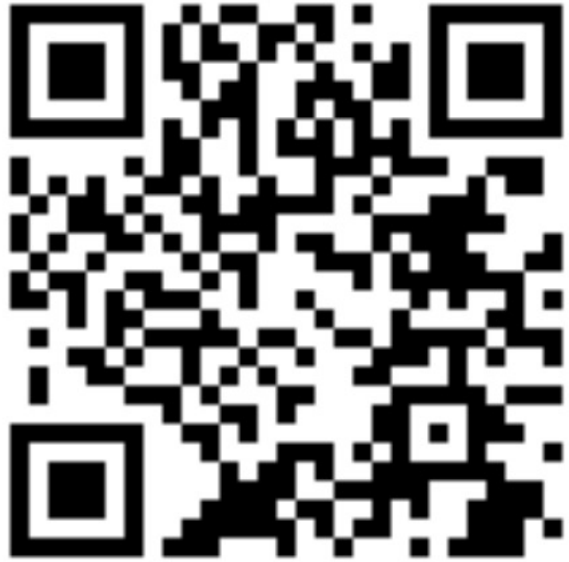
Промышленное программирование на Java. Семинары (2023)

Intro

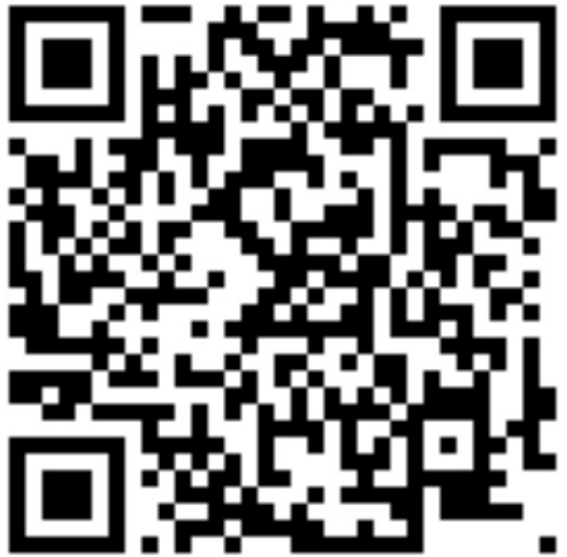
Преподаватель: Гималетдинова Альбина (можно просто *Альбина*)

- HSE’17 (ФКН ПИ), HSE’20 (Менеджмент, Управление проектами)
- Разработчик в Яндекс Маркете 
- Tg: @albina_astr
- Автор курса – Николай Ameличев, Яндекс Облако



- https://t.me/+xH72UVvllZ1iNTli

Важные объявления (пин),
Вопросы и ответы,
Обратная связь



- https://github.com/albina-astr/hse-java-spring-2023

Материалы (презентации, код),
Прогресс всех команд,
Всё о требованиях, дедлайнах.

Расписание

- Вторник, 18:10 – 19:30, дружно-вместе начинаем в **18:15**
- 3 и 4 модуль, 20 семинаров, до 20.06. см. [график учебного процесса]
  - **24.01** – вводный семинар
  - Схема: **семинары + демо-дни**

```
Январь      24.01,      31.01
Февраль     07.02,      14.02,      21.02,      28.02
Март        07.03,      14.03       21.03
-------- Сессия 28.03..03.04 ----------
Апрель      04.04,      11.04⭐,     18.04,      25.04
-------- Каникулы: 01.05, 08.05, 09.05 -------
Май         02.05⭐,     16.05,      23.05⭐,     30.05
Июнь        06.06⭐,     13.06⭐*,    20.06
```

Примерное распределение работ:

- Сбор требований
- Проектирование
- Разработка и тестирование
- Сборка и деплой

- Планируем закончить все проекты к 13.06
- 13.06 подводим итоги, выставляем оценки
- 20.06 последний семинар, свободная тема :)

Коммуникация

- Чатик курса в Telegram. Не стесняемся там задавать вопросы :)
- Ассистенты курса:
  - Аюбджон @starboy369
  - Тимур @team_mur, проходил этот курс ранее :)
- Читаю личку в tg: @albina_astr, но лучше пишите в чат курса. Ваши вопросы могут совпадать.
- Работаем в **GitHub**.
  - Смотрю ваши проекты несколько раз в неделю (ср, пт). Ассистенты будут помогать с Pull Requests (PR) (ревью).
  - Issues
    - Вы: заводите Issues, чтобы не забыть что и кому надо сделать.
    - Я: буду заводить Issues по результатам демо-дней (чек-лист).
  - **Pull Requests (PR)**: по крупным этапам проекта буду смотреть и комментировать + ассистенты
  - **Документация** (Wiki/*.md файлы в Source) — буду смотреть:
    - требования (Product Vision, User Stories/Use Cases, DoD – Definition of Done)
    - высокоуровневое описание архитектуры (результаты ОО-дизайна)
  - **Исходный код** (Source): буду бегло просматривать на первых этапах разработки, потом если попросите/если у меня возникнут вопросы к архитектуре, тестам, сборке и т.п.
- **Включите нотификации в GitHub**, чтобы не пропустить мои комменты и Issues.

Команды и проекты

Уже взятые проекты из списка идей ~~вычёркиваются~~.

Команда	Проект	Ближайший дедлайн	GitHub
[redacted]	5. Статистика по исходному коду на C-like языках программирования	Четвёртый релиз до **21.05**: https://github.com/[redacted]	[redacted]
[redacted]	10. Карточки-запоминалки (Flash Cards)	Четвёртый релиз до **21.05**: https://github.com/[redacted]	[redacted]

* — повышенный уровень сложности

Дедлайны

▶ Прошедшие

Предстоящие:

- **Четвёртый, финальный релиз**: deadline 21.05, hard deadline 28.05. Критерии оценки:
  - **Исправлены все проблемы**, обнаруженные в предыдущих релизах.
  - Проект **упакован любым из следующих способов**: Docker-образ, GraalVM native-image, jlink.
  - В **системе непрерывной интеграции**, интегрированной с GitHub (Github Actions, Travis CI, ...), успешно настроен **запуск тестов** на каждый коммит в ветке `main` и в пул-реквестах.
  - [необязательно] Если в CI дополнительно настроена сборка проекта из `main` в исполняемый артефакт (докер-образ, native-image, jlink), **это будет преимуществом**.»

Общие требования

Цель проекта – создать **простой, но законченный продукт** вида

Продукт должен:

- делать одно дело/один класс дел, но хорошо (Unix way);
- быть нетривиальным — не просто обёрткой над известной Ja
- быть достаточно гибким, чтобы можно было опробовать в нём
  логгирования/сериализации/организации Command-Line Interf

Будем практиковать **итеративную, гибкую (Agile) разработку**. П
который:

Merged **Add pv and us** #1

[redacted] merged 3 commits into `main` from `water` on [redacted]

[redacted] reviewed on [redacted] View changes

[redacted] left a comment Owner Author

А извлекать текст из картинок не планируется? **@nvamelichev**

nvamelichev commented on [redacted] Collaborator

> А извлекать текст из картинок не планируется? **@nvamelichev**

Эту идею можно оставить "на будущее", вдруг ваш проект быстро полетит и вам скучно станет?

А пока OCR мне кажется излишним. Сейчас лучше сосредоточиться на извлечении данных из пары хорошо известных форматов, например, PDF и DOCX.

1

nvamelichev requested changes on [redacted] View changes

user_stories.md

```
... ...  @@ -0,0 +1,8 @@
1  + ### User Stories
2  +
```

Collaborator

рограммиста, который конвертнул 499 файликов а потом в ту же
й конвертации все 500 файлов обрабатывать, если новый/

Третий релиз #10

Closed 3 tasks done **nvamelichev** opened this issue on [redacted] · [redacted] comments

nvamelichev commented on [redacted] • edited by [redacted] Collaborator

- [x] Сделать разбор командной строки (долг со второго релиза, см. **Второй релиз** #4)
- [x] Отдельный режим работы, в котором кешируется соответствие «входной файл (ну, хеш от него) → извлечённый текст»
  Пользовательский сценарий: «…извлекать текст только из новых или изменившихся файлов, например, при регулярной обработке одной и той же папки»
- [x] Сборка с помощью jlink или Docker

Структура семинара

«Режим лекция + практика»	Демо-день
50-60 мин. **Лекция + демо**, возможен **интерактив с аудиторией** :)	50-60 мин. **Выступление всех команд**: демо проектов + вопрос-ответ ~7-10 мин./проект
20-30 мин. **Обсуждение** демо. **Вопросы и ответы** по теме демо	20-30 мин. **Мини-демо** **или короткая презентация** Без интерактива, вопросов-ответов

Структура оценки

Оценка за курс = min(
    1.0,
    round(**оценка за проект** [0.4 .. 0.8] +
        оценка за д/з [0.0 .. 0.4], 1)
)

(суммируем баллы за домашку и проект, округляем до десятых.
больше 1.0 получить нельзя.)

Оценка за проект = 0.6 (базовый балл) -
    суммарный штраф [-0.2 .. 0.0] +
    суммарный бонус [0.0 .. 0.2]

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
  - Идеи проектов (можно взять свою): будут лежать на GitHub.
- **Итеративная, гибкая (Agile)** разработка + Deadline Driven Development
  - Преподаватель – в роли **Product Owner** («владельца продукта») и **заказчика**
  - Утверждает вашу идею, смотрит демки, задаёт вопросы, предлагает варианты развития проекта
  - **Может (и будет!) менять требования** во время разработки :(
  - Взаимодействует с командой через GitHub и Telegram (но в основном GitHub)
  - Первая фаза – **сбор требований** (до 14.02):
    - Выбрать **тему проекта** до 07.02 (выбор можно изменить до 12.02 простым большинством голосов в команде)
    - Сформулировать **Product Vision**, «в**И**дение продукта» до 14.02
    - @see https://leadstartup.ru/db/product-vision, https://intuit.ru/studies/courses/2188/174/lecture/4724?page=2
    - Описать пользовательские истории (**User Stories**)/сценарии использования (**Use Cases**) до 14.02
    - @see https://ru.wikipedia.org/wiki/Пользовательские_истории,
    - @see https://pmclub.pro/articles/user-story-pora-primenyat-pravilno

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

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
  - Идеи проектов (можно взять свою): https://github.com/nvamelichev/hse-java-spring-2022/blob/main/project-ideas.md
- **Итеративная, гибкая (Agile)** разработка + Deadline Driven Development
  - Преподаватель – в роли **Product Owner** («владельца продукта») и **заказчика**
  - Утверждает вашу идею, смотрит демки, задаёт вопросы, предлагает варианты развития проекта
  - **Может (и будет!) менять требования** во время разработки :(
  - Взаимодействует с командой через GitHub и Telegram (но в основном GitHub)
  - Первая фаза – **сбор требований** (до 14.02):
    - Выбрать **тему проекта** до 07.02 (выбор можно изменить до 12.02 простым большинством голосов в команде)
    - Сформулировать **Product Vision**, «в**И**дение продукта» до 14.02
    - @see https://leadstartup.ru/db/product-vision, https://intuit.ru/studies/courses/2188/174/lecture/4724?page=2
    - Описать пользовательские истории (**User Stories**)/сценарии использования (**Use Cases**) до 14.02
    - @see https://ru.wikipedia.org/wiki/Пользовательские_истории,
    - @see https://pmclub.pro/articles/user-story-pora-primenyat-pravilno

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

Командный проект (2)

- **Обязательно**:
  - Стандартная система сборки (Maven или Gradle)
  - Юнит-тесты — обязательно со 2-го демо-дня, желательно с 1-го
  - Исполняемый артефакт (Docker, GraalVM native-image, jlink image) — обязательно с 3-го демо-дня. До этого, **можно** исполняемый JAR-файл + запускать руками/скриптом (java -jar ...)
  - Сборка и деплой в системе непрерывной интеграции (GitHub Actions) — к 4-му демо-дню
- **Можно**:
  - Библиотеки, напр. Google Guava
    - Можно даже библиотеку, которую сделает соседняя команда, но преподаватель должен об этом знать заранее
  - Паттерны, абстракции (без фанатизма :-))
- **Нельзя**:
  - Тривиальная «обёртка» над готовой внешней библиотекой, программой, веб-сервисом...
  - Любая форма плагиата, в т.ч. креативно переработать студенческие проекты прошлого года

Темы семинаров (примерные)

1. **Intro**: 24.01
2. **Build**: Maven, fundamentals, advanced topics & demo — 31.01
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. Singeton, Abstract Factory, Builder, Static Factory

(maybe) DDD?

5. **Testing**: xUnit (JUnit5-vintage). Testing fundamentals (Fowler's test type diagram). AssertJ/GoogleTruth/Hamcrest. Mockito. TDD. (maybe) BDD?
6. **Logging**: slf4j, Logback/Log4j2
7. **(maybe) Java Debugging**: Basic debugging concepts, 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, Static Analysis, (de)serialization, ORMs, etc.

Но возможно, про аннотации будет лекция и всё

9. **Dependency Injection**: 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 (+ basic container implementation details, e.g. chroot and namespaces). 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**: Live 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 (Scrum, XP, Kanban). Pair programming (risks, advantages). Agile WaterfallTM and other management atrocities