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

Массивы в Java – это структура данных, которая хранит упорядоченные коллекции фиксированного размера элементов нужного типа. Элементы массива доступны через индекс. Отсчет индексов ведется от 0. Обращение: array[index].

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

Объявление и инициализация:

```
for (int i = 0; i < size; i++) {
  array[i] = number;
}

for (int i = 0; i < size1; i++) {
  for (int j = 0; j < size2; j++) {
    array[i][j] = number;
}

int array[] = new int[size];
int array[][] = new int[size1][size2];
```

Разница заключается в том, что массив ссылочных хранит ссылки на эти самые объекты в памяти

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

Массив – это та же ссылка, просто склонировать не получится, можно поэлементно для примитивных типов, а для ссылочных типов нужно клонировать еще и сам объект ссылки.

Object.clone()
Arrays.copyOf ()
System.arraycopy ()
Arrays.copyOfRange ()

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?

Массив, содержащий в себе массивы с разным количеством элементов.

.length

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

ArrayIndexOutOfBoundsException – это исключение, появляющееся во время выполнения. Оно возникает тогда, когда мы пытаемся обратиться к элементу массива по отрицательному или превышающему размер массива индексу.

Исключение ArrayStoreException

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”.

Чем проще кусок выполняемой задачи, тем проще понимание и чтение все программы в целом. Метод должен делать что-то одно, либо должен быть разделен не 2 метода.

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

В Java параметры в методы передаются по значению. Но при передаче примитивного типа мы не можем испортить исходное значение, а при передаче объекта мы по значению ссылки и если объект изменяем можем его изменить.

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

```
int[] function(int[] array) {
  return array;
}
```

Длинна массивов неизмена. Можно создать новый нужной длины и скопировть в него старый массив.

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.

Функция, возвращающая значение:

```
int function() {
  return 42;
}
```

Разницы нет, возврат происходит по значению.

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

Быстрая сортировка:

```
public static void quickSort(int[] array, int low, int high) {
        if (array.length == 0)
            return;//завершить выполнение если длина массива равна 0
        if (low >= high)
            return;//завершить выполнение если уже нечего делить
        // выбрать опорный элемент
        int middle = low + (high - low) / 2;
        int opora = array[middle];
        // разделить на подмассивы, который больше и меньше опорного элемента
        int i = low, j = high;
        while (i <= j) {
            while (array[i] < opora) {
                i++;
            }
            while (array[j] > opora) {
                j--;
            }
            if (i <= j) {//меняем местами
                int temp = array[i];
                array[i] = array[j];
                array[j] = temp;
                i++;
                j--;
            }
        }
        // вызов рекурсии для сортировки левой и правой части
        if (low < j)
            quickSort(array, low, j);
        if (high > i)
            quickSort(array, i, high);
    }
```

Вставками:

```
public static void insertionSort(int[] array) {
    for (int i = 1; i < array.length; i++) {
        int current = array[i];
        int j = i - 1;
        while(j >= 0 && current < array[j]) {
            array[j+1] = array[j];
            j--;
        }
        array[j+1] = current;
    }       }
```

Пузырьком:

```
public static void bubbleSort(int[] array) {
    boolean sorted = false;
    int temp;
    while(!sorted) {
        sorted = true;
        for (int i = 0; i < array.length - 1; i++) {
            if (array[i] > array[i+1]) {
                temp = array[i];
                array[i] = array[i+1];
                array[i+1] = temp;
                sorted = false;
            }
        }
    }
}
```

Слиянием:

```
public static void mergeSort(int[] array, int left, int right) {
    if (right <= left) return;
    int mid = (left+right)/2;
    mergeSort(array, left, mid);
    mergeSort(array, mid+1, right);
    merge(array, left, mid, right);
}
public static void merge(int[] array, int low, int mid, int high) {
    int leftArray[] = new int[mid - low + 1];
    int rightArray[] = new int[high - mid];
    for (int i = 0; i < leftArray.length; i++)
        leftArray[i] = array[low + i];
    for (int i = 0; i < rightArray.length; i++)
        rightArray[i] = array[mid + i + 1];
    int leftIndex = 0;
    int rightIndex = 0;
    for (int i = low; i < high + 1; i++) {
        if (leftIndex < leftArray.length && rightIndex < rightArray.length) {
            if (leftArray[leftIndex] < rightArray[rightIndex]) {
               array[i] = leftArray[leftIndex];
               leftIndex++;
            } else {
                array[i] = rightArray[rightIndex];
                rightIndex++;
            }
        } else if (leftIndex < leftArray.length) {
            array[i] = leftArray[leftIndex];
            leftIndex++;
```

```
        } else if (rightIndex < rightArray.length) {
            array[i] = rightArray[rightIndex];
            rightIndex++;
        }
    }
}
```

Сортировка выбором:

```
int[] array = {10, 2, 10, 3, 1, 2, 5};
System.out.println(Arrays.toString(array));
for (int left = 0; left < array.length; left++) {
            int minInd = left;
            for (int i = left; i < array.length; i++) {
                        if (array[i] < array[minInd]) {
                                    minInd = i;
                        }
             }
            swap(array, left, minInd);
}
```

Сортировка Шелла:

```
public void sort (int[] arr)
    {
        for (int inc = arr.length / 2; inc >= 1; inc = inc / 2)
            for (int step = 0; step < inc; step++)
                insertionSort (arr, step, inc);
    }
    private void insertionSort (int[] arr, int start, int inc)
    {
        int tmp;
        for (int i = start; i < arr.length - 1; i += inc)
            for (int j = Math.min(i+inc, arr.length-1); j-inc >= 0; j = j-inc)
                if (arr[j - inc] > arr[j])
                {
                    tmp = arr[j];
                    arr[j] = arr[j-inc];
                    arr[j-inc] = tmp;
                }
                else break;
    }
```