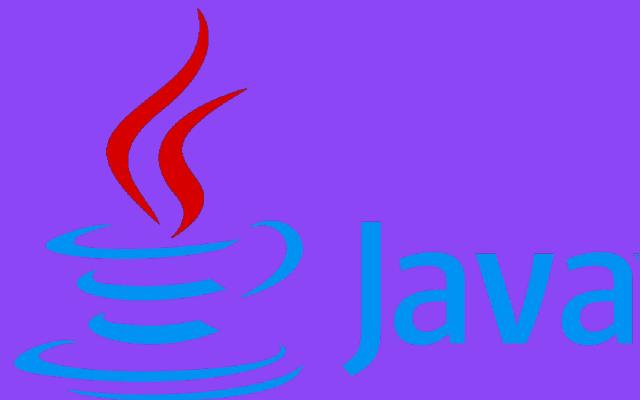


Платформа: история и окружение





Знакомство и содержание урока

Давайте знакомиться!



Иван Овчинников

Начальник группы разработки специального программного обеспечения в АО Российские космические системы

Для РКС - руководитель группы разработчиков. Для ГБ - преподаватель, автор.

- 💥 Информационные системы и системы управления
- 💥 Прошивки аппаратуры спутников и приёмников
- 💥 Цифровая схемотехника космических аппаратов
- 💥 Более 25 потоков, более 3000 студентов

Давайте знакомиться!



Иван Овчинников

Начальник группы разработки специального программного обеспечения в АО Российские космические системы

Для Java - техлид и архитектор СПО наземных автоматизированных комплексов

- 💥 Отраслевые информационные системы
- 💥 Межотраслевые информационные системы
- 💥 База данных компонентов космического применения



План курса

- 1 Устройство платформы
- 2 Фреймворк
- 3 ООП
- 4 Подсистема ввода-вывода
- 5 Работа с терминалом
- 6 Автоматизация сборки





Что будет на уроке сегодня

- 📌 Краткая история (причины возникновения)
- 📌 Базовый инструментарий, который понадобится (выбор IDE)
- 📌 Что нужно скачать, откуда (как выбрать вендора, версии)
- 📌 Из чего все состоит (JDK, JRE, JVM и их друзья)
- 📌 Структура проекта (пакеты, классы, метод main, комментарии)
- 📌 Отложим мышки в сторону (CLI: сборка, пакеты, запуск)
- 📌 Документирование (Javadoc)
- 📌 Автоматизируй это (Makefile, Docker)





Краткая история



TIOBE Programming Community Index

Source: www.tiobe.com

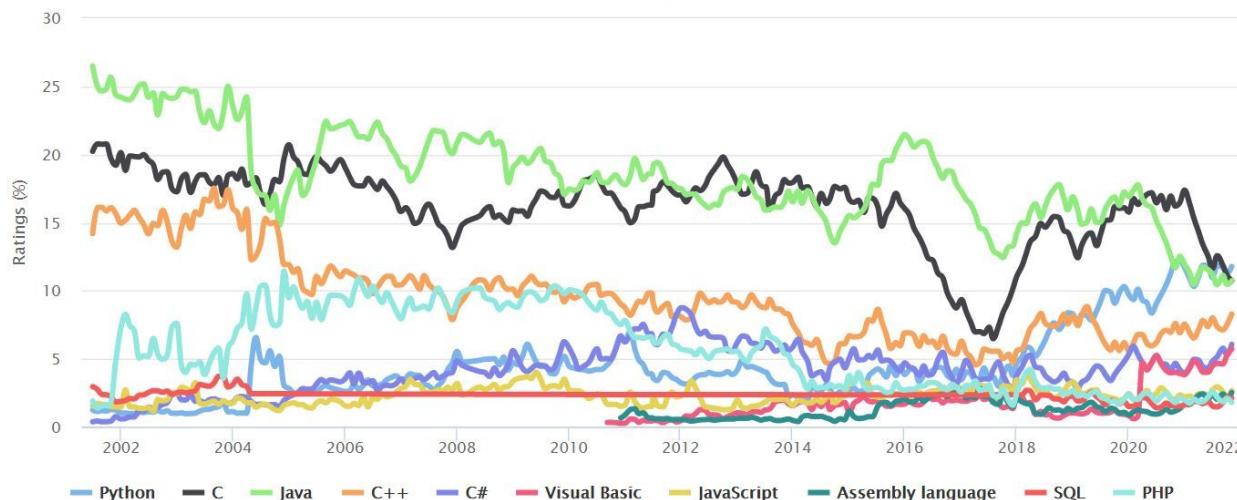


График tiobe (2022)



LANGUAGE	NOV 2017	MAR 2018	NOV- MAR
Java	57	58	+1
JavaScript	54	57	+3
C++	45	46	+1
C#	28	26	-2



Почему Java?



Написано однажды, работает везде

Теперь ваша очередь!



Ответьте на вопрос сообщением в чат

Вопрос:

Как Вы думаете, почему язык программирования Java стал популярен в такие короткие сроки?

1. Существовавшие на тот момент Pascal и C++ были слишком сложными;
2. Java быстрее, чем C++;
3. Однажды написанная на Java программа работает везде.





Базовый инструментарий



Базовый инструментарий

- 📌 Eclipse
- 📌 NetBeans
- 📌 IntelliJ IDEA
- 📌 BlueJ
- 📌 Oracle JDeveloper
- 📌 MyEclipse
- 📌 Greenfoot
- 📌 jGRASP
- 📌 JCreator
- 📌 DrJava





NetBeans





Eclipse





IntelliJ IDEA



IntelliJ IDEA





Android Studio



Теперь ваша очередь!



Ответьте на вопрос сообщением в чат

Вопрос:

1. Как Вы думаете, почему среда разработки IntelliJ IDEA стала стандартом де-факто в коммерческой разработке приложений на Java?
 - a. NetBeans перестали поддерживать;
 - b. Eclipse слишком медленный и тяжеловесный;
 - c. IDEA оказалась самой дружелюбной к начинающему программисту;
 - d. Все варианты верны.





Что нужно скачать и откуда



JDK, среда программирования





JDK, среда, ...



MATLAB





JDK, среда программирования



IntelliJ IDEA





Что нужно скачать

Oracle JDK

OpenJDK by Oracle

Liberica JDK

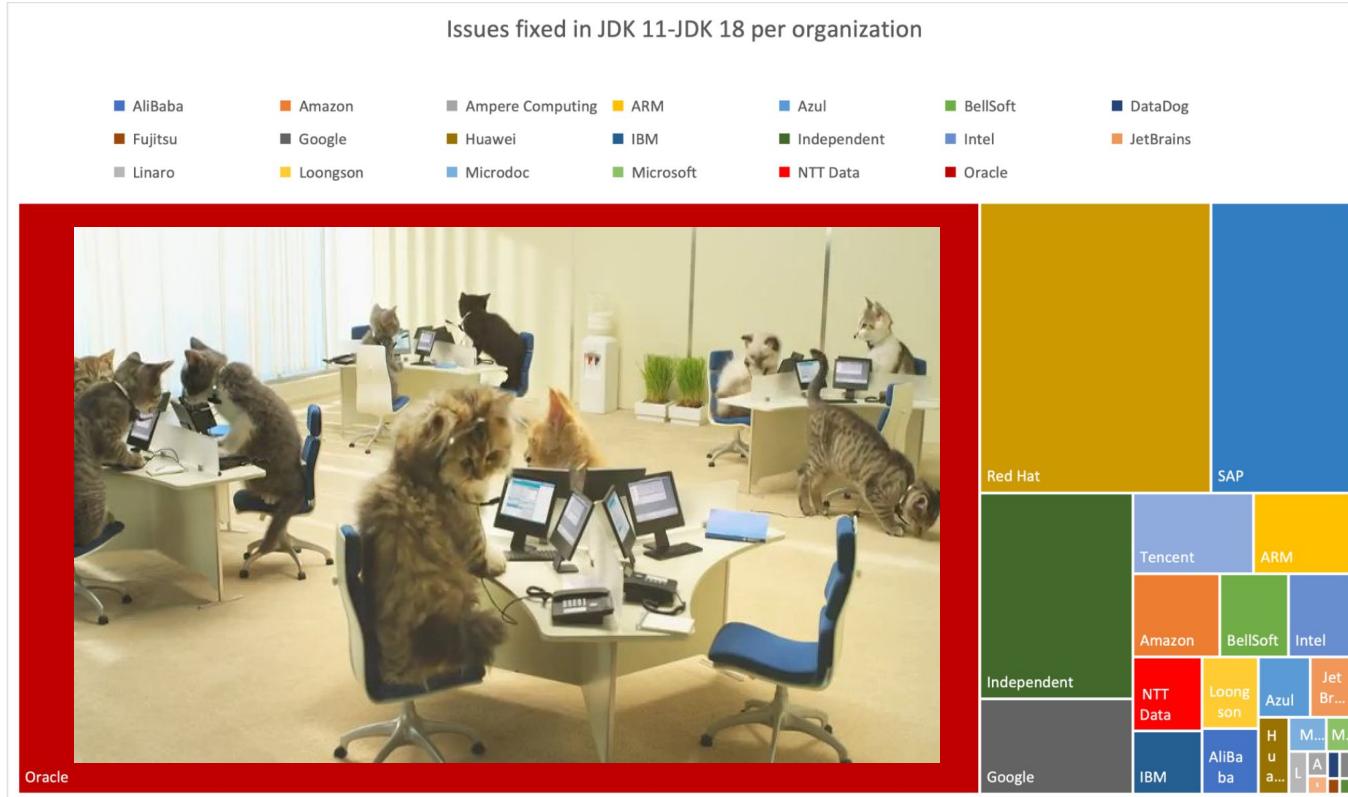
Экзотические

- GOST Java
- AdoptOpenJDK
- Red Hat OpenJDK
- Azul Zulu
- Amazon Corretto





Кто делает для нас JDK?





Выбираем версию JDK

JDK 1.8 

JDK 11 

JDK 13 

JDK 11+ (при использовании 1.8) 

Кошачьи советы





Внимательно прочтите лицензию

JDK 1.8 ✓

JDK 11 ✓

JDK 13 ✓





Переключение версий и вендоров





Иногда нужно по-быстрому (Jupyter notebook + IJava)





Переменные среды

- 📌 PATH
- 📌 JAVA_HOME
- 📌 JRE_HOME
- 📌 J2SDKDIR
- 📌 J2REDIR



Теперь ваша очередь!



Ответьте на несколько вопросов сообщением в чат

Вопросы:

1. Чем отличается SDK от JDK?
2. Какая версия языка (к сожалению) остаётся самой популярной в разработке на Java?
3. Какие ещё JVM языки существуют?





Из чего всё состоит



JDK и его друзья

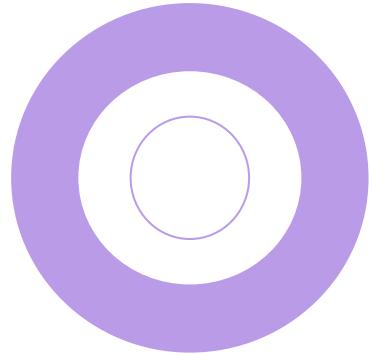
TL;DR:

- 📌 JDK = JRE + инструменты разработчика;
- 📌 JRE = JVM + библиотеки классов;
- 📌 JVM = Native API + механизм исполнения + управление памятью.



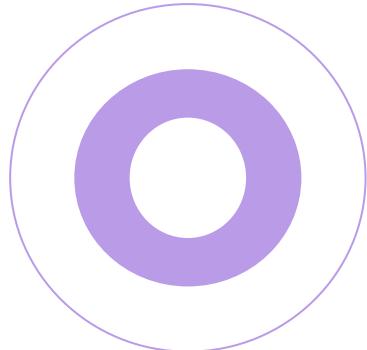


JDK = Java SDK (Java Development Kit, JRE + компилятор)



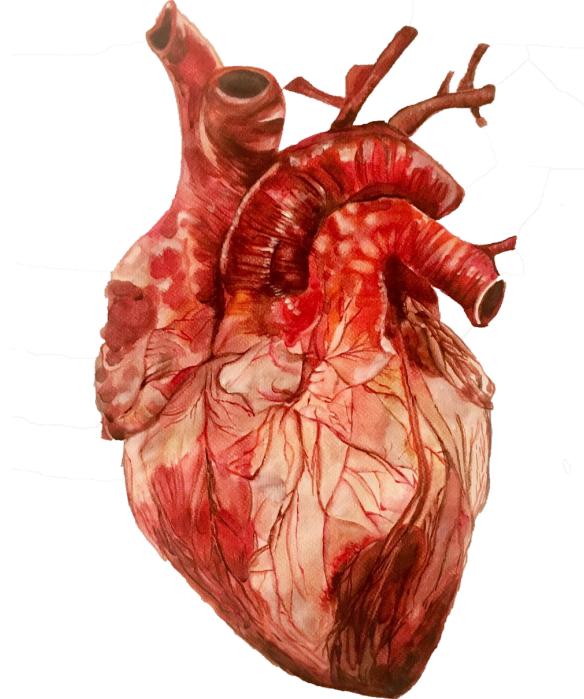
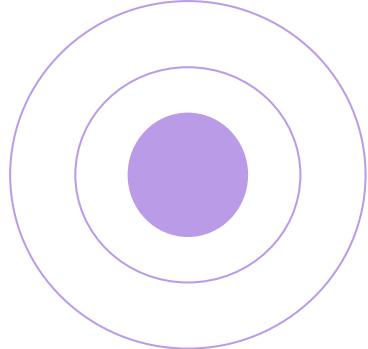


JRE = Java Runtime Environment



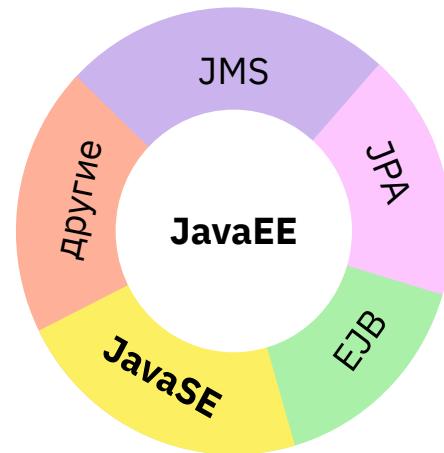
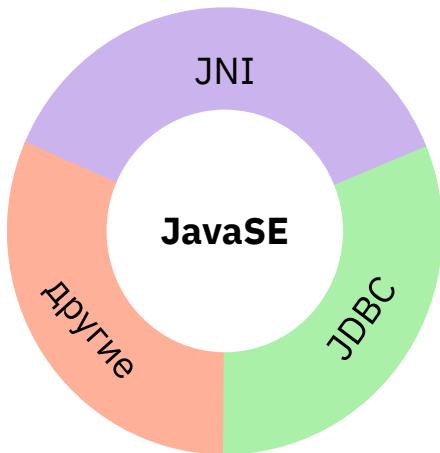


JVM = Java Virtual Machine



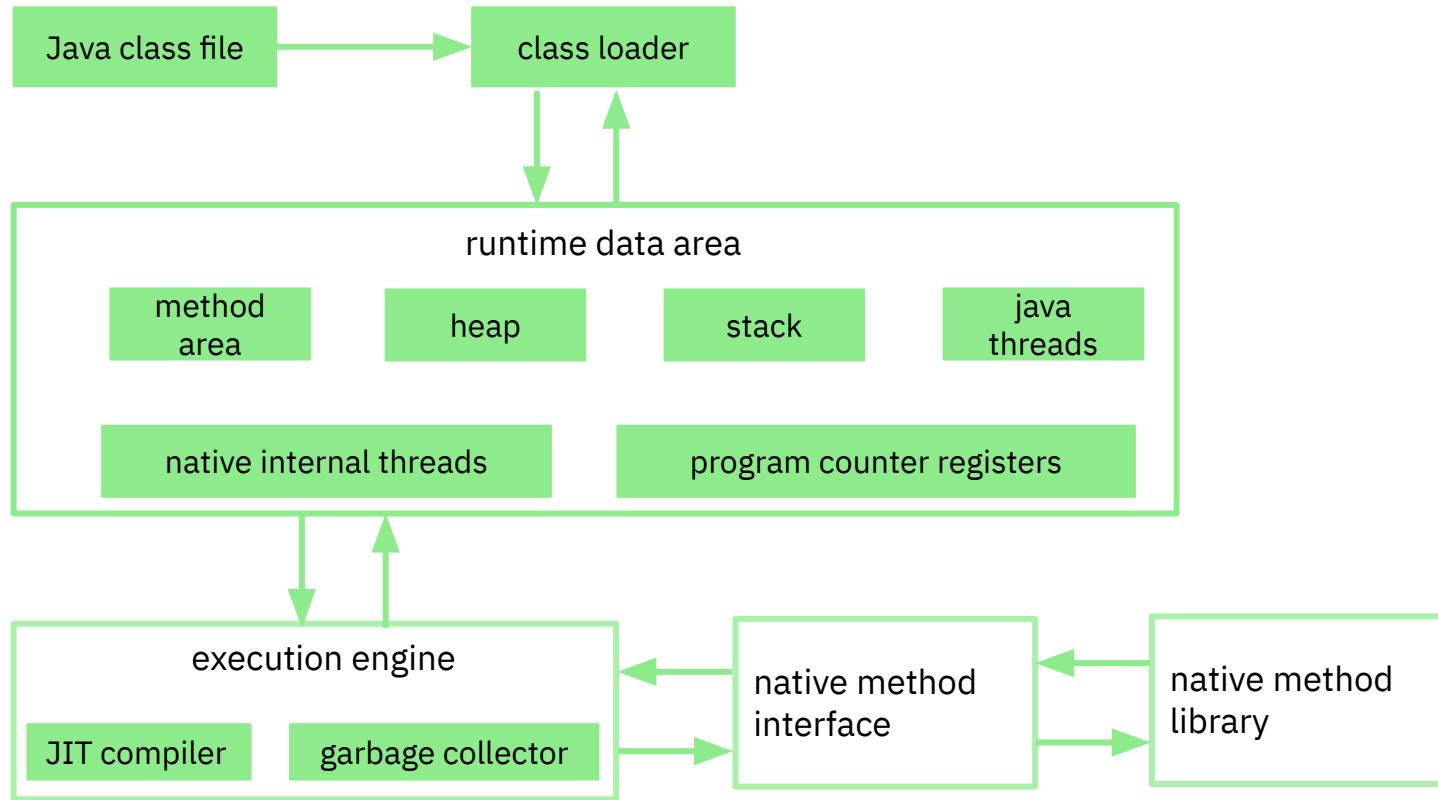


JRE (JavaSE: JNI, JDBC, ...; JavaEE: EJB, JMS, JPA, ...)





Java Virtual Machine (JVM) and what happens inside it



JVM на пальцах





JIT и GC

GraalVM™



Теперь ваша очередь!



Ответьте на несколько вопросов сообщением в чат

Вопросы:

1. JVM и JRE - это одно и тоже?
2. Что входит в состав JDK, но, не входит в состав JRE?
3. Утечки памяти
 - a. Невозможны, поскольку работает сборщик мусора;
 - b. Возможны;
 - c. Существуют только в C++ и других языках с открытым менеджментом памяти.





Структура проекта



Структура проекта

- 📌 Простейшие (один файл)
- 📌 Обычные (несколько пакетов)
- 📌 Шаблонные (формируются сборщиками)
- 📌 Скриптовые (jupyter notebook)





Простейший проект

The screenshot shows a code editor window for a Java file named `Main.java`. The code contains a single class definition:

```
1 > public class Main {  
2 >     public static void main(String[] args) {  
3 >         System.out.println("Hello, world");  
4 >     }  
5 > }  
6
```

Annotations in the code editor highlight specific parts of the code:

- A green arrow icon is placed before the first line of code (the class declaration).
- A green arrow icon is placed before the second line of code (the `main` method declaration).
- A green arrow icon is placed before the third line of code (the `System.out.println` statement).
- A green arrow icon is placed before the fourth line of code (the closing brace of the `main` method).
- A green arrow icon is placed before the fifth line of code (the closing brace of the class).
- A vertical line with a small circle at the top is positioned between the first and second lines of code.
- A vertical line with a small circle at the top is positioned between the second and third lines of code.
- A vertical line with a small circle at the top is positioned between the third and fourth lines of code.
- A vertical line with a small circle at the top is positioned between the fourth and fifth lines of code.

The text "4 usages" is displayed above the code editor window, indicating the number of occurrences of the selected code elements.



Простейший проект

```
[ivan-igorevich@MacBook-Pro-Ivan sources-draft % javac Main.java
[ivan-igorevich@MacBook-Pro-Ivan sources-draft % java Main
Hello, world
ivan-igorevich@MacBook-Pro-Ivan sources-draft % ]
```



Jupyter Notebook





Проект на Java

- 📌 Пакеты
- 📌 Классы
- 📌 Метод main
- 📌 Комментарии
- 📌 Ресурсы





Пакеты



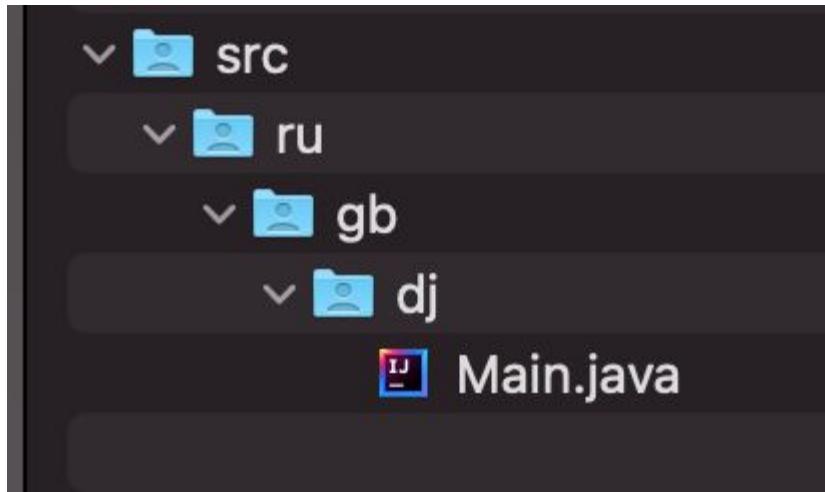


Пакеты





Пакеты





Имена пакетов

если домен gb.ru



то пакет ru.gb

БОЛЬШИЕ БУКВЫ

разделять_слова_нижними_подчеркиваниями

Main.java

```
1 package ru.gb.dj;
```



Классы





Классы





Классы

Программа = класс

1 файл = 1 класс

Класс = существительное в именительном падеже
с большой буквы

Название = UpperCamelCase





Точка входа в программу





JVM ищет точку входа в программу





Точка входа в программу

4 ►



```
public static void main(String[] args) {
```



Комментарии





Комментарии

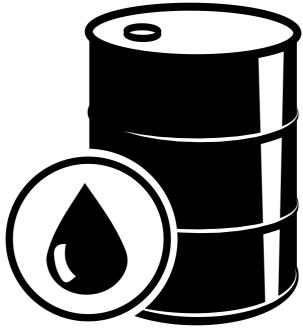
```
public static int sum(int a, int b) {  
    return a + b; // возврат без проверки переполнения  
}
```

```
27     *  
28     * Метод декорирует число, добавляя к нему строку  
29     * при помощи функции форматирования строк  
30     * */
```

```
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      * */
```



Ресурсы



Теперь ваша очередь!



Ответьте на несколько вопросов сообщением в чат

Вопросы:

1. Зачем складывать классы в пакеты?
2. Может ли существовать класс вне пакета? (да/нет)
3. Комментирование кода
 - a. Нужно только если пишется большая подключаемая библиотека;
 - b. Хорошая привычка;
 - c. Захламляет исходники.

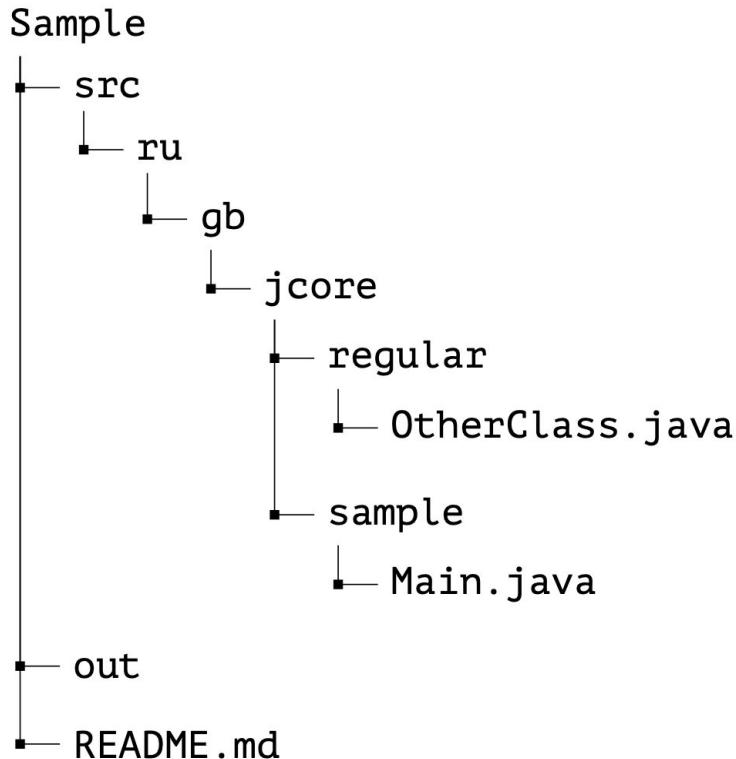




Отложим мышки в сторону



Структура простого проекта





Привет, мир!

```
13 ►     public static void main(String[] args) {  
14         System.out.println("Hello, world!"); // приветствие  
15     }
```



Команды компиляции

```
ivan-igorevich@MacBook-Pro-Ivan sources-draft % javac Main.java
ivan-igorevich@MacBook-Pro-Ivan sources-draft % java Main
Hello, world
[1]
```



Кофе, крошка!

87654321	0011	2233	4455	6677	8899	aabb	ccdd	efff	0123456789abcdef
00000000:	cafe	babe	0000	0037	001d	0a00	0600	0f09	
00000010:	0010	0011	0800	120a	0013	0014	0700	1507	
00000020:	0016	0100	063c	696e	6974	3e01	0003	2829	
00000030:	5601	0004	436f	6465	0100	0f4c	696e	654e	
00000040:	756d	6265	7254	6162	6c65	0100	046d	6169	
00000050:	6e01	0016	285b	4c6a	6176	612f	6c61	6e67	
00000060:	2f53	7472	696e	673b	2956	0100	0a53	6f75	



Два класса в разных пакетах

```
Project Main.java × OtherClass.java ×

Main.java
1 package ru.gb.jcore.sample;
2
3 import ru.gb.jcore.regular.OtherClass;
4
5 /**
6  * Основной класс приложения. Здесь мы можем описать
7  * его основное назначение и способы взаимодействия с ним.
8 */
9 public class Main {
10 /**
11  * Точка входа в программу. С неё всегда всё начинается.
12 */
13 public static void main(String[] args) {
14     System.out.println("Hello, world!");
15     int result = OtherClass.sum(2, 2);
16     System.out.println(OtherClass.decorate(result));
17 }
18
19
20
21
22 }

OtherClass.java
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 sum(int a, int b) {
16     return a + b; // возврат без проверки переполнения
17 }
18
19 /**
20  * Функция декорирования числа для вывода в консоль
21  * в виде строки с преамбулой "Вот Ваше число"
22 *
23 * @param a число, требующее декорации
24 * @return Отформатированная строка.
25 */
26 public static String decorate(int a) {
27     /*
28      * Метод декорирует число, добавляя к нему строку,
29      * при помощи функции форматирования строк
30      */
31     return String.format("Here is your number: %d.", a);
32 }
33
34 }
```



КОД КОМАНД КОМПИЛЯЦИИ

```
ivan-igorevich@MacBook-Pro-Ivan Sample % javac -sourcepath ./src -d out src/ru/gb/jcore/sample/Main.java
ivan-igorevich@MacBook-Pro-Ivan Sample % java -classpath ./out ru.gb.jcore.sample.Main
Hello, world!
```

Теперь ваша очередь!



Ответьте на несколько вопросов сообщением в чат

Вопросы:

1. Что такое javac?
2. Кофе, крошка?
3. Где находится класс в папке назначения работы компилятора?
 - a. В подпапках, повторяющих структуру пакетов в исходниках;
 - b. В корне плоским списком;
 - c. Зависит от ключей компиляции.





Документирование



Javadoc

[OVERVIEW](#) [PACKAGE](#) [CLASS](#) [TREE](#) [DEPRECATED](#) [INDEX](#) [HELP](#)

ALL CLASSES

SEARCH:

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Package ru.gb.jcore.regular

Class OtherClass

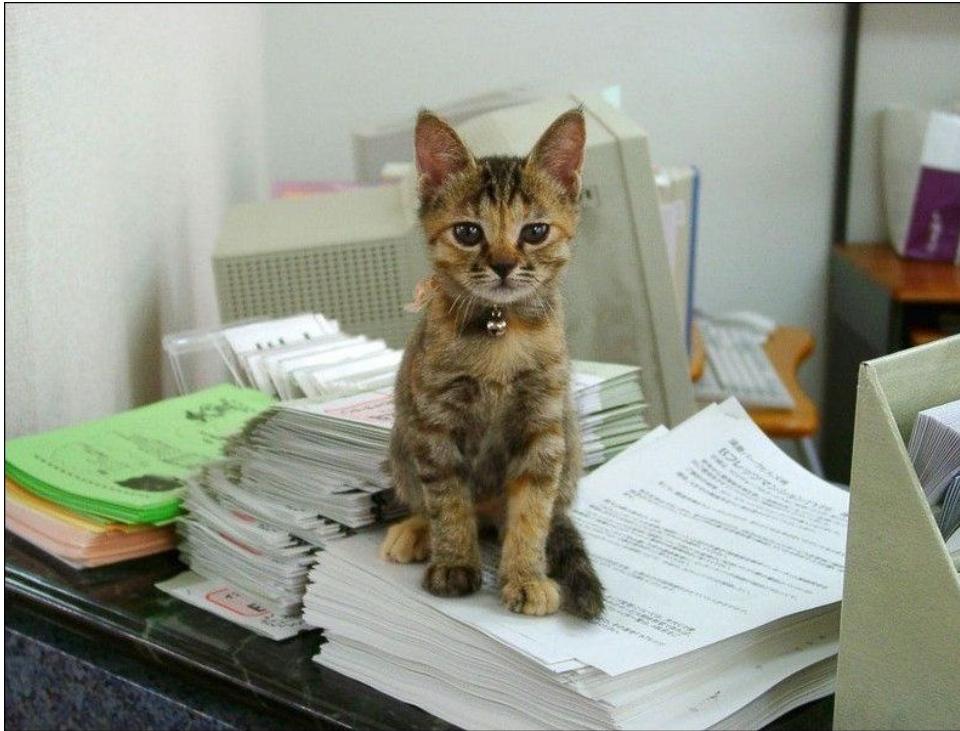
java.lang.Object
ru.gb.jcore.regular.OtherClass

```
public class OtherClass
extends java.lang.Object
```

Другой, очень полезный класс приложения. Здесь мы можем описать его основное назначение и способы взаимодействия с ним.



Документирование



Программист

Программа

Документация

Успех



Программист

Программа

Техпис

Программа

Успех

Почта/чат:
скриншоты/объясне
ния не
программисту
технических
тонкостей



Программист

Техпис



Утилита Javadoc

Основные ключи и аргументы

- 📌 -d docs
- 📌 -sourcepath src
- 📌 -cp out
- 📌 -subpackages
- 📌 ru





Использование Javadoc

```
ivan-igorevich@MacBook-Pro-Ivan Sample % javadoc -d docs -sourcepath src -cp out -subpackages ru
Loading source files for package ru...
Constructing Javadoc information...
Standard Doclet version 11.0.15.1
Building tree for all the packages and classes...
Generating docs/ru/gb/jcore/regular/OtherClass.html...
Generating docs/ru/gb/jcore/sample/Main.html...
src/ru/gb/jcore/sample/Main.java:13: warning: no @param for args
    public static void main(String[] args) {
        ^
Generating docs/ru/gb/jcore/regular/package-summary.html...
Generating docs/ru/gb/jcore/regular/package-tree.html...
Generating docs/ru/gb/jcore/sample/package-summary.html...
Generating docs/ru/gb/jcore/sample/package-tree.html...
Generating docs/constant-values.html...
Building index for all the packages and classes...
Generating docs/overview-tree.html...
Generating docs/index-all.html...
Building index for all classes...
Generating docs/allclasses-index.html...
Generating docs/allpackages-index.html...
Generating docs/deprecated-list.html...
Building index for all classes...
Generating docs/allclasses.html...
Generating docs/allclasses.html...
Generating docs/index.html...
Generating docs/overview-summary.html...
Generating docs/help-doc.html...
1 warning
ivan-igorevich@MacBook-Pro-Ivan Sample %
```



Куда же без особенностей работы с Windows?

- 📌 -locale ru_RU
- 📌 -encoding utf-8
- 📌 -docencoding cp1251



Теперь ваша очередь!



Ответьте на несколько вопросов сообщением в чат

Вопросы:

1. Javadoc находится в JDK или JRE?
2. Что делает утилита Javadoc?
 - a. Создаёт комментарии в коде;
 - b. Создаёт программную документацию;
 - c. Создаёт веб-страницу с документацией из комментариев.

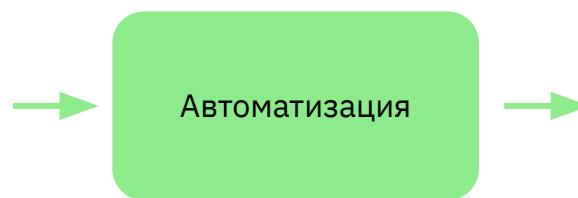




Автоматизирай это



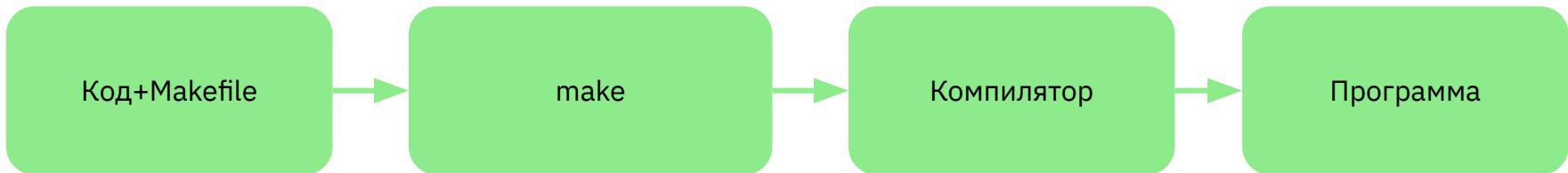
Автоматизация (make)



Makefile

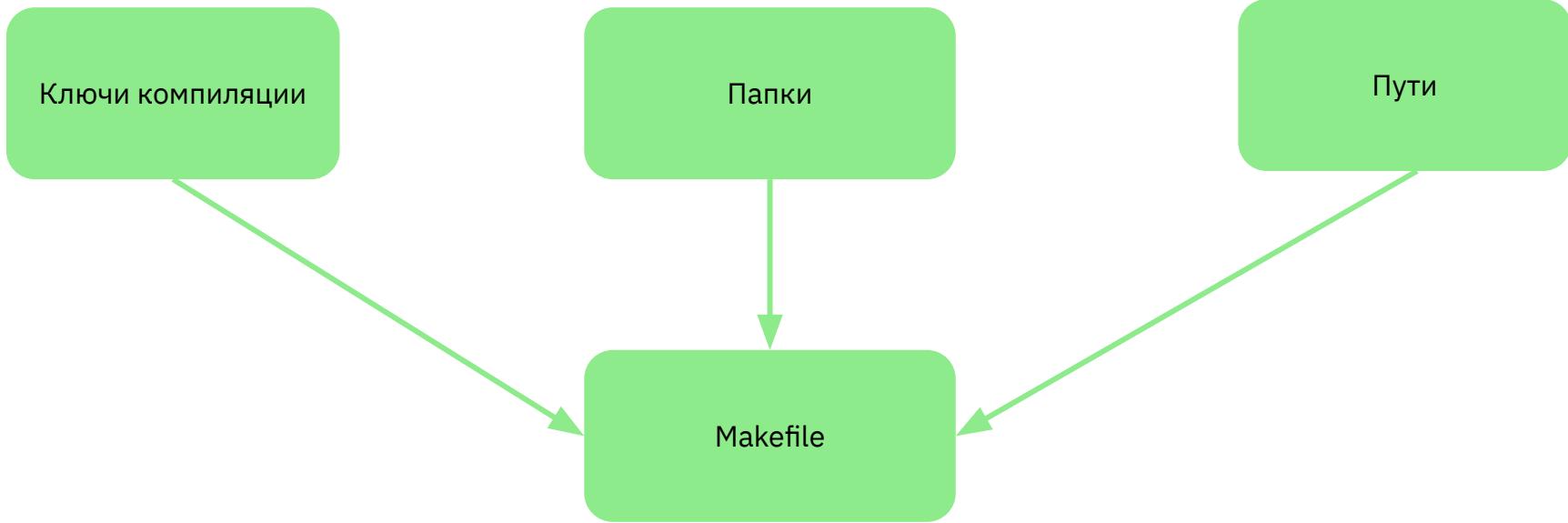


Автоматизация (make)





Makefile





Автоматизация сборки в CLI

```
all:  
    javac -sourcepath .src/ -d out src/ru/gb/dj/Main.java
```



Инициализация и использование переменной в make

```
1 SRC_DIR := src
2 OUT_DIR := out
3
4 JC := javac
5 JCFLAGS := -sourcepath .$(SRC_DIR)/ -d $(OUT_DIR)
```

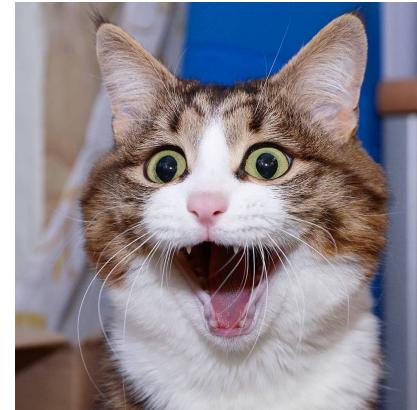
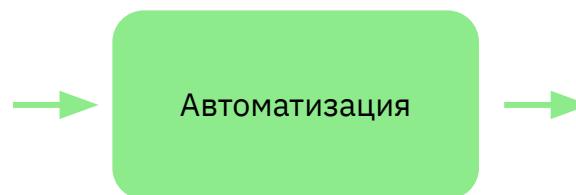
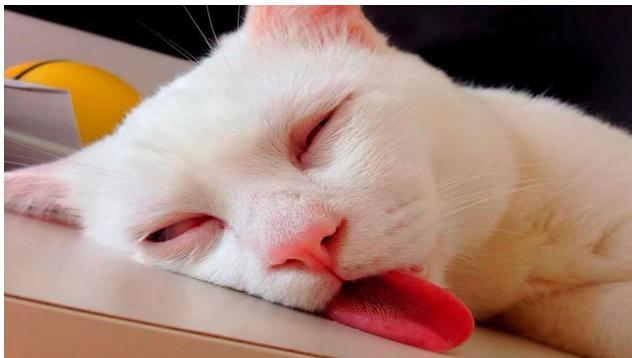


Вызовы make для разных таргетов

```
[ivan-igorevich@MacBook-Pro-Ivan sources-draft % make run
cd out && java ru.gb.dj.Main
Hello, world
[ivan-igorevich@MacBook-Pro-Ivan sources-draft % make clean
rm -R out
[ivan-igorevich@MacBook-Pro-Ivan sources-draft % cd out
cd: no such file or directory: out
[ivan-igorevich@MacBook-Pro-Ivan sources-draft %
```



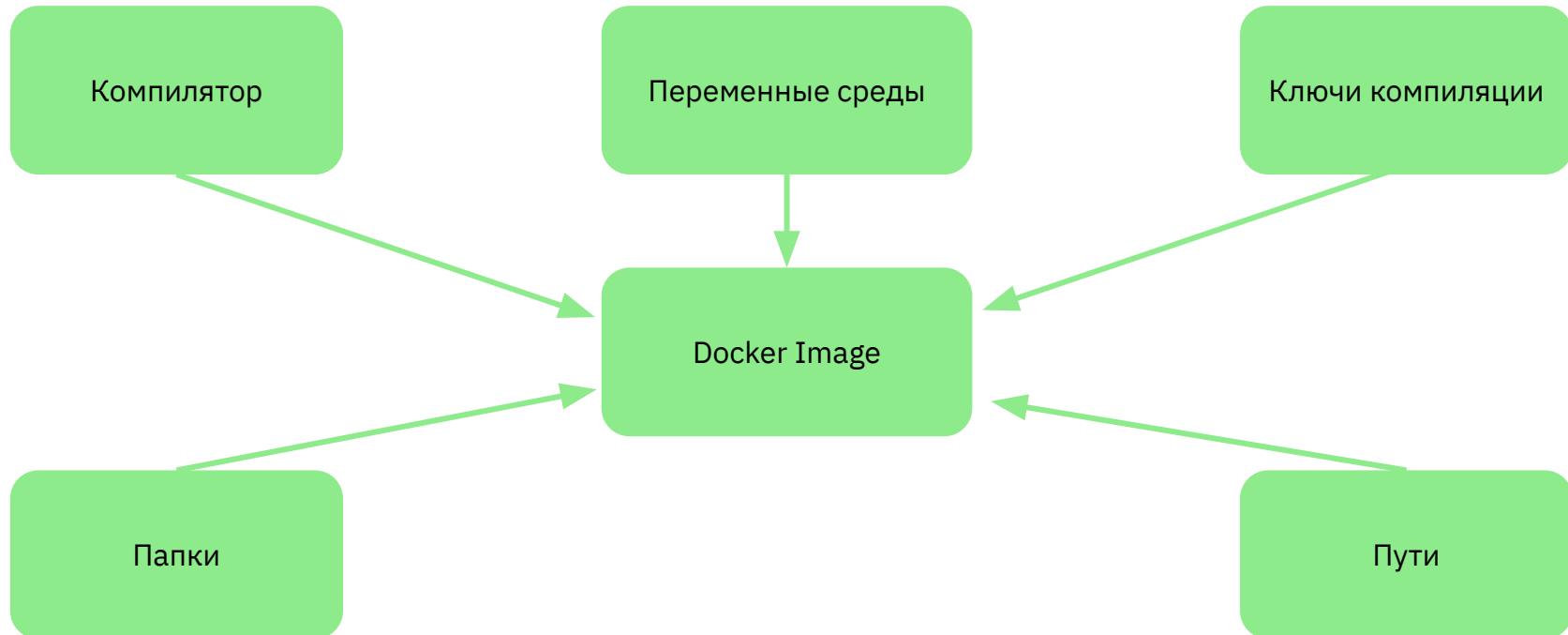
Автоматизация (Docker)



Docker

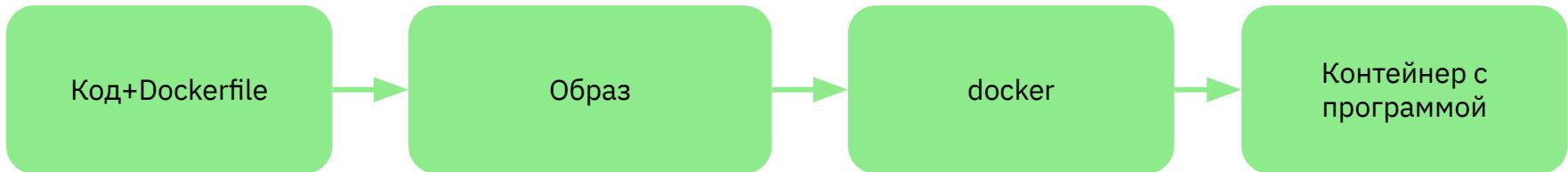


Автоматизация (Docker)





Автоматизация (Docker)





Автоматизация сборки в CLI, контейнеризация

```
1 # syntax=docker/dockerfile:1
2 FROM bellsoft/liberica-openjdk-alpine:11.0.16.1-1
3
4 COPY ./src ./src
5
6 RUN mkdir ./out
7
8 RUN javac -sourcepath ./src -d out ./src/ru/gb/dj/Main.java
9
10 CMD java -classpath ./out ru.gb.dj.Main
```



Команды для сборки и запуска контейнера

```
ivan-igorevich@MacBook-Pro-Ivan sources-draft % docker build . -t hellojava:latest
[+] Building 2.2s (13/13) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 37B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> resolve image config for docker.io/docker/dockerfile:1
=> CACHED docker-image://docker.io/docker/dockerfile:1@sha256:9ba7531bd80fb0a85863
=> [internal] load build definition from Dockerfile
=> [internal] load .dockerignore
=> [internal] load metadata for docker.io/bellsoft/liberica-openjdk-alpine:11.0.16
=> [internal] load build context
=> => transferring context: 1.89kB
=> [1/4] FROM docker.io/bellsoft/liberica-openjdk-alpine:11.0.16.1-1@sha256:050e6ed
=> CACHED [2/4] COPY ./src ./src
=> CACHED [3/4] RUN mkdir ./out
=> CACHED [4/4] RUN javac -sourcepath ./src -d out ./src/ru/gb/dj/Main.java
=> exporting to image
=> => exporting layers
=> => writing image sha256:262fa2090bf812c22d29163497b4c897634bdc93f90c3619ff0cb19
=> => naming to docker.io/library/hellojava:latest
```

```
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn
ivan-igorevich@MacBook-Pro-Ivan sources-draft % docker run --rm hellojava:latest
Hello, world
```



На этом уроке

- 📌 Краткая история
- 📌 Базовый инструментарий
- 📌 Что нужно скачать
- 📌 JDK, JRE, JVM
- 📌 Структура проекта
- 📌 CLI: сборка, пакеты, запуск
- 📌 Javadoc
- 📌 Makefile, Docker





Практическое задание

- 📌 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 веб-страницу с документацией ко всем пакетам;
- 📌 Создать Makefile с задачами сборки, очистки и создания документации на весь проект;
- 📌 *Создать два Docker-образа. Один должен компилировать Java-проект обратно в папку на компьютере пользователя, а второй забирать скомпилированные классы и исполнять их.





Помните, что единственная корочка,
ради которой действительно стоит учиться,
это корочка головного мозга.