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;

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, озвучиает 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 форматирующую 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.

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

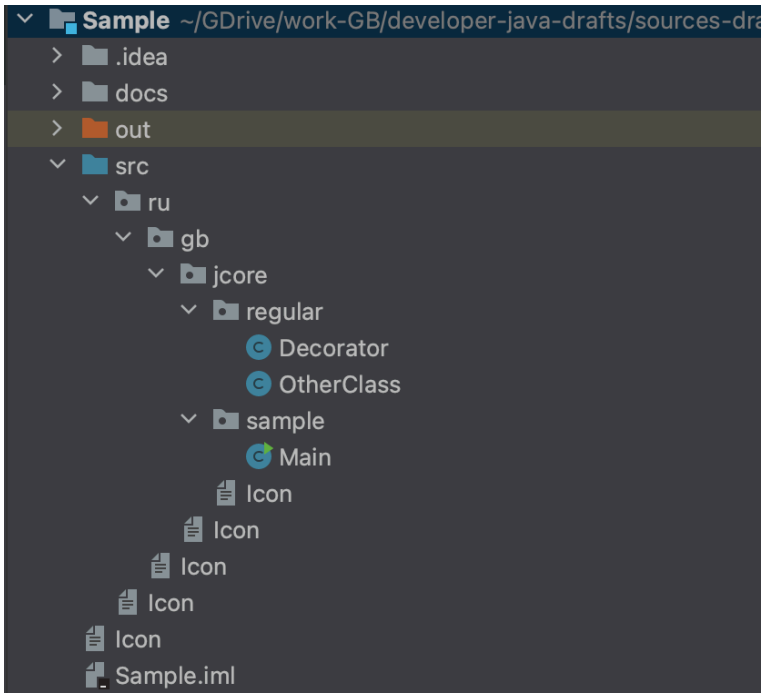

Рис. 1: Структура проекта

Листинг 1: Код основного класса

```
package ru.gb.jcore.sample;

import ru.gb.jcore.regular.Decorator;
import ru.gb.jcore.regular.OtherClass;

/**
* Основной класс приложения. Здесь мы можем описать
* его основное назначение и способы взаимодействия с ним.
* */
public class Main {
  /**
  * Точка входа в программу. С неё всегда всё начинается.
  *
  * @param args стандартные аргументы командной строки
  * */
  public static void main(String[] args) {
    int result = OtherClass.add(2, 2);
    System.out.println(Decorator.decorate(result));
    result = OtherClass.sub(2, 2);
    System.out.println(Decorator.decorate(result));
    result = OtherClass.mul(2, 2);
    System.out.println(Decorator.decorate(result));
    result = OtherClass.div(2, 2);
    System.out.println(Decorator.decorate(result));
  }
}
```

Листинг 2: Код считающего класса

```
package ru.gb.jcore.regular;

/**
 * Другой, очень полезный класс приложения. Здесь мы можем описать
 * его основное назначение и способы взаимодействия с ним.
 * */
public class OtherClass {
    /**
     * Функция суммирования двух целых чисел
     *
     * @param a первое слагаемое
     * @param b второе слагаемое
     * @return сумма a и b, без проверки на переполнение переменной.
     * */
    public static int add(int a, int b) {
        return a + b; // возврат без проверки переполнения
    }

    /**
     * Функция деления двух целых чисел
     *
     * @param a делимое
     * @param b делитель
     * @return частное a и b, без проверки на переполнение переменной.
     * */
    public static int div(int a, int b) {
        return a / b; // возврат без проверки переполнения
    }

    /**
     * Функция умножения двух целых чисел
     *
     * @param a первый множитель
     * @param b второй множитель
     * @return произведение a и b, без проверки на переполнение переменной.
```

```
     * */
    public static int mul(int a, int b) {
        return a * b; // возврат без проверки переполнения
    }

    /**
     * Функция вычитания двух целых чисел
     *
     * @param a уменьшаемое
     * @param b вычитаемое
     * @return разность a и b, без проверки на переполнение переменной.
     * */
    public static int sub(int a, int b) {
        return a - b; // возврат без проверки переполнения
    }
}
```

Листинг 3: Код декоратора

```
package ru.gb.jcore.regular;

/**
 * Декоратор. Он декорирует, то есть, накладывает на результат декорации.
 * Внешний вид важен, поэтому этот класс существует.
 * */
public class Decorator {
    /**
     * Функция декорирования числа для вывода в консоль
     * в виде строки с преамбулой 'Вот Ваше число'
     *
     * @param a число, требующее декорации
     * @return Отформатированная строка.
     * */
    public static String decorate(int a) {
        /*
         * Метод декорирует число, добавляя к нему строку
         * при помощи функции форматирования строк
         * */
        return String.format("Here is your number: %d.", a);
    }
}
```

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

Листинг 4: Команды компиляции

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

Рис. 2: Результат компиляции

Листинг 5: Команда создания документации

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

```

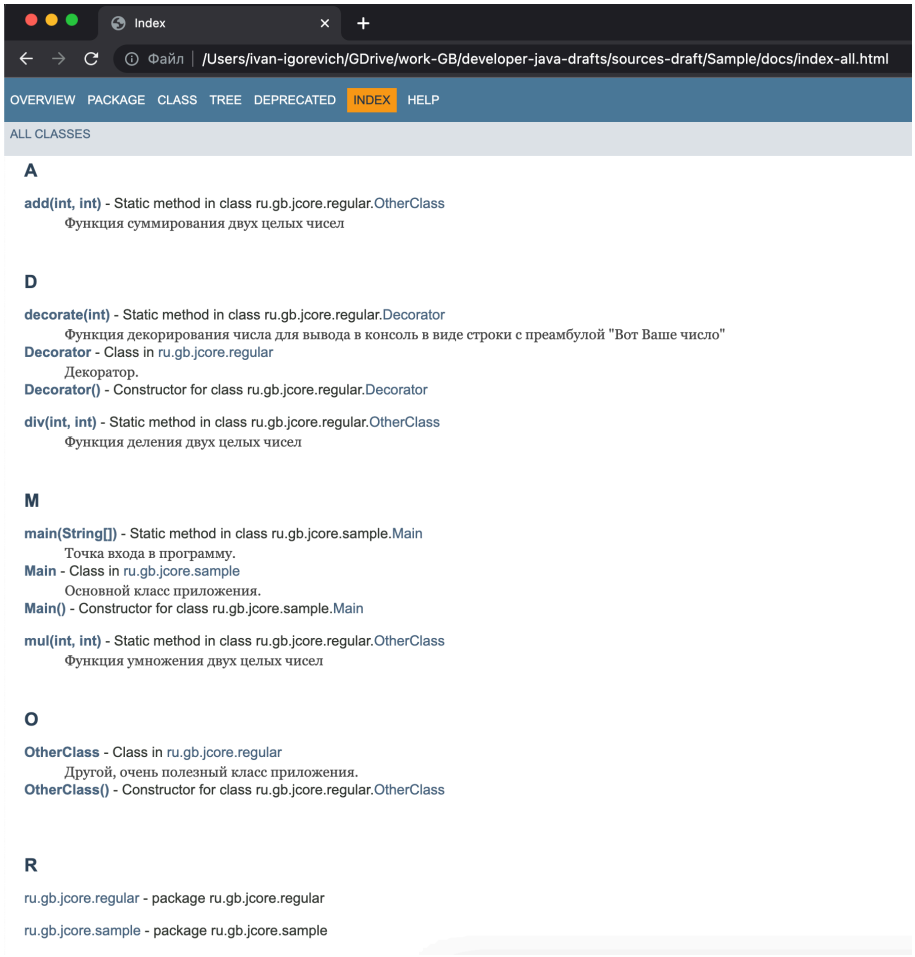

Рис. 3: Результат создания документации

– Создать Makefile с задачами сборки, очистки и создания документации на весь проект.
Вариант решения[1]

Листинг 6: Makefile

```
SRC_DIR := src
OUT_DIR := out
DOC_DIR := doc

JC := javac
JDOC := javadoc
JSRC := -sourcepath ./$(SRC_DIR)
JCLASS := -cp ./$(OUT_DIR)
JCDEST := -d $(OUT_DIR)
JDOCDEST := -d $(DOC_DIR)
MAIN_SOURCE := ru/gb/jcore/sample/Main
MAIN_CLASS := ru.gb.jcore.sample.Main
```

[1]Обратите внимание, что все отступы сделаны не пробелами, а табуляцией, иначе Makefile не работает

```
13
14 all:
15     ${JC} ${JSRC} ${JCDEST} ${SRC_DIR}/${MAIN_SOURCE}.java
16
17 clean:
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 тасков

- *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

1.4.4. Вопросы и ответы

— **Ценность этапа** Задать задание для самостоятельного выполнения между занятиями.
— **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 7: Простейший проект

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

Листинг 8: Команды компиляции

```
javac Main.java
java Main

```

$*_1$ отделить исходный код от скомпилированных файлов

Вариант решения

```
javac -d out Main.java
java -classpath ./out Main

```

$*_2$ 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`.

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

```

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.

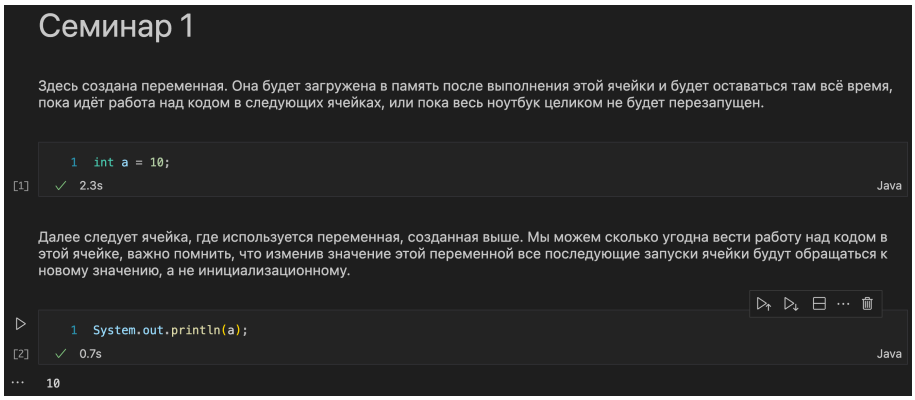

Рис. 5: Вариант решения

$*_1$ осуществить форматированный вывод

Листинг 9: Вариант решения (вторая ячейка)

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

$*_1$ сохранить форматирующую строку в ячейке с переменной

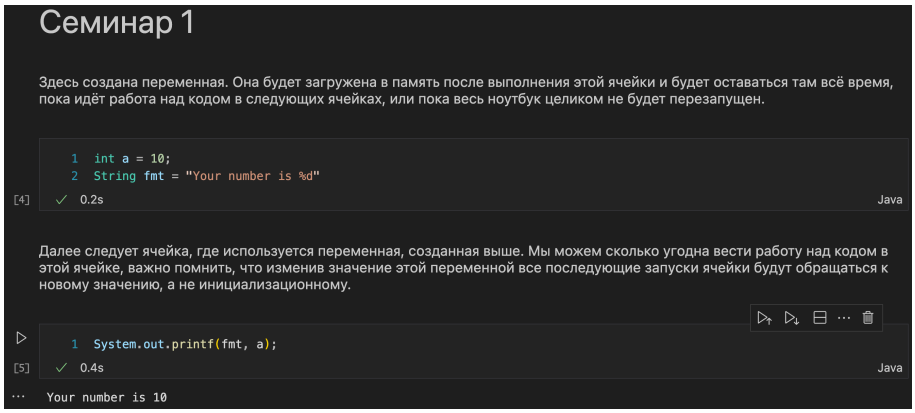

Рис. 6: Вариант решения

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

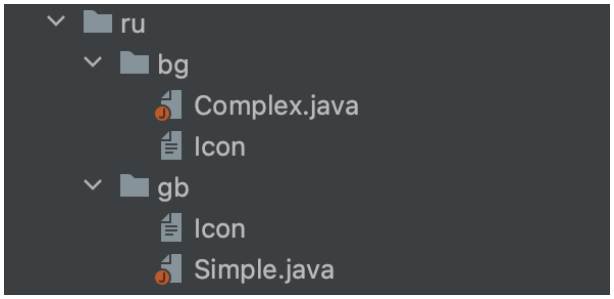

Рис. 7: Иерархия получившегося проекта

Листинг 10: Основной класс

```
package ru.gb;

import ru.bg.Complex;
/**
 * Это простой класс. Он простой настолько, что ничего не делает
 */
public class Simple {
   /**
    * Функция запускающая программу и приветствующая мир.
    * Наверное, самая популярная функция в мире.
    *
    * @param args стандартные аргументы командной строки
    * */
   public static void main(String[] args) {
      System.out.println(Complex.hello());
   }
}
```

Листинг 11: Вспомогательный класс

```
package ru.bg;

```

```java
/**
 * Это уже весьма усложнённый класс, мы будем его вызывать из простого
 */
public class Complex {
   /**
    * Функция, возвращающая какую-то строку. Возможно, даже приветствующую мир.
    *
    * @return строка с приветствием.
    * */
   public static String hello() {
      return "Hello, world!";
   }
}
```

Листинг 12: Команды компиляции и создания документации

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

```

$*_1$ создать документацию на каждый пакет по отдельности

Листинг 13: Вариант решения

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

```

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 ревьювер)
  - Объяснить к кому обращаться за помощью и где искать подсказки
  - Объяснить где взять проект заготовки для дз
- **Задания**
  1. Решить все задания (в том числе «со звёздочкой»), если они не были решены на семинаре, без ограничений по времени;
  2.

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.