1) **10 вопросов по абстрактным классам и интерфейсам с собеседований по Java**

https://javarush.ru/groups/posts/431-10-voprosov-po-abstraktnihm-klassam-i-interfeysam-s-sobesedovaniy-po-jazihku-java

https://javarush.ru/quests/lectures/questcore.level03.lecture07

https://javarush.ru/groups/posts/1985-raznica-mezhdu-abstraktnihmi-klassami-i-interfeysami

**Что такое JVM, JDK, JRE? В чем отличия?**

[Java Development Kit](https://docs.oracle.com/en/java/javase/13/docs/specs/man/index.html) — комплект разработчика приложений на языке Java. Он включает в себя [Java Development Tools](https://docs.oracle.com/javase/6/docs/technotes/tools/index.html) и среду выполнения Java — [JRE](https://www.oracle.com/technetwork/java/javase/jre-8-readme-2095710.html) (Java Runtime Environment).  
  
 Java development tools включают в себя около 40 различных тулов: javac (компилятор), java (лаунчер для приложений), javap (java class file disassembler), jdb (java debugger) и др.  
 Среда выполнения JRE — это пакет всего необходимого для запуска скомпилированной Java-программы. Включает в себя виртуальную машину [JVM](https://docs.oracle.com/javase/6/docs/technotes/guides/vm/index.html) и библиотеку классов Java — Java Class Library.  
 JVM — это программа, предназначенная для выполнения байт-кода. Первое преимущество JVM — это принцип “*Write once, run anywhere”*. Он означает, что приложение, написанное на Java, будет работать одинаково на всех платформах. Это является большим преимуществом JVM и самой Java.  
 До появления Java, многие компьютерные программы были написаны под определенные компьютерные системы, а предпочтение отдавалось ручному управлению памятью, как более эффективному и предсказуемому. Со второй половины 1990-х годов, после появления Java, автоматическое управление памятью стало общей практикой.  
 Существует множество реализаций JVM, как коммерческих, так и с открытым кодом. Одна из целей создания новых JVM — увеличение производительности для конкретной платформы. Каждая JVM пишется под платформу отдельно, при этом есть возможность написать ее так, чтобы она работала быстрее на конкретной платформе. Самая распространённая реализация JVM — это JVM Hotspot от [OpenJDK](https://openjdk.java.net/). Также есть реализации [IBM J9](https://www.ibm.com/support/knowledgecenter/SS4QMC_9.3.0/com.ibm.help.perf.manage.doc/c_FND_PM_IBMJ9JVM.html), [Excelsior JET](https://en.wikipedia.org/wiki/Excelsior_JET).

**Какие типы данных поддерживает Java? Что такое автоупаковка и автораспаковка?**

Java поддерживает примитивные типы (byte, boolean, char, short, int, long, dooble, float.) данных и ссылочные. Даже для примитивных типов данных есть классы-упаковки (int - Integer). Integar x = 10 - автоупаковка. То есть тут 10 упаковывается в объект и х становится этим самым объектом.

int n = x - тут уже это автораспаковка. Тут из объекта х будет вытащено 10 и добавлено в n.

**- Что такое String[] args в методе main()?**

Это аргументы, входные для нашего приложения. В edit configurations в строке Program Arguments эти аргументы можно прописать

- **Какими значения по умолчанию инициализируются поля примитивных типов данных?**

если они локальные (то есть объявлены на уровне методов без присвоения значений), то ничего в них нет.

**- Почему Java не поддерживает указатели, множественное наследование?**

Указателей нет, потому что мы не управляем напрямую памятью, тут есть сборщик мусора и это небезопасно.

Множественное наследование может внести неоднозначность, есть в двух классах, от которых мы бы наследовались, будет метод с одной и той же сигнатурой, но с совершенно разным поведением

**- Что такое циклы? Какие типы циклов вы знаете?**

for, foreach, do while, while.

**- Как можно организовать бесконечный цикл?**

while (true), for(; ;)

**- Что такое break, continue и label?**

break останавливает цикл (только один). Если циклы вложенные, то брейк остановит только внутренний, но есть еще такой вариант:

lab1: - это тот самый label

for (int i = 0; i < 5; i++) {

for (int j = 0; j < 5; j++) {

break lab1;

}

} в этом случае вы выйдем после брейк сразу из всех циклов

**- Что такое пакеты, зачем они нужны? Какие пакеты по умолчанию подключены?**

пакеты служат для группировки кода.

Пакеты помогают для управлениями уровнями доступа (default - можно обращаться внутри пакета, protected - внутри пакета и в наследнике).

Для урегулирования конфликтов с одинаковыми наименованиями классов. Импортировать можно и тот и тот.

По умолчанию работает пакет java.lang

**- Импортируются ли классы из подпакетов при импорте пакета?**

нет. Например, мы импортируем класс java.util\*. К классам, которые находятся внутри этого пакета, мы можем легко обращаться. Но, в этом пакете есть множество других подпакетов. Например concurrent. Без импорта, мы не сможем обратиться к классам, вложенным в пакет concurrent

**- Что такое статический импорт**

это ипорт статических методов и статических полей из любого класса.

Например, мы можем импортировать класс Math (import static java.lang.Math.\*). После этого нам будут доступны все статические методы и переменные. Пример: нам не надо будет писать Math.sin(), достаточно будет написать sin.

ООП

**Что такое классы/ объекты?**

Образно можно сказать, что классы - это некий чертеж, по которому мы можем делать объекты. А объекты - это уже конкретные реализации экземпляров этих классов

**Что такое инкапсуляция, наследование и полиморфизм?**

Тут все понятно

**Что такое абстрактный класс?**

По абстрактному классу нельзя создать объект класса. Абстрактные методы могут находится только в абстрактном классе.

теперь про отличия абстрактных классов от интерфейсов:

- в абстрактном классе могут быть поля / в интерфейсе их нет (там можно только глобальные статические константы объявлять) То есть если мы напишем, например, int i = 10 - это public static final int = 10.

- в абстрактном классе можно реализовывать нормальную логику методов с привязкой к полям / в интерфейсах можно делать дефолтные методы, но это не дает гибкости

- от интерфейсов можно наследоваться в любом количестве / от абстрактного класса можно только от одного, так как это все-таки класс

**Можно ли создавать абстрактные методы в обычном классе? Создавать обычное методы в абстрактном классе?**

Абстрактные методы в обычном классе создавать нельзя. Если мы так попробуем сделать, то придется класс сделать абстрактным. А вот обычные методы в абстрактном классе конечно же можно создавать

**Что такое интерфейс?**

В интерфейсы мы прописываем какие-то методы, которые класс, который имплементирует этот интерфейс, должен реализовать их. Например, если мы имплементируем интерфейс Comparable, то мы знаем, что объекты данного класса мы можем сортировать без проблем

**Бывают ли пустые интерфейсы?**

Да. Например, Serializable

**Можно ли создать объект Flyable f = new Flyable();? Flyable – интерфейс;**

нет. Вернее создастся анонимный внутренний класс, который имлпментит этот интерфейс и по нему будет создан объект

**Что означает ключевое слово static? Можно ли переопределить private или static-методы?**

Статик поля привязаны к классам, а обычные к объектам класса. Для работы со статическими полями и методами нам объекты класса не нужны. Мы к ним обращаемся через класс.

Нельзя переопределять private и static методы. Для переопределения нам нужно наследование. Private и static поля не переходят к наследнику.

**Можно ли получить доступ к нестатическому полю из статического метода?**

нельзя

**Что такое блок static { }?**

Это блок, который при загрузке класса, выполняется первым. Например, для выполнения какой-то предварительной настройки класса, либо проинициализировать все статические поля.

**Когда загружается класс в память?**

Если класс еще не был загружен память, то он подгрузится в следующих случаях:

- когда мы создаем объект этого класса

- когда мы обращаемся к статическому полю этого класса

- когда мы вызываем статический метод у этого класса

**Что такое cast?**

приведение объекта к определенному типу

**Что такое переопределение и перегрузка методов?**

Переопределение связано с наследованием, когда к классах-наследниках, нужно прописать свою реализацию методов. Перегрузка подразумевает написание методов с одним именем, но с разным набором входных элементов

**Что такое конструктор и перегрузка конструктора? Может ли класс иметь несколько конструкторов?**

Конструктор - это метод, который вызывается при создании объекта и служит для того, чтоб этот самый объект проинициализировать. Перегрузка конструкторов допустима.

**Когда вызывается конструктор?**

Один раз при создании экземпляра класса после инициализации статических переменных и статических блоков если такие имеются, потом вызывается конструктор и происходит сперва инициализации полей класса и блоков инициализации и только потом работа тела конструктора.

**Поддерживает ли Java множественное наследование?**

нет

**Какие модификаторы доступа вы знаете и сколько их?**

private внутри класса

default внутри класса + внутри пакета

protected внутри класса + внутри пакета + наследники

public везде

**Что означает ключевое слово this?**

ссылка на текущий объект, у которого будет выполняться метод

**Что означает ключевое слово super?**

ссылка на родителя

**Что означает ключевое слово final?**

если final подключени к методу, то его нельзя переопределить, если к классу, то он него нельзя наследоваться, если к переменной -то это константа

**Какой класс является базовым для всех классов?**

Object

**В чем разница между String, StringBuffer, StringBuilder?**

String - это имутабельная строка

если строка должна постоянно меняться, то лучше взять:

StringBuffer - у него есть внутренняя синхронизация, поэтому используется, когда 2 и более потоков с ним взаимодействуют

StringBuilder - не synchronized

**Может ли метод быть одновременно abstract и final?**

нет, потому что абстракт говорит, что его надо переопределить, а final наоборот, что его нельзя переопределять

Коллекции

**Что такое ArrayList и LinkedList, какая между ними разница?**

ArrayList - это динамический массив, который изменяется. Изначальная длина 10 ячеек (можно ее задать самому). Сам он не уменьшается, но можно это сделать с помощью кода. Работает только с ссылочными типами данных (в случае примитивов используем обёртки). Поиск элементов происходит за константное время (не важно к какому элементу обращаешься). Если мы добавляем элемент в начало, то сдвигается весь массив.

LinkedList - хранит объекты по связям. То есть объект знает кто следующий и кто предыдущий. Поиск по индексу не особо эффективный. Но если мы добавляем, например, в начало листа элементы, то этот массив будет работать быстрее других.

**Что такое Set, HashSet, LinkedHashSet, TreeSet, и в чем между ними разница?**

Set - это интерфейс, который обозначает множество уникальных элементов

HashSet - просто множество, хранимое в случайном порядке

LinkedHashSet - запоминает порядок добавление элементов

TreeSet - сортирует элементы

**Можно ли в TreeSet отдать список объектов, которые не реализуют интерфейс Comparable?**

Можно, но нужно указать компоратор, иначе словим exception.

HashMap - выдает записи "кое-как"

LinkedHashMap - выдает записи в порядке их добавления

TreeMap - выдает записи в порядке сортировки ключа

**В чем разница между Hashtable, HashMap, ConcurrentHashMap, Collection.synchronizedMap(...)?**

Hashtable - синхронизирован. Тут нельзя хранить ключи null.

HashMap - не синхронизирован. Можно хранить ключи null.

ConcurrentHashMap - лучший вариант многопоточного HashMap, потому что он блокриуется частично и дает паралллельный доступ на чтение

Collection.synchronizedMap - можно MAp завернуть в коллекцию, где все методы синхронизированы

**В чем разница между ArrayList и Vector?**

Вектор - это старый аналог ArrayList, в котором все методы синхронихированы

**Существуют ли какие-нибудь требованию к объектам, используемым в качестве ключа в HashMap?**

Они должны переопределить equals и hashcode. Equals и hashcode должный быть согласованы

**Что такое loadFactor и initialCapactity в HashMap?**

loadFactor - коэффициент заполненности. Например, когда в hashMap ёмкость заполнена на 75% - ее емкость вырастает

initialCapactity - начальная ёмкость Map. Она равна степени двойки

**Что такое коллизии и как их избежать?**

Коллизии - это когда несколько записей в HashMap попадают в один и тот же бакет. Чем меньше коллизий, тем HashMap будет работать быстрее.

**Что такое Iterator, чем он отличается от ListIterator?**

Iterator - это объект, который позволяет обойти все элементы коллекции или любого итерируемого набора данных

Iterator умеет "ходить" только слева-направо, а ListIterator может ходить в двух направлениях.

**Чем массив отличается от ArrayList?**

тут все понятно и так

**Что такое hashCode()?**

Это числовое значение, которое ставится в соответствии к объекту

Исключения

**Что означает каждое из ключевых слов: try, catch, finally, throw, throws?**

try - попытка выполнить код

catch - перехват исключения

finally - в любом случае выполняющийся кусок кода

throw - бросить exception

throws - пробросить exception

**Для чего в основном используется блок finally?**

В основном для закрытия ресурсов.

**Что такое try-with-resource?**

try c обязательным закрытием ресурсов. Экономит место в коде, вместо finally

**Покажите примерную схему дерева классов, наследующихся от Throwable;**

https://geekbrains.ru/posts/java\_exceptions

**Что такое checked и unchecked исключения?**

checked-исключения - это исключения, которые компилятор нас заставит обработать, либо пробросить наверх

unchecked - компилятору пофиг, что мы будем с ним делать

**Что происходит при необработанном исключении?**

Печатается стэктрейс и программа "падает"

**В чем разница между Exception и Error?**

Exception - это, как правило, некая внештатная ситуация, которую можно исправить налету

С Error уже ничего не сделать (например, память закончилась)

**Может ли исключение быть снова брошено после перехвата?**

Да. Мы можем перехватить и сразу бросить его дальше или бросить другое исключение

**Примеры подклассов RunTimeException**

ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException

МНОГОПОТОЧНОСТЬ

**В чем разница между процессом и потоком?**

Поток - исполнение в рамках одного приложения. Несколько потоков работают с общей памятью в рамках одного приложения

Процесс как правилоо работает в разных приложениях. У них разные адресные пространства и они друг с другом не взаимодействуют

**Какие существуют способы создания потоков (и сколько их)?**

Thread либо пул потоков

**Какой способ создания потоков предпочтителен?**

**Что такое Deadlock?**

Когда 2 (или больше) потока ждут завершения друг друга и из этого кольца не могут выйти

**В чем разница между notify() и notifyAll()?**

notify будит 1 уснувший поток, ожидающий по монитору, notifyAll все потоки, ожидающие по монитору.

**В чем разница между sleep() и wait()?**

sleep - поток засыпает на определенное количество времени, а wait - пока не выполнится notify. Но в wait можно так же указать таймаут

Так же поток в методе wait может быть разбужен другим потоком, а в sleep() нет. Метод wait() может быть объявлен только в synchronized блоке

**Можно ли перезапустить завершившийся поток?**

нет

**Какие способы синхронизации вы знаете?**

По монитору

**Зачем нужен Semaphore, ExecutorService?**

Семафор - управляет доступом к ресурсы посредством счетчика

ExecutorService – менеджер управления потоками, помогает в создании и завершении потоков, получение результата выполнения потоков, отслеживания хода выполнения задач.

**Что такое daemon thread?**

это поток, который работает, пока работает хотя бы 1 поток в

**Какие преимущества дает использование Lock по сравнению с synchronized blocks?**

В Lock можно устанавливать таймаут и проверять занят он или нет, позволяет более гибко управлять блокировками.

**JDBC**

**Что такое JDBC?**

Это набор интерфейсов для подключения к базам данных.

**Какова роль драйвера в JDBC?**

драйвер - это реализация вышеупомянутых интерфейсов

**Зачем нужен вызов метода Class.forName(…)?**

Class.forName - это обращение в java, чтобы она загрузила класс в память

**В чем разница между Statement и PreparedStatement?**

в Statement мы отдаем строку и на ходу компилируется запрос и отправляется

PreparedStatement предкомпилированный запрос. Он работает немного быстрее. Позволяет выполнять параметризированные запросы, так как он предкомпилированный то может быть эффективно использован несколько раз. Является лучшим выбором чем Statement более быстрый, позволяет писать более читаемый код с использованием геттеров и сеттеров, так же предотвращает SQL injection.

**Что такое Connection Poll?**

Database Connection Pool (dbcp) — это способ решения изложенной выше проблемы. Он подразумевает, что в нашем распоряжении имеется некоторый набор («пул») соединений к базе данных. Когда новый пользователь запрашивает доступ к БД, ему выдаётся уже открытое соединение из этого пула. Если все открытые соединения уже заняты, создаётся новое. Как только пользователь освобождает одно из уже существующих соединений, оно становится доступно для других пользователей. Если соединение долго не используется, оно закрывается.

**Что такое ResultSet?**

Это результат, получаемый в ответ на наш запрос в базу. Чтобы получить его нам нужно у **~~connection~~ Statement, PreparedStetement, CollbackStatement** выполнить execute.query

**Что такое обобщения?**

Используются когда класс должен работать с разными типами данных по одной и той же логике, позволяют избежать многих ошибок на этапе компиляции

**В чем разница между == и equals()?**

== - сравнивает объекты по ссылкам, а equals сравнивает по наполнению

**Что такое сериализация/десериализация?**

сериализация - процесс преобразования объектов в набор байтов

десериализация - восстановление объекта из набора байт

**Что за ключевые слова - transient и volatile?**

transient - ставится, когда мы не хотим какое то поле сериализовывать

volatile - запрет кэширования

**В чем разница между Serializable и Externalizable?**

Serializable делает автоматическую сериализацию и десериализацию

Externalizable - это интерфейс, который позволяет управлять процессом сериализации и десериализации

**Необходим ли установленный JDK для запуска Java-программ?**

нет

**Можно ли в интерфейсе объявить статический метод?**

**Да можно**

**Можно ли увеличить размер массива после его инициализации?**

Напрямую нельзя. Можно создать новый массив

**Что такое локальный класс?**

Это класс, который объявлен внутри метода

**Что такое сборщик мусора (Garbage Collector)?**

Это элемент, который убирает ненужные неиспользуемые объекты. За счет этого освобождается память

**Можно ли с помощью Arrays.sort() отсортировать массив объектов?**

Да, если объекты либо реализуют интерфейс comparable, либо если в метод сорт передать компоратор

**Что означает иммутабельность строк?**

Это значит, что мы не можем изменить строку

**Как создать собственный иммутабельный класс?**

Либо все поля сделать final, либо поля сделать поля не final, но поля не менять и не давать их менять другим классам

**Что делает метод toString()?**

преобразует объект к читаемому виду, то есть к строке

**В чем разница между набором классов Reader/Writer и InputStream/OutputStream?**

Reader/Writer - запись/чтение байтов и преобразование их к символам используется для текстовой информации

InputStream/OutputStream - запись/чтение по байтам

**Что такое области видимости?**

например, мы можем создать такой код:

{

int x = 10

}

х будет виден только в пределах этих скобок.