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

Массив – это набор однотипных переменных, для обращения к которым используется общее имя. Индексация элементов массива начинается с нулевого элемента. Обратиться к i-му элементу массива можно указав имя массива[i].

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

Инициализация одномерного массива: `int[] dayOfMonth = new int[31];`

Инициализация двумерного массива: `int[][] days = new int[12][31];`

**3. Объясните, что значит клонирование массива, как в Java можно клонировать массив, в чем состоит разница в клонировании массивов примитивных и ссылочных типов.**

Клонировать массив означает создать точную его копию.

**Клонировать массив можно при помощи:**

Object.clone() — можно использовать этот метод для **полного копирования** массива. Этот метод не подойдет вам, если вы хотите частично скопировать массив.

System.arraycopy() — лучший способ сделать частичную копию массива.

Arrays.copyOf() — если вы хотите скопировать несколько первых элементов массива или сделать полную копию массива, вы можете использовать этот метод.

Arrays.copyOfRange() — еще один полезный метод частичного копирования массива.

Object.clone() – реализует только поверхностное копирование. Это означает, что копируются только примитивные типы.
Для того, чтобы произвести глубокое клонирование, необходимо в классе переопределить метод clone() и в нём произвести клонирование ссылочных типов.

**4. 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, по сути, представляет собой массив массивов. Рванный массив – это массив с разным количеством элементов для второго измерения (строк). Узнать количество строк и элементов можно при помощи поля length.

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

Ошибка java.lang.ArrayIndexOutOfBoundsException возникает при обращении к отрицательному индексу массива (которого конечно же попросту не существует) или к индексу за пределами длины массива.
Если попытаться записать в ячейку массива ссылку на объект неправильного типа, возникнет исключение ArrayStoreException.

**6. Объясните, зачем при кодировании разделять решаемую задачу на методы. Поясните, как вы понимаете выражение:“Один метод не должен выполнять две задачи”.**

Разделять решаемую задачу на методы необходимо для упрощения процесса разработки, а также повторного использования кода.

Один метод должен определять какое-то поведение объекта. Он должен выполнять одну задачу, что значительно упростит возможность расширения программы и повторного использования кода.

**7. Объясните, как в Java передаются параметры в методы, в чем особенность передачи в метод значения примитивного типа, а в чем ссылочного.**

Java is always pass-by-value - Java передает всё по значению. Для примитивов передаются копии значений. Для объектов в качестве значений выступают ссылки на эти объекты.

**8. Объясните, как в метод передать массив. И как массив вернуть из метода. Можно ли в методе изменить размер переданного массива.**

Чтобы передать массив в метод, его нужно указать в качестве аргумента. А чтобы вернуть – в качестве возвращаемого типа.

```
private int[] sum(int[] array){

}
```

В методе можно лишь создать новый массив с нужным размером и присвоить новую ссылку переменной массива.

**9. Поясните, что означает выражение ‘вернуть значение из метода’. Как можно вернуть значение из метода. Есть ли разница при возврате значений примитивного и ссылочного типов.**

Чтобы вернуть значение из метода используется ключевое слово return;

```
private int[] sum(int[] array){
  return new int[array.length + 1];
}
```

При возврате значений ссылочного типа мы можем вернуть вместо класса родителя класс наследник. В этом отличие от возврата примитивного типа.

**10. Перечислите известные вам алгоритмы сортировки значений, приведите код, реализующий эти алгоритмы.**

Пузырьковая сортировка

```
for (int i = array.length - 1; i > 0; i--) {

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

        if (array[j] > array[j + 1]) {

            int tmp = array[j];
            array[j] = array[j + 1];
            array[j + 1] = tmp;
        }
    }
}
```

Сортировка слиянием

```
private static void sort(int[] array) {

    if (array.length < 2) {
        return;
    }

    int mid = array.length / 2;

    int[] leftHalf = new int[mid];
    int[] rightHalf = new int[array.length - mid];

    for (int i = 0; i < mid; i++) {
        leftHalf[i] = array[i];
    }

    for (int i = mid; i < array.length; i++) {
        rightHalf[i - mid] = array[i];
    }

    sort(leftHalf);
    sort(rightHalf);

    merge(array, leftHalf, rightHalf);

}

private static void merge(int[] array, int[] leftHalf, int[] rightHalf) {
    int leftIndex = 0, rightIndex = 0, index = 0;

    while (leftIndex < leftHalf.length && rightIndex < rightHalf.length) {

        if (leftHalf[leftIndex] <= rightHalf[rightIndex]) {
            array[index++] = leftHalf[leftIndex++];
        } else {
            array[index++] = rightHalf[rightIndex++];
        }
    }

    while (leftIndex < leftHalf.length) {
        array[index++] = leftHalf[leftIndex++];
    }

    while (rightIndex < rightHalf.length) {
        array[index++] = rightHalf[rightIndex++];
```

```
    }

}
```