# Java: знакомство и как пользоваться базовым API (семинары)

### Задание 1. Удаление отрицательных значений из массива

Реализуйте метод, который принимает на вход целочисленный массив и удаляет все отрицательные числа. Метод должен вернуть массив, содержащий только неотрицательные числа.

**Пример**:

[-1, 2, -3, 4, -5, 6]

**Результат:**

[2, 4, 6]

```
import java.util.Arrays;


class FilterNegative {

    public static int[] filterNegative(int[] a) {

        // Напишите свое решение ниже


    }

}


// Не удаляйте этот класс - он нужен для вывода результатов на экран и проверки

public class Printer {

    public static void main(String[] args) {

        int[] a;
```

```
        if (args.length == 0) {
            // При отправке кода на Выполнение, вы можете
варьировать эти параметры
            a = new int[]{-1, 2, -3, 4, -5, 6};
        } else {
            a = Arrays.stream(args[0].split(",
")).mapToInt(Integer::parseInt).toArray();
        }

        FilterNegative answer = new FilterNegative();
        String itresume_res =
Arrays.toString(answer.filterNegative(a));
        System.out.println(itresume_res);
    }
}
```

**Подсказка № 1**

Для удобства работы с элементами, которые нужно сохранить, используйте `ArrayList<Integer>`. Это позволит динамически добавлять элементы и избежать сложностей с изменением размера массива.

**Подсказка № 2**

Пройдитесь по каждому элементу массива с помощью цикла `for`. Если элемент неотрицательный (`num >= 0`), добавьте его в `ArrayList`.

**Подсказка № 3**

После того как вы добавили все нужные элементы в `ArrayList`, преобразуйте его в обычный массив `int[]`. Это можно сделать, создав новый массив нужного размера и скопировав в него элементы из `ArrayList`.

**Подсказка № 4**

После преобразования `ArrayList` в массив, верните полученный массив как результат работы метода.

**Эталонное решение:**

```
import java.util.ArrayList;

import java.util.Arrays;


class FilterNegative {

   public static int[] filterNegative(int[] a) {

       ArrayList<Integer> result = new ArrayList<>();

       for (int num : a) {

           if (num >= 0) {

               result.add(num);

           }

       }

       // Преобразуем ArrayList в массив

       int[] resultArray = new int[result.size()];

       for (int i = 0; i < result.size(); i++) {

           resultArray[i] = result.get(i);

       }

       return resultArray;

   }

}


// Не удаляйте этот класс - он нужен для вывода результатов на экран и проверки

public class Printer {

   public static void main(String[] args) {
```

```
        int[] a;



        if (args.length == 0) {

            // При отправке кода на Выполнение, вы можете варьировать
эти параметры

            a = new int[]{-1, 2, -3, 4, -5, 6};

        } else {

            a = Arrays.stream(args[0].split(",
")).mapToInt(Integer::parseInt).toArray();

        }



        FilterNegative answer = new FilterNegative();

        String itresume_res =
Arrays.toString(answer.filterNegative(a));

        System.out.println(itresume_res);

    }

}
```

### Задача 2. Уникальные числа

Напишите метод, который принимает целочисленный массив и возвращает новый массив, содержащий только уникальные элементы из исходного массива.

**Пример:**

[1, 2, 2, 3, 4, 4, 5]

**Результат:**

[1, 2, 3, 4, 5]

```
import java.util.Arrays;

import java.util.LinkedHashSet;
```

```
class UniqueElements {

    public static int[] getUniqueElements(int[] a) {

        // Напишите свое решение ниже


    }

}


// Не удаляйте этот класс - он нужен для вывода результатов на экран и проверки

public class Printer {

    public static void main(String[] args) {

        int[] a;


        if (args.length == 0) {

            // При отправке кода на Выполнение, вы можете варьировать эти параметры

            a = new int[]{1, 2, 2, 3, 4, 4, 5};

        } else {

            a = Arrays.stream(args[0].split(",
")).mapToInt(Integer::parseInt).toArray();

        }


        UniqueElements answer = new UniqueElements();

        String itresume_res =
Arrays.toString(answer.getUniqueElements(a));

        System.out.println(itresume_res);
```

```
    }

}
```

**Подсказка № 1**

`LinkedHashSet` автоматически исключает дубликаты и сохраняет порядок вставки элементов. Это удобный способ сохранить уникальные элементы и порядок их появления.

**Подсказка № 2**

Пройдитесь по каждому элементу исходного массива и добавьте его в `LinkedHashSet`. Если элемент уже присутствует, `LinkedHashSet` не добавит его повторно.

**Подсказка № 3**

После того как все уникальные элементы будут собраны в `LinkedHashSet`, преобразуйте его в массив. Для этого создайте новый массив нужного размера и скопируйте элементы из `LinkedHashSet`.

**Подсказка № 4**

После преобразования `LinkedHashSet` в массив, верните полученный массив как результат работы метода.

**Эталонное решение:**

```
import java.util.ArrayList;

import java.util.Arrays;



class UniqueElements {

   public static int[] getUniqueElements(int[] a) {

       ArrayList<Integer> result = new ArrayList<>();

       for (int num : a) {

           if (!result.contains(num)) {

               result.add(num);
```

```
            }

        }

        // Преобразуем ArrayList в массив

        int[] resultArray = new int[result.size()];

        for (int i = 0; i < result.size(); i++) {

            resultArray[i] = result.get(i);

        }

        return resultArray;

    }

}


// Не удаляйте этот класс - он нужен для вывода результатов на экран и проверки

public class Printer {

    public static void main(String[] args) {

        int[] a;


        if (args.length == 0) {

            // При отправке кода на Выполнение, вы можете варьировать эти параметры

            a = new int[]{1, 2, 2, 3, 4, 4, 5};

        } else {

            a = Arrays.stream(args[0].split(",
")).mapToInt(Integer::parseInt).toArray();

        }


        UniqueElements answer = new UniqueElements();
```

```
        String itresume_res =
Arrays.toString(answer.getUniqueElements(a));

        System.out.println(itresume_res);

    }

}
```

### Задача 3. Длина слов

Реализуйте метод, который принимает на вход массив строк и возвращает новый массив, содержащий только строки, длина которых больше 3 символов.

**Пример:**

["cat", "elephant", "dog", "giraffe"]

**Результат:**

["elephant", "giraffe"]

```
import java.util.Arrays;

import java.util.stream.Collectors;


class FilterStrings {

    public static String[] filterShortStrings(String[] arr) {

        // Напишите свое решение ниже


    }

}


// Не удаляйте этот класс - он нужен для вывода результатов на
экран и проверки

public class Printer {

    public static void main(String[] args) {
```

```
        String[] arr;


        if (args.length == 0) {
            // При отправке кода на Выполнение, вы можете
варьировать эти параметры
            arr = new String[]{"cat", "elephant", "dog",
"giraffe"};
        } else {
            arr = args[0].split(", ");
        }


        FilterStrings answer = new FilterStrings();
        String itresume_res =
Arrays.toString(answer.filterShortStrings(arr));
        System.out.println(itresume_res);
    }
}
```

**Подсказка № 1**

Создайте `ArrayList<String>`, чтобы хранить строки, которые соответствуют вашему условию (длине больше 3 символов). Это поможет вам динамически добавлять строки без необходимости предварительно определять размер массива.

**Подсказка № 2**

Итерируйте через массив строк с помощью цикла `for`. Для каждой строки проверьте ее длину с помощью метода `length()`. Если длина строки больше 3 символов, добавьте ее в `ArrayList`.

**Подсказка № 3**

После того как вы добавили все строки, длина которых больше 3 символов, в `ArrayList`, преобразуйте его в массив. Для этого создайте новый массив строк и скопируйте элементы из `ArrayList` в этот массив.

**Подсказка № 4**

После преобразования `ArrayList` в массив, верните полученный массив как результат работы метода.

**Эталонное решение:**

```
import java.util.ArrayList;

import java.util.Arrays;


class FilterStrings {

   public static String[] filterShortStrings(String[] arr) {

       ArrayList<String> result = new ArrayList<>();

       for (String s : arr) {

           if (s.length() > 3) {

               result.add(s);

           }

       }

       // Преобразуем ArrayList в массив

       String[] resultArray = new String[result.size()];

       for (int i = 0; i < result.size(); i++) {

           resultArray[i] = result.get(i);

       }

       return resultArray;

   }

}
```

```java
// Не удаляйте этот класс - он нужен для вывода результатов на экран и проверки

public class Printer {

   public static void main(String[] args) {

       String[] arr;


       if (args.length == 0) {

           // При отправке кода на Выполнение, вы можете варьировать эти параметры

           arr = new String[]{"cat", "elephant", "dog", "giraffe"};

       } else {

           arr = args[0].split(", ");

       }


       FilterStrings answer = new FilterStrings();

       String itresume_res = Arrays.toString(answer.filterShortStrings(arr));

       System.out.println(itresume_res);

   }

}
```

### Задача 4*. Среднее значение массива

Напишите метод, который принимает массив целых чисел и возвращает среднее значение элементов массива, округленное до ближайшего целого числа.

**Пример**:

[4, 2, 7, 5, 1]

**Результат:**

4

```
import java.util.Arrays;


class AverageCalculator {

    public static int calculateAverage(int[] a) {

        // Напишите свое решение ниже


    }

}


// Не удаляйте этот класс - он нужен для вывода результатов на экран и проверки

public class Printer {

    public static void main(String[] args) {

        int[] a;


        if (args.length == 0) {

            // При отправке кода на Выполнение, вы можете варьировать эти параметры

            a = new int[]{4, 2, 7, 5, 1};

        } else {

            a = Arrays.stream(args[0].split(", ")).mapToInt(Integer::parseInt).toArray();

        }


        AverageCalculator answer = new AverageCalculator();
```

```
        int result = answer.calculateAverage(a);

        System.out.println(result);

    }

}
```

**Подсказка № 1**

Используйте цикл `for`, чтобы пройтись по всем элементам массива и подсчитать их сумму. Это поможет вам узнать, сколько в сумме составляют все элементы массива.

**Подсказка № 2**

Используйте свойство `length` массива для получения количества элементов. Это значение понадобится вам для вычисления среднего значения.

**Подсказка № 3**

Чтобы получить среднее значение, разделите сумму всех элементов на количество элементов. Поскольку результат может быть дробным, вам нужно преобразовать его в тип `double` для точного вычисления.

**Подсказка № 4**

Используйте метод `Math.round()` для округления среднего значения до ближайшего целого числа. Метод `Math.round()` возвращает тип `long`, поэтому вам нужно преобразовать его в `int`.

**Подсказка № 5**

Если массив пустой (`a.length == 0`), вычисление среднего значения приведет к ошибке. В этом случае можно вернуть 0.

**Эталонное решение:**

```
import java.util.Arrays;


class AverageCalculator {

   public static int calculateAverage(int[] a) {

       if (a.length == 0) return 0;
```

```
        int sum = 0;

        for (int num : a) {

            sum += num;

        }

        return (int) Math.round((double) sum / a.length);

    }

}



// Не удаляйте этот класс - он нужен для вывода результатов на экран и проверки

public class Printer {

    public static void main(String[] args) {

        int[] a;



        if (args.length == 0) {

            // При отправке кода на Выполнение, вы можете варьировать эти параметры

            a = new int[]{4, 2, 7, 5, 1};

        } else {

            a = Arrays.stream(args[0].split(",
")).mapToInt(Integer::parseInt).toArray();

        }



        AverageCalculator answer = new AverageCalculator();

        int result = answer.calculateAverage(a);

        System.out.println(result);

    }

}
```