## ДЗ-1 Настройка окружения

#### Практическое задание

1. Настроить переменные окружения и среду разработки
2. Создать свой репозиторий в github.com
3. Создать и закоммитить проект для intellij idea под управлением maven.
4. Добавить в pom.xml зависимости на jetty и freemarker:

<dependency>  
 <groupId>org.eclipse.jetty</groupId>  
 <artifactId>jetty-server</artifactId>  
 <version>9.1.0.M0</version>          
</dependency>    
<dependency>  
 <groupId>org.freemarker</groupId>  
 <artifactId>freemarker</artifactId>  
 <version>2.3.20</version>  
</dependency>

***Список теоретических вопросов***

* История и особенности языка Java.
* Платформа Java. Java Editions. GC, JVM, JDK, JRE. ByteCode.
* Правила наименований в Java
* UML диаграммы наследования и реализации интерфейсов
* class Object и class Class<?>. Объект класса Class.
* Простые типы в Java. Классы обертки простых типов.
* Generic programming в Java.
* Коллекции в Java. Iterator, Iterable.

## ДЗ-2 Авторизация пользователя

#### Практическое задание

1. При помощи Jetty запустить свой web сервер (при обращении на <http://localhost:8080/> возвращать “Hello Server!”)
2. Написать class PageGenerator на основе freemarker. Написать страницу, которая перезапрашивает себя раз в секунду с сервера (<https://github.com/vitaly-chibrikov/tp_java_2013_09/tree/master/L01-2>).
3. Подключить библиотеку сервлетов для Jetty

<groupId>org.eclipse.jetty</groupId>

<artifactId>jetty-webapp</artifactId>

<version>9.1.0.M0</version>

1. Написать class Frontend extends HttpServlet и обрабатывать в нем все запросы (context.addServlet(new ServletHolder(frontend), "/\*"); Переписать “Hello server” на сервлетах.
2. Каждому пользователю присваивать уникальный идентификатор. Идентификатор хранить в HttpSession.
3. Подключить в Jetty выдачу статики. Создать страницу index.html с картинкой и возвращать ее при открытии страницы.
4. Сделать страницу с «авторизацией» пользователя. Пользователя нужно спросить имя и пароль и проверить их в объекте Frontend (можно захардкодить). Если пользователь узнан – вернуть ему его id.

В результате выполнение задания у вас должен быть веб сервер, на localhost:8080/, который при открытии index.html возвращает картинку и форму с запросом логина и пароля. После ввода правильного логина и пароля редиректит на страницу с таймером, на которой также выведен id сесии пользователя.

#### Список теоретических вопросов

* Инструменты разработки и библиотеки (git, maven, jetty, freemarker)

## ДЗ-3 Регистрация в базе данных.

#### Практическое задание

1. Расширить приложение так чтобы пользователь мог с начальной страницы перейти на страницу регистрации. На этой странице должна быть форма, отправка которой инициирует на сервере создание записи в БД для нового пользователя.
2. Логин пользователя должен проходить через чтение данных пользователя из БД.
3. Добавить файл с описанием схемы базы в проект.
4. На стороне БД создать таблицу по описанной схеме.
5. Код приложения должен содержать DAO и DataSet для таблицы авторизации. DAO может быть написан на "чистом" JDBC или с помощью Hibernate.

#### Список теоретических вопросов

* JDBC API.
* class Connection.
* Statement.
* ResultSet.
* DataSet.
* DAO.
* Annotation.
* Object Relational Mapping.

## ДЗ-4 Тестирование

#### Практическое задание

Подготовка к написанию тестов. Все тесты должны храниться в том же package, что и тестируемый код, но не в папке src, а в папке - test.

1. Написать тест на создание страниц: приветствия, успешной и неуспешной авторизации.
2. Написать тесты на роутинг, используя Mock, Fake, Stub или Spy дублёров.
3. Написать тест на регистрацию пользователя: удачный и неудачный вариант.
4. Написать тест на авторизацию пользователя: удачный и неудачный вариант.
5. Сделать единый Runner тестов с анализом покрытия кода тестами.
6. Написать тест на авторизацию пользователя, используя Selenium. \*
7. Написать нагрузочный тест на авторизацию. \*\*

В результате выполнения задания, code coverage в модулях frontend’а и базы данных должен быть выше 80%.

Под тестом на роутинг понимается тест следующего вида: в функцию передаётся путь вида “/“,”/auth” или ”/reg”. Функция проверят настройки роутинга для вашего приложения и дает ответ, есть ли такой маршрут или нет. В будущем тест нужно будет развить, проверяя какое именно действие было произведено для того или иного пути.

#### Список теоретических вопросов

* Виды тестирования: интеграционное, функциональное, юнит.
* Нагрузочное тестирование: jMeter, Yandex.Танки.
* Code coverage.
* Мутационное тестирование.
* Test doubles (Mock, Fake, Stub, Spy).
* jUnit. Tricks and tips: ожидание исключения, временные файлы, таймаут.
* Continuous Integration.
* Selenium.

## ДЗ-5 Многопоточность

#### Практическое задание

1. Изменить class Frontend, так чтобы его объект можно было запустить в отдельном потоке (implements Runnable)
2. Добавить в класс переменную int handleCount = 0.
3. Запустить Frontend в отдельном потоке. В методе run() писать значение handleCount в лог каждые 5 секунд.
4. В методе handle() кроме формирования страницы, увеличивать значение handleCount.

#### Тесты

#### Список теоретических вопросов

* Processes and Threads.
* Interface Runnable. class Thread. Методы start() и run().
* Методы класса Thread: sleep(), interrupt(), join().
* Ключевые слова final, volatile и synchronized.
* Atomic variables.
* Проблемы многопоточного доступа к данным. Race condition.
* Методы класса Object: wait(), notify() и notifyAll().

## ДЗ-6 Система обмена сообщениями

#### Практическое задание

1. Вынести работу с БД в отдельный сервис (DBService) и запустить его в отдельном потоке.
2. Производить аутентификацию пользователя с помощью сообщений, через MessageSystem.
3. Пока пользователь проходит авторизацию – выдавать сообщение «Ждите авторизации».
4. Добавить AddressService, запусить второй DBSerivce.

#### Тесты

#### Список теоретических вопросов

* Плюсы и минусы многопоточных приложений.
* Способы взаимодействия потоков.
* java.util.concurrent
* MessageSystem. Address и Abonent.

## ДЗ-7 Ресурсная система

#### Практическое задание

1. Написать виртуальную файловую систему (VFS).
2. VFS  НЕ должна загружать файлы в память при старте сервера. Задача этой системы скрывать детали работы с файлами.
3. Написать ресурсную систему сервера (ResourceFactory).
4. Все константы, которые использует сервер вынести в ресурсные xml файлы. Написать классы, объекты которых можно восстановить на основе этих ресурсных xml файлов. Код, который использует константу, должен спрашивать у ResourceFactory объект ресурса по пути к файлу.

#### Тесты

#### Список теоретических вопросов

* I/O Streams. Потоки байт и потоки символов.
* Сериализация/десериализация. Interface Serializable.
* Ключевое слово transient.
* Reflection. Роль reflection в сериализации.
* SAX парсер.
* DOM парсер.
* ResourceFactory. Роль ресурсов в игре.

## ДЗ-8

#### Практическое задание

1. Добавить еще один абонент и поток - GameMechanics.
2. Добавить адрес этого абонента в AddressService.
3. Написать абстрактное сообщение MsgToGM и реализацию - MsgStartGameSession.
4. Frontend должен посылать сообщение MsgStartGameSession на GameMechanics в тот момент, когда к серверу присоединилось достаточно пользователей для старта новой игры (не менее двух). Получив это сообщение, игровая механика должна создать игровую сессию для пользователей, Id которых пришли в сообщении.
5. Написать сообщение с механики на Frontend, и пересылать в нем из GameMechanics реплику состояния игровой сессии.
6. Запоминать время старта игровой партии, пересылать из игровой механики текущую длительность игровой партии и показывать ее на странице.

#### Тесты

#### Список теоретических вопросов

* Архитектура игрового сервера.
* Процесс работы игрового сервера, репликация.