

# Настройки приложения

ПРЕПОДАВАТЕЛЬ



# Юрий Костяной

**Java/Kotlin backend-разработчик**

- 3+ года опыта в коммерческой разработке
- 2+ года опыта в преподавании
- Проекты по интеграции сторонних платформ, CRM
- Проблемно-ориентированный подход в преподавании



# ВАЖНО:

- Камера должна быть включена на протяжении всего занятия.
- Если у Вас возник вопрос в процессе занятия, пожалуйста, поднимите руку и дождитесь, пока преподаватель закончит мысль и спросит Вас, также можно задать вопрос в чате или когда преподаватель скажет, что начался блок вопросов.
- Организационные вопросы по обучению решаются с кураторами, а не на тематических занятиях.
- Вести себя уважительно и этично по отношению к остальным участникам занятия.
- Во время занятия будут интерактивные задания, будьте готовы включить камеру или демонстрацию экрана по просьбе преподавателя.

# ПЛАН ЗАНЯТИЯ

1. Основной блок
2. Вопросы по основному блоку
3. Домашняя работа

1

# ОСНОВНОЙ БЛОК

# Введение

- Главное – настрой



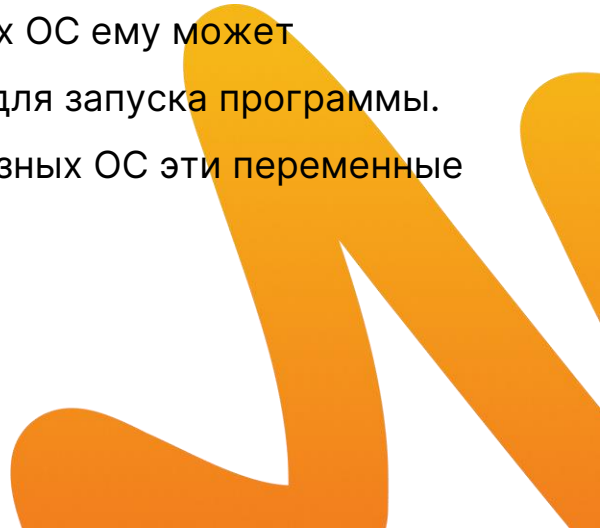
# Проблема

Большинство современных приложений являются мультязычными, имеют тёмную и светлую темы, могут сохранять временные файлы в папке пользователя, если доступ к папкам системного диска ограничен ОС.

При этом логика самого приложения не меняется.

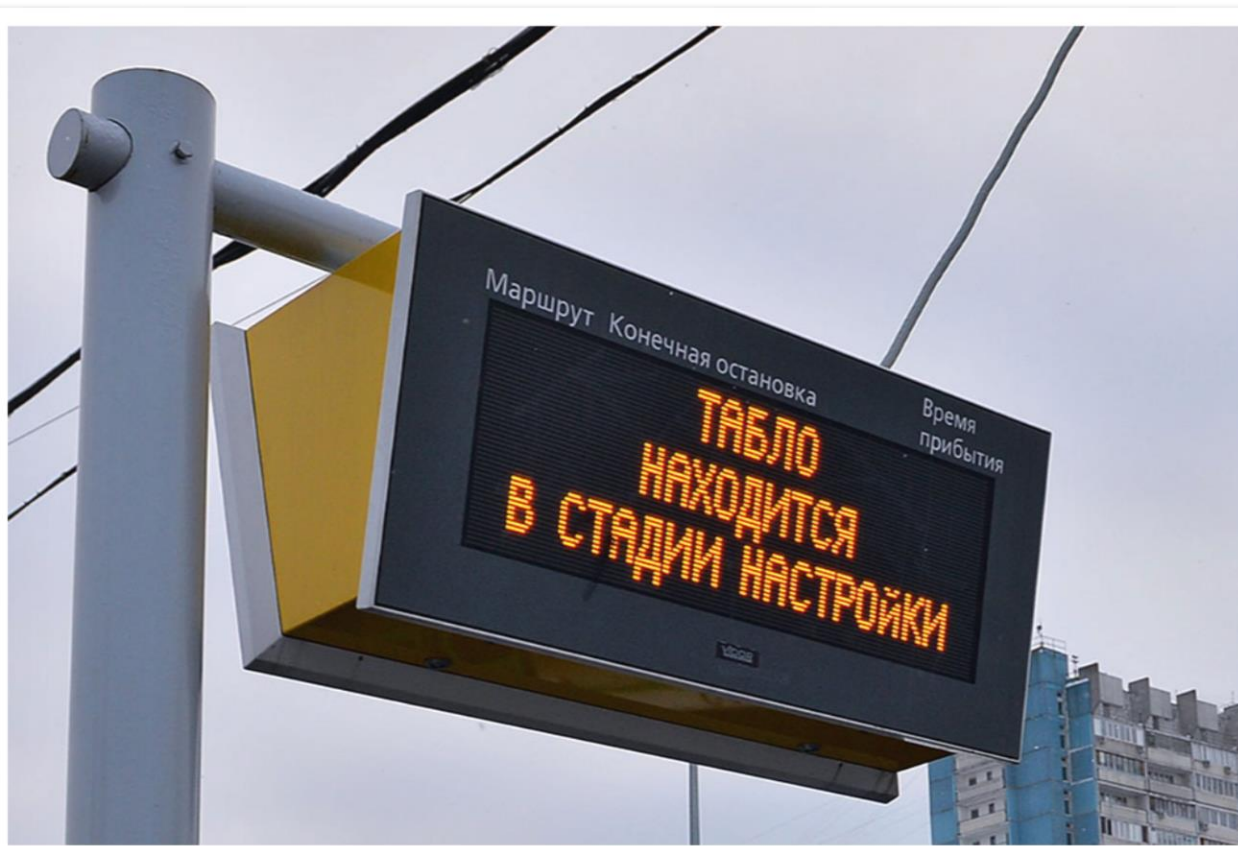
Если приложение является кроссплатформенным, то для разных ОС ему может потребоваться выполнить разные подготовительные действия для запуска программы. Например, прочитать значение переменных окружения. Но в разных ОС эти переменные могут отличаться.

*Каким образом можно добиться необходимой гибкости?*



# Ответ

Создать настройки  
приложения



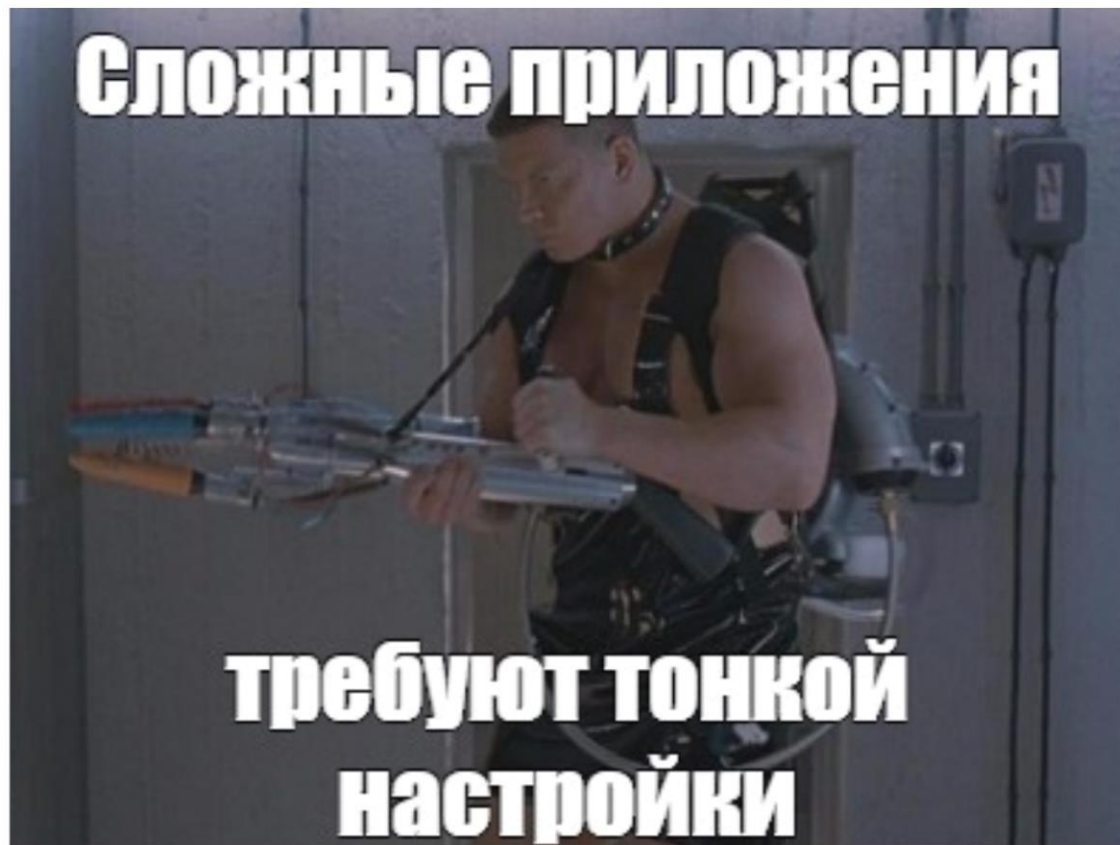


# Главное – настрой

## Способы настройки приложения

Существует несколько способов задать настройки для приложения:

- с помощью аргументов командной строки, указанных при запуске программы;
- с помощью переменных окружения, хранящихся в ОС;
- с помощью файла настроек.



# Главное – настрой

## Аргументы командной строки



Самый простой способ передать в программу данные – это аргументы командной строки.

Внутри метода `main()` мы можем работать с аргументами как с массивом строк.

```
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        System.out.println("args: " + Arrays.toString(args));
        boolean isIncognito = args.length > 0 && Boolean.parseBoolean(args[0]);
        if (isIncognito) {
            System.out.println("Программа запущена в режиме инкогнито");
        }
    }
}
```

Вывод:

*args: [true, arg1, arg2, someOtherArg, theLastOne]*

*Программа запущена в режиме инкогнито*

# Главное – настрой

## Переменные окружения

Для получения переменных и параметров окружения используются класс *System*:

- *getenv()* – метод получения системных переменных;
- *getProperties()* – метод получения параметров окружения.

```
import java.util.Arrays;
import java.util.Map;
import java.util.Properties;

public class Main {
    public static void main(String[] args) {
        Map<String, String> envMap = System.getenv();
        System.out.println("\nПеременные окружения:");
        envMap.forEach((k, v) -> System.out.printf("\t%s=%s\n", k, v));
        Properties props = System.getProperties();
        System.out.println("Настройки окружения:");
        props.forEach((k, v) -> System.out.printf("\t%s=%s\n", k, v));
    }
}
```

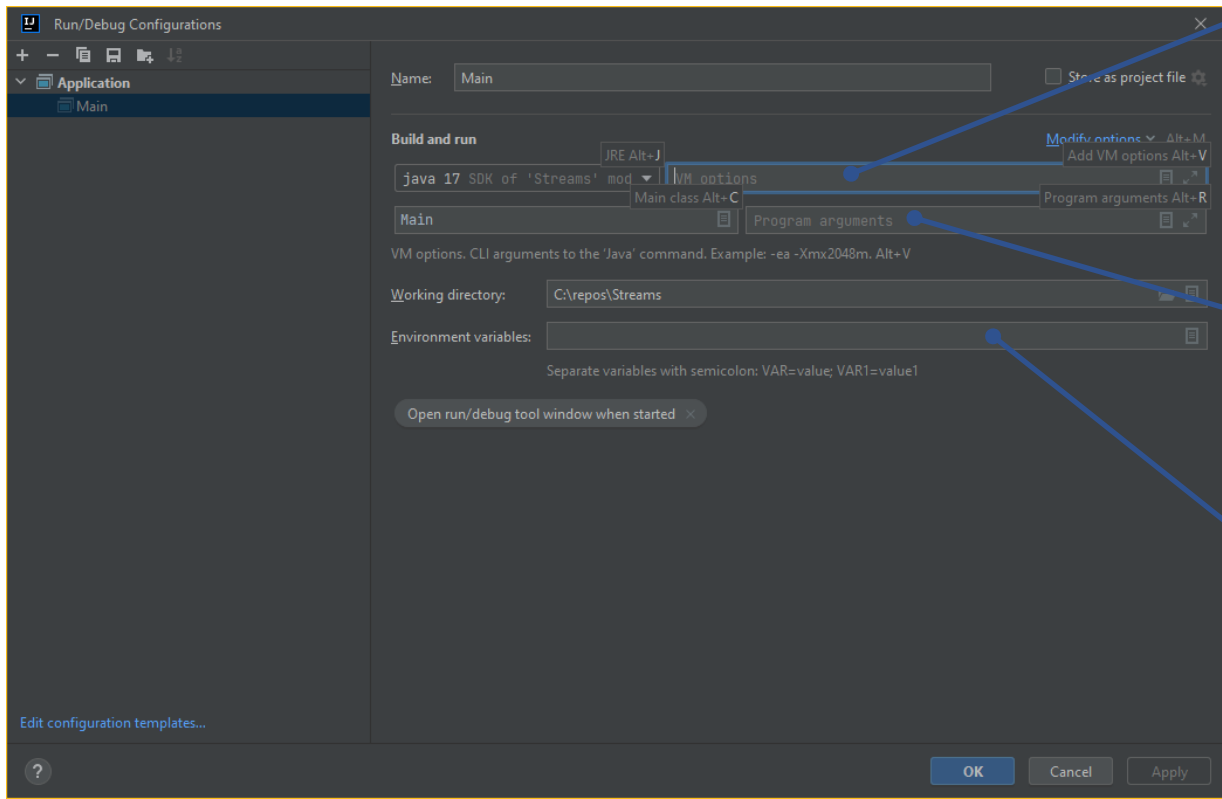


Влияние  
окружения необратимо

# Главное – настрой

## Способ задать параметры окружения

Переменные окружения можно задать в ОС или непосредственно в IDEA.



Настройки для запуска JVM

Аргументы, которые попадут в args[]

Переменные окружения, попадающие в getEnv()

# Задание

- 1 Создайте программу, которая принимает набор чисел и выводит их в отсортированном виде. В зависимости от аргументов запуска программы данные принимаются от пользователя или читаются из файла, путь, к которому передан в аргументах.
- 2 Выведите в консоль все переменные и параметры окружения.
- 3 Создайте переменную окружения, в которой поместите ссылку на техподдержку (просто на любой сайт). В программе выведите в консоль наименование операционной системы, сообщение, что данная ОС не поддерживается и ссылку на «техподдержку».

# Главное – настрой Файл конфигурации



Хранение настроек в переменных окружения и аргументах часто является неудобным и менее безопасным с точки зрения доступа. Поэтому для приложений чаще применяют файлы настроек.

В Java процесс получения настроек основан на механизме работы с файлами. А основной функционал по конвертированию настроек из строки в программную сущность выполняет класс *Properties*, который расширяет класс *HashTable* (старая версия *HashMap*).

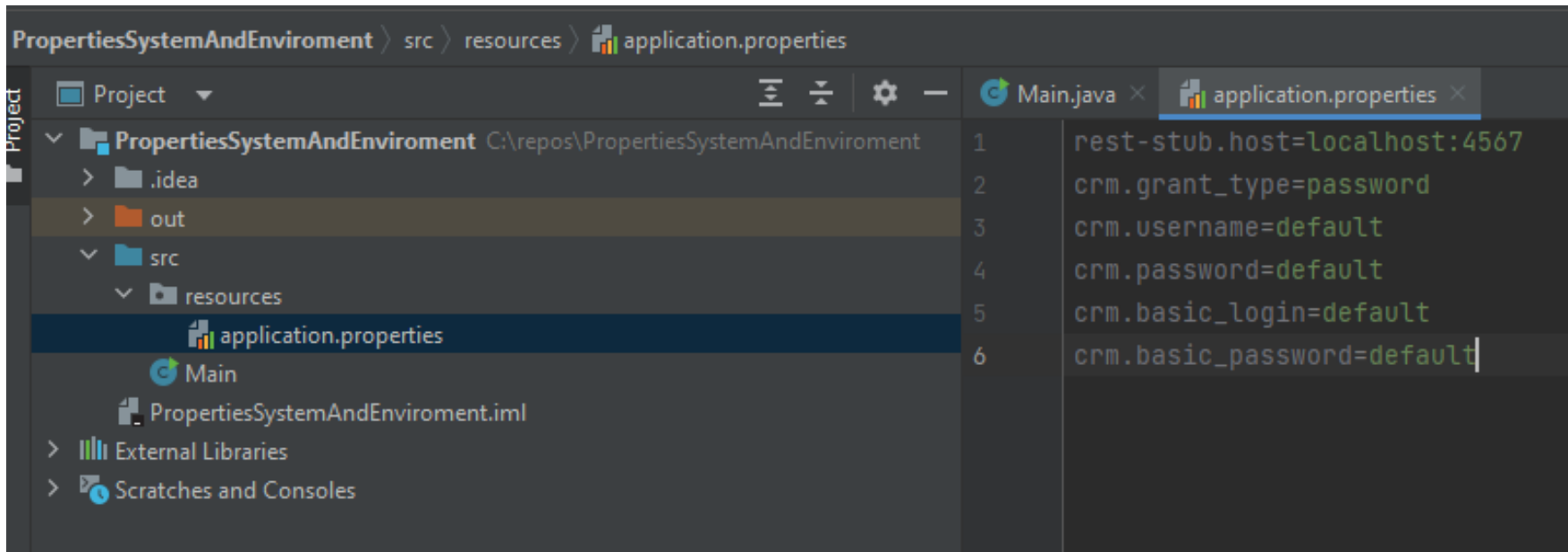
# Главное – настрой Файл конфигурации

```
import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

public class Main {
    public static void main(String[] args) {
        Properties appProps = getApplicationProperties();
        System.out.println("\nНастройки приложения из файла:");
        appProps.forEach((k, v) -> System.out.printf("\t%s=%s\n", k, v));
    }
    public static Properties getApplicationProperties() {
        Properties appProperties = new Properties();
        try (InputStream in = Thread.currentThread().getContextClassLoader()
            .getResourceAsStream("resources/application.properties")) {
            appProperties.load(in);
            return appProperties;
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

# Главное – настрой Файл конфигурации

Пример содержимого файла конфигурации



The screenshot shows an IDE interface with a project named 'PropertiesSystemAndEnviroment'. The left sidebar displays the project structure, including folders like '.idea', 'out', 'src', and 'resources'. The 'resources' folder is expanded, showing the 'application.properties' file. The main editor window displays the content of 'application.properties' with the following properties:

```
1 rest-stub.host=localhost:4567
2 crm.grant_type=password
3 crm.username=default
4 crm.password=default
5 crm.basic_login=default
6 crm.basic_password=default
```

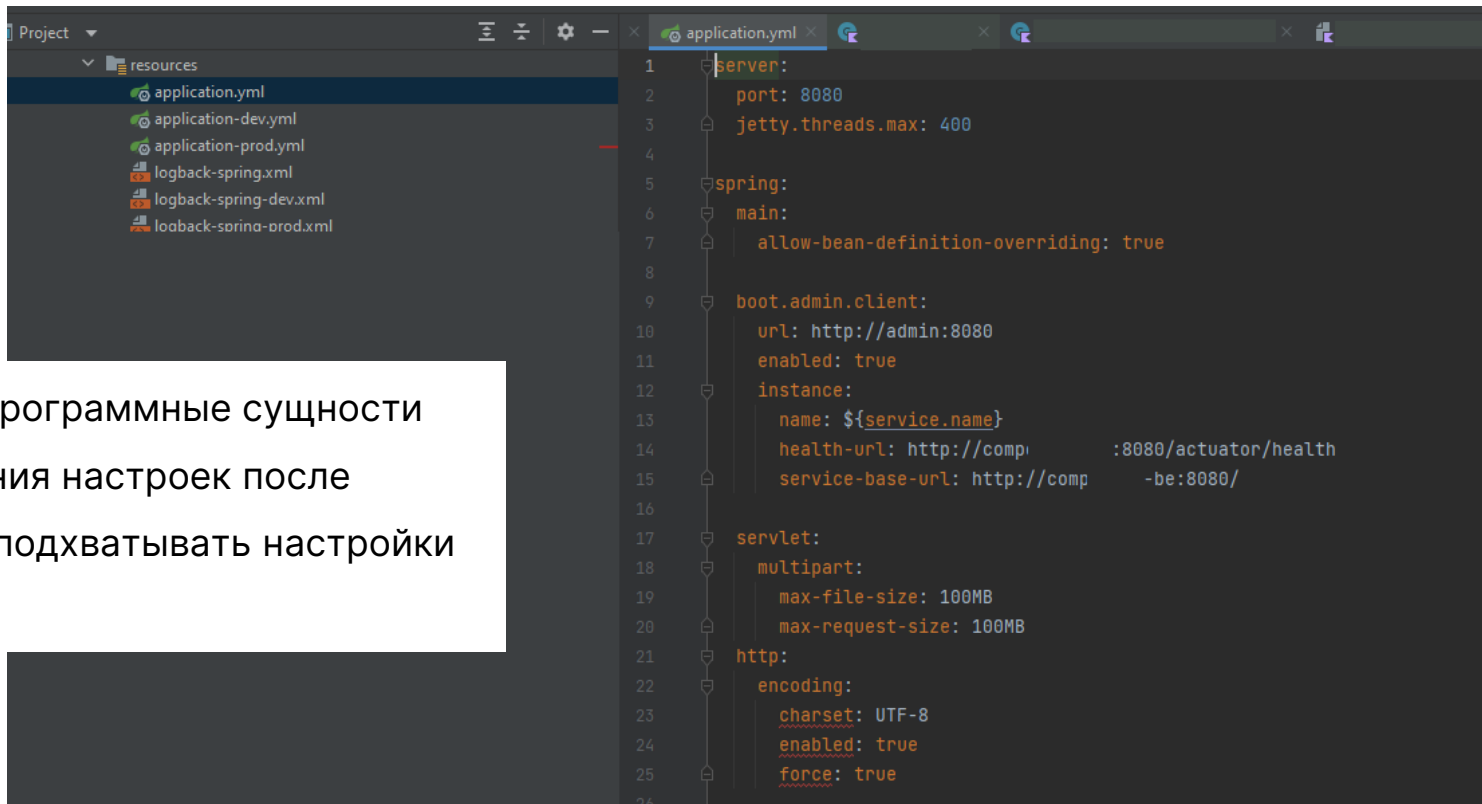


# Главное – настрой

# Файл конфигурации в Spring



В Spring для файлов конфигурации используется более современный язык разметки – YAML.



Spring использует программные сущности (классы) для хранения настроек после прочтения и умеет подхватывать настройки «на лету».

# Задание

Дополните программу из предыдущего задания. В зависимости от того, какая настройка указана в файле настроек, отсортированный список чисел должен быть выведен в консоль или сохранён в выходной файл. Имя файла должно быть задано в файле настроек.



2

# ВОПРОСЫ ПО ОСНОВНОМУ БЛОКУ

3

# Домашнее задание

# Домашнее задание

1 Создайте класс `Logger` – класс для логирования событий в программе (дата, время и текст сообщения). У каждого объекта данного класса должны быть методы `debug()`, `info()`, `warning()`, `error()`. Класс должен уметь выводить лог в консоль или в файл. В методе `main()` создайте объект класса `Logger`. Вызовите все доступные методы.

2 Залогируйте все доступные параметры и переменные окружения при старте программы.

3 Создайте файл настроек для вашей программы. В зависимости от указанных настроек программа должна включать заданный уровень логирования:

*debug* – логировать всё

*info* – логировать `info`, `warning` и `error` сообщения

*warning* – логировать `warning` и `error` сообщения

*error* – логировать только `error` сообщения.

В настройках должно быть указано, куда нужно выводить логи – в консоль или в файл.

# ЗАКЛЮЧЕНИЕ

