C:\Users\USER1\IdeaProjects\seminar1\src\main>docker run calculator:v1
Result: 4
Result: 0
Result: 4
Result: 1
C:\Users\USER1\IdeaProjects\seminar1\src\main>

```

## Шаг № 5 Запустим программу из созданного образа в интерактивном режиме (вход, выполнение программы и продолжение работы)

C:\Users\USER1\IdeaProjects\seminar1\src\main>**docker run -it calculator:v1**

### В докере можно посмотреть файловую систему



## ИСПОЛЬЗОВАНИЕ ДОКЕР ДЛЯ КОМПИЛЯЦИИ ПРИЛОЖЕНИЯ

### Шаг № 1 Создадим скрипт, позволяющий выполнить компиляцию приложения.

```

FROM bellsoft/liberica-openjdk-alpine:11.0.10 as BuildProject
WORKDIR /app
COPY ./java ./src
RUN mkdir ./out
RUN javac -sourcepath ./src -d out src/ru/geekbrains/sample/Main.java

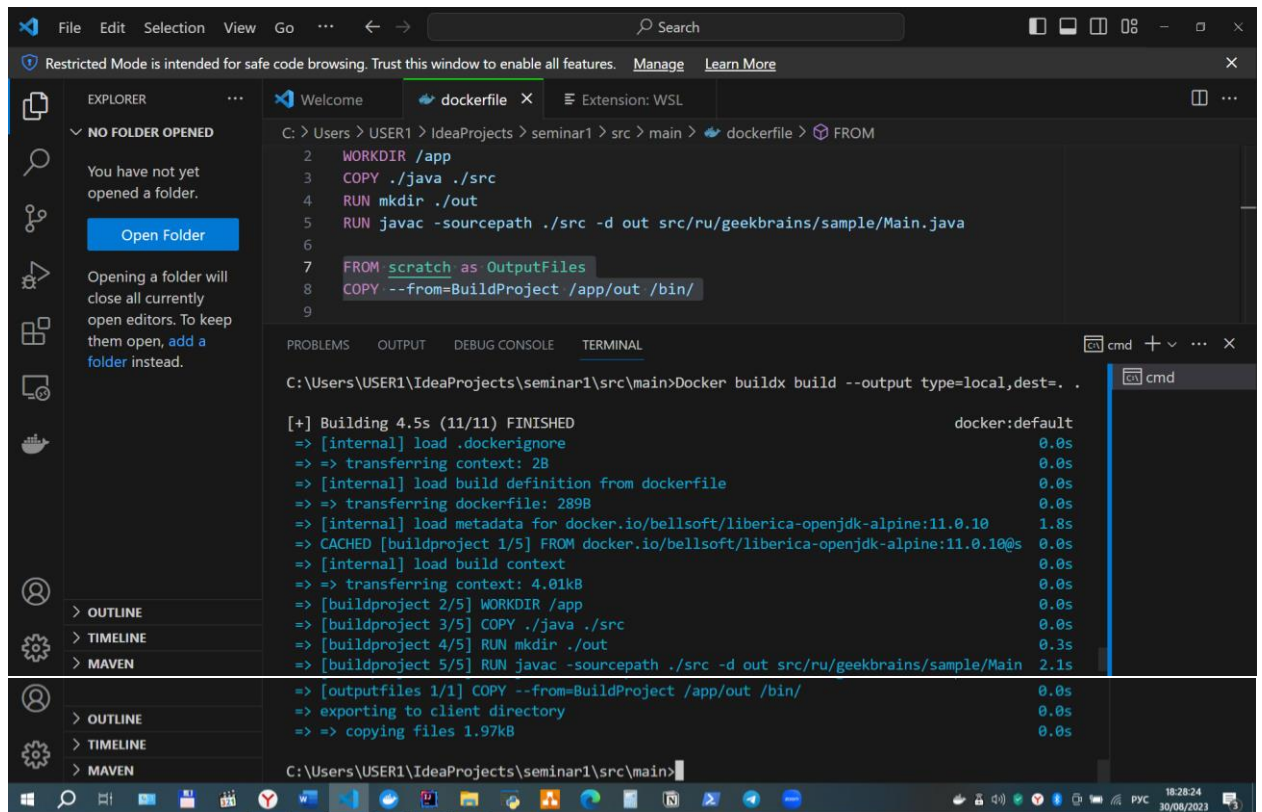
```

**Шаг № 2 Создадим скрипт, выполняющий сценарий копирования файлов, которые сохранены в контейнере назад в компьютер.**

```
FROM scratch as OutputFiles
COPY --from=BuildProject /app/out /bin/
```

**Шаг № 3 Запустим скрипт с использованием расширенных возможностей по компиляции**

**Docker buildx build --output type=local,dest=.**



**Шаг № 4 Проверим выполнение задания: в папку bin две минуты назад скопированы файлы**

