| Architecture | SOLID принципы | Подробнее о первом или/и последнем |
| --- | --- | --- |
| Architecture | Патерны ООП. Задача про уток | <http://second-ext.inttrust.ru/Site/itforum.nsf/0/8f5cfcbd13982149c32571fd00322142/$FILE/DesignPatterns-01(V.Panov).pdf>  Соответствие решения SOLID принципам |
| Architecture | REST архитектура. Основные принципы |  |
| Java (Validation) |  | Существующие валидаторы. Где и как применяются. Создание пользовательского валидатора   * **JPA валидация** обладает ограниченной функциональностью, но является хорошим выбором для простейших ограничений в классах сущности, если такие ограничения могут быть отображены на DDL. * **Bean Validation** — гибкий, лаконичный, декларативный, многоразовый и удобный для чтения способ настроить большинство проверок в классах предметной области. В большинстве случаев это лучший выбор, если не нужно запускать валидации внутри транзакций. * **Валидация по контракту** - это bean validation, но для вызовов методов. Используйте ее для входных и выходных параметров метода, например, в контроллерах REST. * **Entity listeners**: хотя они и не так декларативны, как аннотации Bean Validation, они отлично подходят для проверки больших графов объектов или проверок внутри транзакции БД. Например, когда нужно считать данные из БД для принятия решения. У Hibernate есть аналог таких слушателей. * **Transaction listeners** — опасное, но мощное оружие, работающее внутри контекста транзакции. Используйте его, когда в процессе исполнения нужно решить, какие объекты должны быть проверены или когда нужно проверить много разных типов сущностей по одному и тому же алгоритму валидации. |
| Java (Threads) | Механизмы синхронизации. Как обеспечить асинхронный вызов и вернуть при этом результат (Java, Spring) | Spring - @Async. Под капотом Java - CompletableFuture. В отличии от Future, не останавливаем основной поток для получение результата - get(), isDone() в цикле, а подключаем callback. Под капотом ForkJoinPool - принцип (конкурентность + параллелизм)  thenAccept, thenApply, thenCompose |
| Spring (Boot) | Что такое стартер? Какие вы знаете аннотации для настройки автоконфигурации? Как создать собственный стартер, который включает в себя внешние зависимости (jar файлы с бинами)? |  |
| Spring (Framework) | Как выполнить метод асинхронно и транзакционно (при этом чтоб игнорировалась внешняя транзакция)? |  |
| Spring (Framework) | В бин A (singleton) инжектирован бин B (prototype). Какая область видимости бина B? Как сделать так, чтобы внутри бина A при обращении к бину B каждый раз создавался новый экземпляр B? |  |
| Spring (Framework) | Что произойдет, если использовать циклические зависимости между бинами? | Если инжектить ссылки в конструкторе, вызовется исключение *BeanCurrentlyInCreationException.* В случае если инжектить через поля, исключения не будет (ссылки инициируются не бинами, а **прокси)**. Но если будет обращение к ссылке, то получим *StackOverflowException* |
| Spring (Framework) | Dependency Injection - best practices. Как есть способы инжектить ссылки? Какие предпочтительнее? | Конструктор, поля или сеттеры. Конструктор ()   * защищает от циклических ссылок (bad architecture ); * нет привязки к контейнеру (POJO). Можно использовать в тестах без контейнера или переключить DI Framework; * есть возможность создавать абстрактный базовый класс (pojo) и в наследниках использовать ссылки на бины, переданные в конструкторе абстрактного класса; * легче отследить нарушение принципа единственной ответственности. Конструктор с большим количеством зависимостей - хороший индикатор; * защита от NullPointerException |
| Spring (Cloud) | Что такое Spring Cloud? Какие возможности модуль включается в себя? |  |
| Architecture | Теорема CAP | Пример Postgres.   * Репликация Master-Slave — одно из распространенных решений * Синхронизация с Master в асинхронном / синхронном режиме * Система транзакций использует двухфазный коммит для обеспечения consistency * Если возникает partition, не можем взаимодейстовать с системой (в основном случае)   Cистема не может продолжать работу в случае partition, но обеспечивает strong consistency и availability. Это система CA! |
| Architecture | Монолитная и микросервисная архитектура. Преимущества и недостатки. Синхронизация микросервисов - возможные решения. |  |
| Architecture | Реактивное программирование. Патерн реактивная система |  |
| Sping | Что такое bean? Жизненный цикл bean? Как создать bean, какие scope bean бывают? |  |
| Database | ACID |  |
| Database | Уровни изоляции и какие проблемы решают? |  |

Основные принципы SOLID, только ли для классов?  
low coupling high cohesion - что это как работает.   
Dependency Injection, Dependency Inversion связь между DI (патрерн реализация) и IOC (принцип)  
Spring DI. Способы DI (конструктор, методы в Java) что дает спринг и что упрощает?  
bean factory post processor  
Spring Data - основные принципы работы  
Scope Bean - какие бывают  
Spring Boot - что такое и в чем его основная фишка  
нормальные формы баз данных  
уровни изоляции транзакций  
микросервисная архитектура - плюсы минусы Паттерн Saga(https://habr.com/ru/post/427705/)  
Принципы REST  
Java 8 отличия от 7 (Что такое Optional)  
Стримы в java  
parallel в стримах это ForkJoinPool  
когда можно когда нельзя использовать параллельность в стримах  
один spring repository управляет одной entity  
isolation и propagation в transactional

Common  
Какие книги читали, Блох?  
Слышали ли про Domain Driven Design  
Что такое ООП, принципы, что такое полиморфизм, статический полиморфизм  
Какие парадигмы программирования есть: функциональное, императивное. декларативное, ООП  
Основные принципы SOLID, только ли для классов?  
GRASP - паттерны  
Паттерны проектирования  
Coupling Cohesion - что это

Java  
Java как функцоинальный язык. Какие инструменты добавились, какие функциональные интерфейсы есть?  
Java стрим API, сплитератор.  
Есть массив массивов, его нужно умножить на константк и получить одномерный  
Отличия map от flatmap. Сигнатура методов.  
Что такое функция высшего порядка.  
Коллекции, интерфейсы.Map отличия от Collection.  
Различие checked unchecked исключения. Иерархия интерфейсов исключений.  
final finally finlalize  
Java 8 отличия от 7 (Что таоке Optional)  
Стримы в java  
Как работают в параллели стримы?

Spring  
stateless и stateful бины. как решается проблема стейтфул бинов. как работает loadbalancer, какие принципы и алгоритмы в основе?  
разница между Spring Controller и RESTResponce Controller  
разница между аннотациями Component и Service  
Dependency Injection, Dependency Inversion связь между DI (патрерн реализация) и IOC (принцип)  
Spring DI. Способы DI (конструктор, методы в Java) что дает спринг и что упрощает?  
bean factory post processor  
Spring Data - основные принципы работы  
Scope bean - какие бывают  
Spring Boot - что такое и в чем его основная фишка  
Писали ли стартеры, принцип работы стартера  
как изменять уровень изоляции в Spring  
нормальные формы баз данных  
микросервисная архитектура - плюсы минусы, Паттерн Saga  
Принципы REST

Hibernate   
Статусы жизни сущности Hibernate  
Session и транзакция - связь. можно ли несколько транзакций на одну сессию

Шаблоны GRASP состоят из 9 паттернов:

1. Information Expert — информацию обрабатываем там, где она содержится.
2. Creator — создаем объекты там, где они нужны.
3. Controller — выносим логику многопоточности в отдельный класс или компонент.
4. Low Coupling 5) High Cohesion — проектируем классы с однородной бизнес-логикой и минимальным количеством связей между собой.
5. Polymorphism — различные варианты поведения системы при необходимости оформляем в виде полиморфных вызовов.
6. Pure Fabrication (чистая выдумка) — не стесняемся создавать классы, не имеющие аналог в предметной области, если это необходимо для соблюдения Low Coupling и High Cohesion.
7. Indirection (посредник) — любой класс вызываем через его интерфейс.
8. Protected Variations — применяя все вышесказанное, получаем устойчивый к изменениям код.

Common

Какие книги читали, Блох?

Слышали ли про Domain Driven Design

Что такое ООП, что такое полиморфизм, статический полиморфизм

Какие парадигмы программирования есть: функциональное, императивное. декларативное, ООП

Основные принципы SOLID, только ли для классов?

GRASP - паттерны. Отличия репозитория от дао.

Закон Диметры (компоненты не зависят от деталей внутренней реализации)

Закон Деметры говорит нам о том же, о чем в детстве говорили родители: «Не разговаривай с незнакомцами». А разговаривать можно вот с кем:  
  
— С методами самого объекта.  
— С методами объектов, от которых объект зависит напрямую.  
— С созданными объектами.  
— С объектами, которые приходят в метод в качестве параметра.  
— С глобальными переменными (что лично мне не кажется верным, так как глобальные переменные во многом увеличивают общую сложность)

Паттерны проектирования

Coupling Cohesion - что это

Java

Java как функцоинальный язык. Какие инструменты добавились, какие функциональные интерфейсы есть?

Java стрим API, сплитератор.

Есть массив массивов, его нужно умножить на константк и получить одномерный

Отличия map от flatmap. Сигнатура методов.

Что такое функция высшего порядка.

Коллолекции, интерфейсы.Map отличия от Collection.

Различие checked unchecked исключения. Иерархия итерфейсов исключений.

final finally finlalize

Java 8 отличия от 7 (Что таоке Optional, почему Optional, это плохая практика в сигнатуре методов - в сигнатуре методов не уберегает от NPE, но в качестве возвращаемого типа Ок)

Стримы в java

Как работают в параллели стримы?

что такое deadlock

@Component

public class ObjectService {

public volatie HashMap<Object, Object> map;

}

какие проблемы с кодом

Spring

stateless и stateful бины. как решается проблема стейтфул бинов. как работает loadbalancer, какие принципы и алгоритмы в основе?

разница между Spring Controller и RESTResponce Controller

разница между аннотациями Component и Service

Dependency Injection, Dependency Inversion связь между DI (патрерн реализация) и IOC (принцип)

Spring DI. Способы DI (конструктор, методы в Java) что дает спринг и что упрощает?

как устроена аннотация Transactional (прокси над всем объектом, что будет если методы внутри обхекта вызывать друг из друга, если они помечены Transactional)

уровни изоляции транзакций

bean factory post processor

statement vs prepared statement

Spring Data - основные принципы работы

Scope bean - какие бывают, разница между прототипом и реквестом

Spring Boot - что такое и в чем его основная фишка

Писали ли стартеры, принцип работы стартера

как изменять уровень изоляции в Spring

нормальные формы баз данных

микросервисная архитектура - плюсы минусы, Паттерн Saga

Принципы REST - методы get-post - почему нельзя создание делать через get.

Юнит и интеграционное тестирование, разница.

знаковмство с мок интерфейсом

Hibernate

Статусы жизни сущности Hibernate

Session и транзакция  - связь. можно ли несколько транзакций на одну сессию

проблема n+1

жизненный цикл entity

findById и Get - один кидает исключение.

констрейнт и ограничения в базе данных.  первичные ключи и foreign

пессимистичная и оптимистичная блокировка.

Каfka - принципиальное отличие от других очередей типа rabbit

«Чем меньше вещей нужно держать в голове, глядя на отдельный участок кода, тем меньше сложность этого кода». То есть программный код нужно организовывать так, чтобы можно было безопасно работать с отдельными фрагментами по очереди (по возможности не думая об остальных фрагментах).

Как кэшируются запросы в БД hibernate, зачем.

Принципы REST

Логирование запросов пользователей (ElasticSearch, Logstash, Kibana и Prometheus)

Свойства распределенных баз данных Base

В **NoSQL**вместо **ACID** может рассматриваться набор свойств **BASE**:

•**базовая доступность** (англ. **Basic availability**) —каждый запрос гарантированно завершается (успешно или безуспешно).

•**гибкое состояние** (англ. **Soft stat**e) —состояние системы может изменяться со временем, даже без ввода новых данных, для достижения согласования данных.

•**согласованность в конечном счёте** (англ. **Eventual consistency**) —данные могут быть некоторое время рассогласованы, но приходят к согласованию через некоторое время

Шардирование, репликация БД (sticky session).

Monitor, состояния flat/inflated

Слабые ссылки в java

Сколько может быть Proxy у bean

Контейнер сервлетов

Пытается ли очистить память java перед stackoverflow

1**Семафор** - примитив синхронизации работы процессов и потоков, выраженный счетчиком количества потоков, у которых единовременный доступ к объекту.

2. ***Мьютекс*** - одноместный семафор (доступ только у одного потока)

3. **Монитор** - скрытая логика, реализация java-машиной блокировки-разблокировки доступа к объекту. Выражена ключевым словом synchronized. Избавляет от необходимости ручного использования примитивов синхронизации.

Сильные и слабые ссылки в java(«Слабые» ссылки и «мягкие» ссылки (WeakReference, SoftReference)

Сколько используется Proxy при использовании Transactional, собственных аннотаций (создается матрешка на каждую аннотацию свой Proxy).

Описать контейнер сервлетов.

**C проекта ВТБ**

вопросы:

**Collections Framework**

* Иерархия
* Зачем необходим интерфейс Set, что это такое, как получить элемент множества?
* Какие реализации интерфейса Set знаете, зачем нужен TreeSet, как работает, в чем отличие HashSet от LinkedHashSet

**Интерфейс Map**

* реализации
* HashMap - как устроен в двух словах
* Есть ли какие-нибудь требования к ключам HashMap, как потерять объект в HashMap

**Иерархия Exception**

* Checked и Unchecked, в чем разница
* Error - какие виды ошибок вы знаете, какие встречались на практике, OutOfMemoryError

(В каких случаях можем получить OutOfMemoryError,  можем ли мы, например, перехватить OutOfMemoryError и запустить поток на исполнение, что в этот момент произойдет с JVM)

**Общие вопросы**

* Какие модификаторы доступа вы знаете в java
* Что такое неизменяемый объект, зачем нужен, как содать (final class, private final поля и публичный конструктор)
* в чем отличие абстрактного класса от интерфейса, есть необходимость переопределять методы интерфейса, если мы реализуем интерфейс в абстрактном классе

**Spring**

* SOLID(принципы ООП) - инверсия зависимостей, в чем заключается смысл принципа ООП, какая реализация используется в Spring (DI)
* Какие есть Виды(@Scope) Бинов, в чем отличие? Scope prototype - использовали на практике? можно ли Prototype Bean инжектировать в Singleton Bean
* Какие существуют варианты внедрения зависимостей? Почему внедрение зависимостей через setter является плохой практикой?
* AOP в spring, прокси объекты: как работает класс с транзакционными методами (помеченные аннотацией @Transactional)

**Hibernate**

* Что такое JPA? Как Hibernate зависит от JPA?
* Как бы вы спроектировали сущности Entity (родитель и ребенок, связь один ко многим), какие аннотации необходимо прописать в родительском классе
* FetchType: Lazy, Eager - какие проблемы с производительностью могут возникнуть и в том и в другом случае (Lazy : проблема N+1 запрос в базу), как правильно работать с Lazy
* Что будет если из нетранзакционного метода в том же классе вызвать тарнзакционный(@Transactional), какие проблемы могут возникнуть?

**Многопоточность:**

* volatile - зачем нужен
* Hapens before - что это такое?
* На примере синглтона - как реализовать синглтон с двойной проверкой (почему двойная проверка является антипаттерном), если синхронизировать доступ к  методу getInstance(), нужно ли оставлять volatile у поля instance или можно убрать?

public class Singleton {

   public static volatile Singleton instance;

   private Singleton() {}

   public static Singleton getInstance() {

       Singleton local = instance;

      if (local == null) {

           synchronized(Singleton.class) {

               local = instance;

               if (local == null) {

                  instance = local = new Singleton();

              }

          }

      }

      return local;

   }

}

* Как запустить поток в java, в чем отличия runnable от callable? как запустить сallable
* CompletableFuture - использует ли пул потоков? какой?
* В чем отличие Lock от synchronized?

**Spring rest api**

* Как бы Вы спроектировали простейщий rest-сервис (например: CRUD операции на Employee)? Зачем нужен DTO(data transfer object)?

**Микросервисная архетикура**

* В чем отличие оркестровки от хореографии?
* Знаете ли паттерн SAGA - в чем суть?
* Circuit breaker - в чем суть? зачем нужен

**SQL**

* inner join us left join
* зачем нужны идексы в бд? какие виды индексов знаете?
* необходимо написать запрос:

есть табличка cities

[Export to CSV](https://wiki.corp.dev.vtb/pages/viewpage.action?pageId=245377212)

| **id** | **city** |
| --- | --- |
| 1 | Moscow |
| 2 | Novgorod |
| 3 | Vladimir |
| 4 | Moscow |

Как удалить дубликаты из таблички?

delete

from

cities

where

id in

(

select

city1.max\_id

from

(

select

city, max(id) as max\_id

from

cities

group by

city

having

count(city) > 1

) city1

)

**Задачка на ООП:**

При запуске алгоритм генерирует подземную парковку  
Т/C  
Автомобиль (3 единицы)  
Мотоцикл (2 единицв)  
Велосипед(1 единица)  
  
У каждого тс, есть свой набор полей(марка тс, цвет)   
Велосипед- марка и цвет  
Мотоцикл - марка, цвет, мощность двигателя  
Автомобиль марка, цвет, мощность двигателя, коробка передач

Необходимо написать алгоритм, при этом на выходе получить количество тс на парковке и количество Мотоциклов,  
а так же описание всех тс на парковке, например (Автомобиль : Audi, Red, 110 h/p, manual)

[Soft-скилы - Developer](https://wiki.corp.dev.vtb/pages/viewpage.action?pageId=75703477)

[Перейти к концу метаданных](https://wiki.corp.dev.vtb/pages/viewpage.action?pageId=75703477#page-metadata-end)

* Создал(а) [Абакумов Константин Сергеевич](https://wiki.corp.dev.vtb/display/~VTB227073@region.vtb.ru) [июн 13, 2020](https://wiki.corp.dev.vtb/pages/viewpreviousversions.action?pageId=75703477)

[Переход к началу метаданных](https://wiki.corp.dev.vtb/pages/viewpage.action?pageId=75703477#page-metadata-start)

Инновационность и адаптивность

1. Приведите пример, когда Вы смогли предложить инновационный подход к работе над привычной задачей?
   1. Как Вам удалось сформулировать этот подход?
   2. В чем была инновационность подхода?
   3. Как часто Вы сталкивались с подобными задачами до этого?
   4. Какого результата добились?
2. Когда Вы сталкиваетесь со сложно разрешимой задачей, как Вы ищете способ ее решения? Опишите конкретную ситуацию
3. Был ли у Вас период, когда Вам приходилось работать в условиях высокой степени неопределенности? Что вы делали в этой ситуации?

Влияние и убеждение

1. Часто бывает внешкомандное взаимодействие?
   1. Часты конфликты?
   2. Как решаете?
   3. Был ли случай, когда приходилось убеждать сотрудников из других команд?

Отношения

1. Приведите пример конфликтной ситуации участником которой Вы стали.
   1. Почему она возникла?
   2. Что делали, чтобы ее решить?
   3. Что могли бы сделать по-другому?
2. Опишите ситуацию, когда Вам было крайне сложно наладить рабочие отношения с другим человеком.
   1. Что именно было сложным?
   2. Как Вы действовали?
   3. Как Вы оцениваете результат?
3. Расскажите о недавней идее, которую Вы разработали совместно с другим коллегой/членом команды

Ориентация на результат, принятие решений и стрессоустойчивость

1. Что для Вас является стрессом на работе? Приведите пример, когда у Вас опустились руки?
2. Была ли цель, которую так и не удалось достичь?
3. Опишите Ваше самое рискованное решение, ответственность за которое Вы взяли на себя. Какую информацию Вы рассматривали перед принятием решения?