**01. Вводный модуль**

Полезные ссылки для работы с IntelliJ IDEA:

* [Горячие клавиши](https://resources.jetbrains.com/storage/products/intellij-idea/docs/IntelliJIDEA_ReferenceCard.pdf) IntelliJ IDEA
* [Основы работы](https://www.kv.by/blog/users/fetisovvs/1049285-osnovy-raboty-s-intellij-idea-interfeys-programmy) с IntelliJ IDEA

**02. Синтаксис языка**

**03. Объекты и классы**

**04. Числа, строки и даты**

**Дополнительный материал**

* Примеры работы методов найдите в [репозитории](https://github.com/skillbox-java/M4) (проект snippets, директория M4\_1)
* Ссылки на классы и их методы смотрите в официальной документации к JDK:

[String](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html)  
[Integer](https://docs.oracle.com/javase/8/docs/api/java/lang/Integer.html)  
[Character](https://docs.oracle.com/javase/8/docs/api/java/lang/Character.html)

**Дополнительный материал**

* [Объяснение, как устроены числа double и float в любом языке программирования](https://habr.com/ru/post/337260/)
* [Особенности float и double в Java](https://habr.com/ru/post/219595/)
* [Калькулятор для экспериментов с битами](https://www.h-schmidt.net/FloatConverter/IEEE754.html)

Прочитать [документацию к классу Math](https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html) на официальном сайте Oracle

Примеры работы с методами indexOf(), lastIndexOf(), substring() и trim() вы найдёте в [репозитории](https://github.com/skillbox-java/M4/tree/master/snippets/src/ru/skillbox/M4_4).

**05. Массивы и коллекции**

**06. Наследование и полиморфизм**

**07. Особенности ООП в Java**

**Полезные ссылки:**

* Новшества Java 8 — <https://www.youtube.com/watch?v=6bN1HcRhse4>
* Лямбда-выражения в Java 8 — <https://habr.com/ru/post/224593/>
* Java Generics — <http://www.quizful.net/post/java-generics-tutorial>
* Apache Maven — основы — <https://habr.com/ru/post/77382/>
* Аннотации в Java: обзор синтаксиса и создание собственных — <https://habr.com/ru/post/139736/>
* Lombok — <https://habr.com/ru/post/345520/>

**08. Исключения, отладка, тестирование и логирование**

**Полезные ссылки**

* Исключения в Java — часть 1 — <https://habr.com/ru/company/golovachcourses/blog/223821/>
* Исключения в Java — часть 2 — <https://habr.com/ru/company/golovachcourses/blog/225585/>
* Юнит-тестирования для чайников — <https://habr.com/ru/post/169381/>

Пример файла конфигурации log4j2 — <https://stackoverflow.com/questions/17418494/log4j2-xml-configuration-log-to-file-and-console-with-different-levels>

**09. Работа с файлами и сетью**

**Рекомендации**

* Все варианты подключения библиотеки jsoup в проект [на странице скачивания библиотеки](https://jsoup.org/download)
* Для получения ссылок воспользуйтесь префиксом **abs** при получении атрибута **src** у тега **img**. [Пример из официальной документации](https://jsoup.org/cookbook/extracting-data/working-with-urls).

**Рекомендации**

* По умолчанию Jsoup читает 2 Мб данных с запрашиваемой страницы. Чтобы снять это ограничение, необходимо добавить вызов метода maxBodySize(0), устанавливающий максимальное значение получаемых данных:

Document doc = Jsoup.connect(URL).**maxBodySize(0)**.get();

значение 0 означает, что нет ограничений на принимаемый объём данных.

* При изучении кода страницы удобно использовать консоль разработчика в браузере. Для этого нажмите F12, перейдите во вкладку Elements и найдите тег <div id="metrodata">. В нём содержатся таблицы с линиями, станциями и пересадками. Обращайте внимание на классы, напишите селекторы на основе найденных классов. Посмотрите, как получать элементы по селекторам в [документации JSoup](https://jsoup.org/cookbook/extracting-data/selector-syntax).
* Для более быстрого поиска нужных селекторов используйте онлайн сервис jsoup <https://try.jsoup.org/>

**10. Работа с MySQL в Java**

**Полезные ссылки**

* Официальная документация по последней версии [Hibernate](https://hibernate.org/orm/documentation/5.4/).
* Примеры работы с [Hibernate](https://www.boraji.com/category/hibernate-orm).
* Hibernate Criteria Queries (официальная документация [на сайте JBoss](https://docs.jboss.org/hibernate/entitymanager/3.5/reference/en/html/querycriteria.html)).

Для получения значения месяца из даты в числовом значении используйте функцию

MONTH( date\_value )

<https://oracleplsql.ru/function-sql-server-month.html>

**Таблица должна содержать следующие поля:**

* student\_id
* course\_id

В таблице нет отдельной колонки с привычным идентификатором записи. В роли идентификатора здесь выступает пара student\_id и course\_id.  Это значит, что пара значений student\_id и course\_id уникальна для каждой записи.

Связка student\_id и course\_id называется «составной ключ» (Composite key).

Один из вариантов создания составного ключа — написать отдельный класс, в котором поля будут содержать значения полей, входящих в составной ключ.

На примере класса Subscription:

public class Key implements Serializable {

@Column(name = "student\_id")

    private int studentId;

@Column(name = "course\_id")

    private int courseId;

 //setters, getters, equals(), hashcode()

}

Класс составного ключа обязан:

* быть public;
* иметь публичный конструктор по умолчанию;
* реализовывать собственные equals(), hashCode(), публичные геттеры и сеттеры;
* имплементировать Serializable.

 Используется класс-ключ в @Entity следующим образом:

@Entity

@Table(name = "Subscriptions")

public class Subscription {

  @EmbeddedId

  private Key id;

  @Column(name = "student\_id", insertable = false, updatable = false)

  private int studentId;

  @Column(name = "course\_id", insertable = false, updatable = false)

  private int courseId;

...

Аннотация @EmbeddedId говорит, что данный параметр является составным ключом.

Если поля ключа использовать и в основном классе @Entity, то необходимо запретить использовать поля для вставки и обновления данных дополнительными параметрами в аннотации @Column.

Для получения объекта из базы данных по составному ключу необходимо в метод get() объекта Session передавать экземпляр составного ключа, например:

Subscription subscription = session.get(Subscription.class, new PK(studentId, courseId));

аналогично и для создания новой записи.

Дополнительные примеры найдёте в [статье](https://easyjava.ru/data/hibernate/pervichnye-klyuchi-v-hibernate/).

**11. Многопоточность**

**Что нужно сделать**

* Согласно [инструкции](https://www.notion.so/sendel/IDEA-Maven-671db52e43a6483cb2fbfb3736cdc7cf) включить поддержку сборщика maven в свой проект.
* Внедрить библиотеку Imgscalr в проект.
* Использовать метод для уменьшения изображения.

public static BufferedImage resize(BufferedImage src, int targetWidth,

            int targetHeight, BufferedImageOp... ops)

* Удостоверьтесь, что ваша программа работает в многопоточном режиме. Для этого можете использовать утилиту [visualVM](https://visualvm.github.io/).

**12. Разработка веб-приложений**

Полезные ссылки:

Building an Application with [Spring Boot](https://spring.io/guides/gs/spring-boot/).

Полезные ссылки:

[Паттерны для новичков: MVC vs MVP vs MVVM](https://habr.com/ru/post/215605/).

Полезные ссылки:

* 10 Best Practices for Better RESTful API. (статья в приложенном файле)
* [REST API Best Practices](https://habr.com/ru/post/351890/).
* Принципы построения [REST JSON API](https://habr.com/ru/post/447322/).
* [Коды состояний HTTP](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%BA%D0%BE%D0%B4%D0%BE%D0%B2_%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D1%8F_HTTP).

**13. Нереляционные (NoSQL) базы данных**

**14. Производительность и оптимизация**

**15. Распределённые хранилища и вычисления**

**16. Заключительный модуль**

**17. Бонус-модуль. Паттерны проектирования**

**18. Бонус-модуль. Алгоритмы**

**19. Бонус-модуль. Структуры данных**

**20. Бонус-модуль. Протокол HTTP**