

# 1. Семинар: компиляция и интерпретация кода

## 1.1. Инструментарий

- Презентация для преподавателя, ведущего семинар;
- Фон GeekBrains для проведения семинара в Zoom;
- Jupyter Notebook для практики и примеров используется Jupyter notebook (потребуется установить Python и ядро IJava) и любой терминал операционной системы (bash, zsh, cmd);
- JDK любая 11 версии и выше;
- Docker, make.

## 1.2. Цели семинара

- Закрепить полученные на лекции знания, касающиеся компиляции, интерпретации кода и создания программной документации;
- Получить практический навык настройки терминала ОС для компиляции и исполнения кода, установки сторонних библиотек для интерпретации;
- Попрактиковаться в написании терминальных команд и простых проектов. Лучше понять принцип импортирования кода и сборки проекта.

## 1.3. План-содержание

Что происходит	Время	Слайды	Описание
Организационный момент	5	1-4	Преподаватель ожидает студентов, поддерживает активность и коммуникацию в чате, озвучивает цели и планы на семинар. Важно упомянуть, что выполнение домашних заданий с лекции является, фактически, подготовкой к семинару
Quiz	5	3-14	Преподаватель задаёт вопросы викторины, через 30 секунд демонстрирует слайд-подсказку и ожидает ответов (4 вопроса, по минуте на ответ)
Рассмотрение ДЗ	15	15-18	Преподаватель демонстрирует свой вариант решения домашнего задания с лекции, возможно, по предварительному опросу, демонстрирует и разбирает вариант решения одного из студентов
Вопросы и ответы	10	19	Преподаватель ожидает вопросов по теме прошедшей лекции, викторины и продемонстрированной работы

Что происходит	Время	Слайды	Описание
Задание 1	10	20-22	Создать, скомпилировать, запустить и продемонстрировать простой проект без использования среды разработки. Показать выполненные терминальные команды, результат компиляции. (* отделить исходный код от скомпилированных файлов, ** сложить исходный код в пакет)
Перерыв (если нужен)	5	26	Преподаватель предлагает студентам перерыв на 5 минут (студенты голосуют)
Задание 2	10	23-25	Настроить окружение Jupyter Notebook с ядром Java, создать одну ячейку с переменной, а вторую с выводом значения этой переменной стандартным способом. Дополнить ячейки описанием markdown. (* осуществить форматированный вывод, ** сохранить форматизирующую строку в ячейке с переменной)
Задание 3	15	27-29	К проекту из задания 1 добавить ещё один класс в соседнем пакете, как это было показано на лекции и комментарии в стиле Javadoc. Комментарии необходимо добавить как к методам, так и к классам. Сгенерировать программную документацию. (* создать документацию на каждый пакет по отдельности)
Домашнее задание	5	39	Объясните домашнее задание, подведите итоги урока
Рефлексия	10	40-42	Преподаватель запрашивает обратную связь
Длительность	90		

## 1.4. Подробности

### 1.4.1. Организационный момент

- **Цель этапа:** Позитивно начать урок, создать комфортную среду для обучения.
- **Тайминг:** 3-5 минут.
- **Действия преподавателя:**
  - Презентует название курса (первый раз) и семинара;
  - Рассказывает немного о себе;
  - Запрашивает активность от аудитории в чате;
  - Презентует цели курса и семинара;
  - Презентует краткий план семинара и что студент научится делать.

### 1.4.2. Quiz

- **Цель этапа:** Вовлечение аудитории в обратную связь.
- **Тайминг:** 5-7 минут (4 вопроса, по минуте на ответ).
- **Действия преподавателя:**
  - Преподаватель задаёт вопросы викторины, представленные на слайдах презентации;
  - через 30 секунд демонстрирует слайд-подсказку и ожидает ответов.
- **Вопросы и ответы:**
  1. Какой механизм используется для непосредственного исполнения скомпилированного кода? (3)
    - (a) JDK;
    - (b) JRE;
    - (c) JVM.
  2. Какая сущность только объединяет классы по смыслу? (1)
    - (a) Пакеты;
    - (b) Библиотеки;
    - (c) Фреймворки.
  3. Основная единица исходного кода программы – это? (2)
    - (a) Функция;
    - (b) Класс;
    - (c) Файл.
  4. Какой ключ используется для указания папки назначения? (1)
    - (a) -d;
    - (b) -out;
    - (c) -to.

### 1.4.3. Рассмотрение ДЗ

- **Цель этапа:** Пояснить неочевидные моменты в формулировке ДЗ с лекции, синхронизировать прочитанный на лекции материал к началу семинара.
- **Тайминг:** 15-20 минут.
- **Действия преподавателя:**
  - Преподаватель демонстрирует свой вариант решения домашнего задания из лекции;
  - возможно, по предварительному опросу, демонстрирует и разбирает вариант решения одного из студентов.
- **Домашнее задание из лекции:**
  - 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.

#### **Вариант решения**

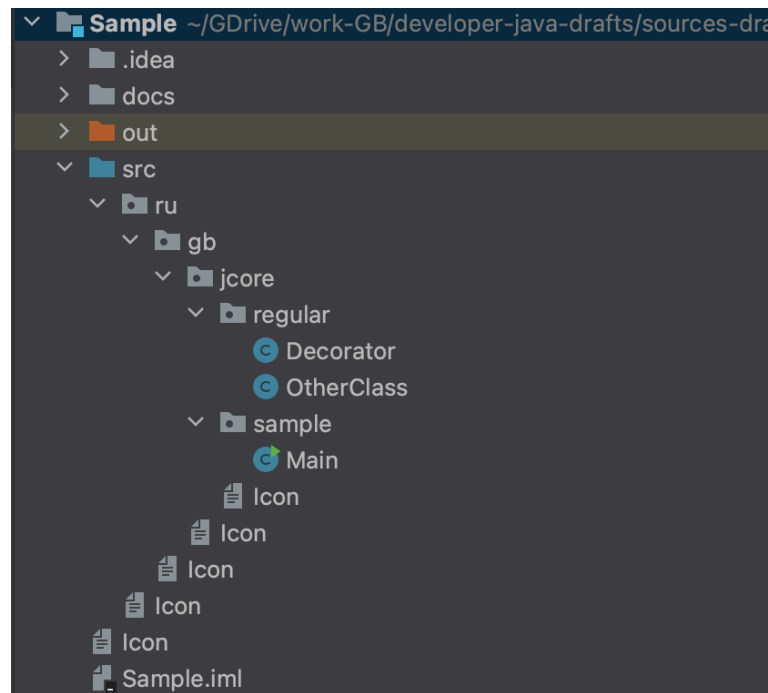


Рис. 1: Структура проекта

Листинг 1: Код основного класса

```

1 package ru.gb.jcore.sample;
2
3 import ru.gb.jcore.regular.Decorator;
4 import ru.gb.jcore.regular.OtherClass;
5
6 /**
7  * Основной класс приложения. Здесь мы можем описать
8  * его основное назначение и способы взаимодействия с ним.
9  * */
10 public class Main {
11     /**
12      * Точка входа в программу. С неё всегда всё начинается.
13      *
14      * @param args стандартные аргументы командной строки
15      * */
16     public static void main(String[] args) {
17         int result = OtherClass.add(2, 2);
18         System.out.println(Decorator.decorate(result));
19         result = OtherClass.sub(2, 2);
20         System.out.println(Decorator.decorate(result));
21         result = OtherClass.mul(2, 2);
22         System.out.println(Decorator.decorate(result));
23         result = OtherClass.div(2, 2);
24         System.out.println(Decorator.decorate(result));
25     }
26 }

```

Листинг 2: Код считающего класса

```

1 package ru.gb.jcore.regular;
2
3 /**
4  * Другой, очень полезный класс приложения. Здесь мы можем описать
5  * его основное назначение и способы взаимодействия с ним.
6  * */
7 public class OtherClass {

```

```

8  /**
9   * Функция суммирования двух целых чисел
10  *
11  * @param a первое слагаемое
12  * @param b второе слагаемое
13  * @return сумма a и b, без проверки на переполнение переменной.
14  * */
15  public static int add(int a, int b) {
16      return a + b; // возврат без проверки переполнения
17  }
18
19  /**
20  * Функция деления двух целых чисел
21  *
22  * @param a делимое
23  * @param b делитель
24  * @return частное a и b, без проверки на переполнение переменной.
25  * */
26  public static int div(int a, int b) {
27      return a / b; // возврат без проверки переполнения
28  }
29
30  /**
31  * Функция умножения двух целых чисел
32  *
33  * @param a первый множитель
34  * @param b второй множитель
35  * @return произведение a и b, без проверки на переполнение переменной.
36  * */
37  public static int mul(int a, int b) {
38      return a * b; // возврат без проверки переполнения
39  }
40
41  /**
42  * Функция вычитания двух целых чисел
43  *
44  * @param a уменьшаемое
45  * @param b вычитаемое
46  * @return разность a и b, без проверки на переполнение переменной.
47  * */
48  public static int sub(int a, int b) {
49      return a - b; // возврат без проверки переполнения
50  }
51  }

```

### Листинг 3: Код декоратора

```

1  package ru.gb.jcore.regular;
2
3  /**
4   * Декоратор. Он декорирует, то есть, накладывает на результат декорации.
5   * Внешний вид важен, поэтому этот класс существует.
6   * */
7  public class Decorator {
8      /**
9       * Функция декорирования числа для вывода в консоль
10      * в виде строки с преамбулой 'Вот Ваше число'
11      *
12      * @param a число, требующее декорации
13      * @return Отформатированная строка.
14      * */
15      public static String decorate(int a) {
16          /*
17           * Метод декорирует число, добавляя к нему строку
18           * при помощи функции форматирования строк

```

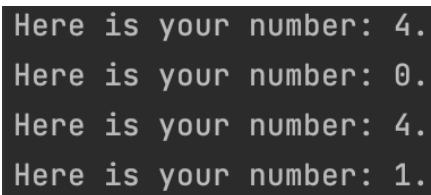
```
19     * */
20     return String.format("Here is your number: %d.", a);
21 }
22 }
```

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

### **Вариант решения**

#### Листинг 4: Команды компиляции

```
1 javac -sourcepath ./src -d out src/ru/gb/jcore/sample/Main.java
2 java -classpath ./out ru.gb.jcore.sample.Main
3
```



```
Here is your number: 4.
Here is your number: 0.
Here is your number: 4.
Here is your number: 1.
```

Рис. 2: Результат компиляции

#### Листинг 5: Команда создания документации

```
1 javadoc -d docs -sourcepath ./src -cp ./out -subpackages ru
2
```

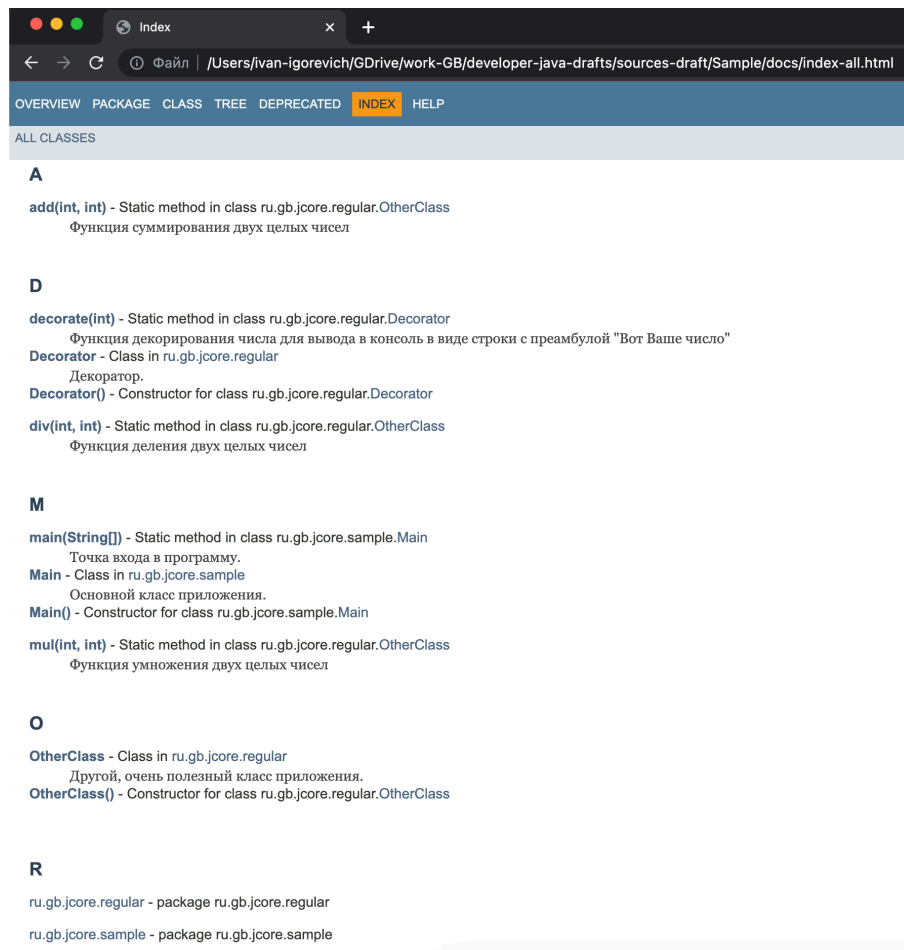


Рис. 3: Результат создания документации

- Создать Makefile с задачами сборки, очистки и создания документации на весь проект.

### Вариант решения<sup>1</sup>

#### Листинг 6: Makefile

```

1 SRC_DIR := src
2 OUT_DIR := out
3 DOC_DIR := doc
4
5 JC := javac
6 JDOC := javadoc
7 JSRC := -sourcepath ./${SRC_DIR}
8 JCLASS := -cp ./${OUT_DIR}
9 JCDEST := -d ${OUT_DIR}
10 JDOCDEST := -d ${DOC_DIR}
11 MAIN_SOURCE := ru/gb/jcore/sample/Main
12 MAIN_CLASS := ru.gb.jcore.sample.Main
13
14 all:
15     ${JC} ${JSRC} ${JCDEST} ${SRC_DIR}/${MAIN_SOURCE}.java
16
17 clean:
```

<sup>1</sup>Обратите внимание, что все отступы сделаны не пробелами, а табуляцией, иначе Makefile не работает

```

18     rm -R ${OUT_DIR}
19
20 run:
21     cd out && java ${MAIN_CLASS}
22
23 docs:
24     ${JDOC}) ${JDOCDEST} ${JSRC} ${JCLASS} -subpackages ru

```

```

ivan-igorevich@MacBook-Pro-Ivan Sample % make all
javac -sourcepath ./src -d out src/ru/gb/jcore/sample/Main.java
ivan-igorevich@MacBook-Pro-Ivan Sample % make run
cd out && java ru.gb.jcore.sample.Main
Here is your number: 4.
Here is your number: 0.
Here is your number: 4.
Here is your number: 1.
ivan-igorevich@MacBook-Pro-Ivan Sample % make docs
make: `docs' is up to date.
ivan-igorevich@MacBook-Pro-Ivan Sample % make clean
rm -R out
ivan-igorevich@MacBook-Pro-Ivan Sample %

```

Рис. 4: Результат выполнения задач

- \*Создать два Docker-образа. Один должен компилировать Java-проект обратно в папку на компьютере пользователя, а второй забирать скомпилированные классы и исполнять их.

#### Вариант решения

Для упрощения был использован docker compose, вместо чистого Docker. Файлы, компилирующие и исполняющие программу представлены в листингах ниже. Оба эти файла запускаются из корня папки проекта командами

```

docker compose -f docker-compose-class.yml up
docker compose -f docker-compose-exec.yml up

```

СООТВЕТСТВЕННО.

Листинг 7: docker-compose-class.yml

```

1 services:
2   app:
3     image: bellsoft/liberica-openjdk-alpine:11.0.16.1-1
4     command: javac -sourcepath /app/src -d /app/out /app/src/ru/gb/jcore/
sample/Main.java
5     volumes:
6       - ./bin:/app/out
7       - ./src:/app/src

```

Листинг 8: docker-compose-exec.yml

```

1 services:
2   app:
3     image: bellsoft/liberica-openjdk-alpine:11.0.16.1-1
4     command: java -classpath /app/out ru.gb.jcore.sample.Main
5     volumes:
6       - ./bin:/app/out

```



#### 1.4.4. Вопросы и ответы

- **Ценность этапа** Вовлечение аудитории в обратную связь, пояснение неочевидных моментов в материале лекции и другой проделанной работе.
- **Тайминг** 5-15 минут
- **Действия преподавателя**
  - Преподаватель ожидает вопросов по теме прошедшей лекции, викторины и продемонстрированной работы;
  - Если преподаватель затрудняется с ответом, необходимо мягко предложить студенту ответить на его вопрос на следующем семинаре (и не забыть найти ответ на вопрос студента!);
  - Предложить и показать пути самостоятельного поиска студентом ответа на заданный вопрос;
  - Посоветовать литературу на тему заданного вопроса;
  - Дополнительно указать на то, что все сведения для выполнения домашнего задания, прохождения викторины и работы на семинаре были рассмотрены в методическом материале к этому или предыдущим урокам.

#### 1.4.5. Задание 1

- **Ценность этапа** Создание, компиляция и запуск проектов без использования среды разработки.
- **Тайминг** 10-20 минут.
- **Действия преподавателя**
  - Пояснить студентам ценность этого опыта (запуск приложений на сервере, в контейнерах, настройка CI/CD в пет-проектах);
  - Выдать задание группам студентов, показать где именно следует искать терминал ОС;
  - Если группа студентов справилась с заданием, а времени осталось более 5 минут, выдавать группе задания «со звёздочкой».
- **Задания:**
  - Создать, скомпилировать, запустить и продемонстрировать простой проект без использования среды разработки.

##### Вариант решения

Листинг 9: Простейший проект

```
1 public class Main {  
2     public static void main(String[] args) {  
3         System.out.println("Hello, world!");  
4     }  
5 }
```

Листинг 10: Команды компиляции

```
1 javac Main.java  
2 java Main  
3
```

- \*<sub>1</sub> отделить исходный код от скомпилированных файлов

#### Вариант решения

```
1 javac -d out Main.java
2 java -classpath ./out Main
3
```

- \*<sub>2</sub> сложить исходный код в пакет с глубиной иерархии не менее 3.

#### Вариант решения

Вручную создать соответствующие вложенные папки, переместить в них файл с исходным кодом `Main.java` и написать оператор `package` первой строкой файла `Main.java`.

```
1 javac -d out ru/gb/jcore/Main.java
2 java -classpath ./out ru.gb.jcore.Main
3
```

### 1.4.6. Задание 2

- **Ценность этапа** Настройка и изучение дополнительного инструментария для создания проектов и описания работы фрагментов кода в виде Jupyter notebook.
- **Тайминг** 10-15 минут.
- **Действия преподавателя**
  - Пояснить студентам ценность этого опыта (использование скриптовых возможностей среды разработки, таких как написание простых фрагментов кода без необходимости создавать большой проект в тяжеловесной среде разработки);
  - Пояснить студентам в каком виде выполнять и сдавать задания;
  - Выдать задание группам студентов, показать где и как скачивать необходимый инструментарий, если он ещё не установлен;
  - Если группа студентов справилась с заданием, а времени осталось более 5 минут, выдавать группе задания «со звёздочкой».
- **Задания**
  - Настроить окружение Jupyter Notebook с ядром IJava, создать одну ячейку с переменной, а вторую с выводом значения этой переменной стандартным способом. Дополнить ячейки описанием markdown.

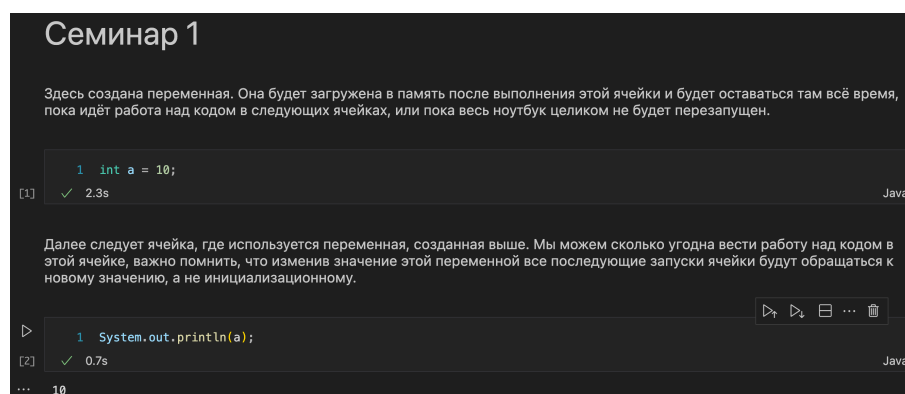


Рис. 5: Вариант решения

- \*<sub>1</sub> осуществить форматированный вывод

## Листинг 11: Вариант решения (вторая ячейка)

```
1 System.out.print("Your number is " + a);
```

\*<sub>2</sub> сохранить форматирующую строку в ячейке с переменной

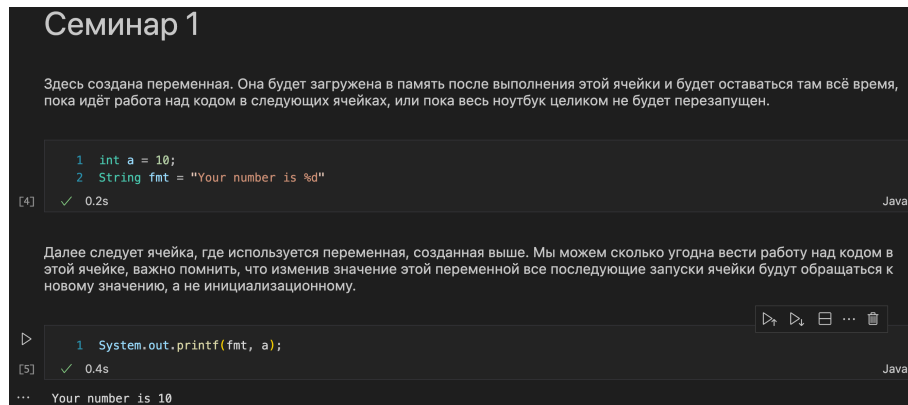


Рис. 6: Вариант решения

### 1.4.7. Задание 3

- **Ценность этапа** Закрепление навыков создания стандартной программной документации.
- **Тайминг** 15-20 минут
- **Действия преподавателя**
  - Пояснить студентам ценность этого опыта (описание пет-проектов для потенциальных соисполнителей, создание базы знаний по проекту на случай длительных пауз в работе)
  - Выдать задание группам студентов
  - Если группа студентов справилась с заданием, а времени осталось более 5 минут, выдать группе задание «со звёздочкой».
- **Задания**
  - К проекту из задания 1 добавить ещё один класс в соседнем пакете, как это было показано на лекции, и комментарии в стиле Javadoc. Комментарии необходимо добавить как к методам, так и к классам. Сгенерировать общую программную документацию.

#### Вариант решения

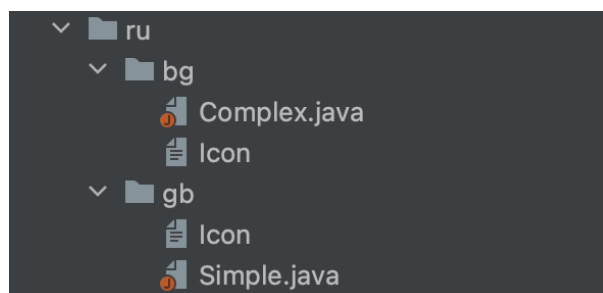


Рис. 7: Иерархия получившегося проекта

### Листинг 12: Основной класс

```
1 package ru.gb;
2
3 import ru.gb.Complex;
4 /**
5  * Это простой класс. Он простой настолько, что ничего не делает
6  */
7 public class Simple {
8     /**
9      * Функция запускающая программу и приветствующая мир.
10     * Наверное, самая популярная функция в мире.
11     *
12     * @param args стандартные аргументы командной строки
13     */
14     public static void main(String[] args) {
15         System.out.println(Complex.hello());
16     }
17 }
```

### Листинг 13: Вспомогательный класс

```
1 package ru.gb;
2
3 /**
4  * Это уже весьма усложнённый класс, мы будем его вызывать из простого
5  */
6 public class Complex {
7     /**
8      * Функция, возвращающая какую-то строку. Возможно, даже приветствующую мир.
9      *
10     * @return строка с приветствием.
11     */
12     public static String hello() {
13         return "Hello, world!";
14     }
15 }
```

### Листинг 14: Команды компиляции и создания документации

```
1 javac -sourcepath . -d out ru.gb/Simple.java
2 java -classpath ./out ru.gb.Simple
3 javadoc -d doc -sourcepath . -cp ./out -subpackages ru
4
```

\*<sub>1</sub> создать документацию на каждый пакет по отдельности

### Листинг 15: Вариант решения

```
1 javadoc -d doc_gb -sourcepath . -cp ./out ru.gb
2 javadoc -d doc_bg -sourcepath . -cp ./out ru.bg
3
```

## 1.4.8. Домашнее задание

- **Ценность этапа** Задать задание для самостоятельного выполнения между занятиями.
- **Тайминг** 5-10 минут.
- **Действия преподавателя**
  - Пояснить студентам в каком виде выполнять и сдавать задания
  - Уточнить кто будет проверять работы (преподаватель или ревьюер)

- Объяснить к кому обращаться за помощью и где искать подсказки
- Объяснить где взять проект заготовки для дз

— **Задания**

5-25 мин Решить все задания (в том числе «со звёздочкой»), если они не были решены на семинаре, без ограничений по времени;

Все варианты решения приведены в тексте семинара выше

10-15 мин Создать docker-контейнер для формирования полной документации по проекту

Листинг 16: docker-compose-class.yml

```
1 services:
2   app:
3     image: bellsoft/liberica-openjdk-alpine:11.0.16.1-1
4     command: javadoc -sourcepath /app/src -cp /app/out -d /app/doc -
5       subpackages ru
6     volumes:
7       - ./bin:/app/out
8       - ./src:/app/src
9       - ./doc:/app/doc
```

#### 1.4.9. Рефлексия и завершение семинара

- **Цель этапа:** Привести урок к логическому завершению, посмотреть что студентам удалось, что было сложно и над чем нужно еще поработать
- **Тайминг:** 5-10 минут
- **Действия преподавателя:**
  - Запросить обратную связь от студентов.
  - Подчеркните то, чему студенты научились на занятии.
  - Дайте рекомендации по решению заданий, если в этом есть необходимость
  - Дайте краткую обратную связь студентам.
  - Поделитесь ощущением от семинара.
  - Поблагодарите за проделанную работу.