

Решение задач

Из лекций 13 и 14

1. Задача номер 3 из лекции 13

```
3 public static void main(String[] args) {
4     print(negativeCount(new int[]{-1, 3, -2, 4}));
5     print(negativeCount(new int[]{1, 3, -2, 4}));
6     print(negativeCount(new int[]{1, 3, 2, 4}));
7     print(negativeCount(new int[]{}));
8 }
9
10 @ private static int negativeCount(int[] array) {
11     int count = 0;
12     if (array.length < 1) {
13         print("array is empty");
14         count = -1;
15     } else {
16         for (int number : array) {
17             if (number < 0) {
18                 count++;
19             }
20         }
21     }
22     return count;
23 }
24
```

Main > negativeCount()

Run: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...

2

1

0

array is empty

-1

Process finished with exit code 0

Ну здесь все просто. Проверка на пустоту и после чего циклом проходим по массиву. Используем без индекса, ибо он нам ни к чему.

2. Задача 4 из лекции 13

```
3 public static void main(String[] args) {
4     print(numberCount( number: 4, new int[]{-1, 3, -2, 4}));
5     print(numberCount( number: 1, new int[]{1, 3, -2, 4, 1}));
6     print(numberCount( number: 6, new int[]{1, 3, 2, 4}));
7     print(numberCount( number: 90, new int[]{}));
8 }
9
10 private static int numberCount(int number, int[] array) {
11     int count = 0;
12     if (array.length < 1) {
13         print("array is empty");
14         count = -1;
15     } else {
16         for (int num : array) {
17             if (num == number) {
18                 count++;
19             }
20         }
21     }
22     return count;
23 }
```

Main > main()

Run: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...

1
2
0
array is empty
-1

Здесь тоже просто. Берем решение предыдущей задачи, меняем аргументы и условие. Готово.

3. Задача номер 5 из лекции 13

```
3 public static void main(String[] args) {
4     print(emptyStringsCount(new String[]{"a", "", "sd", ""}));
5     print(emptyStringsCount(new String[]{""}));
6     print(emptyStringsCount(new String[]{" " }));
7     print(emptyStringsCount(new String[]{}));
8 }
9
10 @ private static int emptyStringsCount(String[] array) {
11     int count = 0;
12     if (array.length < 1) {
13         print("array is empty");
14         count = -1;
15     } else {
16         for (String string : array) {
17             if (string.isEmpty()) {
18                 count++;
19             }
20         }
21     }
22     return count;
23 }
```

Main > main()

Run: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...

2

1

0

array is empty

-1

Process finished with exit code 0

Аналогично первой задаче меняю тип и условие, готово. Ничего сложного. Если вам что-то не понятно, то задавайте вопросы.

4. Задача 1 из лекции 14

```
3 public static void main(String[] args) {
4     showNumbersDividedBy3( max: 10);
5 }
6
7 private static void showNumbersDividedBy3(int max) {
8     for (int i = 0; i < max; i += 3) {
9         print(i);
10    }
11 }
12
```

Main > showNumbersDividedBy3()

Run: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...

0
3
6
9

Мы считаем числа от нуля и до параметра который максимум. Чтобы не писать условие кратности, мы сразу меняем наш шаг на +3. Т.е. сначала у нас будет первый цикл с $i=0$, потом +3 т.е. $i=3$, $i=6$ и так далее, до max .

5. Задача 2 из лекции 14

```
3 public static void main(String[] args) {
4     showNumbersDividedByNumber( max: 10, arg: 4);
5 }
6
7 private static void showNumbersDividedByNumber(int max, int arg) {
8     for (int i = 0; i < max; i += arg) {
9         print(i);
10    }
11 }
12
```

Main > main()

Run: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...

0
4
8

В этой задаче все то же самое, за исключением что мы выводим числа кратные не конкретно 3, а тому, которое передали. Вы бы могли написать иначе. Инкрементируя по единице и проверяя в условии. Но так проще и оптимальнее. Ведь согласитесь, каждый раз когда вы пишете условие, там ваш компилятор останавливается на какое-то время. А нам нужно чтобы код выполнялся быстро. Если хотите проверьте это на гигантских размерах.

6. Задача 3 из лекции 14

```
2
3 public static void main(String[] args) {
4     showFibbonachi( max: 10);
5 }
6
7 private static void showFibbonachi(int max) {
8     int item0 = 1;
9     int item1 = 1;
10    for (int i = 0; i < max; i++) {
11        if (i == 0 || i == 1) {
12            print(1);
13        } else {
14            print(item0 + item1);
15            int item10ld = item1;
16            item1 = item0 + item1;
17            item0 = item10ld;
18        }
19    }
20 }
```

Main > showFibbonachi()

Run: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...

1
1
2
3
5
8
13
21
34
55

Process finished with exit code 0

Последовательность Фиббоначи это когда новый член последовательности является суммой предыдущий 2 членов. Но первые 2 члена это единицы. Потому у нас будет проверка. Хотя вы могли бы просто вывести две единицы и написать цикл начиная с индекса 2.

```
private static void showFibbonachi(int max) {
    int item0 = 1;
    int item1 = 1;
    print(item0);
    print(item1);
    for (int i = 2; i < max; i++) {
        print(item0 + item1);
        int item10ld = item1;
        item1 = item0 + item1;
        item0 = item10ld;
    }
}
```

7. Задача 4 из лекции 14

```
2
3 public static void main(String[] args) {
4     print(differenceBetweenMaxAndMin(new int[]{5, 3}));
5     print(differenceBetweenMaxAndMin(new int[]{15, 2, 10, -3, 1, -13}));
6 }
7
8 @
9 private static int differenceBetweenMaxAndMin(int[] array) {
10     int difference = 0;
11     if (array.length < 2) {
12         print("array doesn't contain at least 2 items");
13     } else {
14         final int item0 = array[0];
15         final int lastItem = array[array.length - 1];
16
17         boolean isFirstItemMin = item0 < lastItem;
18         int min = isFirstItemMin ? item0 : lastItem;
19         int max = isFirstItemMin ? lastItem : item0;
20
21         for (int i = 1; i < array.length - 1; i++) {
22             if (array[i] < min) {
23                 min = array[i];
24             } else if (array[i] > max) {
25                 max = array[i];
26             }
27         }
28         difference = max - min;
29     }
30     return difference;
31 }
32
33 Main > differenceBetweenMaxAndMin()
Run: Main x
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
2
28
Process finished with exit code 0
```

Здесь немного сложно. Сначала проверим что в массиве есть хотя б 2 элемента. После уже находим мин и максимум среди первого и последнего. После проходим циклом между вторым элементом и предпоследним. Сравним каждый член этой подгруппы с текущими минимумом и максимумом и перезапишем значения если это число окажется меньше минимума или больше максимума. Вы бы могли поступить проще конечно. Сначала найти минимум полным циклом и потом максимум полным циклом. Но мы за оптимизацию и проходим весь массив за раз. А еще существует более простой способ. Отсортировать массив и просто найти разницу между первым членом и последним. т.е. после сортировки у вас минимум будет первым, а максимум последним или же наоборот если вы отсортируете в обратном порядке. Об этом мы поговорим в следующей лекции. А пока что такое решение. Вы можете в мейне написать больше тестов и проверить наш метод.

8. Задача 5 из лекции 14

Эту задачу мы можем решить прямо сейчас, но я бы не хотел это делать используя только те знания, которые у нас есть. Поэтому если вы решили эту задачу, то молодцы. А я подожду пока мы пройдем еще пару тем и тогда уже напишу решение. Просто зная циклы нужно будет несколько раз пройти по массиву. А я бы не хотел этого. Запомните эту задачу, мы к ней вернемся еще не раз.

9. Задача 6 из лекции 14



```
2
3 public static void main(String[] args) {
4     showItemsReverted(new int[]{1,2,3});
5 }
6
7 private static void showItemsReverted(int[] array) {
8     if (array.length == 0) {
9         print("array is empty");
10    } else {
11        for (int i = array.length - 1; i >= 0; i--) {
12            print(array[i]);
13        }
14    }
15 }
16
```

Main > main()

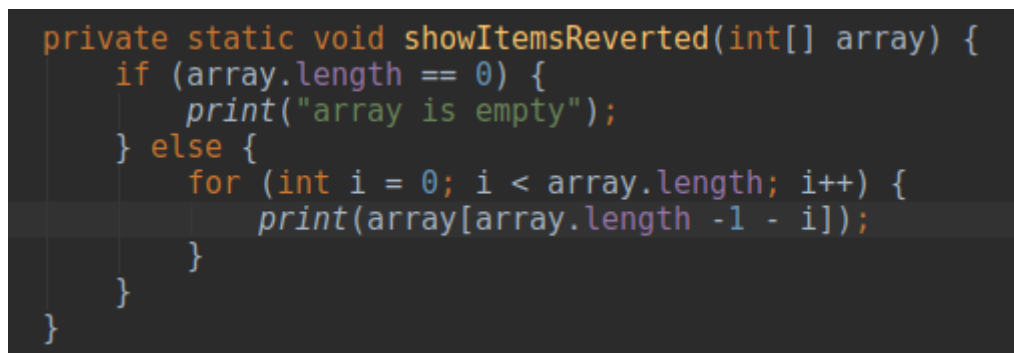
Run: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...

3
2
1

Process finished with exit code 0

Здесь все просто. Вы могли бы написать иначе. Например вот так



```
private static void showItemsReverted(int[] array) {
    if (array.length == 0) {
        print("array is empty");
    } else {
        for (int i = 0; i < array.length; i++) {
            print(array[array.length - 1 - i]);
        }
    }
}
```

Разницы совсем нет. Главное чтобы работало одинаково хорошо

10. Задача 7 из лекции 14

```
public static void main(String[] args) {
    showIndex(new String[] { "a", "b", "this", "2", "this", "this" });
    showIndex(new String[] { "a", "b", "this", "2" });
}

private static void showIndex(String[] array) {
    if (array.length == 0) {
        print("array is empty");
    } else {
        String indexes = "";
        for (int i = 0; i < array.length; i++) {
            if ("this".equals(array[i])) {
                if (!indexes.isEmpty()) {
                    indexes += ", ";
                }
                indexes += i;
            }
        }
        print(indexes);
    }
}
```

Main > main()

un: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...

2, 4, 5

2

Process finished with exit code 0

Здесь по сути наверно самое интересное это конкатенация индексов если их больше чем 1. Я рекомендую проверять сначала на пустоту и уже потом добавлять запятую и пробел. Ведь если у вас будет 1 индекс, то зачем сразу добавлять запятую и пробел? Чтобы в конце проверить строку еще раз и удалить запятую? А мы умеем это? Запомните, лучше избегайте этой операции. Проще добавлять чем удалять в принципе.

11. Задача 8 из лекции 14

```
2
3 public static void main(String[] args) {
4     int[] newArray = abs(new int[]{9, -1, 2, -5, -6, 0, 2});
5     for (int number : newArray) {
6         print(number);
7     }
8 }
9
10 @ private static int[] abs(int[] array) {
11     int[] result = new int[array.length];
12
13     for (int i = 0; i < result.length; i++) {
14         result[i] = array[i] > 0 ? array[i] : -1 * array[i];
15     }
16     return result;
17 }
18
```

Main > main()

Run: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...

9
1
2
5
6
0
2

В этой задаче все тоже просто. Создаем массив той же длины что и входящий аргумент. Проходим по нему и кладем в новый массив значения. Элементарно. Кто-то скажет, что мы забыли проверку на пустой массив, но она и не нужна здесь. У нас нет конфликтов по этому вопросу. Но если хотите, то добавьте условие в начале метода.

12. Задача 9 из лекции 14

```
2
3 public static void main(String[] args) {
4     int[] newArray = abs(new int[]{1, 2, 3, 4, 5});
5     for (int number : newArray) {
6         print(number);
7     }
8 }
9
10 @ private static int[] abs(int[] array) {
11     int[] result = new int[array.length];
12
13     for (int i = 0; i < result.length - 1; i++) {
14         result[i] = array[i] + array[i + 1];
15     }
16     result[result.length - 1] = array[array.length - 1];
17     return result;
18 }
19
```

Main > abs()

Run: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...

3
5
7
9
5

Мы проходим по массиву чисел, но не по всем элементам, а кроме последнего. И чтобы последний элемент тоже присутствовал в нашем массиве мы просто копируем его в результирующий. Не нужны никакие условия тем более внутри цикла. Хотя да, здесь проверка на пустоту действительно нужна, потому что мы обращаемся к элементу по индексу. А в случае пустого будет ошибка. Про ошибки мы поговорим позже.

13. Задача 10 из лекции 15

```
3 public static void main(String[] args) {
4     int[] newArray = abs(new int[]{1, 2, 3, 4, 3, 2, 1});
5     for (int number : newArray) {
6         print(number);
7     }
8 }
9
10 private static int[] abs(int[] array) {
11     int[] result = new int[array.length];
12     for (int i = 0; i < result.length; i++) {
13         result[i] = array[i] + array[array.length - 1 - i];
14     }
15     return result;
16 }
17
```

Main

Run: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...

2
4
6
8
6
4
2

Process finished with exit code 0

Давайте поговорим о вот какой штуке. Смотрите, у нас задание где нужно складывать первый и последний элемент массива. И о чем нужно подумать в первую очередь? А о том, что если у нас четное количество элементов, то все будет супер. Каждому элементу есть пара. Если массив состоит из 4 элементов, то 2 пары будет – 1 и 4, 2 и 3. А если в массиве нечетное количество элементов? Тогда наш средний элемент будет складываться с самим собой! Посмотрите. 1 2 3 4 3 2 1 стало 2 4 6 8 6 4 2. Т.е. 4 сложилось само с собой. И если бы я не упомянул в задании это, вы бы могли и не делать ничего. Или же сами додуматься о том, чтобы не складывать число с собой же. Здесь все зависит на самом деле от реальных условий задач. Так что вы всегда должны думать о всех возможных вариантах. Во-первых всегда проверять входные аргументы у метода и во-вторых думать над состоянием этого аргумента.

Это нужно для того, чтобы проверять свои методы. Далее мы подробно остановимся на тестах и тестировании методов. Но пока просто знайте, что если вы написали метод, который принимает аргументом массив чисел, то во-первых попробуйте передать туда пустой массив. Во-вторых массив с 1 элементом. И так как у нас идет работа с парами, любое четное и нечетное количество элементов. Тогда вы будете уверены что все написали правильно.

14. Задача 11 из лекции 15

```
3 public static void main(String[] args) {
4     int[] newArray = doubleArray(new int[]{1, 2, 3, 4});
5     String text = "";
6     for (int number : newArray) {
7         text += number + " ";
8     }
9     print(text);
10 }
11
12 @ private static int[] doubleArray(int[] array) {
13     int[] result = new int[array.length * 2];
14     int j = 0;
15     for (int i = 0; i < result.length; i += 2) {
16         result[i] = array[j];
17         result[i + 1] = array[j];
18         j++;
19     }
20     return result;
21 }
22
```

Main > main()

Run: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
1 1 2 2 3 3 4 4

Process finished with exit code 0

Итак, нам нужен массив с размером $2 \times$ размер первого. Значит нужно его создать. Теперь, как его заполнить? У нас будет явная путаница с индексами. Я просто создал счетчик для первого массива и прошелся по итоговому массиву и заполнил его данными. Все просто, если не усложнять ничего. Согласен, можно было написать иначе. И если у вас получилось все правильно, то я рад за вас. Можем обсудить ваше решение.

Кстати, у меня возник вопрос – а какая максимальная длина массива? Как бы я мог найти ответ на него? Наверно я бы посмотрел на то, что мне возвращает `array.length` и увидел бы что там `int`. Значит ли это что я могу положить в массив максимум 2 миллиарда значений?

Давайте проверим на деле.

```
public static void main(String[] args) {
    int[] array = new int[Integer.MAX_VALUE + 10000];
}
```

И среда разработки сразу нам говорит что нельзя к максимальному значению прибавлять еще. Запустите код и вы увидите ошибку в консоли. Про ошибки как и обещал попозже

поговорим. Так что создание переменной `int j` вполне оправдано. Мы же не знаем какого размера массив приходит в аргумент метода.

А еще у меня возник такой вопрос: а какой длины может быть строка? У нее же тоже `string.length` есть и там тоже `int`, так? А как мы можем проверить? Давайте в цикле конкатенировать строку. Осторожно! Это может занять некоторое время, лучше для этих целей используйте облачную вычислительную машину. Ваш компьютер может встять надолго. У кого компьютер остался жив после этого отпишитесь мне в личку ;) Но если боитесь то не рискуйте, компуктер может взорваться! (ха-ха, шутка. Или нет ;)).

15. Задача 12 из лекции 15

Эту задачу я бы тоже оставил на потом, если вы ее решили, то молодцы. Если же нет, то ничего страшного. Мы ее решим когда у нас будут знания о более удобных штуках нежели обычные массивы и циклы.

16. Задача 13 из лекции 15

```
3 public static void main(String[] args) {
4     int[] array = productArrays(new int[]{1, 2, 3, 4, 5}, new int[]{6, 7, 8, 9, 10});
5     for (int item : array) {
6         print(item);
7     }
8     int[] failed = productArrays(new int[]{1, 3}, new int[]{5});
9     failed = productArrays(new int[]{Integer.MAX_VALUE}, new int[]{2});
10    for (int item: failed) {
11        print (item);
12    }
13 }
14
15 private static int[] productArrays(int[] array1, int[] array2) {
16     int[] result;
17     final int arrayLength = array1.length;
18     if (arrayLength == array2.length) {
19         result = new int[arrayLength];
20
21         for (int i = 0; i < arrayLength; i++) {
22             result[i] = array1[i] * array2[i];
23         }
24     } else {
25         print("arrays are different size");
26         result = new int[]{};
27     }
28     return result;
29 }
30
31
```

Main main()

Run: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...

6
14
24
36
50
arrays are different size
-2

Process finished with exit code 0

Итак, мы получаем в итоге массив, где каждый элемент это произведение членов двух массивов. И здесь нужно подумать над тем, а что если наши массивы разной длины? А еще нужно подумать над тем, а что будет, если мы перемножим числа сверх максимального значения. Как видите код выполняется, но мы получаем неверный результат. Я уже говорил ранее, что если вы производите действия над числами, то имейте ввиду что они могут выйти за рамки допустимых значений. Что же здесь нужно было сделать? Вернуть массив другого типа. Да.

```
2
3 public static void main(String[] args) {
4     double[] array = productArrays(new int[]{1, 2, 3, 4, 5}, new int[]{6, 7, 8, 9, 10});
5     for (double item : array) {
6         print(item);
7     }
8     double[] failed = productArrays(new int[]{1, 3}, new int[]{5});
9     failed = productArrays(new int[]{Integer.MAX_VALUE}, new int[]{2});
10    for (double item : failed) {
11        print(item);
12    }
13 }
14
15 private static double[] productArrays(int[] array1, int[] array2) {
16     double[] result;
17     final int arrayLength = array1.length;
18     if (arrayLength == array2.length) {
19         result = new double[arrayLength];
20
21         for (int i = 0; i < arrayLength; i++) {
22             result[i] = ((double) array1[i]) * array2[i];
23         }
24     } else {
25         print("arrays are different size");
26         result = new double[]{};
27     }
28     return result;
29 }
30
31
Main > main()
Run: Main x
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
6.0
14.0
24.0
36.0
50.0
arrays are different size
4.294967294E9
Process finished with exit code 0
```

Как видите все вывелось правильно. Так что когда вы пишете код, то проверяйте его на граничных условиях. Если метод принимает аргументом число, то проверьте что этот метод работает и с самым большим значением и с самым меньшим и с нулем. И с положительным и с отрицательным. Если мы говорим про строку, то проверьте что метод работает если на вход получил пустую строку или строку с 1 символом. С кириллицей или же символы вместо букв. В этом вся суть программирования. Написать такой код, который бы работал в любом случае и не давал бы сбоев. Т.е. вы должны всегда видеть

PROCESS FINISHED WITH EXIT CODE 0