// JavaJub

Собес в Сбере

Стажировка SberSeasons

Java · Стажёр

Вопросы, задачи и подготовка
к интервью на студенческую стажировку

Стек

Java Core · ООП · Коллекции · SQL · Алгоритмы · Spring · Git

// содержание

Содержание

01 О стажировке и формат собеса

SberSeasons — основная программа оплачиваемых стажировок Сбера для студентов очной формы (бакалавриат с 2 курса, специалитет, магистратура, аспирантура). Проходит два раза в год — весной и осенью. Стажировка длится 3–6 месяцев, а самое главное: 78% стажёров получают оффер в штат. То есть это не «отметка в резюме», а реальная дверь в IT-карьеру.

В этом гайде разобраны все вопросы и задачи, которые встречаются на интервью на стажировку по направлению Java. Уровень — стажёр / trainee (фактически Junior без коммерческого опыта). Сбер сам прямо подчёркивает: «не нужен опыт production-кода, нужны фундамент и желание учиться». Поэтому фокус — на основах Java Core, коллекциях, SQL, алгоритмах и умении думать вслух.

Параметры программы

Параметр	Значение
Программа	SberSeasons — основная стажировка Сбера для студентов
Кому	Студенты очной формы: бакалавриат с 2 курса, специалитет, магистратура, аспирантура
Длительность	3–6 месяцев, обычно по 20+ часов в неделю
Зарплата	~45 000 ₽/мес (средняя стипендия, для аспирантов выше)
Формат	Офис / гибрид / удалёнка — зависит от команды и города
Оффер после	78% стажёров получают предложение в штат
Когда подают	Заявки два раза в год — весной и осенью на sberstudent.ru

Как устроен отбор

Процесс полностью онлайн, 4–6 этапов. По времени растягивается на 2–4 недели.

Этап	Длит.	Что проверяют
1. Анкета на sberstudent.ru	1 раз	Резюме + мотивация + выбор направления. Здесь отсеивается больше всего кандидатов!
2. Онлайн-тесты	30–60 мин	Логика, английский, иногда базовый Java/SQL
3. Видео-интервью с HR	20–30 мин	Мотивация, опыт по STAR, базовый английский
4. Техническое интервью	30–90 мин	Java Core + алгоритмы (live coding) + SQL + Spring базово + soft
5. Доп. задание (опц.)	дни	Pet-задача: написать REST-сервис, тесты, выложить в Git
6. Финал / fit-интервью	30–45 мин	Рассказ о проекте + взаимная оценка с тимлидом

ВНИМАНИЕ · Анкета — главный отсев

Большинство кандидатов отсеивается на анкете. Это не формальность. Отнесись к ней как к резюме: распиши учебные проекты, курсовые, хакатоны, ссылки на GitHub, любой пет-проект.

Если у тебя есть хоть один проект — обязательно укажи ссылку. Один работающий проект на GitHub стоит десяти строк в разделе «навыки».

На что Сбер обращает особое внимание

1. Фундамент. ООП, коллекции, JVM «на пальцах», базовый SQL. Глубина промышленных знаний не нужна — нужна крепкая база.
2. Алгоритмы и структуры данных. Сбер сам подчёркивает: эта секция часто весомее, чем вопросы по Spring. К live coding надо тренироваться отдельно.
3. Способность думать вслух. На лайвкодинге интервьюер слушает ход мысли. Молчаливое верное решение хуже громкого неверного с правильными рассуждениями.
4. Готовность учиться. Вопросы намеренно задают за пределами твоих знаний, чтобы посмотреть, как ты реагируешь. Честное «не знаю, но логически предположил бы…» — лучше выдуманного ответа.
5. Адекватность. Не приукрашивай резюме. Опытные руководители на 5-минутной беседе видят, где правда, а где надувательство.
6. Командность. Расскажи про учебные проекты как про командную работу: «как договаривались», «как делили задачи», «как разрешали конфликты».

ФИШКА · Что говорить про Сбер на вопрос «почему именно мы»

Подготовь 2–3 факта: один из крупнейших работодателей IT в России, своя экосистема (СберМаркет, Самокат, Сбер3доровье, СберДевайсы), Platform V — собственная low-code платформа, активная разработка AI (GigaChat, Kandinsky). Можешь упомянуть конкретный продукт Сбера, которым пользуешься.

Главное — не говорить «потому что большая зарплата». Это правда, но это плохой ответ.

Что Сбер НЕ ждёт от стажёра

Это важно знать, чтобы не нервничать. От стажёра не ждут:

- Глубокого знания System Design. Это не для стажировки.
- Production-опыта с Kafka, Kubernetes, микросервисами. Достаточно «знаю концепции, могу рассказать общими словами».
- Знания специфики банка (АБС, ЦБ РФ, проводки, межбанк) — научат на месте.
- Глубоких алгоритмов уровня LeetCode Hard. Задачи на стажировке — Easy с LeetCode.
- Идеального ответа на каждый вопрос. Честное «не знаю» — лучше попытки выкрутиться.

02 Стек: к чему готовиться

Сбер — большая компания с тысячами продуктовых команд. Стек разнится от команды к команде, но базовый набор, к которому проверяют стажёра, у всех одинаковый. Ниже — три группы: то, что точно спросят (must-have), то, что встречается часто, и то, что в плюс.

Точно спросят (фундамент)

- Java Core: ООП, примитивы, String, исключения, Stream API, Optional, лямбды, функциональные интерфейсы.
- Коллекции: List / Set / Map, устройство HashMap, ArrayList vs LinkedList.
- JVM на пальцах: где живут примитивы и объекты, GC, разница JVM / JRE / JDK.
- Многопоточность базово: что такое поток, deadlock, synchronized, volatile.
- SQL: SELECT, JOIN, GROUP BY, HAVING, индексы, ACID, транзакции и уровни изоляции.
- Алгоритмы: сортировки, бинарный поиск, Big-O, стек, очередь.
- Git: базовый набор команд, что такое Pull Request.

Часто встречается

- Spring и Spring Boot — на уровне «знаю, что такое DI и IoC, как работает @Autowired».
- Hibernate / JPA — спецификация vs реализация, @Entity, Lazy vs Eager, N+1 проблема.
- HTTP и REST — методы, идемпотентность, статус-коды, REST vs SOAP.
- Maven — pom.xml, жизненный цикл, scopes.
- Паттерны: Singleton, Builder, Factory, Observer, Proxy vs Decorator.
- Тесты: JUnit 5, Mockito, виды тестов (unit / integration).
- Liquibase / Flyway — что это и зачем (миграции БД).

В плюс, если знаешь

- Kafka — основные понятия (topic, broker, producer, consumer). От стажёра не ждут глубины, но термины должны быть на слух.
- Docker — что такое контейнер, отличие от VM.
- Redis — что это (key-value хранилище).
- NoSQL базово — MongoDB (документная), Cassandra (column-family).

> **ФИШКА · Сбер использует Oracle И PostgreSQL**
>
> Это особенность Сбера — много legacy на Oracle, новые сервисы на PostgreSQL. Поэтому на собесе SQL могут спросить через призму обеих БД: «в Oracle есть SEQUENCE, а в Postgres что?» (SERIAL / IDENTITY).
>
> На стажёре глубокая разница не нужна, но 2–3 факта стоит знать: в Postgres нет Read Uncommitted, минимум — Read Committed; в Oracle есть BEFORE / AFTER триггеры; auto_increment в обеих БД делается по-разному.

Что отличает Сбер

Платформа V — внутренняя облачная платформа Сбера, на которой работает значительная часть продуктов. Подразделение, в которое возьмут стажёра, скорее всего связано с одним из крупных направлений: Sber AI (GigaChat, Kandinsky), Платформа V (PaaS / IaaS), Экосистема (СберМаркет, СберЗдоровье, СберДевайсы), Корпоративные системы, Сопровождение IT-блока.

Знать всё это не нужно, но полезно зайти на developers.sber.ru и пробежать новости — это даст контекст и понимание, чем занимается компания.

> **ВНИМАНИЕ · Сбер не пользуется LeetCode Hard**
>
> По словам самих ребят из Сбера (статья на Хабре), на стажёре дают Easy-задачи с LeetCode. Это значит — сортировки, поиск, разворот строки, FizzBuzz, две суммы, палиндром.
>
> Не тратить время на Medium / Hard. Тратить на 30–50 Easy-задач, чтобы рука была набита.

03 Java Core и ООП

Самый большой блок на любом Junior-собесе. На стажировку в Сбер ждут уверенного владения базой: 4 принципа ООП, что такое JVM/JRE/JDK, как работают модификаторы доступа, в чём разница final / finally / finalize, как работает try-with-resources. Каждый ответ ниже — это минимум, который должен звучать на собеседовании.

ООП — 4 принципа

- Назови 4 принципа ООП.

  → *Инкапсуляция — скрытие внутреннего состояния объекта, доступ только через методы. Наследование — один класс получает поля и методы другого через extends. Полиморфизм — один интерфейс, разные реализации; вызов метода по ссылке на родителя выполняет нужный метод потомка. Абстракция — выделение существенного, скрытие деталей реализации (абстрактные классы и интерфейсы).*

- Что такое DRY, KISS, YAGNI?

  → *DRY — Don't Repeat Yourself. Не дублируй код, выноси в отдельный метод/класс. KISS — Keep It Simple, Stupid. Простое решение лучше навороченного. YAGNI — You Aren't Gonna Need It. Не пиши код «на будущее», который не нужен сейчас.*

SOLID

- Расшифруй SOLID.

  → *S — Single Responsibility (один класс — одна ответственность). O — Open/Closed (открыт для расширения, закрыт для модификации — добавляй наследников, не правь оригинал). L — Liskov Substitution (наследника можно поставить вместо родителя без поломок поведения). I — Interface Segregation (много специализированных интерфейсов лучше одного большого). D — Dependency Inversion (зависим от абстракций, не от конкретных реализаций — это основа DI в Spring).*

JVM, JRE, JDK

- Чем отличаются JVM, JRE и JDK?

  → *JVM (Java Virtual Machine) — виртуальная машина, которая исполняет байткод. Содержит GC и JIT-компилятор. JRE (Java Runtime Environment) — JVM + стандартные библиотеки (java.lang, java.util и т.д.), то что нужно для запуска. JDK (Java Development Kit) — JRE + компилятор javac + инструменты (jar, javadoc, jdb). JDK нужен для разработки, JRE — только для запуска.*

Примитивы и обёртки

- Какие примитивные типы есть в Java?

  → *Их 8: byte (1 байт), short (2 байта), int (4 байта), long (8 байт), float (4 байта), double (8 байт), char (2 байта, UTF-16), boolean (1 бит). У каждого примитива есть класс-обёртка: Integer, Long, Boolean и так далее.*

- Где хранятся примитивы и где объекты?

  → *Примитивы как локальные переменные — на стеке. Если примитив — поле объекта, он лежит внутри объекта в куче. Все объекты — в куче (heap). Статические поля и метаданные классов — в Metaspace (с Java 8 заменил PermGen).*

- Что такое автобоксинг и unboxing? В чём подвох?

→ *Автоматическое преобразование примитива в обёртку и обратно: Integer i = 5; — это int 5 заворачивается в Integer (boxing). int j = i; — обратно (unboxing). Подвох в том, что если обёртка null, при unboxing будет NullPointerException: Integer i = null; int j = i; — НПЕ.*

> **ФИШКА · Integer cache**
>
> Integer кэширует значения от -128 до 127. Поэтому Integer.valueOf(100) == Integer.valueOf(100) даёт true (одна и та же ссылка из кэша). А Integer.valueOf(200) == Integer.valueOf(200) — false (разные объекты).
>
> Правило: для обёрток ВСЕГДА используй equals(), не ==. Это любимый подвох на собесах.

Object и его методы

- От какого класса наследуются все классы в Java?

  → *От java.lang.Object. Это корень иерархии — даже массивы и enum'ы наследуются от него косвенно.*

- Какие основные методы у Object?

  → *equals(Object) — проверка равенства. hashCode() — целочисленный хэш. toString() — строковое представление. getClass() — Class-объект. clone() — копия объекта (нужно implements Cloneable). wait(), notify(), notifyAll() — для синхронизации потоков. finalize() — устаревший, deprecated с Java 9.*

equals и hashCode (классика)

- Расскажи контракт equals и hashCode.

  → *Контракт equals: рефлексивность (a.equals(a) = true), симметричность (a.equals(b) ⇔ b.equals(a)), транзитивность (a=b и b=c → a=c), консистентность (повторный вызов даёт тот же результат), equals(null) = false. Главное правило связки: если a.equals(b), то a.hashCode() == b.hashCode(). Обратное не обязано выполняться.*

- Что будет, если переопределить только equals, но не hashCode?

  → *Сломаются hash-структуры: HashMap, HashSet, Hashtable. Положили объект — посчитался один hashCode, переопределённый equals говорит «они равны», но другой hashCode значит другой бакет. Не найдёшь, что положил. Поэтому при переопределении equals — обязательно переопределяй hashCode.*

- Что будет, если hashCode возвращает константу (например, return 1)?

  → *Все объекты попадут в один бакет HashMap. Поиск превратится в перебор связного списка (или дерева с Java 8) — деградация с O(1) до O(n) или O(log n). Карта работает, но медленно. Это любимый подвох на собесе.*

String и его особенности

- Почему String immutable?

  → *Безопасность: пароли, URL, classpath — нельзя случайно изменить. String Pool — иммутабельность нужна для безопасного шеринга. Хэш-структуры: hashCode у String кэшируется, нельзя ломать. Потокобезопасность: иммутабельный объект можно безопасно шарить между потоками без синхронизации.*

- == vs equals для строк?

  → *== сравнивает ссылки. equals — содержимое. Из-за String Pool два литерала "abc" == "abc" дадут true, но new String("abc") == "abc" — false (new создаёт новый объект вне пула). На собесе всегда отвечать «equals для содержимого».*

- В чём отличие String, StringBuilder, StringBuffer?

→ *String — immutable. Каждая конкатенация создаёт новый объект, в цикле это O(n²) по памяти. StringBuilder — mutable, не потокобезопасный, быстрый. StringBuffer — mutable + synchronized методы, потокобезопасный, но медленнее. В одном потоке всегда нужен StringBuilder.*

Модификаторы доступа

- Какие модификаторы доступа есть в Java?

  → *private — только внутри своего класса. package-private (без модификатора, default) — внутри своего пакета. protected — пакет + наследники из других пакетов. public — отовсюду.*

- Как работает protected через пакеты?

  → *protected-метод виден всему своему пакету (как package-private) И всем наследникам, даже из других пакетов. То есть из другого пакета можно получить доступ только через наследование, не напрямую через ссылку. Тонкость на собесе.*

static, final

- Что такое static?

  → *Принадлежит классу, а не объекту. static-поле общее для всех экземпляров. static-метод можно вызвать без объекта: ClassName.method(). static-блок выполняется при загрузке класса. Часто используется для констант (static final), фабричных методов, утилит.*

- Что делает final для класса, метода, переменной?

  → *Для класса — нельзя унаследовать (final class String). Для метода — нельзя переопределить в наследнике. Для переменной — можно присвоить один раз. Для поля — становится константой (часто вместе со static).*

final, finally, finalize — три разных слова

- В чём разница final, finally, finalize?

  → *final — модификатор. final class / method / field. finally — блок в try-catch-finally, выполняется ВСЕГДА (даже при исключении или return). Используется для освобождения ресурсов. finalize() — метод Object, который раньше вызывал GC перед удалением объекта. Deprecated с Java 9, использовать нельзя. Сейчас для очистки используют try-with-resources и Cleaner.*

abstract class vs interface

- В чём разница абстрактного класса и интерфейса?

  → *Абстрактный класс может иметь состояние (поля), конструктор, реализованные методы. От него можно унаследоваться ТОЛЬКО одного. Интерфейс — контракт, описывающий «что умеет». С Java 8 могут быть default-методы (с реализацией) и static-методы. Можно реализовать ЛЮБОЕ количество интерфейсов. Использовать абстрактный класс — когда есть общая база и состояние. Интерфейс — когда нужно описать роль или behaviour.*

- Что такое default-методы в интерфейсах?

  → *С Java 8 в интерфейсе можно объявить метод с реализацией через ключевое слово default. Это решает проблему расширения интерфейсов без поломки совместимости: добавляешь новый метод с default-реализацией, и существующие реализации не ломаются. Так в Java 8 в Collection появился stream() — default-метод.*

Исключения

- Расскажи иерархию исключений.

  → *Throwable — корень. Делится на Error (OOM, StackOverflow — не ловить) и Exception. Exception → checked (IOException, SQLException — обязаны быть в throws) и RuntimeException → unchecked (NullPointerException, IllegalArgumentException — не обязаны).*

- Что такое checked и unchecked исключения?

  → *Checked — компилятор требует обработать (catch) или объявить в throws. Используются для ожидаемых внешних ошибок (IO, БД). Unchecked (RuntimeException и наследники) — обрабатывать не обязательно. Используются для ошибок программирования (NPE, IllegalArgumentException).*

- Можно ли создать свой exception от RuntimeException?

  → *Да, и это типовая практика. Большинство современных проектов делают свои исключения от RuntimeException — не загромождают сигнатуру методов throws. От Error не рекомендуется наследоваться — это «системные» ошибки JVM, бизнес-код их трогать не должен.*

- Что такое try-with-resources?

  → *Конструкция, которая гарантирует вызов close() у ресурсов, реализующих AutoCloseable. Заменяет try-finally. При исключении в try ресурсы всё равно закроются, а если close() сам бросит — оно станет suppressed внутри основного.*

```
try (BufferedReader reader = new BufferedReader(new FileReader("file.txt"))) {
    return reader.readLine();
}  // reader.close() вызовется автоматически
   // Даже если упадёт readLine — close всё равно сработает
```

Optional

- Для чего нужен Optional?

  → *Контейнер, который может содержать или не содержать значение. Появился в Java 8 как альтернатива null. Помогает явно показывать в сигнатуре методов, что результат может отсутствовать, и заставляет вызывающего обработать оба случая. Цель — уменьшить NullPointerException.*

- В чём разница orElse и orElseGet?

  → *orElse(value) — eager: значение вычисляется ВСЕГДА, даже если Optional не пустой. orElseGet(Supplier) — lazy: вызывается только если Optional пустой. Если default-значение получается дорогим вызовом — использовать orElseGet.*

```
Optional<User> user = repo.findById(id);

// ❌ Плохо: heavyDefault() вызывается всегда
User u1 = user.orElse(heavyDefault());

// ✅ Хорошо: heavyDefault() вызывается только при пустом Optional
User u2 = user.orElseGet(() -> heavyDefault());
```

Лямбды и функциональные интерфейсы

- Что такое функциональный интерфейс?

  → *Интерфейс с РОВНО ОДНИМ абстрактным методом (default и static не считаются). Можно реализовать лямбдой. @FunctionalInterface — необязательная аннотация для compile-time проверки.*

- Какие функциональные интерфейсы из java.util.function ты знаешь?

  → *Consumer<T> — принимает T, ничего не возвращает (T → void). Supplier<T> — без параметров, возвращает T (() → T). Predicate<T> — принимает T, возвращает boolean. Function<T,R> — T → R. BiFunction<T,U,R> — два параметра. UnaryOperator<T> = Function<T,T>. BinaryOperator<T> = BiFunction<T,T,T>.*

- Какие переменные можно захватывать в лямбде?

  → *Только final или effectively final (не меняются после присваивания). Поля класса можно использовать без ограничений. Локальные переменные — только если они эффективно финальные. Это нужно для гарантии корректности при асинхронном вызове.*

// JavaJub

Это только начало гайда.

Сбер · SberSeasons · Стажировка

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