1. Что такое Throwable?

Throwable - это суперкласс в Java для всех ошибок и исключений. В иерархии Java любой объект, который может быть выброшен (thrown) с помощью оператора throw, должен быть экземпляром класса Throwable или одного из его подклассов.

2. Что такое Error?

Error - это подкласс Throwable, который указывает на серьёзные проблемы, с которыми приложение, как правило, не должно пытаться справиться. Ошибки обычно связаны с ситуациями, выходящими за пределы контроля программы, например, OutOfMemoryError, когда виртуальная машина Java (JVM) не может выделить дополнительную память.

3. В чем отличие checked и unchecked? Можно ли поймать unchecked exception?

Checked исключения - это исключения, на которые компилятор требует явно реагировать, либо с помощью блока try-catch, либо с помощью указания в сигнатуре метода с throws. Unchecked исключения (все наследники RuntimeException) не требуют обязательной обработки и могут быть пойманы, но не требуют от разработчика специального указания на их обработку.

4. В чем разница throw и throws? Где какой используется?

throw используется для явного выброса исключения, а throws указывается в сигнатуре метода для объявления того, что метод может выбросить исключение. Вы используете throw внутри метода, а throws - в определении метода.

5. Что можно поймать в catch? В каком порядке отрабатываются catch в try/catch/finally?

В блоке catch можно поймать любой объект, который является экземпляром Throwable или его подклассов. Блоки catch отрабатывают в порядке их следования в коде, поэтому исключение будет обработано первым подходящим блоком catch, который его может обработать. Блок finally исполняется после блоков try/catch, независимо от того, было ли выброшено исключение.

6. Для чего нужен finally? Всегда ли отработает finally?

finally используется для выполнения важного кода, такого как закрытие ресурсов, который должен выполняться независимо от того, произошло исключение или нет. Блок finally выполняется всегда, за исключением случаев, когда JVM выходит из строя (например, System.exit() или критическая ошибка машины).

7. В чем разница между final/finally/finalize?

final - это модификатор, который можно использовать для классов, методов и переменных и который указывает, что что-то нельзя изменить (класс нельзя наследовать, метод нельзя переопределить, переменная после присвоения не может изменить своё значение).

finally - это блок, который содержит код, который должен быть выполнен после блоков try/catch.

finalize - это метод, который вызывается перед сборкой мусора для выполнения последних операций очистки объекта перед его уничтожением.

8. Если поймать OutOfMemoryError в catch и запустить System.gc(), высвободит ли это память и поможет ли программе?

Поймать OutOfMemoryError возможно, но это не означает, что мы сможем восстановить программу до полностью работоспособного состояния, поскольку ошибка OutOfMemoryError указывает на то, что JVM не смогла выделить память для дальнейших нужд. Вызов System.gc() просит JVM выполнить сборку мусора, но не гарантирует освобождение достаточного количества памяти или немедленное её освобождение. К тому же, в момент возникновения OutOfMemoryError система может быть уже слишком нестабильна, чтобы продолжить нормальную работу.

9. Можно положить Exception внутрь Exception?

Да, в Java исключения могут быть вложенными. Мы можем "завернуть" одно исключение в другое, используя конструктор исключения, который принимает другое исключение в качестве аргумента. Это называется "цепочкой исключений" (exception chaining) и используется для сохранения информации о первопричине исключения при его перебрасывании на более высокий уровень. Пример кода:

```
try {

} catch (IOException e) {

   throw new RuntimeException("Произошла ошибка ввода-вывода", e);

}
```

Здесь IOException "заворачивается" в RuntimeException, при этом исходное исключение сохраняется в качестве его причины (cause).