Ответы нужно давать лаконичные, достаточно продемонстрировать понимание предмета.

Ф.И.О.	Салахеев Данис Фанисович
Java	
Что такое переопределение метода?	Реализация метода в интерфейсах или абстрактных классах или расширение функционала метода родительского класса в классах-наследниках. Внедрение полиморфизма через реализацию метода.
Какие бывают виды классов?	1) *внутренний (nested)* – собственностью внешнего класса, и создание производится **только** через экземпляр внешнего класса, inner имеет доступ ко всем его полям и методам. (нет статики) 2) *вложенный (static)* – считается *вполне* самостоятельным, однако создание объекта производится по имени внешнего класса, имеет доступ к его статическим полям и методам.(есть статика). 3) *локальный (local)* – может быть объявлен внутри фигурных скобок (методы, блоки, стат. блоки, т.д.). Их «скоуп» ограничен фигурными скобками. 4) *анонимный* – база для реализации лямбд создается при помощи ключ. слова *new,* отсутствует имя экземпляра и повторно использоваться не может: «сам себе класс и объект»
Как и зачем можно использовать модификатор final?	1) final поля и имена ссылок пишутся в *верх. регистре*: - константы (примитивы): для хранения постоянных значений: *final int G = 9.82;* 2) final ссылки на объект – служит для запрета на смену на другую ссылка объекта того е типа в памяти, однако его состояние можно изменить; *final String[] ANSWERS = {"Yeap", "Fine", "Nice"};* *ANSWERS[2] = "Nope!";* **Частный случай:** экземпляры final-классов: *final String RUS_CAPITAL = "Moscow";* *RUS_CAPITAL = "Berlin" // error* 3) методы – запрет на переопределение метода в классах-наследниках: для безопасности окончательно-реализованного метода: final int calculateAge() {….} 4) классы – запрет на создание наследников, как

	правило это: утилитарные классы, классы для создания неизменяемых (immutable) объектов
Какие есть варианты использования ключевого слова try?	1) Классический try-catch-finally – служит для обработки исключений, finally (необязательный) – для закрытия ресурсов и прочих завершающий операций; 2) try-with-resources – позволяет автоматическое закрытие ресурсов, которые реализуют интерфейс AutoCloseable, без ручного использования finally для этого.
Какие есть стандартные реализации интерфейса List и в каких ситуациях их нужно использовать?	1) ArrayList – динамический массив, в тех случаях когда требуется частый вызов по индексу, частое добавление в конец массива (при переполнении массива производится автоматическое расширение и копирование). 2) LinkedList – двунаправленный список (связанные узлы), быстрые операции при вставке и удалении в середину списка (при котором у узлов производится смена ссылок на добавляемый с список объект)
В чём основная идея стримов из пакета java.util.stream?	Реализация функционального программирования в java, гибкое взаимодействие с данными с применением конвейерной (inline) обработки функциями высшего порядка, тем самым конечный код получается компактным и часто легко читаемый.
Разное	
Каким критериям должна удовлетворять «хорошая» хэш-функция?	Такая хэш-функция должна рассчитать некоторую значение (int) за «оптимальное» время и «через себя» отобразить запрашиваемый объект с минимальными коллизиями в применяемых хэш-функции структурах данных.
В чём причина популярности и широкого распространения кодировки UTF-8?	Служит для унификации и корректного отображения всех букв, символов, иероглифов. Популярен из-за того, что имеет широкий диапазон значений, который содержит для кодировки всех символов мира, компактна, а также имеет обратную совместимость со старыми стандартами.
Сравните форматы XML и JSON. Когда какой использовать?	1) XML – более старый формат, в основе которого имеются тэги. Использование: если имеются некоторые ограничения и правила описания (через схемы и root-схемы) набора данных и конфигурация, которые можно описать как атрибутов, активно используется в некоторых фреймворках: Maven, Liquibase, Hibernate config (вариативно) и т.д.

	Имеет больший размер, чем JSON, что будет медленнее передача больших данных, плохо читаем, но в более старых проектах REST используется XML. 2) JSON – гибкий и свободный формат, более удобен для использования и передачи данных на фронтэнд, компактный и полностью совместим по структуре и синтаксису с JS-объектами. В настоящее время активно используется в REST API из-за простоты и меньшего объема и общение микросервисов, описание настроек в Docker, Kubernetesи т.д.
Опишите что будет происходить «под капотом» после ввода адреса сайта в браузере и нажатия Enter?	1) После этого DNS-сервер производит проверку на корректность домена, и ищет в регистре нужный нам первичный IP, по которому можно найти конечный сервер. 2. После получения IP-адреса, по умолчанию порты по протоколам, в данном случае это: http: 80 или https: 443 3. Далее задача состоит в том, что нам необходимо этому серверу IP:port сообщить чего мы от него хотим. 4. При обращении через браузер заполнение и передача данных производится неявно самим браузером, т.к. браузер при обращении на какой-либо ресурс применяет только один метод http. 5. Сам запрос может состоять из: - стартовая строка (Starting line), обязательна, она определяет тип сообщения и представляет собой следующий паттерн, например: > GET /person/1 HTTP/1.1 - заголовки (Headers); - тело (Body); - optional 6. После этого сервер получает наш запрос и начинает построчно считывать этот запрос. Первым он видит метод GET, потом URI ресурса, который он должен вернуть клиенту. При отсутствии URI (например /) будет возвращена index.html, хотя это зависит он настройки самого сервера. 8. Далее браузер узнает дополнительную информацию из заголовок. Например следующая является строка HOST: где определяется адрес сайта в виде домена. При этом, на одном IP-адресе могут находится несколько серверов с несколькими доменными именами. 9. В заголовках может содержаться другая информация необходимая серверу, >HOST: yandex.ru >User-Agent: Google Chrome >Accept: text/html

	>Accept-Encoding: gzip, deflate >Connection: close После заголовок пустая строка и пусток тело (в данном случае для GET-метода). 10. Далее по полученным данным от клиента, сервер начинает формировать ответ, первая строка состоит из пробелом: `Версия HTTP протокола 'пробел' версии протокола 'пробел' код состояния 'пробел' пояснение ответа`. Например: > HTTP/1.1 200 OK Далее следует заголовки в виде дополнительной информации: Например: Connection: keep-alive Content-Encoding: gzip Content-Type: application/json Date: Fri, 07 May 2021 09:42:35 GMT NEL: {"report_to":"cf-nel","max_age":604800} Далее следует пустая строка и тело ответа от сервера на запрошенный ресурс клиентом. В теле может быть любая информация, тоже html, xml, json, file, картинки. 11. Клиент (браузер) получает ответ и начинает потрочно считывать. Он выполняет полученные действия в зависимости от кода состояния ответа от сервера. Далее смотрит на заголовки, при необходимости получает cookies и производит дополнительную операцию. 12. Браузер доходит до пустой строки и начинает парсить и рендерить тело ответа в зависимости от Content-Type и Content-Encoding из заголовка ответа. Если в теле имеются другие ссылки, то браузер делает еще по http-запросу по этим ресурсам для получения дополнительных указаний для рендеринга исходного ответа от сервера.