OTUS

Специальная цена

Java Developer. Professional

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

Длительность курса: 160 академических часов

1 Язык и платформа Java

1	**Подготовка к курсу. ДЗ**	познакомиться с программой курса; изучить основные инструменты. Домашние задания 1 Проект gradle с модульной структурой Цель: нучиться 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**	углубить знания о gradle; познакомиться с текущей ситуацией в мире java.
3	**QA и тестирование**	познакомиться с junit и mockito; объяснить на примере, что такое "тестируемое приложение"
4	**Контейнеры и алгоритмы. ДЗ**	познакомиться с Generic-ами в Java и со стандартными коллекциями. Домашние задания 1 DIY ArrayList Цель: изучить как устроена стандартная коллекция ArrayList. Попрактиковаться в создании своей коллекции. Написать свою реализацию ArrayList на основе массива. class DIYarrayList<T> implements List<T>{...} Проверить, что на ней работают методы из java.util.Collections: Collections.addAll(Collection<? super T> c, T... elements) Collections.static <T> void copy(List<? super T> dest, List<? extends T> src) Collections.static <T> void sort(List<T> list, Comparator<? super T> c) 1) Проверяйте на коллекциях с 20 и больше элементами. 2) DIYarrayList должен имплементировать ТОЛЬКО ОДИН интерфейс - List. 3) Если метод не имплементирован, то он должен выбрасывать исключение UnsupportedOperationException.
5	**Инструменты для преобразования контейнеров, unsafe, jmh**	объяснить на примере принципы создания коллекций; познакомиться с пакетом unsafe, утилитой JMH и популярными библиотеками коллекций.
6	**Аннотации. ДЗ**	познакомиться с механизмом Reflection; объяснить, что такое Аннотации и как их можно сделать. 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 **Lombok**

познакомиться с технологиями annotation Processor и возможностями Lombok.

---

8 **Сборщик мусора. ДЗ**

познакомиться со сборщиком мусора в Java.

Домашние задания

1 Сравнение разных сборщиков мусора

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

Написать приложение, которое следит за сборками мусора и пишет в лог количество сборок каждого типа
(young, old) и время которое ушло на сборки в минуту.

Добиться OutOfMemory в этом приложении через медленное подтекание по памяти
(например добавлять элементы в List и удалять только половину).

Настроить приложение (можно добавлять Thread.sleep(...)) так чтобы оно падало
с OOM примерно через 5 минут после начала работы.

Собрать статистику (количество сборок, время на сборки) по разным GC.

!!! Сделать выводы !!!
ЭТО САМАЯ ВАЖНАЯ ЧАСТЬ РАБОТЫ:
Какой gc лучше и почему?

Выводы оформить в файле Conclusions.md в корне папки проекта.

Результаты измерений сведите в таблицу.

Попробуйте провести этот эксперимент на небольшом хипе порядка 256Мб, и на максимально возможном, который у вас может быть.

---

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

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

---

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

познакомиться с принципами работы виртуальной машины Java, ClassLoader-ами и 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	**Java 8**	изучить введение в функциональное программирование (ФП); познакомиться с возможностями ФП, которые появились в Java 8.

2 Проектирование

1 **Концепты проектирования ООП. ДЗ**

объяснить принципы SOLID и общие критерии идеальной архитектуры.

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**

объяснить поведенческие паттерны проектирования

---

3 **Creational patterns**

изучить "создающие" паттерны проектирования

---

4 **Structural patterns. ДЗ**

изучить структурные паттерны проектирования

Домашние задания

1 Обработчик сообщений

Цель: Применить на практике шаблоны проектирования.

Реализовать todo из модуля homework.

3 Работа с окружением

1 **Сериализация. ДЗ**

познакомиться с функционалом сериализации объектов.

Домашние задания

1 Свой json object writer

Цель: Научиться сериализовывать объект в json, попрактиковаться в разборе структуры объекта.

Напишите свой json object writer (object to JSON string) аналогичный gson на основе javax.json.

Gson это делает так:
```
Gson gson = new Gson();
AnyObject obj = new AnyObject(22, "test", 10);
String json = gson.toJson(obj);
```

Сделайте так:
```
MyGson myGson = new MyGson();
AnyObject obj = new AnyObject(22, "test", 10);
String myJson = myGson.toJson(obj);
```

Должно получиться:
```
AnyObject obj2 = gson.fromJson(myJson, AnyObject.class);
System.out.println(obj.equals(obj2));
```

Поддержите:
- примитивные типы и Wrapper-ы (Integer, Float и т.д.)
- строки
- массивы примитивных типов
- коллекции (interface Collection)

Не забываться, что obj может быть null :)

---

2 **NIO. Логирование**

познакомиться с методами логирования в Java;
познакомиться с NIO.

---

3 **JDBC. ДЗ**

познакомиться с транзакцией в реляционной СУБД и jdbc.

Домашние задания

1 Самодельный ORM

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

Работа должна использовать базу данных H2.

Создайте в базе таблицу User с полями:

- id bigint(20) NOT NULL auto_increment
- name varchar(255)
- age int(3)

Создайте свою аннотацию @Id

Создайте класс User (с полями, которые соответствуют таблице, поле id отметьте аннотацией).

Реализуйте интерфейс JdbcMapper<T>, который умеет работать с классами, в которых есть поле с аннотацией @Id.
JdbcMapper<T> должен сохранять объект в базу и читать объект из базы.
Для этого надо реализовать оставшиеся интерфейсы из пакета mapper.
Таким обзазом, получится надстройка над DbExecutor<T>, которая по заданному классу умеет генерировать sql-запросы.
А DbExecutor<T> должен выполнять сгенерированные запросы.

Имя таблицы должно соответствовать имени класса, а поля класса - это колонки в таблице.

Проверьте его работу на классе User.

За основу возьмите класс HomeWork.

Создайте еще одну таблицу Account:
• no bigint(20) NOT NULL auto_increment
• type varchar(255)
• rest number

Создайте для этой таблицы класс Account и проверьте работу JdbcMapper на этом классе.

---

4 **Общие вопросы работы с СУБД, myBatis**

рассмотреть CAP-теорему;
рассмотреть методы организации блокировок;
познакомиться с MyBatis.

---

5 **Hibernate**

познакомиться с Hibernate.

---

6 **JPQL. ДЗ**

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

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 H2.

Возьмите за основу предыдущее ДЗ (Самодельный ORM), используйте предложенный на вебинаре api (пакет api) и реализуйте функционал сохранения и чтения объекта

User через Hibernate.
(Рефлексия больше не нужна)
Конфигурация Hibernate должна быть вынесена в файл.

Добавьте в User поля:
адрес (OneToOne)
class AddressDataSet {
private String street;
}
и телефон (OneToMany)
class PhoneDataSet {
private String number;
}

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

ВАЖНО.
1) Hibernate должен создать только три таблицы: для телефонов, адресов и пользователей.
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**

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

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

---

9 **Web сервер. ДЗ**

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

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 пользователями.
На этой странице должны быть доступны следующие функции:
- создать пользователя
- получить список пользователей

4 Spring

1	**Dependency injection. ДЗ**	объяснить фундаментальные основы Inversion of Control (IoC) и Dependency Injection (DI) 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**	познакомиться со Spring Boot; объяснить, что такое Spring Boot и как им пользоваться; объяснить Spring MVC.
3	**Asynchronous Web applications**	познакомиться с устройством стартеров Spring Boot. объяснить как можно сделать асинхронный web-сервис на java;
4	**Spring Data Jdbc. ДЗ**	познакомитться с фреймворком 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**

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

---

2 **JMM**

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

---

3 **Executors. ДЗ**

познакомиться с пулами потоков в Java.

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 **Message System. ДЗ**

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

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**

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

Домашние задания

1 MessageServer

		Цель: Научиться разрабатывать сетевые приложения. Сервер из предыдущего ДЗ про MessageSystem разделить на три приложения: • MessageServer • Frontend • DBServer Запускать Frontend и DBServer из MessageServer. Сделать MessageServer сокет-сервером, Frontend и DBServer клиентами. Пересылать сообщения с Frontend на DBService через MessageServer. Запустить приложение с двумя серверами фронтенд и двумя серверами баз данных на разных портах. Если у вас запуск веб приложения в контейнере, то MessageServer может копировать root.war в контейнеры при старте
6	**NIO**	изучить основы сетевых возможностей NIO.
7	**Netty**	изучить основные принципы работы Netty.

6 Проектная работа

1	**Выбор темы и организация проектной работы**	выбрать и обсудить тему проектной работы; спланировать работу над проектом; ознакомиться с регламентом работы над проектом. 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	**Консультация по проектам и домашним заданиям**	получить ответы на вопросы по проекту, ДЗ и по курсу.
3	**Защита проектных работ**	защитить проект и получить рекомендации экспертов.