// JavaJub

Собес в ITK Academy

Java Junior

Вопросы, задачи и подготовка
к первому техническому интервью

Стек
Java · Spring · Spring Boot · Docker · SQL · Git · OAuth

// содержание

Содержание

01 Про ITK Academy и формат собеса

ITK Academy — это IT-компания из Таганрога, работающая по принципу аутстаффинга. То есть напрямую к ним «джавистом» ты не идёшь: они дообучают, помогают с резюме и портфолио, а затем устраивают на проекты компаний-партнёров. Это значит, что собес у тебя будет двухэтапный.

Как обычно устроен путь до оффера

Этап	С кем	Что проверяют
1. Знакомство	Менеджер ITK	Мотивация, стек, ожидания, опыт обучения
2. Тех-собес внутри	Техлид/ментор ITK	Java Core, базовые коллекции, ООП, твой pet-проект
3. Подготовка	Ментор	Доработка резюме, дообучение пробелов, моки
4. Собес у клиента	Тех-команда заказчика	Глубокий тех-собес под стек заказчика
5. Оффер	HR заказчика + ITK	Условия, выход на проект

> **ФИШКА · Главный нюанс**
>
> Ты готовишься не к одному собесу, а к двум разным. Внутри ITK будут смотреть, насколько тебя «продаваемо» можно отправить клиенту. У клиента — насколько ты вписываешься в их стек и команду. Готовиться нужно по широкой базе.

02 Стек по вакансии

ITK честно расписывает требования к Junior Java на Хабр Карьере. Вот что нужно знать на момент собеса:

Обязательный минимум

- Java/Kotlin на уровне core: типы, ООП, коллекции, исключения
- Git: основные команды, работа с ветками
- Spring: основные модули — web, security, data, cloud, test (на уровне «для чего нужны»)
- Spring Boot: стартеры, зачем они нужны, как написать свой
- Docker: контейнер vs образ, базовые команды
- CI/CD в GitLab или GitHub: что это и как пайплайн собирается
- ACID в реляционных БД, индексы — когда применять
- Pet-проект — обязательно

Будет плюсом

- OAuth 2.0: общие принципы (clients, scopes, grant types)
- Keycloak: если щупал руками — отдельный жирный плюс
- Java 11+ (по описанию вакансии — это базовый минимум)
- REST API: умение спроектировать простой эндпоинт
- PostgreSQL и MySQL — базовые отличия и использование

> **ВНИМАНИЕ · Что бросается в глаза**
>
> Стек для Junior довольно широкий — Spring Security, Cloud, OAuth, Keycloak обычно ждут от Middle. Это значит: глубоко знать всё не нужно, но «своими словами объяснить, что это и зачем» — обязательно.

03 Java Core & ООП

Это база — без неё дальше не пускают. Для Junior спрашивают концепты, а не тонкости JMM.

Типовые вопросы

- Расскажите про принципы ООП.

  → *Инкапсуляция, наследование, полиморфизм, абстракция. На Junior достаточно объяснить «своими словами» с примерами из жизни.*

- В чём разница между JDK, JRE и JVM?

  → *JVM — виртуальная машина (исполняет байт-код). JRE = JVM + стандартные библиотеки (нужно для запуска). JDK = JRE + инструменты разработки (javac, jdb, jar).*

- Java — компилируемый или интерпретируемый язык?

  → *И то, и другое. Сначала компилируется в байт-код (.class), потом JVM его интерпретирует и компилирует «горячие» места JIT-компилятором в нативный код.*

- Какие есть примитивные типы и их размеры?

  → *byte (1), short (2), int (4), long (8), float (4), double (8), char (2), boolean (~1, JVM-зависимо).*

- Что такое автобоксинг и распаковка?

  → *Автоматическое преобразование примитива в обёртку (int → Integer) и обратно. Имеет цену: создание объекта, возможный NullPointerException при распаковке null.*

- Почему String в Java immutable?

  → *Безопасность (передача в файлы, БД, сеть), потокобезопасность, кэширование hashCode, работа String Pool.*

- Что такое String Pool?

  → *Область в heap (раньше — в PermGen), где хранятся уникальные строковые литералы. String s = "hi" использует пул, new String("hi") — нет.*

- Разница между == и equals() для строк.

  → *== сравнивает ссылки. equals() сравнивает содержимое. Для строк, созданных литералом, == может вернуть true из-за String Pool, но полагаться на это нельзя.*

- Что вернёт someObj.equals(null)?

  → *false по контракту. equals не должен бросать NPE на null-аргумент.*

- Контракт equals/hashCode.

  → *1) Если a.equals(b), то a.hashCode() == b.hashCode(). 2) Если хэшкоды разные — объекты точно не равны. Если переопределяешь один — переопределяй и второй.*

- Что произойдёт, если положить объект в HashSet, а потом изменить поле, участвующее в hashCode?

  → *Потеряешь объект — найти его через contains() будет уже нельзя. Классическая ошибка с mutable-объектами в коллекциях.*

- Разница между checked и unchecked исключениями.

  → *Checked наследуются от Exception (но не RuntimeException) — компилятор требует обработки или throws. Unchecked — от RuntimeException и Error — обрабатывать необязательно.*

- Назови несколько unchecked исключений.

  → *NullPointerException, ArrayIndexOutOfBoundsException, IllegalArgumentException, ClassCastException, NumberFormatException.*

- Можно ли в catch ловить Throwable? Стоит ли?

  → *Можно, но не стоит — поймаешь и Error (OutOfMemoryError, StackOverflowError), которые обычно нельзя восстанавливать. Лови конкретные исключения.*

- Что такое try-with-resources?

  → *Конструкция try (Resource r = ...) {}, которая автоматически вызовет r.close() в finally. Ресурс должен реализовывать AutoCloseable.*

- Можно ли переопределить static-метод?

  → *Нет. Static-методы не участвуют в полиморфизме. Если в подклассе будет метод с такой же сигнатурой — это сокрытие (hiding), а не переопределение.*

- К каким конструкциям применим модификатор final?

  → *К классу (нельзя наследовать), методу (нельзя переопределить), переменной (нельзя переприсваивать).*

- Что такое абстрактный класс? Чем отличается от интерфейса?

  → *Абстрактный класс может содержать состояние и реализованные методы, наследуется только один. Интерфейс — контракт, можно реализовать несколько. С Java 8 в интерфейсах появились default-методы.*

- Расскажи про модификаторы доступа.

  → *private — только в классе. (default/package-private) — в пакете. protected — в пакете + наследникам. public — везде.*

> **СОВЕТ · Лайфхак**
>
> На вопрос про ООП всегда заготовь жизненный пример. Например, инкапсуляция — это банкомат: ты не знаешь, как он внутри хранит деньги, у тебя есть только публичные методы «снять», «положить», «проверить баланс». Запоминается и тебе, и интервьюеру.

04 Коллекции

Спрашивают всегда. HashMap и ArrayList — обязательно. Готовься рассказывать «как устроено».

- Какие основные интерфейсы в Collection Framework?

  → *Collection (List, Set, Queue), Map (отдельно). List — упорядоченный с дубликатами. Set — без дубликатов. Map — пары ключ-значение.*

- Как устроен ArrayList внутри?

  → *Динамический массив. При нехватке места создаёт новый массив в 1.5 раза больше и копирует данные через Arrays.copyOf.*

- ArrayList vs LinkedList — что когда использовать?

  → *ArrayList: быстрый доступ по индексу O(1), быстрая итерация. LinkedList: быстрая вставка/удаление в начале O(1), но поиск O(n). На практике почти всегда выигрывает ArrayList.*

- Как устроен HashMap?

  → *Массив бакетов (Node[]). Бакет выбирается по хэшу ключа: (n - 1) & hash, где n — размер массива. При коллизии — связный список, а с 8 элементов в бакете и размере таблицы ≥ 64 он превращается в красно-чёрное дерево.*

- Что такое load factor?

  → *Порог заполнения, по умолчанию 0.75. При size >= capacity * loadFactor происходит resize: новый массив вдвое больше + перехеширование всех элементов.*

- Сложность операций HashMap.

  → *put, get, remove — O(1) в среднем. В худшем случае O(log n) благодаря дереву (Java 8+).*

- Можно ли в HashMap положить null-ключ или null-значение?

  → *В HashMap — да (один null-ключ хранится в bucket 0). В ConcurrentHashMap — нельзя ни ключ, ни значение.*

- Что будет, если использовать как ключ изменяемый объект и потом его изменить?

  → *Объект «потеряется». hashCode станет другим, и при поиске мы попадём не в тот bucket.*

- HashMap vs TreeMap vs LinkedHashMap.

  → *HashMap — быстрый, без порядка. TreeMap — отсортирован по ключам, O(log n). LinkedHashMap — сохраняет порядок вставки или access order.*

- HashSet — на основе чего реализован?

  → *На HashMap, где значение — заглушка (PRESENT). Все операции делегируются HashMap.*

- Что такое fail-fast итератор?

  → *Итератор, который кидает ConcurrentModificationException, если коллекция изменена не через сам итератор. Так работают итераторы у HashMap, ArrayList.*

- Как из List сделать неизменяемый?

  → *List.copyOf(list) (Java 10+) или Collections.unmodifiableList(list) — обёртка, которая бросит UnsupportedOperationException на add/remove.*

- В чём разница между List.of(1,2,3) и new ArrayList<>()?

  → *List.of возвращает immutable-список. Любая попытка изменить — UnsupportedOperationException. И в нём нельзя null.*

05 Исключения и многопоточность (база)

Для Junior спрашивают самое начало, без JMM и lock-free алгоритмов.

Исключения

- Иерархия исключений в Java.

  → *Throwable → Error (системные, не ловим) и Exception. От Exception → checked и RuntimeException (unchecked).*

- Можно ли в одном catch ловить несколько типов исключений?

  → *Да, multi-catch с Java 7: catch (IOException | SQLException e). Переменная при этом effectively final.*

- Что выполнится в finally, если в try return?

  → *Finally выполнится перед фактическим возвратом. Если в finally тоже return — он перебьёт значение из try (анти-паттерн).*

- Какое исключение лучше — выбросить своё или использовать стандартное?

  → *Если ситуация уникальна для домена — своё (наследник RuntimeException обычно). Если стандартное (например, IllegalArgumentException) подходит — лучше его.*

Многопоточность — база

- Чем процесс отличается от потока?

  → *Процесс — единица ОС со своей памятью. Поток — единица исполнения внутри процесса, потоки делят общую память.*

- Способы создать поток в Java.

  → *extends Thread, implements Runnable, через Callable + ExecutorService, через CompletableFuture. На Java 21+ — Virtual Threads.*

- Что такое synchronized?

  → *Ключевое слово для захвата монитора объекта. Гарантирует, что в блок одновременно войдёт только один поток. Можно на методе или на блоке кода.*

- Зачем нужен volatile?

  → *Гарантирует видимость изменений переменной между потоками (без кэширования в регистрах). НЕ гарантирует атомарность операций типа i++.*

- Состояния потока.

  → *NEW (создан), RUNNABLE (готов или выполняется), BLOCKED (ждёт монитор), WAITING / TIMED_WAITING (ждёт сигнал/время), TERMINATED.*

- Что делает Thread.sleep() и Thread.yield()?

  → *sleep — приостанавливает поток на указанное время (не освобождает мониторы). yield — намекает планировщику «дай поработать другим», но это лишь подсказка.*

06 Spring и Spring Boot

В вакансии ITK прямо упомянуты Spring web, security, data, cloud, test и Spring Boot стартеры. На Junior достаточно базы.

Базовые вопросы

- Что такое Spring и зачем он нужен?

  → *Фреймворк для упрощения разработки enterprise-приложений на Java. Главные фичи: Inversion of Control (IoC), Dependency Injection (DI), AOP, готовые модули для web, data, security.*

- Что такое IoC и DI?

  → *IoC — принцип: контейнер сам управляет жизненным циклом объектов. DI — реализация: зависимости в объект внедряются извне (через конструктор / сеттер / поле).*

- Какой способ DI предпочтительнее?

  → *Constructor injection. Поля можно сделать final, явно видны обязательные зависимости, легко тестировать без Spring-контекста.*

- Чем @Component отличается от @Service, @Repository, @Controller?

  → *Технически почти ничем — все они помечают бин для регистрации в контексте. Но семантически: @Service — бизнес-логика, @Repository — работа с БД (плюс перевод исключений в DataAccessException), @Controller — веб-слой.*

- Что такое @Autowired и @Qualifier?

  → *@Autowired — внедрение зависимости. Если бинов несколько — Spring не знает какой выбрать → нужен @Qualifier("имяБина") или @Primary на одном из бинов.*

- Какие scopes у бинов?

  → *singleton (по умолчанию — один на контекст), prototype (новый при каждом запросе), request, session, application, websocket — для веб-приложений.*

- Что такое Spring Boot и чем он отличается от Spring?

  → *Spring Boot — это Spring + автоконфигурация + встроенный сервер (Tomcat/Jetty) + удобные стартеры. Минимизирует boilerplate. По сути «Spring без боли с XML».*

- Что такое стартер в Spring Boot?

  → *Готовый набор зависимостей под задачу. Например, spring-boot-starter-web подтянет Spring MVC, Tomcat, Jackson и т.д. Один импорт — и веб-приложение работает.*

- Как Spring Boot узнаёт, что и как настраивать?

  → *Через @EnableAutoConfiguration (входит в @SpringBootApplication) + @Conditional. Конфигурации перечислены в META-INF/spring/...AutoConfiguration.imports (или старом spring.factories).*

- Что делает @SpringBootApplication?

  → *Это композиция трёх аннотаций: @Configuration + @EnableAutoConfiguration + @ComponentScan.*

- Как сделать REST-эндпоинт?

  → *Класс с @RestController, метод с @GetMapping/@PostMapping. Параметры — @PathVariable, @RequestParam, @RequestBody. Возвращаемый объект сериализуется в JSON через Jackson.*

- Как обработать исключение в REST-контроллере?

  → *@ExceptionHandler в самом контроллере или глобально через @RestControllerAdvice + @ExceptionHandler.*

- Что такое Spring Data JPA?

  → *Модуль, который позволяет работать с БД через интерфейсы-репозитории. Достаточно унаследоваться от JpaRepository<Entity, ID> — Spring сам сгенерирует реализацию.*

- Как Spring создаёт реализацию репозитория, если ты только написал интерфейс?

→ *Через прокси, который генерируется в рантайме. Имя метода (findByEmail) парсится и превращается в SQL/JPQL.*

- Что такое @Transactional?

  → *Аннотация, которая открывает транзакцию перед методом и коммитит после или откатывает при исключении. По умолчанию откат на RuntimeException и Error, но не на checked.*

Про Spring Security и Cloud (на пальцах)

- Что делает Spring Security?

  → *Защищает приложение: аутентификация (кто ты) и авторизация (что тебе можно). Работает через цепочку фильтров перед твоим контроллером.*

- Что такое Spring Cloud и для чего он?

  → *Набор инструментов для микросервисов: service discovery (Eureka), конфиг-сервер, API Gateway, Circuit Breaker. На Junior достаточно знать «что это и зачем».*

// JavaJub

Это только начало гайда.

ITK Academy · Junior

Полная версия — бесплатно в Telegram-канале:

- все вопросы с ответами по грейду
- задачи: live-coding, SQL, System Design — с разбором
- чек-лист «за 2 часа до собеса»

@java_jub

t.me/java_jub · подписка бесплатная