<table>
<tr><td>Swap no temp- slower as JVM has to create the variable XOR</td><td>

```
x = x ^ y;
y = x ^ y;
x = x ^ y;
```

</td></tr>
<tr><td>Swap no temp better using addition and subtraction.</td><td>

```
a = 10, b = 7

a = a - b ; // a = 3
b = b + a ; // b = 10
a = b - a ; // a = 7

 JVM wouldn't need to create extra temporary variables to store
intermediate parts of expressions.
```

</td></tr>
<tr><td>Frequency and Rotation</td><td>Collections.rotate() and Collections.frequency.</td></tr>
<tr><td>GCD</td><td>BigInteger gcd(BigInteger val)</td></tr>
<tr><td>Merge Two Arrays</td><td>System.arraycopy(a, 0, c, 0, aLen);</td></tr>
<tr><td>Concat two arrays</td><td>

```
IntStream a= Arrays.stream(arr);
IntStream b= Arrays.stream(numbers);
IntStream c = IntStream.concat(a, b);
```

</td></tr>
<tr><td>Rotate array</td><td>
1. Args int nums[] and k rotations
2. Create temp array-a
3. Iterate
4. temp[i+k%original.legnth] = nums[i]

```
int [] temp = null;


for(int i =0; i<arr.length;i++){

      temp[i+k+arr.length] =arr[i];


}
```

</td></tr>
<tr><td>Re-arrange Odd Even</td><td>
1. Temp variables j=-1, temp
2. Iterate
3. If arr is even, %2==0
4. Increment j++
5. Swap
   a. Temp = arr[i]
   b. Arr[i]=arr[j]

```
int j =-1;  //this is -1 so that we will always have the evens
arranged to teh right.
int temp;

      for(int i=0; i<arr.length;i++){
```

</td></tr>
</table>

	`` if(arr[i] % 2==0){ //move the counter by one to the right j++; //do the swap arr[i] = arr[i] ^ arr[j]; arr[j] = arr[i] ^ arr[j]; arr[i] = arr[i] ^ arr[j]; } ```
Equilibrium	1. Int sum, int left sum 2. Iterate and sum+=arr[i] 3. Nested iterate and sum =-arr[i] 4. If leftsum ==sum return 1 5. Leftsum += arr[i]
Reverse Array	1. Iterate length/2 2. Temp = arr[i] 3. Arr[i] = arr[length -1 -i] 4. arr[length -1 -i] = temp
Swap	• Temp = arr[i] • Arr[i] = arr[b] • Arr[b] = temp
HashMap put	1. Create hash 2. Entry entry = table[hash] 3. If entry != null A. If entry.key same value k B. Entry value =v; C. Else (collision) D. While entry.next !=null Entry bucket = new Entry Entry.next =bucket 4. Else 5. Entry newBucket = new Entry(k,v) Table[hash] =newBucket
Greatest Profit between two elemetns	``` public int maxProfit(int[] prices) { if(prices==null||prices.length<=1) return 0; int min=prices[0]; // min so far int max=0; ```

	`` for(int i=1; i<prices.length; i++){ max = Math.max(result, prices[i]-min); min = Math.min(min, prices[i]); } return max; } ```
Prime Number	``` //checks whether an int is prime or not. boolean isPrime(int n) { //check if n is a multiple of 2 if (n%2==0) return false; //if not, then just check the odds for(int i=3;i*i<=n;i+=2) { if(n%i==0) return false; } return true; } ```
PrimeNumber JDK	```BigInteger.valueOf(1235).isProbablePrime(1)```
Stack	``` public MyStack(int s) { maxSize = s; stackArray = new long[maxSize]; top = -1; } public void push(long j) { stackArray[++top] = j; } public long pop() { return stackArray[top--]; } public long peek() { return stackArray[top]; } public boolean isEmpty() { return (top == -1); } public boolean isFull() { return (top == maxSize - 1); } public static void main(String[] args) { MyStack theStack = new MyStack(10); theStack.push(10); theStack.push(20); theStack.push(30); theStack.push(40); theStack.push(50); ```

	`while (!theStack.isEmpty()) {` `long value = theStack.pop();` `System.out.print(value);` `System.out.print(" ");` `}` `System.out.println("");` `}` `}`
Find distinct	`int[] unique = Arrays.stream(arr).distinct().toArray();`
Max difference beteween two array elements	`int maxDifference(int arr[], int n)` `{` `int min_element=arr[0];` `int diff=arr[1]-arr[0];` `for(i=1;i<n;i++)` `{` `if(arr[i]-min_element>diff)` `diff=arr[i]-min_element;` `if(arr[i]<min_element)` `min_element=arr[i];` `}` `return diff;` `}`
Find duplicates	`String[] strArray = {"Java", "JSP", "Servlets", "Java", "Struts", "JSP", "JDBC"};` `HashSet<String> set = new HashSet<String>();` `for (String arrayElement : strArray)` `{` `if(!set.add(arrayElement))` `{` `System.out.println("Duplicate Element is : "+arrayElement);` `}`
Array Highest and Lowest	`int[] large=new int[] {47498, 14526, 74562, 42681, 75283, 45796};` `int [] data=Arrays.stream(large).sorted().toArray();` `System.out.println(data[0]); //lowest` `System.out.println(data[data.length-1]);  //highest`
Access an array element	`System.out.println(data[0]); //lowest`

Find all elements with sum of k	`int count = 0;// Initialize result` `// Consider all possible pairs and check their sums` `for (int i = 0; i < arr.length; i++)` `for (int j = i + 1; j < arr.length; j++)` `if ((arr[i] + arr[j]) == sum)` `count++;`
intersection of two arrays	`HashSet<String> set1 = new HashSet<String>(Arrays.asList(inputArray1));` `HashSet<String> set2 = new HashSet<String>(Arrays.asList(inputArray2));` `set1.retainAll(set2);`
intersection of multiple arrays	`HashSet<Integer> unionSet = new HashSet<Integer>();` `Then add to the set`
Frequence	`Counter elementCountMap.put(i, elementCountMap.get(i)+1);`
Filter object	`public List<Article> getAllJavaArticles() {` `return articles.stream()` `.filter(article -> article.getTags().contains("Java"))` `.collect(Collectors.toList());` `}`
Array Copy Range	`Arrays.copyOfRange(oldArray, startIndex, endIndex);`

```
public List<Article> getAllJavaArticles() {
  return articles.stream()
    .filter(article -> article.getTags().contains("Java"))
    .collect(Collectors.toList());
  }
```

https://www.geeksforgeeks.org/java-tricks-competitive-programming-java-8/