// JavaJub

Собес в Бизнес Технологиях

Global ERP · альтернатива SAP

Java · Junior

Часть 1 · бесплатная версия · @Java_Jub

Вопросы, задачи и подготовка
к интервью: Java Core + Алгоритмы + Обход деревьев

Стек по вакансии

Java SE · ООП · Коллекции · HTTP · PostgreSQL

equals/hashCode · String Pool · Collections · Алгоритмы · Деревья · REST

Содержание

Разделы 01–09 — открыты. *10–22 — в Java Jub Pro (показаны темы и вопросы).*

01 Бизнес Технологии и Global ERP

Бизнес Технологии — российский разработчик корпоративного ПО с офисом в Санкт-Петербурге на Коломяжском, 25 лет на рынке. Главный продукт — Global ERP: платформа для управления крупным предприятием, позиционируется как реальная отечественная альтернатива SAP и Oracle E-Business Suite. В сценарии импортозамещения это один из ключевых игроков рынка, особенно для машиностроения, судостроения, нефтегаза и химии.

Технологически Global ERP — это трёхзвенная архитектура: PostgreSQL как СУБД, сервер приложений собственной разработки на Java, веб-клиент в браузере. Никакого Spring «из коробки» — у компании свой framework, написанный годами под их задачи. Поэтому на собесе тебя проверяют на знание базы Java и алгоритмов, а не на «расскажи Spring Boot 3».

Что важно понимать про продукт

Что такое Global ERP в одной фразе?

Это российская комплексная информационная система для управления предприятием: финансы, склад, закупки, производство, кадры, документооборот — всё на единой платформе, работающей через браузер. Позиционируется как замена SAP ERP / S4HANA и Oracle EBS. В декабре 2024 платформа выдержала нагрузочный тест на 15 тысяч одновременных пользователей в одной базе — это уровень крупных предприятий.

Какие ещё продукты у компании?

Линейка Global: Global ERP (управление предприятием), Global MES (управление производственными процессами на цеховом уровне — внедрено на НПП «Радар ммс»), Global EAM (управление ремонтами и оборудованием), Global HRM (кадры и расчёт зарплаты), Global-FrameWork (платформа цифровизации управленческих процессов). Все продукты включены в реестр российского ПО.

На каких отраслях специализируется компания?

Машиностроение, судостроение, химическая промышленность, нефтегазовая отрасль, энергетика — то есть тяжёлая индустрия и крупные корпорации с десятками тысяч сотрудников. Это не банковский / e-commerce домен: задачи другие — длинные бизнес-операции, сложная учётная политика, интеграции с ФНС, банками, ГИС, 1С.

Почему у них собственный framework, а не Spring?

Платформа разрабатывалась более 10 лет назад, когда команда сознательно выбрала путь собственных решений, чтобы получить полный контроль над архитектурой и независимость от внешних вендоров. Свой framework позволяет глубоко кастомизировать поведение системы под каждого крупного заказчика — это критично для ERP, где у каждого предприятия свои бизнес-правила. Для тебя это значит: на работе будешь учить их внутренние библиотеки, а не Spring.

Что значит «нет legacy» в описании вакансии?

У компании нет наследия старого Java-кода 2005 года с нечитаемыми JSP-страницами и Struts 1. Платформа писалась с нуля под современный стек, постоянно рефакторится силами команды senior+ уровня. Тебе как Junior это плюс: код будет читабельный, архитектура внятная, ментор в команде есть. Минус: высокая планка к качеству твоего кода тоже.

Чем компания отличается от 1С:ERP и Галактики?

1С и Галактика — это другие российские ERP-системы, исторически сильные в среднем бизнесе и бухгалтерии. Global ERP более ориентирован на крупный enterprise с тяжёлой индустрией и пытается закрыть нишу импортозамещения именно SAP-уровня, где 1С исторически не доминирует. Технологически тоже разные: 1С строится на собственном языке программирования 1С, Global ERP — это Java + PostgreSQL.

Что значит «25 лет на рынке» для меня как Junior?

Компания стабильная, не стартап. Есть процессы, есть ментор, есть документация. Минус — может быть консервативный темп: ревью кода долгое, релизы по графику, эксперименты ограничены. Плюс — белая зарплата, ДМС, аккредитация IT-компании (отсрочка от армии), реальные крупные клиенты, чьи имена идут в резюме весомо.

> **СОВЕТ · Что почитать перед собесом про продукт**
>
> Зайди на global-system.ru — посмотри страницу Global ERP, прочитай 2–3 абзаца про модули и архитектуру. На собесе оценивают, понимаешь ли ты, куда идёшь работать. Полезно знать слова: трёхзвенная архитектура, импортозамещение, реестр российского ПО, машиностроение, судостроение.

Команда и условия

Сколько человек в компании и где офисы?

Около 400+ сотрудников. Главный офис в Санкт-Петербурге (Коломяжский, 33к2, около метро Пионерская), есть офисы в Перми, Нижнем Новгороде, Вологде. Вакансия Junior — конкретно СПб офис, удалёнка не предлагается. График 5/2 с 9:00 до 18:00, есть гибкость ±1 час по началу дня.

Что в команде проекта?

Команда состоит из senior и middle+ разработчиков — это значит, что Junior — единственный новичок, остальные опытные. С одной стороны, есть у кого учиться: тебя не оставят один на один с фреймворком. С другой стороны, планка кода высокая, на ревью будут указывать на много мелочей. Это нормально и полезно для роста.

Что важно знать про условия?

Оформление по ТК РФ, полностью белая зарплата от 80 000 рублей на руки для Junior, ДМС после трёх месяцев испытательного срока. Компания аккредитована Минцифры как IT-компания (важно для отсрочки и IT-ипотеки). Есть корпоративные и спортивные мероприятия, pizza-day в последнюю пятницу месяца — это про культуру, и упоминание этого на собесе показывает, что ты читал вакансию.

02 Формат собеседования

Junior-собес в Бизнес Технологиях идёт стандартно для российских продуктовых компаний с офисом: 2 этапа, суммарно 1,5–2 часа, может добавиться тестовое задание. Знание этого формата заранее снимает половину волнения — ты приходишь с пониманием, что именно от тебя ждут на каждом этапе.

Этапы и длительность

Этап	Длительность	С кем	Содержание
1. HR-скрининг	20–30 мин	Рекрутер	Опыт, мотивация, ожидания, готовность к офису СПб
2. Техническое интервью	1–1,5 часа	Тимлид + CTO / senior разработчик	Java Core + ООП + коллекции + JVM + алгоритмы + SQL + Code Review + pet-проекты + soft
3. Тестовое задание (опц.)	1–3 дня дома	Самостоятельно	Реализация структуры данных, простой REST-сервис, парсер строки или CSV

Этап 1 — HR-скрининг

Что спрашивает HR на скрининге?

Стандартный набор: расскажи о себе, что заканчивал, какие пет-проекты делал, почему Java, почему именно Бизнес Технологии, готов ли работать в офисе на Пионерской пять дней в неделю. Дополнительно — ожидания по зарплате (вилка от 80 000 на руки, не стесняйся назвать чуть выше нижней границы — например, 90 000), готовность к full-time. Длится 20–30 минут, обычно по телефону или Zoom без видео.

На что HR смотрит, кроме ответов?

На вменяемость, на готовность учиться, на стабильность (не прыгаешь ли каждые полгода между работами / курсами). Для Junior из вуза или после курсов важна мотивация: почему именно разработка, почему Java, что делал руками. Чёткие короткие ответы лучше длинных монологов. Если HR спрашивает «есть ли вопросы», задавай — спроси про процесс адаптации, про менторство, про технологии команды.

> **ВНИМАНИЕ · Готовность к офису — фильтр**
>
> Вакансия офисная, удалёнки нет даже частичной. Если ты живёшь не в СПб или не готов ездить на Пионерскую каждый день — этот вопрос HR закроет разговор сразу. Заранее проверь, сколько ехать от твоего дома до Коломяжского, 33к2. На собесе уверенно скажи: «да, готов, рассматриваю офис как плюс — общение с командой важно для Junior».

Этап 2 — техническое интервью

Кто проводит техническое интервью?

Чаще всего тимлид направления + либо CTO, либо senior разработчик команды. Иногда без CTO, только тимлид. Атмосфера обычно спокойная — компания не банковская, без давления и

стрессовых техник. Спрашивают по конкретным темам по очереди: Java Core, коллекции, JVM, многопоточность, алгоритмы, БД. В конце — практическая задача на коде.

Какие блоки точно будут на техническом?

Java Core (примитивы, Object, equals/hashCode, String, Optional) — всегда. ООП с примерами и SOLID — всегда. Коллекции (HashMap внутри, ArrayList vs LinkedList) — почти всегда. JVM минимально (heap / stack / GC) — часто. Алгоритмы и обход деревьев — точно, это в требованиях вакансии. SQL базовый — точно, ERP без БД не работает. Тесты (JUnit / Mockito) — спросят, особенно про юнит-тесты.

Будет ли тестовое задание?

Может быть, не обязательно. Для Junior часто заменяют живой задачей на собесе — пишешь код в IDE на компьютере собеседующего или в общем документе. Если тестовое всё-таки дадут, оно будет небольшое: реализовать структуру данных (свой ArrayList, стек, очередь), написать простой REST-сервис, парсер CSV или JSON. Срок — 1–3 дня.

Спрашивают ли что-то нестандартное?

Иногда — логические задачи или вопросы на сообразительность («как определить фальшивую монету за 2 взвешивания», «сколько мячей помещается в школьный автобус»). Это не часто, но возможно. Главное — не зависать молча, рассуждать вслух. На IT-собесах ценят процесс мышления, а не правильный ответ с первой попытки.

Как себя вести во время технического?

Если не знаешь ответ — не выдумывай, скажи честно «не сталкивался, но логично предположить, что...». Если что-то знаешь поверхностно — обозначь это: «слышал термин, но в практике не использовал». Хорошие Junior-кандидаты честны: senior-собеседующие сразу видят, когда ты пытаешься натянуть знания, и это минус. Лучше показать знание границ своего опыта, чем городить ложь.

▸ Этап 3 — тестовое (если будет)

Что обычно дают в тестовом для Junior?

Три типичных варианта: (1) реализовать структуру данных с нуля — свой ArrayList или LinkedList с методами add, get, remove. (2) Написать маленький REST-сервис на Java SE с одним эндпоинтом, который что-то парсит и возвращает JSON. (3) Парсер — на вход CSV или строка «key1=value1;key2=value2», на выходе структура и операции с ней. Срок 1–3 дня, в реальности занимает 4–6 часов работы.

На что смотрят при проверке тестового?

Чистота кода (читаемые имена переменных, отсутствие магических чисел, разбиение на методы), наличие тестов (хотя бы 3–5 юнитов через JUnit 5), корректность работы на edge cases (пустой вход, null, отрицательные числа), README с описанием решения. Бонус — продуманная архитектура: разделение на слои, интерфейсы для абстракций, обработка исключений. Не дописывай Spring Boot, если в задании про чистую Java SE.

Совет по структуре тестового: сделай отдельный пакет model для классов с данными, service для бизнес-логики, util для парсеров. Покрой 3–5 юнит-тестами через JUnit 5. В README напиши, что реализовал, какая сложность алгоритма, какие edge cases учёл, как запустить. Это уровень Middle+ для тестового — впечатлишь команду.

03 Стек по вакансии: что готовить

Стек вакансии Junior — намеренно базовый. Конкретный фреймворк (Spring, Hibernate) в требованиях не назван — у компании свой framework, его будут учить тебя на месте. На собесе проверяют фундамент Java, без которого ни один фреймворк не освоить.

Что обязательно к подготовке

Тема	Уровень подготовки	Где спросят
Java SE	Уверенно — без подсказок	Тех. собес, тестовое
ООП и принципы	С примерами из жизни / кода	Тех. собес
SOLID, DRY, KISS	Расшифровка + пример	Тех. собес
Коллекции	HashMap внутри, ArrayList vs LinkedList	Тех. собес
JVM / память / GC	На пальцах	Тех. собес
Многопоточность	Базово (Thread, synchronized, volatile)	Тех. собес
Алгоритмы и Big-O	Сложности коллекций + классика	Тех. собес, live coding
Обход деревьев	In/Pre/Post/Level-order — код наизусть	Live coding
SQL базовый	SELECT, JOIN, GROUP BY, агрегаты	Тех. собес
JDBC	Statement vs PreparedStatement	Тех. собес
Тесты (JUnit + Mockito)	Базово	Тех. собес
Git	Базовые команды	HR + тех. собес
Maven	pom.xml, фазы жизненного цикла	Тех. собес

Что желательно (плюс к offer)

Pet-проекты на Java — даже учебные: backend на чистой Java SE, REST через HttpServer, парсер CSV или JSON, мини-чат через сокеты. Главное — твой код, который ты сам объяснишь.

Fullstack web-разработка базово — HTML, CSS, JavaScript, представление о DOM, fetch, JSON. Компания делает веб-клиент, понимание фронта — плюс.

PostgreSQL — основной СУБД продукта. Базовый SQL обязателен, но если знаешь индексы и EXPLAIN — отличный сигнал.

Понимание ERP-домена — что это, какие модули, чем отличается от банка и e-commerce. Это покажет осознанный выбор компании.

Знакомство с Selenium / Selenide / Playwright — потому что в обязанностях упомянуты «визуальные тесты». Не обязательно уметь, но понимать слова «UI-тест», «headless-браузер».

Чего НЕ ждут от Junior

Знания Spring Boot 3, Hibernate, JPA на глубоком уровне — у компании свой framework, спросят базу Java вместо этого.

Опыта коммерческой разработки — допустимы только pet-проекты.

Опыта с Kafka, RabbitMQ, Kubernetes — это уровень Middle.

Архитектурных решений и system design — для Junior это не стандарт.

Знания конкретных СУБД-нюансов (MVCC, EXPLAIN ANALYZE, плановщик запросов) — спросят максимум базовое.

> **ВНИМАНИЕ · Особенности этой вакансии**
>
> 60 процентов времени — на Java Core, коллекции, ООП, SOLID. 20 процентов — на алгоритмы и обход деревьев. 10 процентов — на SQL базовый. 10 процентов — на остальное (JVM, многопоточность, паттерны, тесты, git). Если за вечер успеешь пройти первые два блока — закроешь 80 процентов вопросов на собесе.

04 Java Core

Это база, без которой ни один Junior-собес не пройти. В Бизнес Технологиях нет Spring Boot — они смотрят, насколько ты владеешь самим языком. Все вопросы ниже типичны для российских собесов уровня Junior; знание ответов даёт стабильную базу для перехода в любую Java-команду.

Примитивы и обёртки

Сколько примитивных типов в Java и какие?

Восемь: byte (1 байт), short (2), int (4), long (8) — это целые; float (4) и double (8) — вещественные; char (2 байта, символ Unicode) и boolean. Каждый примитив имеет тип-обёртку: Integer, Long, Boolean и так далее. Примитивы хранятся в стеке (если это локальная переменная) и не могут быть null. Обёртки — это объекты в heap, ссылки на них могут быть null.

Чем опасны типы-обёртки?

Тремя вещами. Первое — NullPointerException при автораспаковке: если Integer x = null, и ты пишешь int y = x, поймаешь NPE. Второе — расход памяти: Integer занимает 16 байт против 4 байт у int, для больших коллекций это ощутимо. Третье — кэш Integer от -128 до 127: для значений в этом диапазоне Integer.valueOf вернёт один и тот же объект, и == даст true; для значений вне диапазона — разные объекты, и == даст false. Это классическая ловушка собеса.

Где живут примитивы и где живут объекты?

Примитивы как локальные переменные методов лежат в стеке вместе с другими данными вызова — это быстро и автоматически освобождается при возврате из метода. Если примитив является полем класса, он лежит в heap внутри объекта-владельца. Объекты всегда в heap, а в стеке только ссылки на них. Статические поля (включая примитивы) живут в Metaspace, в области класса.

Object, equals и hashCode

Какие методы есть у Object?

Их девять: equals, hashCode, toString, getClass, clone, finalize (deprecated с Java 9), wait (три перегрузки), notify, notifyAll. Из них на собесе чаще всего спрашивают про первые три — equals, hashCode, toString. Остальные про многопоточность и про reflection.

Что возвращает toString по умолчанию?

Строку вида ИмяКласса@хэшкод_в_шестнадцатеричном — например, com.example.User@1b6d3586. Это не равно hashCode объекта напрямую: метод Object.toString вызывает Integer.toHexString(System.identityHashCode(this)), то есть идентичность объекта, а не результат твоего переопределённого hashCode. Если переопределяешь equals/hashCode, обязательно переопределяй и toString — это сильно облегчает отладку и логи.

Какой контракт у equals и hashCode?

Три правила. Первое — если a.equals(b) равно true, то a.hashCode должен быть равен b.hashCode (обратное не обязательно). Второе — equals должен быть рефлексивным (a.equals(a) = true), симметричным (a.equals(b) ⇔ b.equals(a)), транзитивным (a=b, b=c ⇒ a=c) и согласованным (повторный вызов возвращает тот же результат). Третье — a.equals(null) должно вернуть false, а не бросать NPE. Нарушение этих правил ломает HashMap, HashSet и любые коллекции на хешах.

Как правильно писать equals?

По шаблону: (1) если this == other, вернуть true (быстрая проверка идентичности). (2) Если other == null, вернуть false. (3) Если getClass() != other.getClass() или other instanceof не подходит — вернуть false. (4) Кастуй other к своему типу. (5) Сравни все значимые поля через Objects.equals (для объектов) или ==/Double.compare (для примитивов). Параллельно переопредели hashCode через Objects.hash(field1, field2, ...) — это стандартный путь.

Когда хэш-коды равны, а equals не равны — как такое возможно?

Hash-коды — это int, то есть всего 4 миллиарда возможных значений на бесконечное число возможных объектов. По принципу Дирихле обязательно найдутся два разных объекта с одинаковым hashCode — это называется коллизия. HashMap справляется с этим: при коллизии в одном бакете лежат несколько элементов в виде связного списка или дерева, и при get вызывается equals для точного сравнения. Поэтому равные hashCode — это норма, неравные equals при равных hashCode — тоже норма.

▸ String

Почему String immutable?

Это сделано из соображений безопасности, потокобезопасности и оптимизации. Immutable String может безопасно разделяться между потоками без синхронизации, может быть ключом в HashMap (hashCode не меняется), может попадать в String Pool и переиспользоваться. Если бы String была изменяемой, передача пути к файлу или пароля в метод позволяла бы изменить значение под капотом — это дыра в безопасности.

Что такое String Pool?

Это специальная область внутри heap, где JVM хранит уникальные String-литералы. Когда ты пишешь String s = "hello", строка попадает в pool, и при повторном литерале "hello" в любом месте программы возвращается ссылка на тот же объект. Это экономит память. String s = new String("hello") создаёт отдельный объект в heap мимо pool, а метод s.intern() возвращает каноническую версию из pool.

String vs StringBuilder vs StringBuffer — что выбрать?

String — immutable, конкатенация в цикле через + создаёт N+1 промежуточных объектов и медленная. StringBuilder — изменяемый, не потокобезопасен, используй для конкатенации в одном потоке (стандартный случай). StringBuffer — то же самое, но методы synchronized; нужен только если действительно несколько потоков пишут в одну строку, что встречается редко. По умолчанию выбирай StringBuilder.

Почему сравнение строк через == опасно?

Потому что == сравнивает ссылки, а не содержимое. String a = "hello"; String b = "hello"; a == b даст true (оба из pool). Но String c = new String("hello"); a == c даст false — это другой объект в heap. Правильное сравнение всегда через equals. Это классическая ловушка на собесах и баг в коде Junior-разработчиков.

▸ Optional

Зачем нужен Optional?

Чтобы явно показать в сигнатуре метода, что результат может отсутствовать. Optional<User> findUser сразу говорит читающему: «может вернуть пусто, обработай это». Это лучше, чем возвращать null, потому что заставляет вызывающего явно проверить через ifPresent / orElse / map, а не забыть и поймать NPE. Optional существует с Java 8 и предназначен прежде всего для возвращаемых значений, а не для полей класса.

// JavaJub

Это только начало гайда.

Бизнес Технологии · 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