


# Вводный семинар по Java

Преподаватель: **Николай Амеличев**

- Ведущий разработчик (бэкенд) @ **Yandex**  **Cloud**
- Можно просто *Коля*



Чат курса: **TBD**

Личка: [@nvamelichev](https://t.me/nvamelichev)



GitHub с материалами по семинарам:

<https://github.com/nvamelichev/hse-java-spring-2022/>

# Организационное

- Расписание семинаров: **пятница, 09:30–10:50**
- **Всегда по Zoom**, т. к. физически я в СПб. Буфер: жду до 09:**35**, дальше начинаю занятие
- **???** семинаров, **21.01–???.05**
  - **21.01** — вводный семинар
  - **???.05** — выставление оценок, **???.05** — последний семинар, разговор на свободную тему
  - => **???** полноценных практических занятий
- Присутственные часы:
  - По будням читаю Telegram (чатик и личку) **минимум** дважды в день (12:00–13:00 и 19:00–21:00)
  - Два раза в неделю в ваших GitHub-репозиториях смотрю Issues и Pull Requests, пишу комментарии. Включите нотификации и не пропустите :-)
- Вопросы? Пишите в чат курса, вам помогут студенты-ассистенты [@team\\_mur](#) и [@starboy369](#)
  - **???** Тимур проходил этот же курс в прошлом году, может подсказать по технической части и пояснить, что от вас требуется на практических занятиях :-)
- Фидбэк: в чате курса пишите конструктивную критику, голосуйте за темы лекций и семинаров:-)

# Структура семинара (80 мин.)

«Режим лекции»

40-50 мин.

**Основная  
презентация**

30-40 мин.

**Мини-демо**  
**[+ Вопросы-ответы**  
**по теме презентации]**

«Режим практики»

50-60 мин.

**Большое демо,**  
**возможен интерактив**  
**с аудиторией :-)**

20-30 мин.

**Обсуждение демо.**  
**Вопросы и ответы**  
**по теме демо**

Демо-день

50-60 мин.

**Выступление всех команд:**  
**демо проектов + вопрос-ответ**

~7-10 мин./проект

20-30 мин.

**Мини-демо**  
**или короткая презентация.**  
**Без** интерактива,  
вопросов-ответов

# Командный проект (1)

- **Команда из 2-4 человек (оптимально – 3 человека)**

Если не договоритесь, члены команды будут выбраны с помощью random.org :-)

- Цель – сделать **простой, но законченный продукт** вида «Java-библиотека + CLI к ней».

- **Не** мобильное и **не** веб-приложение

- Некоторые идеи: <https://github.com/nvamelichev/hse-java-spring-2022/blob/main/project-ideas.md>

- Преподаватель – в роли **Product Owner** («владельца продукта»).

Утверждает вашу идею, смотрит демки, задаёт вопросы, предлагает варианты развития проекта, **может (и будет!) менять требования** во время разработки

- **Итеративная, гибкая (Agile) разработка**

- **Product Vision** («вИдение продукта») начинайте обдумывать уже сейчас.

- @see <https://leadstartup.ru/db/product-vision>

- Активная разработка начнётся, когда вы изучите основы Java (синтаксис, управляющие конструкции, основы ООП в Java, дженерики, коллекции, исключения).

- До этого – формирование команд, обсуждение и выбор темы проекта, сбор требований, ОО-проектирование.

- На каждом **демо-дне** от каждой команды – мини-демо проекта на 5-7 мин.

И будем обсуждать приоритеты по проекту, мои комментарии к PR/Issues и т.п.

# Командный проект (2)

- Взаимодействие с преподавателем по проекту – GitHub: Pull Requests, Issues.
- Основные решения, принятые **внутри команды**, и их обоснование – документируйте в GitHub (хотя бы в PR/Issues, а лучше – в Wiki.)
- **Обязательно:**
  - Юнит-тесты
  - Сборка Maven или Gradle (предпочтительнее Maven, т. к. у преподавателя с ним больше опыта работы :-))
  - Continuous Integration (GitHub Actions)
  - Docker-образ/Native Image. На начальном этапе **можно** исполняемый JAR-файл + скрипт для запуска
- **Можно:**
  - Популярные библиотеки, напр. Google Guava
  - Паттерны, абстракции (без фанатизма :-))
- **Нельзя:**
  - Сделать тривиальную «обёртку» над готовой внешней библиотекой
  - Скопипастить готовый командный проект, который делали в прошлом году :-D

# Примерные темы семинаров

1. **Build 1:** Maven, fundamentals — **21.01**
2. **Build 2:** Maven, advanced topics & demo — **28.01**
3. **OOD 1:** Object-Oriented Design. Class-Responsibility-Collaborators (CRC) Cards. Basic UML Diagrams (Class, Sequence, Activity/Statechart). SOLID, DRY, YAGNI, KISS
4. **OOD 2:** GoF Patterns and how to read the GoF book. Strategy, Decorator, Proxy. Iterator, Visitor, Observer. Singleton, Abstract Factory, Builder, Static Factory (maybe) DDD?
5. **Logging:** slf4j, Logback/Log4j2. TDD. (maybe) BDD?
6. **Testing:** xUnit (JUnit5-vintage). Testing fundamentals (Fowler's test type diagram). AssertJ/GoogleTruth/Hamcrest. Mockito
7. **(maybe) Java Debugging:** Basic debugging ideas, basic debugger features w/demo. Old-style Profilers (JVisualVM) w/demo (maybe) async-profiler and flame graphs? (maybe) Remote debugging?
8. **(maybe) Annotations and How to Use Them:** @Override, @Nonnull, @Nullable, @Json...: validation, code checkers, (de)serialization, ORMs, etc. A возможно, про аннотации будет лекция и всё
9. **Dependency Injection:** Basics (Inversion of Control). Service Locator vs Dependency Injection. Roll-your-own DI. @Inject. Demo: Google Dagger
10. **Packaging Java for VMs:** 1: Uberjar (aka fat jar). maven-assembly-plugin. The Dark Art of Shading (and why you mostly do not need it)
11. **Packaging Java for Containers 2:** Docker Containers, Images and Registries. Manual Dockerfile. Fabric8 docker-maven-plugin. Google Jib (Java Image Builder). (maybe) Docker Compose and k8s concepts
12. **(maybe) Packaging & Containerization 3:** GraalVM native-image. Static Java Problems & Perspectives (Excelsior JET, Project Leyden)
13. **Continuous Integration/Continuous Deployment:** Demo using GitHub Actions
14. **(maybe) Code Quality:** Sun Code Style guidelines. JavaDoc. Test Coverage (via IntelliJ). Checkstyle. maven-enforcer-plugin. Sonar, Coverity...
15. **(maybe) Methodology:** Elements of Agile methodologies (Scrum, XP, Kanban). Pair (and DESpair) programming in the pandemic age. Agile Waterfall(TM) and other managerial atrocities.