10 Вопросов, которые обязательно задают работодатели

Мы составили базу вопросов, которые задают на собеседовании. В ней более 500 вопросов, которые позволяют нашим студентам уверенно проходить собеседования и быстро устраиваться в IT компании.

1.Чем различаются JVM, JRE и JDK?

JVM, Java Virtual Machine (Виртуальная машина Java) — основная часть среды времени исполнения Java (JRE). Виртуальная машина Java исполняет байт-код Java, предварительно созданный из исходного текста Java-программы компилятором Java. JVM может также использоваться для выполнения программ, написанных на других языках программирования.

JRE, Java Runtime Environment (Среда времени выполнения Java) - минимально-необходимая реализация виртуальной машины для исполнения Java-приложений. Состоит из JVM и стандартного набора библиотек классов Java.

JDK, Java Development Kit (Комплект разработки на Java) - JRE и набор инструментов разработчика приложений на языке Java, включающий в себя компилятор Java, стандартные библиотеки классов Java, примеры, документацию, различные утилиты.

Коротко: JDK - среда для разработки программ на Java, включающая в себя JRE - среду для обеспечения запуска Java программ, которая в свою очередь содержит JVM - интерпретатор кода Java программ.

2. Какие существуют модификаторы доступа?

private (приватный): члены класса доступны только внутри класса. Для обозначения используется служебное слово `private`.

default, package-private, package level (доступ на уровне пакета): видимость класса/членов класса только внутри пакета. Является модификатором доступа по умолчанию - специальное

обозначение не требуется.

protected (защищённый): члены класса доступны внутри пакета и в наследниках. Для обозначения используется служебное слово `protected`.

public (публичный): класс/члены класса доступны всем. Для обозначения используется служебное слово `public`.

Последовательность модификаторов по возрастанию уровня закрытости: public, protected, default, private.

Во время наследования возможно изменения модификаторов доступа в сторону большей видимости (для поддержания соответствия принципу подстановки Барбары Лисков).

3. Что вы знаете о функции `main()`?

Метод `main()` — точка входа в программу. В приложении может быть несколько таких методов. Если метод отсутствует, то компиляция возможна, но при запуске будет получена ошибка _\`Error: Main method not found\`_.

```java
public static void main(String[] args) {}
```

4.Почему в некоторых интерфейсах вообще не определяют методов?

Это так называемые маркерные интерфейсы. Они просто указывают что класс относится к определенному типу. Примером может послужить интерфейс `Clonable`, который указывает на то, что класс поддерживает механизм клонирования.

5. Дайте определение понятию «интерфейс». Какие модификаторы по умолчанию имеют поля и методы интерфейсов?

Ключевое слово `interface` используется для создания полностью абстрактных классов. Основное предназначение интерфейса - определять каким образом мы можем использовать класс, который его реализует. Создатель интерфейса определяет имена методов, списки аргументов и типы возвращаемых значений, но не реализует их поведение. Все методы неявно объявляются как `public`.

Начиная с Java 8 в интерфейсах разрешается размещать реализацию методов по умолчанию `default` и статических `static` методов.

Интерфейс также может содержать и поля. В этом случае они автоматически являются публичными `public`, статическими `static` и неизменяемыми `final`.

6. Почему `char[]` предпочтительнее `String` для хранения пароля?

С момента создания строка остаётся в пуле, до тех пор пока не будет удалена сборщиком мусора. Поэтому, даже после окончания использования пароля, он некоторое время продолжает оставаться доступным в памяти и способа избежать этого не существует. Это представляет определённый риск для безопасности, поскольку кто-либо, имеющий доступ к памяти сможет найти пароль в виде текста.

В случае использования массива символов для хранения пароля имеется возможность очистить его сразу по окончанию работы с паролем, позволяя избежать риска безопасности, свойственного строке.

7. Каким образом передаются переменные в методы, по значению или по ссылке?

В Java параметры всегда передаются только по значению, что определяется как «скопировать значение и передать копию». С примитивами это будет копия содержимого. Со ссылками - тоже копия содержимого, т.е. копия ссылки. При этом внутренние члены ссылочных типов через такую копию изменить возможно, а вот саму ссылку, указывающую на экземпляр - нет.

8.Расскажите про вложенные классы. В каких случаях они применяются?

Класс называется вложенным (Nested class), если он определен внутри другого класса. Вложенный класс должен создаваться только для того, чтобы обслуживать обрамляющий его класс. Если вложенный класс оказывается полезен в каком-либо ином контексте, он должен стать классом верхнего уровня. Вложенные классы имеют доступ ко всем (в том числе приватным) полям и методам внешнего класса, но не наоборот. Из-за этого разрешения использование вложенных классов приводит к некоторому нарушению инкапсуляции.

Существуют четыре категории вложенных классов:

+ Static nested class (Статический вложенный класс);
+ Member inner class (Простой внутренний класс);
+ Local inner class (Локальный класс);
+ Anonymous inner class (Анонимный класс).

Такие категории классов, за исключением первого, также называют внутренними (Inner class). Внутренние классы ассоциируются не с внешним классом, а с экземпляром внешнего.

Каждая из категорий имеет рекомендации по своему применению.

9. Что такое «анонимные классы»? Где они применяются?

Это вложенный локальный класс без имени, который разрешено декларировать в любом месте обрамляющего класса, разрешающем размещение выражений. Создание экземпляра анонимного класса происходит одновременно с его объявлением. В зависимости от местоположения анонимный класс ведет себя как статический либо как нестатический вложенный класс.

Анонимные классы имеют несколько ограничений:

+ Их использование разрешено только в одном месте программы - месте его создания;
+ Применение возможно только, если после порождения экземпляра нет необходимости на него ссылаться;
+ Реализует лишь методы своего интерфейса или суперкласса, т.е. не может объявлять каких-либо новых методов, так как для доступа к ним нет поименованного типа.

Анонимные классы обычно применяются для:

+ создания объекта функции (function object), например реализация интерфейса `Comparator`;
+ создания объекта процесса (process object), такого как экземпляры классов `Thread`, `Runnable` и подобных;
+ в статическом методе генерации;
+ инициализации открытого статического поля `final`, которое соответствует сложному перечислению типов, когда для каждого экземпляра в перечислении требуется отдельный подкласс.

10.Для чего нужен сборщик мусора?

Сборщик мусора (Garbage Collector) должен делать всего две вещи:

+ Находить мусор - неиспользуемые объекты. (Объект считается неиспользуемым, если ни одна из сущностей в коде, выполняемом в данный момент, не содержит ссылок на него, либо цепочка ссылок, которая могла бы связать объект с некоторой сущностью приложения, обрывается);
+ Освобождать память от мусора.

Существует два подхода к обнаружению мусора:

Reference counting и Tracing

Reference counting (подсчёт ссылок). Суть этого подхода состоит в том, что каждый объект имеет счетчик. Счетчик хранит информацию о том, сколько ссылок указывает на объект. Когда ссылка уничтожается, счетчик уменьшается. Если значение счетчика равно нулю, - объект можно считать мусором. Главным минусом такого подхода является сложность обеспечения точности счетчика. Также при таком подходе сложно выявлять циклические зависимости (когда два объекта указывают друг на друга, но ни один живой объект на них не ссылается), что приводит к утечкам памяти.

Главная идея подхода Tracing (трассировка) состоит в утверждении, что живыми могут считаться только те объекты, до которых мы можем добраться из корневых точек (GC Root) и те объекты, которые доступны с живого объекта. Всё остальное - мусор.

Существует 4 типа корневых точки:

+ Локальные переменные и параметры методов;
+ Потоки;
+ Статические переменные;
+ Ссылки из JNI.

Самое простое java приложение будет иметь корневые точки:

+ Локальные переменные внутри `main()` метода и параметры `main()` метода;
+ Поток который выполняет `main()`;
+ Статические переменные класса, внутри которого находится `main()` метод.

Таким образом, если мы представим все объекты и ссылки между ними как дерево, то нам нужно будет пройти с корневых узлов (точек) по всем рёбрам. При этом узлы, до которых мы сможем добраться - не мусор, все остальные - мусор. При таком подходе циклические зависимости легко выявляются. HotSpot VM использует именно такой подход.

Для очистки памяти от мусора существуют два основных метода:

Copying collectors Mark-and-sweep

При copying collectors подходе память делится на две части «from-space» и «to-space», при этом сам принцип работы такой:

+ Объекты создаются в «from-space»;
+ Когда «from-space» заполняется, приложение приостанавливается;
+ Запускается сборщик мусора. Находятся живые объекты в «from-space» и копируются в «to-space»;
+ Когда все объекты скопированы «from-space» полностью очищается;
+ «to-space» и «from-space» меняются местами.

Алгоритм работы mark-and-sweep можно описать так:

+ Объекты создаются в памяти;
+ В момент, когда нужно запустить сборщик мусора приложение приостанавливается;
+ Сборщик проходится по дереву объектов, помечая живые объекты;
+ Сборщик проходится по всей памяти, находя все не отмеченные куски памяти и сохраняя их в «free list»;
+ Когда новые объекты начинают создаваться они создаются в памяти доступной во «free list».

Сборщики мусора HotSpot VM используют комбинированный подход Generational Garbage Collection, который позволяет использовать разные алгоритмы для разных этапов сборки мусора. Этот подход опирается на том, что:

+ большинство создаваемых объектов быстро становятся мусором;
+ существует мало связей между объектами, которые были созданы в прошлом и только что созданными объектами.

Остальные ответы на вопросы доступны по ссылке после авторизации

БАЗА ВОПРОСОВ НА СОБЕСЕДОВАНИИ

Это не просто подготовка к собеседованию!
Это повышение квалификации, что позволяет стать специалистом и претендовать на достойные вакансии

С уважением, школа JavaGuru.