OTUS

Java Developer. Professional

Продвинутое программирование на Java: все, что надо знать и уметь Middle+ специалисту

Длительность курса: 170 академических часов

1 Язык и платформа Java

1 Подготовка к курсу. ДЗ

Цели занятия:

познакомиться с программой курса;
рассмотреть основные инструменты.

Краткое содержание:

решение организационных вопросов;
знакомство с инструментами: Git, Idea, gradle.

Домашние задания

1 Проект gradle с модульной структурой

Цель: научиться создавать проект Gradle (Maven), подготовиться к выполнению домашних заданий.

1) Создайте аккаунт на github.com (если еще нет)
2) Создайте репозиторий для домашних работ
3) Сделайте checkout репозитория на свой компьютер
4) Создайте локальный бранч hw01-gradle
5) Создать проект gradle
6) В проект добавьте последнюю версию зависимости
<groupId>com.google.guava</groupId>
<artifactId>guava</artifactId>
7) Создайте модуль hw01-gradle
8) В модуле сделайте класс HelloOtus
9) В этом классе сделайте вызов какого-нибудь метода из guava
10) Создайте "толстый-jar"
11) Убедитесь, что "толстый-jar" запускается.
12) Сделайте pull-request в gitHub
13) Ссылку на PR отправьте на проверку (личный кабинет, чат с преподавателем).

При желании, можно сделать maven-проект и далее на курсе работать с maven-ом.
Для Maven инструкция аналогичная (просто в тексте замените Gradle на Maven).

2 **Дополнение к gradle, история изменения языка**

Цели занятия:

углубить знания о gradle;
познакомиться с текущей ситуацией в мире java.

Краткое содержание:

управление зависимостями в gradle;
версии Java, изменения в политике Oracle.

---

3 **QA и тестирование**

Цели занятия:

познакомиться с junit и mockito;
объяснить на примере, что такое "тестируемое приложение"

Краткое содержание:

виды тестов;
инструменты: junit, mockito.

---

4 **Контейнеры и алгоритмы. ДЗ**

Цели занятия:

познакомиться с Generic-ами в Java и со стандартными коллекциями.

Краткое содержание:

Generics;
стандартные коллекции JDK.

Домашние задания

1 Применение коллекций

Цель: попрактиковать различные аспекты коллекций.
познакомиться с реализациями Map.

надо сделать todo в классах из пакета homework.
Все тесты должны проходить.
Предполагается использование встроенного в jdk функционала, поэтому реализация методов должна быть буквально из нескольких строк.

---

5 **Инструменты для преобразования контейнеров, unsafe, jmh**

Цели занятия:

объяснить на примере принципы создания коллекций;
познакомиться с пакетом unsafe, утилитой JMH и популярными библиотеками коллекций.

Краткое содержание:

Unsafe;
утилита JMH;
принципы построения hashMap;
Apache Commons;
Google Guava.

---

6 **Аннотации. ДЗ**

Цели занятия:

познакомиться с механизмом Reflection;
объяснить, что такое Аннотации и как их можно сделать.

Краткое содержание:

Reflection;
аннотации.

Домашние задания

1 Свой тестовый фреймворк

Цель: научиться работать с reflection и аннотациями,
понять принцип работы фреймворка junit.

Написать свой тестовый фреймворк.

Поддержать свои аннотации @Test, @Before, @After.

Запускать вызовом статического метода с именем класса с тестами.

Т.е. надо сделать:
1) создать три аннотации - @Test, @Before, @After.
2) Создать класс-тест, в котором будут методы, отмеченные аннотациями.
3) Создать "запускалку теста". На вход она должна получать имя класса с тестами, в

котором следует найти и запустить методы отмеченные аннотациями и пункта 1.
4) Алгоритм запуска должен быть следующий::
метод(ы) Before
текущий метод Test
метод(ы) After
для каждой такой "тройки" надо создать СВОЙ объект класса-теста.
5) Исключение в одном тесте не должно прерывать весь процесс тестирования.
6) На основании возникших во время тестирования исключений вывести статистику выполнения тестов (сколько прошло успешно, сколько упало, сколько было всего)
7) "Запускалка теста" не должна иметь состояние, но при этом весь функционал должен быть разбит на приватные методы. Надо придумать, как передавать информацию между методами.

---

7 **Lombok**

Цели занятия:

познакомиться с технологиями annotation Processor и возможностями Lombok.

Краткое содержание:

проблема Boilerplate-кода в Java-проектах;
дерево интеграции функционала;
как Annotation Processor может взломать компилятор?;
Lombok – интеграция в проект;
Record’ы для Java 13-;
mutable-Record’ы;
декларативные Builder’ы;
различные мелочи.

---

8 **Сборщик мусора. ДЗ**

Цели занятия:

познакомиться со сборщиком мусора в Java.

Краткое содержание:

концепция сборки мусора в JVM;
виды сборщиков мусора;
мониторинг работы сборщиков;
примеры проблем производительности, связанных с мусором.

Домашние задания

1 Определение нужного размера хипа

Цель: на примере простого приложения понять какое влияние оказывают сборщики мусора

Есть готовое приложение (модуль homework)
Запустите его с размером хипа 256 Мб и посмотрите в логе время выполнения.

Пример вывода:
spend msec:18284, sec:18

Увеличьте размер хипа до 2Гб, замерьте время выполнения.
Результаты запусков записывайте в таблицу.
Определите оптимальный размер хипа, т.е. размер, превышение которого,
не приводит к сокращению времени выполнения приложения.

Оптимизируйте работу приложения.
Т.е. не меняя логики работы (но изменяя код), сделайте так, чтобы приложение работало быстро с минимальным хипом.
Повторите измерения времени выполнения программы для тех же значений размера хипа.

9 **Углубленные основы (примитивные типы, Remote debug, Hot swap)**

Цели занятия:

объяснить детали устройства типов данных в Java;
познакомиться с механизмами Remote Debug и Hot swap;
познакомиться с утилитой Jol.

Краткое содержание:

примитивные типы, строки, массивы;
память, которую занимают объекты;
изменение размера объекта, утилита JOL;
remote Debug;
hot swap.

---

10 **Байт код, class-loader, инструментация, asm. ДЗ**

Цели занятия:

познакомиться с принципами работы виртуальной машины Java, ClassLoader-ами и байт-кодом.

Краткое содержание:

байт код. Содержание .class. декомпиляция;
Class Loader: примеры Class Lloader'ов;
самодельный простой Cloass Loader;
Instrumentation;
ASM – инструмент для анализа и манипуляций с байт-кодом.

Домашние задания

1 Автоматическое логирование.

Цель: Понять как реализуется AOP, какие для этого есть технические средства.

Разработайте такой функционал:
метод класса можно пометить самодельной аннотацией @Log, например, так:

```
class TestLogging {
@Log
public void calculation(int param) {};
}
```

При вызове этого метода "автомагически" в консоль должны логироваться значения параметров.

Например так.

```
class Demo {
public void action() {
new TestLogging().calculation(6);
}
}
```

В консоле дожно быть:
executed method: calculation, param: 6

Обратите внимание: явного вызова логирования быть не должно.

Учтите, что аннотацию можно поставить, например, на такие методы:
public void calculation(int param1)
public void calculation(int param1, int param2)
public void calculation(int param1, int param2, String param3)

P.S.
Выбирайте реализацию с ASM, если действительно этого хотите и уверены в своих силах.

---

11 **Функциональное программирование в Java**

Цели занятия:

рассмотреть введение в функциональное программирование (ФП);
познакомиться с возможностями ФП, которые появились в Java 8.

Краткое содержание:

чистые функции;
немутирующие данные;
лямбда-функции;
монады;
Streams.

2 Проектирование

1 Концепты проектирования ООП. ДЗ

Цели занятия:

объяснить принципы SOLID и общие критерии идеальной архитектуры.

Краткое содержание:

идеальная архитектура;
Coupling и Cohesion;
полиморфизм;
SOLID Принципы.

Домашние задания

1 Эмулятор банкомата

Цель: Применить на практике принципы SOLID.

Написать эмулятор ATM (банкомата).

Объект класса ATM должен уметь:
- принимать банкноты разных номиналов (на каждый номинал должна быть своя ячейка)
- выдавать запрошенную сумму минимальным количеством банкнот или ошибку если сумму нельзя выдать
Это задание не на алгоритмы, а на проектирование.
Поэтому оптимизировать выдачу не надо.
- выдавать сумму остатка денежных средств

В этом задании больше думайте об архитектуре приложения.
Не отвлекайтесь на создание таких объектов как: пользователь, авторизация, клавиатура, дисплей, UI (консольный, Web, Swing), валюта, счет, карта, т.д.
Все это не только не нужно, но и вредно!

2 **Behavioral patterns**

Цели занятия:

объяснить поведенческие паттерны проектирования

Краткое содержание:

паттерны: Observer, Command, Chain of responsibility, Memento, State, Strategy, Visitor.

---

3 **Creational patterns**

Цели занятия:

рассмотреть "создающие" паттерны проектирования.

Краткое содержание:

паттерны: Factory Method, Abstract, FactoryBuilder, Prototype, Singleton, Object Pool.

---

4 **Structural patterns. ДЗ**

Цели занятия:

рассмотреть структурные паттерны проектирования

Краткое содержание:

паттерны:
- Adapter
- Decorator
- Bridge
- Composite
- Facade
- Flyweight
- Proxy

Домашние задания

1 Обработчик сообщений

Цель: Применить на практике шаблоны проектирования.

Реализовать todo из модуля homework.

3 Работа с окружением

1 **Сериализация. ДЗ**

Цели занятия:

познакомиться с функционалом сериализации объектов.

Краткое содержание:

Java I/O;
что такое сериализация?;
Java API для работы с JSON;
GSON;
Google ProtoBuf.

Домашние задания

1 Обработчик json-ов

Цель: Научиться обрабатывать json, научиться работать с файлами

Некая система:
- принимает входящий json файл;
- обрабатывает данные из файла;
- формирует ответный файл.

Надо реализовать недостающий функционал
Более подробно смотрите в примерах к вебинару.

---

2 **NIO. Логирование**

Цели занятия:

познакомиться с методами логирования в Java;
познакомиться с NIO.

Краткое содержание:

организация логирования в Java: logback, sl4j;
NIO, файловые операции.

---

3 **JDBC. ДЗ**

Цели занятия:

познакомиться с транзакцией в реляционной СУБД и jdbc.

Краткое содержание:

ACID;
транзакции в СУБД;
JDBC, роль JDBC в стеке технологий;
база данных в Docker;
Connection Pool, HikariCP;
Testcontainers;
ORM Pattern “Executor”.

Домашние задания

1 Самодельный ORM

Цель: Научиться работать с jdbc.
На практике освоить многоуровневую архитектуру приложения.

Работа должна использовать базу данных в docker-контейнере .

В модуле homework реализуйте классы:
- EntityClassMetaData
- EntitySQLMetaData
- DataTemplateJdbc

Метод main в классе HomeWork должен работать без ошибок.

4	**Общие вопросы работы с СУБД, архитектура РСУБД**	**Цели занятия:** оптимизировать SQL-запросы **Краткое содержание:** - архитектура РСУБД - нормализация - планы запросов - статистика - типы JOIN (hash, loop, merge) - индексы - уровни изоляции - оптимизация запросов, основные проблемы sql-запросов
5	**Hibernate**	**Цели занятия:** познакомиться с Hibernate. **Краткое содержание:** Hibernate, место Hibernate в современном стеке технологий; конфигурирование Hibernate в коде и в XML; Java persistency query language; DBSevice pattern.

6 **JPQL. ДЗ**

Цели занятия:

познакомиться с Connection Pool;
проанализировать методы конструирования запросов в Hibernate.

Краткое содержание:

что такое Connection Pool и для чего он нужен?;
изучение HikariCP;
Hibernate: Entity, Fetch, JPQL, SQL.

Домашние задания

1 Использование Hibernate

Цель: На практике освоить основы Hibernate. Понять как аннотации-hibernate влияют на формирование sql-запросов.

Работа должна использовать базу данных в docker-контейнере .

За основу возьмите пример из вебинара про JPQL (class DbServiceDemo).
Добавьте в Client поля:
адрес (OneToOne)
class Address {
private String street;
}
и телефон (OneToMany)
class Phone {
private String number;
}

Разметьте классы таким образом, чтобы при сохранении/чтении объека Client каскадно сохранялись/читались вложенные объекты.

ВАЖНО.
1) Hibernate должен создать только три таблицы: для телефонов, адресов и клиентов.
2) При сохранении нового объекта не должно быть update-ов.
Посмотрите в логи и проверьте, что эти два требования выполняются.

7 **Типы ссылок. Кэширование. ДЗ**

Цели занятия:

объяснить, какие в java есть виды ссылок и для чего они нужны;
оценить, как устроены кэши;
познакомиться с "промышленным" кэшом Ehcache.

Краткое содержание:

виды ссылок в Java;
пример самодельного кэша;
Ehcache.

Домашние задания

1 Свой cache engine

Цель: Научится применять WeakHashMap, понять базовый принцип организации кеширования.

Закончите реализацию MyCache из вебинара. Используйте WeakHashMap для хранения значений.

Добавьте кэширование в DBService из задания про Hibernate ORM или "Самодельный ORM".
Для простоты скопируйте нужные классы в это ДЗ.

Убедитесь, что ваш кэш действительно работает быстрее СУБД и сбрасывается при недостатке памяти.

8 **No SQL**

Цели занятия:

познакомиться с noSQL базами данных;
объяснить отличия SQL от noSQL, когда и что следует использовать;
познакомиться с MongoDB.

Краткое содержание:

SQL базы данных;
noSQL базы данных;
SQL vs noSQL;
Redis;
Cassandra;
Neo4J;
MongoDB;
MongoDB java, реактивное программирование в Java.

9 **Web сервер. ДЗ**

Цели занятия:

объяснить на примере Jetty принципы работы Web-сервера и servlet API

Краткое содержание:

встроенный веб сервер;
сервлеты: servlet API, жизненный цикл сервлета;
Jetty: устройство, работа, подключение сервлетов.

Домашние задания

1 Веб сервер

Цель: Научиться создавать серверный и пользовательский http-интерфейсы.
Научиться встраивать web-сервер в уже готовое приложение.

Встроить веб-сервер в приложение из ДЗ про Hibernate ORM (или в пример из вебинара встроить ДЗ про Hibernate :)).
Сделать стартовую страницу, на которой админ должен аутентифицироваться.
Сделать админскую страницу для работы с клиентами.
На этой странице должны быть доступны следующие функции:
- создать клиента
- получить список клиентов

4 Spring

1 Dependency injection. ДЗ

Цели занятия:

объяснить фундаментальные основы Inversion of Control (IoC) и Dependency Injection (DI)

Краткое содержание:

понятия Inversion of Control (IoC) и Dependency Injection (DI);
что такое Spring-а;
что такое контекст Spring-а;
несколько способов его конфигурирования контекста.

Домашние задания

1 Домашнее задание (Собственный IoC контейнер)

Цель: В процессе создания своего контекста понять как работает основная часть Spring framework.

Обязательная часть:

- Скачать заготовку приложения тренажера таблицы умножения из репозитория с примерами
- В классе AppComponentsContainerImpl реализовать обработку, полученной в конструкторе конфигурации, основываясь на разметке аннотациями из пакета appcontainer.
Так же необходимо реализовать методы getAppComponent
- В итоге должно получиться работающее приложение. Менять можно только класс AppComponentsContainerImpl

Дополнительное задание (можно не делать):
- Разделить AppConfig на несколько классов и распределить по ним создание компонентов. В AppComponentsContainerImpl добавить конструктор, который обрабатывает несколько классов-конфигураций

Дополнительное задание (можно не делать):
- В AppComponentsContainerImpl добавить конструктор, который принимает на вход имя пакета, и обрабатывает все имеющиеся там классы-конфигурации (см. зависимости в pom.xml)

2 **Spring Boot. Spring MVC**

Цели занятия:

познакомиться со Spring Boot;
объяснить, что такое Spring Boot и как им пользоваться;
объяснить Spring MVC.

Краткое содержание:

посмотрим, что такое Spring Boot и Spring MVC;
на примере посмотрим применение Thymeleaf.
Стартеры Spring Boot более подробно будут рассмотрены на следующем вебинаре

---

3 **Asynchronous Web applications**

Цели занятия:

познакомиться с устройством стартеров Spring Boot.
объяснить как можно сделать асинхронный web-сервис на java;

Краткое содержание:

стартеры Spring Boot.
AJAX;
Long polling;
Websockets;

---

4 **Spring Data Jdbc. ДЗ**

Цели занятия:

познакомитться с фреймворком Spring Data Jdbc

Краткое содержание:

чем Spring Data Jdbc отличается от Spring Data, MyBatis и т.д.
основные приемы использования Spring Data Jdbc

Домашние задания

1 Веб-приложение на Spring Boot

Цель: Нучиться создавать CRUD-приложения на Spring Boot

- Взять за основу ДЗ к вебинару Занятие «Web сервер. ДЗ», но без страница логина.
- Вместо Jetty использовать Spring Boot
- Работу с базой данных реализовать на Spring Data Jdbc
- В качестве движка шаблонов использовать Thymeleaf
Если Thymeleaf не нравится, используйте чистый HTML и JavaScript

Авторизацию и аутентификацию делать не надо.

5 Многопоточность

1 Thread

Цели занятия:

познакомиться с основными принципами многопоточности;
объяснить как управлять потоками в Java.

Краткое содержание:

многопоточность;
класс и объект Thread;
создание многопоточного приложения;
проблемы многопоточного доступа.

---

2 JMM

Цели занятия:

познакомиться с основными проблемами многопоточности;
объяснить зачем придумали JMM;
рассмотреть основные положения JMM.

Краткое содержание:

основная проблема многопоточности;
«железные» оптимизации;
«программные» оптимизации;
JMM;
Volatile;
Happens before;
Lock-free алгоритмы, CAS.

---

3 **Executors. ДЗ**

Цели занятия:

познакомиться с пулами потоков в Java.

Краткое содержание:

поддержка многопоточности в стандартной библиотеке;
Workers. Executors;
Fork/Join.

Домашние задания

1 Последовательность чисел

Цель: Освоить базовые механизмы синхронизации.

Два потока печатают числа от 1 до 10, потом от 10 до 1.
Надо сделать так, чтобы числа чередовались, т.е. получился такой вывод:
Поток 1:1 2 3 4 5 6 7 8 9 10 9 8 7 6 5 4 3 2 1 2 3 4....
Поток 2: 1 2 3 4 5 6 7 8 9 10 9 8 7 6 5 4 3 2 1 2 3....

Всегда должен начинать Поток 1.

4 **Потокобезопасные коллекции. ДЗ**

Цели занятия:

познакомиться с потокобезопасными контейнерами;
познакомиться с паттерном - "система обмена сообщениями".

Краткое содержание:

потокобезопасные контейнеры;
применение очередей для взаимодействия потоков.

Домашние задания

1 MessageSystem

Цель: На практике освоить архитектурный подход "Система сообщений".

Добавить систему обмена сообщениями в ДЗ про веб сервер с IoC контейнером
(тут проще использовать Spring Boot).
Пересылать сообщения из вебсокета в DBService и обратно.

Как работать с вебсокетами смотрите пример из вебинара «Asynchronous Web applications».

MessageSystem добавьте как модуль, по примеру из вебинара.

---

5 **Многопроцессные приложения. ДЗ**

Цели занятия:

проанализировать сетевое взаимодействие в java.
объяснить принципы работы "клиент-серверного" приложения в Java

Краткое содержание:

Сокеты;
RMI;
разбор примера многопроцессного приложения;
gRPC.

Домашние задания

1 gRPC клиент-серверное приложение или "Убить босса"

Цель: Научиться разрабатывать сетевые приложения с gRPC.

Разработать клиент-серверное приложение с применением технологии gRPC.

1) Серверная часть.
сервер по запросу клиента генерирует последовательность чисел.
запрос от клиента содержит начальное значение (firstValue) и конечное(lastValue).
Раз в две секунды сервер генерирует новое значение и "стримит" его клиенту:
firstValue + 1
firstValue + 2
...
lastValue

2) Клиентская часть.
клиент отправдяет запрос серверу для получения последовательности чисел от 0 до 30.
клиент запускает цикл от 0 до 50.
раз в секунду выводит в консоль число (currentValue) по такой формуле:
currentValue = [currentValue] + [ПОСЛЕДНЕЕ число от сервера] + 1
начальное значение: currentValue = 0
Число, полученное от сервера должно учитываться только один раз.
Обратите внимание, сервер может вернуть несколько чисел, надо взять именно ПОСЛЕДНЕЕ.

Должно получиться примерно так:
currentValue:1
число от сервера:2
currentValue:4 <--- число от сервера учитываем только один раз
currentValue:5 <--- тут число от сервера уже не учитывается.
число от сервера:3
currentValue:9
currentValue:10
new value:4
currentValue:15
currentValue:16

Для коммуникации используйте gRPC.
Клиент и сервер не обязательно разделять по модулям.
Можно сделать один модуль с двумя main-классами для клиента и сервера.

Пример лога работы клиента (new value - это значение полученное от сервера)

```
21:44:04.782 [main] INFO
ru.otus.numbers.client.NumbersClient - numbers
Client is starting...
21:44:04.932 [main] INFO
ru.otus.numbers.client.NumbersClient -
currentValue:1
21:44:05.140 [grpc-default-executor-0] INFO
r.o.n.client.ClientStreamObserver - new value:2
21:44:05.933 [main] INFO
ru.otus.numbers.client.NumbersClient -
currentValue:4
21:44:06.933 [main] INFO
ru.otus.numbers.client.NumbersClient -
currentValue:5
21:44:07.113 [grpc-default-executor-0] INFO
r.o.n.client.ClientStreamObserver - new value:3
21:44:07.934 [main] INFO
ru.otus.numbers.client.NumbersClient -
currentValue:9
21:44:08.934 [main] INFO
ru.otus.numbers.client.NumbersClient -
currentValue:10
21:44:09.112 [grpc-default-executor-0] INFO
r.o.n.client.ClientStreamObserver - new value:4
21:44:09.935 [main] INFO
ru.otus.numbers.client.NumbersClient -
currentValue:15
21:44:10.935 [main] INFO
ru.otus.numbers.client.NumbersClient -
currentValue:16
21:44:11.113 [grpc-default-executor-0] INFO
r.o.n.client.ClientStreamObserver - new value:5
21:44:11.935 [main] INFO
ru.otus.numbers.client.NumbersClient -
currentValue:22
21:44:12.936 [main] INFO
ru.otus.numbers.client.NumbersClient -
currentValue:23
21:44:13.113 [grpc-default-executor-0] INFO
r.o.n.client.ClientStreamObserver - new value:6
21:44:13.936 [main] INFO
ru.otus.numbers.client.NumbersClient -
currentValue:30
21:44:14.937 [main] INFO
ru.otus.numbers.client.NumbersClient -
currentValue:31
21:44:15.114 [grpc-default-executor-0] INFO
r.o.n.client.ClientStreamObserver - new value:7
21:44:15.938 [main] INFO
ru.otus.numbers.client.NumbersClient -
```

currentValue:39
21:44:16.938 [main] INFO
ru.otus.numbers.client.NumbersClient -
currentValue:40
21:44:17.113 [grpc-default-executor-0] INFO
r.o.n.client.ClientStreamObserver - new value:8
21:44:17.939 [main] INFO
ru.otus.numbers.client.NumbersClient -
currentValue:49
21:44:18.939 [main] INFO
ru.otus.numbers.client.NumbersClient -
currentValue:50
21:44:19.113 [grpc-default-executor-0] INFO
r.o.n.client.ClientStreamObserver - new value:9
21:44:19.940 [main] INFO
ru.otus.numbers.client.NumbersClient -
currentValue:60
21:44:20.940 [main] INFO
ru.otus.numbers.client.NumbersClient -
currentValue:61
21:44:21.114 [grpc-default-executor-0] INFO
r.o.n.client.ClientStreamObserver - new value:10
21:44:21.119 [grpc-default-executor-0] INFO
r.o.n.client.ClientStreamObserver - request
completed

---

6 **Применение RabbitMQ**

Цели занятия:

познакомиться с очередями на примере RabbitMQ и SpringBoot

Краткое содержание:

Очереди;
Виды Exchange;
Работа с RabbitMQ из SpringBoot

---

7 **NIO**

Цели занятия:

рассмотреть основы сетевых возможностей NIO.

Краткое содержание:

NIO для сокетов.

---

8 **Netty**

Цели занятия:

рассмотреть основные принципы работы Netty.

Краткое содержание:

архитектура Netty;
примеры применения.

---

9 **Реактивное программирование. Spring Webflux. ДЗ**

Цели занятия:

познакомится с реактивным программированием в Java

Краткое содержание:

что такое реактивное программирование и для чего оно нужно;
Spring webflux

Домашние задания

1 Реактивное Веб-приложение на Spring Boot

Цель: Нучиться создавать реактивное CRUD-приложения на Spring Boot

Взять за основу ДЗ "Веб-приложение на Spring Boot".
Использовать spring webflux и mongo качестве базы данных

---

10 **Применение Kafka**

Цели занятия:

познакомиться с платформой Kafka и возможностями применения в приложениях на java.

Краткое содержание:

что такое Kafka и для чего это надо.
как пользовать Kafka в программах на java

6 Проектная работа

1 **Основы CI/CD**

Цели занятия:

познакомится с основами развертывания приложени

Краткое содержание:

Проблемы "ручной" сборки приложения и деплоя
что такое CI/CD, что это дает
популярные инструменты: Jenkins, TeamCity, GitLab
понятие CI/CD pipeline
CI/CD pipeline на примере GitLab

Домашние задания

1 Проектная работа

Цель: выбрать тему проекта;
закрепить тему в чат с преподавателем.

Заключительный месяц курса посвящен проектной работе. Свой проект это то, что интересно писать студенту. То, что можно создать на основе знаний, полученных на курсе.
При этом не обязательно закончить его за месяц. В процессе написания по проекту можно получить консультации преподавателей.

Проект должен стать примером кода, который можно показывать потенциальным работодателям.

Примеры тем проекта:
- web сервер (разберите протокол)
- socket сервер на NIO (как netty)
- свой ORM
- распределенный кэш
- кэш для hibernate

2 **Знакомство с Kubernetes**

Цели занятия:

узнать, что такое Kubernetes и чем он полезен java-разработчику

Краткое содержание:

что такое Kubernetes
основные части (абстракции)
требования к приложению
демонстрация Minikube
демонстрация managed Kubernetes

---

3 **Защита проектных работ**

Цели занятия:

защитить проект и получить рекомендации экспертов.

Краткое содержание:

презентация проектов перед комиссией;
вопросы и комментарии по проектам.