Algorithmization

1. Массив — это структура данных, хранящая набор значений (элементов массива). Индексация элементов массива начинается с 0. Обращаться к I элементу mas[i-1] -1 потому что отчет идет с 0. Размер массива может быть фиксированный, например, mas[12]. Массив в Джава однороден и может хранить значения только одного с ним типа.

2. Пример объявления одномерного массива int[] twoDimArray квадратные скобки могут быть около типа массива или около названия int twoDimArray[]. Сколько мерный массив решается количеством []. Инициализация может быть несколькими способами, например, {11,12,12} или int[][] twoDimArray = new int[3][4]; String[] seasons = {"Winter", "Spring", "Summer", "Autumn"}; пример объявления ссылочного типа данных, как видим, разницы нет.

3. Клонирование массива означает вызов переопределеного метода clone(). В java проводится поэлементное копирование. Если клонировать массив ссылочных типов, то объект, который скопировал массив, будет иметь те же ссылки, а примитивный тип нет.

4. Двумерный массив в Джава — это массив массивов, то есть в каждой ячейке блоков первого уровня массива находится массив, например, [1][5] мы обратились к пятому элементу первого массива. Рваный массив — это такой двумерный массив, когда в количество массивов в блоках первого уровня может разнится, например, в [1][2] в первой ячейке только 3 элемента может быть, а вот в [2][5] этого же массива как мы видим 5. array[0].lenght при помощи этой команды можно узнать сколько элементов в данной ячейке.

5. Ошибка java.lang.ArrayIndexOutOfBoundsException возникает в том случае, если мы выходим за пределы массива (пытаемся обратится к элементу, которого нет). java.lang.ArrayStoreException данная ошибка возникает в случае если мы пытаемся добавить в массив элемент не того типа, для которого он создан (массив интов, а мы пытаемся добавить float).

6. Улучшает читабельность кода, его гибкость, появляется возможность разделения задач. Действия слишком различны по природе, которые не могут быть совмещенными (например: сравнение и печатать).

7. Параметры в методы передаются по значению. Для примитивных типов — вы передаете копию текущего значения, для ссылок на объекты — вы передаете копию ссылки (дистанционного управления). Никогда не передается объект. Все объекты хранятся в куче. Всегда.

8. В java массив считается объектов, следовательно передается значение ссылки. Для передачи массива нужно его указать в принимаемых

значениях метода, а для его возврата, нужно указать тип возвращаемого значения и использовать оператор return. Да, так как вы работаете со значением ссылки, а не самой ссылкой.

9. Это означает, что в сигнатуре метода есть тип возвращаемых данных и оператор return. Вернуть значение можно при помощи оператора return и типа, указанного в методе. Всегда возвращает ссылку (если это не примитивный тип).

10. Сортировка пузырьком

```
int[] array = {10, 2, 10, 3, 1, 2, 5};
for (int i = 1; i < array.length; i++) {
    if (array[i] < array[i - 1]) {
        swap(array, i, i-1);
    }
}

private void swap(int[] array, int ind1, int ind2) {
   int tmp = array[ind1];
   array[ind1] = array[ind2];
   array[ind2] = tmp;
}
```

Сортировка выбором

```
int[] array = {10, 2, 10, 3, 1, 2, 5};
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 вставками

```
int[] array = {10, 2, 10, 3, 1, 2, 5};
for (int left = 0; left < array.length; left++) {
    // Вытаскиваем значение элемента
```

```
    int value = array[left];
    // Перемещаемся по элементам, которые перед вытащенным элементом
    int i = left - 1;
    for (; i >= 0; i--) {
      // Если вытащили значение меньшее — передвигаем больший элемент дальше
      if (value < array[i]) {
            array[i + 1] = array[i];
      } else {
            // Если вытащенный элемент больше — останавливаемся
            break;
      }
    }
    // В освободившееся место вставляем вытащенное значение
    array[i + 1] = value;
}
```

Сортировка Шелла

```
int[] array = {10, 2, 10, 3, 1, 2, 5};
// Высчитываем промежуток между проверяемыми элементами
int gap = array.length / 2;
// Пока разница между элементами есть
while (gap >= 1) {
   for (int right = 0; right < array.length; right++) {
      // Смещаем правый указатель, пока не сможем найти такой, что
      // между ним и элементом до него не будет нужного промежутка
      for (int c = right - gap; c >= 0; c -= gap) {
         if (array[c] > array[c + gap]) {
            swap(array, c, c + gap);
         }
      }
   }
   // Пересчитываем разрыв
   gap = gap / 2;
}
```