

---

# Java Core (семинары в записи)

## *Урок 1. Компиляция и интерпретация кода*

---

### **Оглавление**

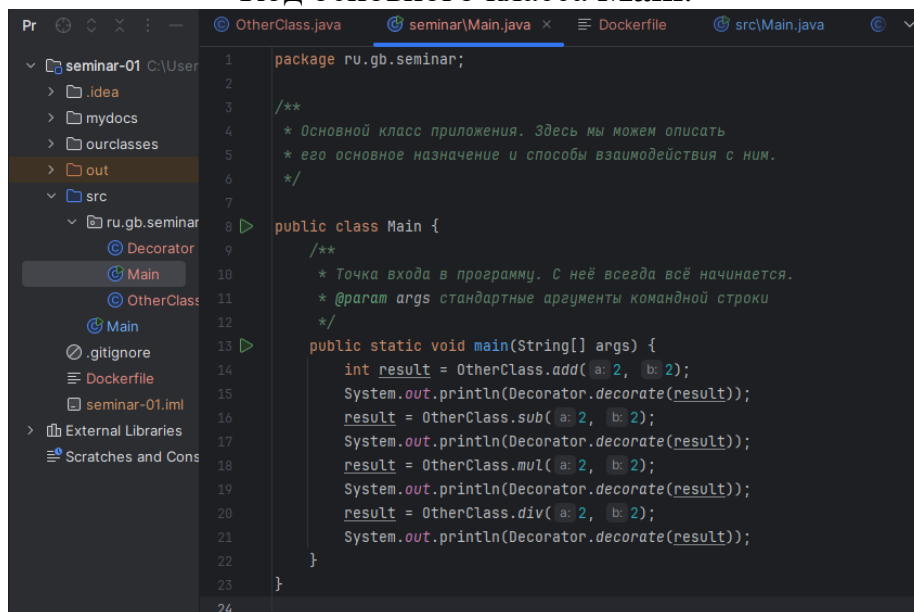
Семинар. Разбор д/з с лекции " Урок 1. Платформа история и окружение" .....	2
Домашняя работа .....	5

## Семинар. Разбор д/з с лекции " Урок 1. Платформа история и окружение"

- 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;
- Скомпилировать проект, а также создать для этого проекта стандартную вебстраницу с документацией ко всем пакетам;

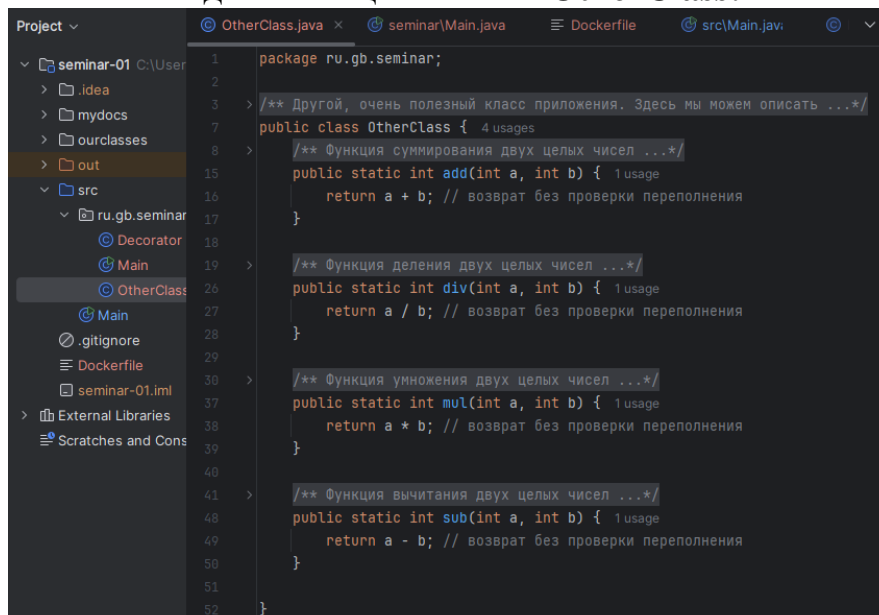
1) Создать проект из трёх классов (основной с точкой входа и два класса в другом пакете), которые вместе должны составлять одну программу, позволяющую производить четыре основных математических действия и осуществлять форматированный вывод результатов пользователю.

### Код основного класса Main:



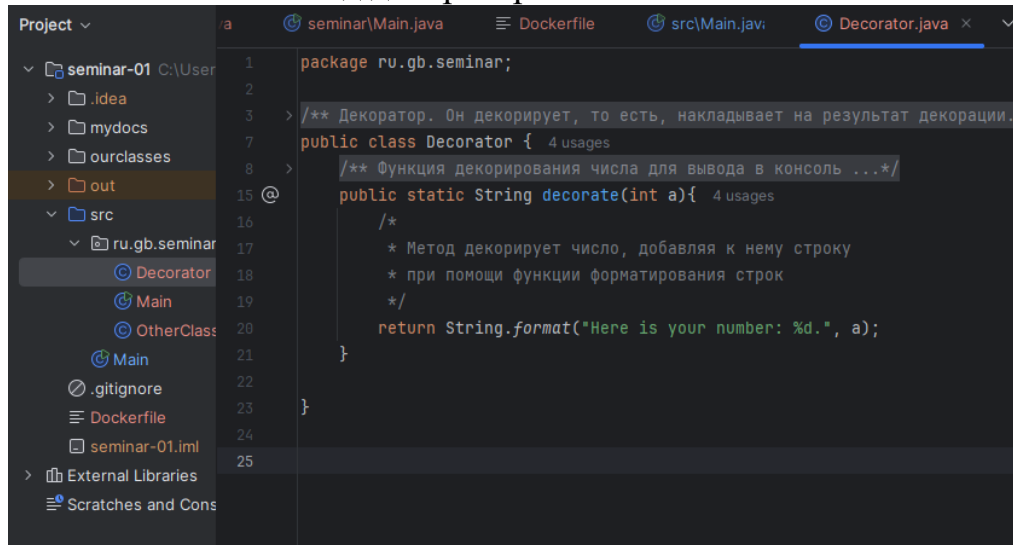
```
1 package ru.gb.seminar;
2
3 /**
4  * Основной класс приложения. Здесь мы можем описать
5  * его основное назначение и способы взаимодействия с ним.
6  */
7
8 public class Main {
9     /**
10      * Точка входа в программу. С неё всегда всё начинается.
11      * @param args стандартные аргументы командной строки
12      */
13     public static void main(String[] args) {
14         int result = OtherClass.add(a: 2, b: 2);
15         System.out.println(Decorator.decorate(result));
16         result = OtherClass.sub(a: 2, b: 2);
17         System.out.println(Decorator.decorate(result));
18         result = OtherClass.mul(a: 2, b: 2);
19         System.out.println(Decorator.decorate(result));
20         result = OtherClass.div(a: 2, b: 2);
21         System.out.println(Decorator.decorate(result));
22     }
23 }
24
```

### Код считающего класса OtherClass:



```
1 package ru.gb.seminar;
2
3 /** Другой, очень полезный класс приложения. Здесь мы можем описать ...*/
4
5 public class OtherClass {
6     /** Функция суммирования двух целых чисел ...*/
7     public static int add(int a, int b) {
8         return a + b; // возврат без проверки переполнения
9     }
10
11     /** Функция деления двух целых чисел ...*/
12     public static int div(int a, int b) {
13         return a / b; // возврат без проверки переполнения
14     }
15
16     /** Функция умножения двух целых чисел ...*/
17     public static int mul(int a, int b) {
18         return a * b; // возврат без проверки переполнения
19     }
20
21     /** Функция вычитания двух целых чисел ...*/
22     public static int sub(int a, int b) {
23         return a - b; // возврат без проверки переполнения
24     }
25 }
26
```

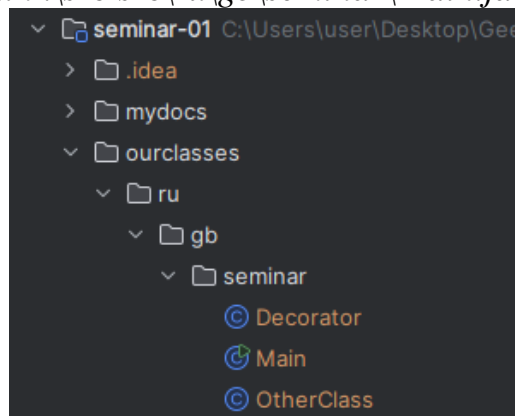
## Код декоратора Decorator:



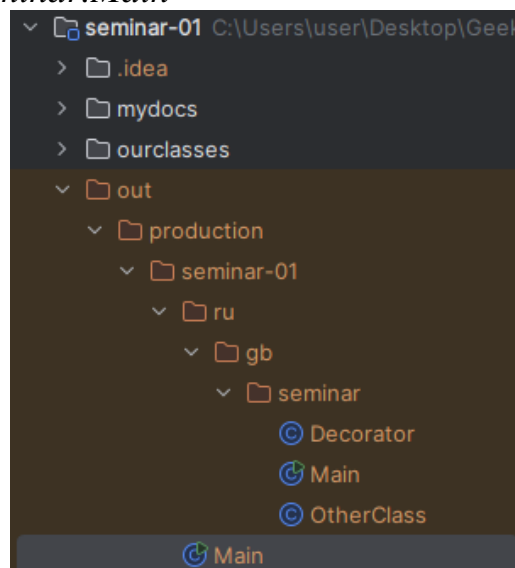
- 2) Скомпилировать проект, а также создать для этого проекта стандартную веб-страницу с документацией ко всем пакетам.

Команды (в терминале) для компиляции:

`javac -d ourclasses -sourcepath .\src src\ru\gb\seminar\Main.java`



`java -classpath .\out ru.gb.seminar.Main`



Команда (в терминале) для создания документации:

*javadoc -d mydocs -sourcepath src -cp .\ourclasses\ -subpackages ru*

The screenshot displays an IDE interface with three main components:

- Project Explorer (Left):** Shows a project structure with folders like `.idea`, `mydocs`, `legal`, `resource-files`, `ru`, `script-files`, `ourclasses`, `out`, and `src`. The `src` folder contains the `ru.gb.seminar` package, which includes the `Decorator` class.
- Code Editor (Center):** Displays the `index.html` file generated by Javadoc. The code includes a DOCTYPE declaration, HTML tags for language and viewport, a title "Generated Documentation", and a package declaration `package ru.gb.seminar`. It also lists classes and their descriptions.
- Preview (Right):** Shows a visual representation of the generated HTML documentation. It features a search bar, a title "Package ru.gb.seminar", and a list of classes. The `Decorator` class is highlighted, showing its description: "Декоратор." (Decorator).

## Домашняя работа

### Создать docker-контейнер для формирования полной документации по проекту

#### 1. Создание Dockerfile:

В корне проекта создайте файл Dockerfile, который будет содержать инструкции по созданию образа контейнера. В нем можно указать базовый образ (например, java:11-openjdk-11), установить необходимые инструменты для генерации документации (например, Javadoc), указать точку входа (например, команду для запуска генерации документации).

```
1
2 FROM bellsoft/liberica-openjdk-alpine:11
3
4 # Копируем исходный код в контейнер
5 COPY ./src /usr/app/src
6
7 # Создаем директории для выходных файлов и документации
8 RUN mkdir -p /usr/app/out /usr/app/doc
9
10 # Компилируем исходный код и создаем javadoc
11 RUN javac -d /usr/app/out /usr/app/src/com/regular/*.java /usr/app/src/com/simple/*.java \
12 && javadoc -d /usr/app/doc /usr/app/src/com/regular/*.java /usr/app/src/com/simple/*.java
13
14 # Устанавливаем рабочую директорию
15 WORKDIR /usr/app
16
17 # Запускаем Java приложение и открываем страницу javadoc в браузере
18 CMD ["sh", "-c", "java -classpath /usr/app/out com.simple.Main && cat /usr/app/doc/indexall.
19 html"]
```

#### 2. Создание образа Docker:

С помощью команды (в терминале) **docker build -t <имя\_образа> .** создается образ контейнера из Dockerfile.

```
5 # Копируем исходный код в контейнер
6 COPY ./src /usr/app/src
7
8 # Создаем директории для выходных файлов и документации
9 RUN mkdir -p /usr/app/out /usr/app/doc
10
11 # Компилируем исходный код и создаем javadoc
12 RUN javac -d /usr/app/out /usr/app/src/ru/gb/seminar/*.java /usr/app/src/ru
13 && javadoc -d /usr/app/doc /usr/app/src/ru/gb/seminar/*.java /usr/app/src/r
14
15 # Устанавливаем рабочую директорию
16 WORKDIR /usr/app
17
18 # Запускаем Java приложение и открываем страницу javadoc в браузере
19 CMD ["sh", "-c", "java -classpath /usr/app/out ru.gb.seminar.Main && cat /u
20
```

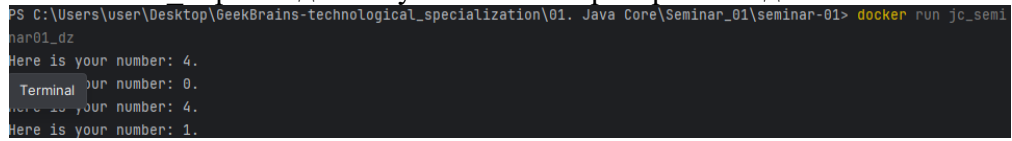
```
Building 8.1s (5/10)
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 934B
=> [internal] load metadata for docker.io/library/openjdk:11
=> [auth] library/openjdk:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 302B

=> => naming to docker.io/library/jc_seminar01_dz:latest
=> => unpacking to docker.io/library/jc_seminar01_dz:latest

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/obozx8seddgx6iuiym9nag3uv
```

### 3. Запуск контейнера:

Используйте команду `docker run -it <имя_образа>` для запуска контейнера в интерактивном режиме или `docker run -d <имя_образа>` для запуска контейнера в режиме демона.



```
PS C:\Users\User\Desktop\GeekBrains-technological_specialization\01. Java Core\Seminar_01\seminar-01> docker run jc_semi
har01_dz
Here is your number: 4.
Terminal your number: 0.
4
Here is your number: 1.
```

---

Ссылка на репозиторий:

[https://github.com/olgashenkel/GeekBrains-  
technological\\_specialization/tree/main/01.%20Java%20Core](https://github.com/olgashenkel/GeekBrains-technological_specialization/tree/main/01.%20Java%20Core)