1. **Дайте определение массиву. Как осуществляется индексация элементов массива. Как необходимо обращаться к i-му элементу массива?**

Массив — это структура данных, в которой хранятся элементы одного типа. Его можно представить, как набор пронумерованных ячеек, в каждую из которых можно поместить какие-то данные (один элемент данных в одну ячейку). Доступ к конкретной ячейке осуществляется через её номер. Номер элемента в массиве также называют **индексом**.

Массивы в Java всегда **однородны**.

Индексация осуществляется с 0-го номера.

Обратиться к i-тому массиву: array[i].method(); или array[i] = ?;

При создании массива с помощью new, в него будут записаны значения по умолчанию.

1. **Приведите способы объявления и инициализации одномерных и двумерных массивов примитивных и ссылочных типов. Укажите разницу, между массивами примитивных и ссылочных типов.**

Объявляются двумя способами:

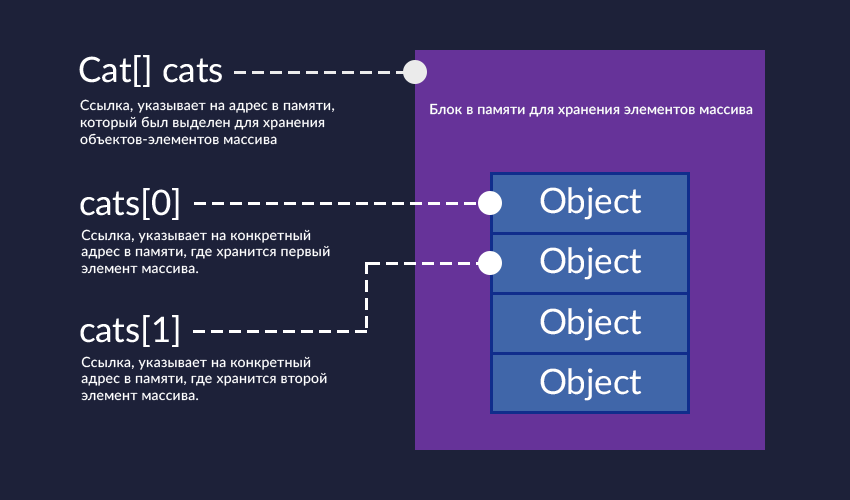
**Int[] array**; или **int array[];** Первый способ лучше, второй – наследие С++.

Инициализация массива — это заполнение его конкретными данными (не по умолчанию).

Инициализация двумя способами: через new и после указания значения каждой «ячейке», или через «{ }» скобочки.

**Массивы хранят значения примитивов или ссылки на объекты (не сами объекты).**

Элементы массива в памяти размещаются в едином блоке. Это сделано для более эффективного и быстрого доступа к ним. Таким образом, ссылка cats указывает на блок в памяти, где хранятся все объекты — элементы массива. А cats[0] — на конкретный адрес внутри этого блока.



**3. Объясните, что представляет собой двумерных массив в Java, что такое “рваный массив”. Как узнать количество строк и количество элементов в каждой строке для “рваного” массива?**

Двумерный массив в Java – это массив массивов. Рваный массив – массив с массивами разной длинны.

Узнать количество строк – array.length();

Узнать количество элементов в строке – array[i].length();

1. **Объясните ситуации, когда в java-коде могут возникнуть следующие исключительные ситуации java.lang.ArrayIndexOutOfBoundsException и java.lang.ArrayStoreException.**

AIOBE – может возникнуть из-за выхода за пределы массива, например при длине 10 пытаться взять длину от 11-го элемента массива.

**ArrayStoreException -** возникает при несоответствии типов в массивах.

1. **Что такое динамические структуры данных? Какие динамические структуры данных вы знаете? Приведите пример реализации любой динамической структуры ‘с нуля’.**

Что такое структура данных? Структура данных — это контейнер, который хранит данные в определенном макете. Этот «макет» позволяет структуре данных быть эффективной в некоторых операциях и неэффективной в других.

**Линейные**, элементы образуют последовательность или линейный список, обход узлов линеен. Примеры: Массивы. Связанный список, стеки и очереди.  
  
**Нелинейные**, если обход узлов нелинейный, а данные не последовательны. Пример: граф и деревья.

## Основные структуры данных.

1. Массивы
2. Стеки (LIFO)
3. Очереди (FIFO)
4. Связанные списки (хранят ссылки на соседей, или только на след.эдемент) Пример LinkedList.
5. Графы
6. Деревья
7. Префиксные деревья (т9)
8. Хэш таблиц (ключ значения)

Пример реализации самый популярный ArrayList. Создан на основе массива. По умолчанию размер = 10.

Добавление: происходит проверка на то хватает ли места. Если места хватает, то добавляется новый элемент на след. Места после последнего существующего (те добавка в конец).

Если вставим в середину, все что идут за ним сдвинутся на 1 элемент.

Если места не хватает создается новый массив в 1.5 + 1 раза больше чем предыдущий. На пред. Не остается ссылок и он соберется GC.

При удалении все сдвигается влево.

1. **Дайте определение таким понятиям как ”класс” и “объект”. Приведите примеры объявления класса и создания объекта класса. Какие спецификаторы можно использовать при объявлении класса.**

Private static class Cat {

}

Объявление класса.

Cat someX – создание переменной someX типа Cat

… = new Cat(); - создание объекта класса.

Спецификаторы: модификаторы доступа, статичный не статичный, имплементация и наследование.

1. **Как вы определяете, какие поля и методы необходимо определить в классе, приведите пример. Какие спецификаторы можно использовать с полями, а какие с методами (и что они значат)?**

Определим логически исходя из потребостей.

|  |  |  |  |
| --- | --- | --- | --- |
| **Модификатор** | **Класс** | **Поле класса** | **Метод** |
| **Доступа** | **+** | **+** | **+** |
| **Static** | **+** | **+** | **+** |
| **final** | **+** | **+** | **+** |
| **synchronized** | **-** | **-** | **+** |
| **transient** | **-** | **+** | **-** |
| **volatile** | **-** | **+** | **-** |
| **Abstract** | **+** | **-** | **+** |

1. **Что такое конструктор? Как вы отличите конструктор от любого другого метода? Сколько конструкторов может быть в классе? Что такое конструктор по умолчанию, может ли в классе совсем не быть конструкторов? Объясните, что делает оператор this() в конструкторе?**

Конструктор - это специальный метод, который вызывается при создании нового объекта.

Конструкторов может быть сколько угодно. К-р по умолчанию = конструктору без параметров. Если переопределен другой конструктор, то по умолчанию не работает, нужно указывать явно. Не быть конструктора не может, но может быть приватный (паттерн синглтон)

This –означает обращаение к этому объекту, используется в след случаях:

Public User(String name, int age) {

This.name = name;

This.age = age;

}

**9. Приведите правила, которым должен следовать компонент java-bean.**

*Bean по-английски боб. А Java Beans – это, стало быть, кофейные бобы (Java – сорт кофе). Такой айтишный юмор*

Правила описания гласят:

* Класс должен иметь [конструктор](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D0%BE%D1%80_(%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) без параметров, с модификатором доступа public. Такой конструктор позволяет инструментам создать объект без дополнительных сложностей с параметрами.
* Свойства класса должны быть доступны через get, set и другие методы (так называемые [методы доступа](https://ru.wikipedia.org/wiki/Accessor)), которые должны подчиняться стандартному соглашению об именах. Это легко позволяет инструментам автоматически определять и обновлять содержание bean’ов. Многие инструменты даже имеют специализированные редакторы для различных типов свойств.
* Класс должен быть [сериализуем](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B8%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F" \l "Java" \o "Сериализация). Это даёт возможность надёжно сохранять, хранить и восстанавливать состояние bean независимым от платформы и виртуальной машины способом.
* Класс должен иметь переопределенные методы equals(), hashCode() и toString().