|  |  |
| --- | --- |
|  | Министерство образования и науки Российской Федерации |
| Федеральное государственное автономное образовательное учреждение высшего образования |
| «Уральский федеральный университет имени первого Президента России Б. Н. Ельцина» |
| Институт радиоэлектроники и информационных технологий – РтФ |
|  |
|  | |

**ОТЧЕТ**

О ВЫПОЛНЕНИИ

ЛАБОРАТОНОЙ РАБОТЫ № 4

Руководитель ст. пр. Н. А. Архипов

Студент гр. РИМ-130971 Е.В. Треглазов

Екатеринбург 2024

Репозиторий: https://github.com/kuk86kuk/java\_ufru\_2024.

Example1

public class Example1 {  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("Введите размер массива");  
 int size = in.nextInt();  
  
 int[] arr = new int[size];  
  
 Random random = new Random();  
  
 for (int i = 0; i < size; i++) {  
 arr[i] = random.nextInt();  
 }  
  
 System.*out*.println("Массив arr : ");  
 System.*out*.println(Arrays.*toString*(arr));  
  
 int[] arrResult = *filterEvenNumbers*(arr);  
  
 System.*out*.println("Массив arrResult : ");  
 System.*out*.println(Arrays.*toString*(arrResult));  
 }  
  
  
 public static int[] filterEvenNumbers(int[] arr) {  
 return Arrays.*stream*(arr).filter(x -> x % 2 == 0).toArray();  
 }  
}

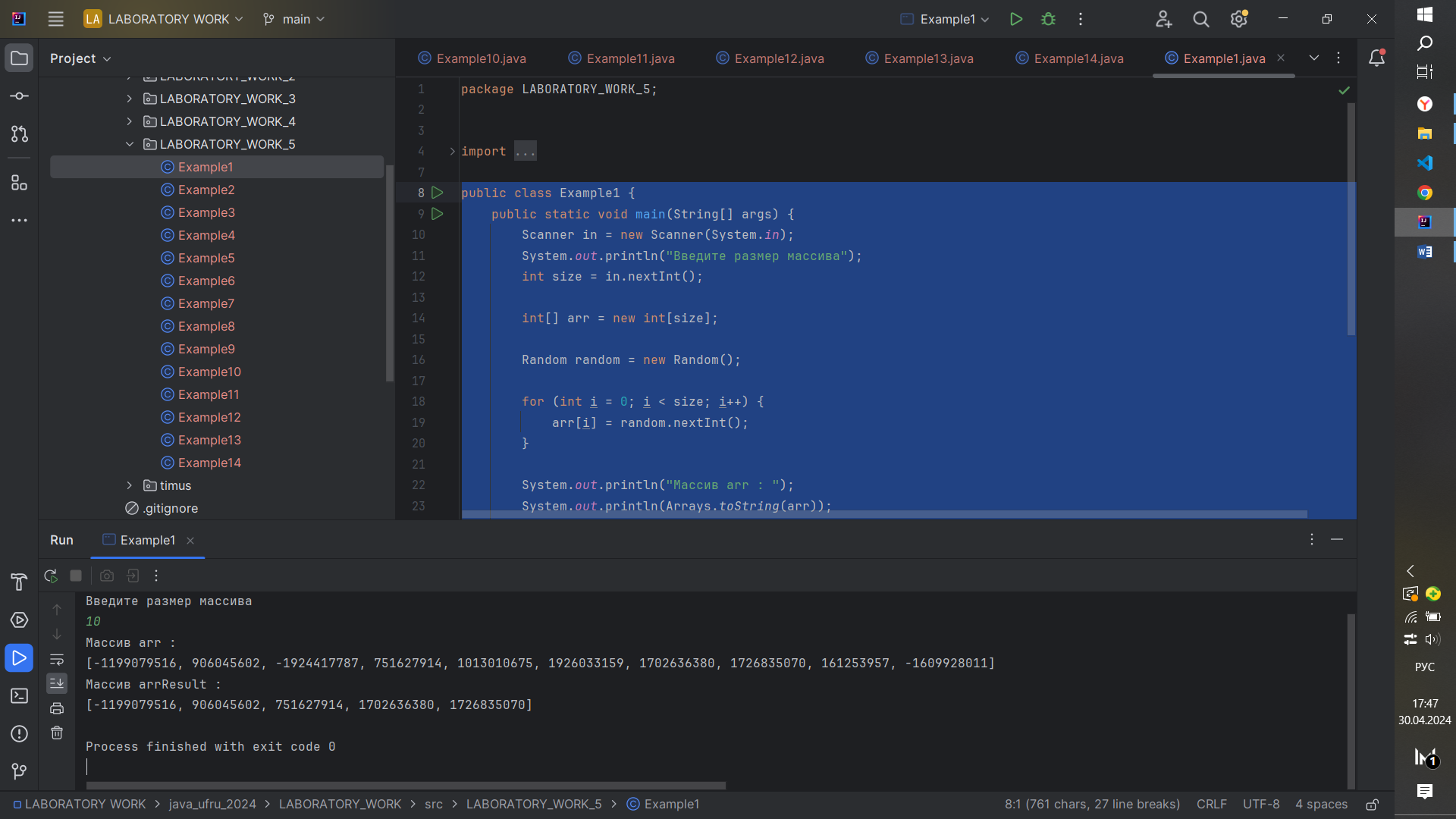
Код создает массив случайных целых чисел заданного размера, затем фильтрует этот массив, оставляя только четные числа, и выводит оба массива на консоль.

В методе main:

* Пользователю предлагается ввести размер массива.
* Создается массив arr с размером, заданным пользователем.
* Заполняется массив arr случайными целыми числами с помощью Random.
* Выводится исходный массив arr на консоль.
* Вызывается метод filterEvenNumbers для получения массива только с четными числами из arr.
* Выводится массив arrResult с четными числами на консоль.

Метод filterEvenNumbers:

* Принимает массив целых чисел.
* Возвращает новый массив, содержащий только четные числа из исходного массива, используя java.util.stream для фильтрации и преобразования в новый массив.



Example2

public class Example2 {  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("Введите размер массива");  
 int size = in.nextInt();  
  
 int[] arr1 = new int[size];  
 int[] arr2 = new int[size];  
 Random random = new Random();  
  
 for (int i = 0; i < size; i++) {  
 arr1[i] = random.nextInt(200);  
 arr2[i] = random.nextInt(200);  
 }  
 System.*out*.println("Массив arr1 : ");  
 System.*out*.println(Arrays.*toString*(arr1));  
 System.*out*.println("Массив arr2 : ");  
 System.*out*.println(Arrays.*toString*(arr2));  
  
 int[] arrResult = *findCommonElements*(arr1, arr2);  
  
 System.*out*.println("Массив arrResult : ");  
 System.*out*.println(Arrays.*toString*(arrResult));  
 }  
  
 public static int[] findCommonElements(int[] arr1, int[] arr2) {  
 return Arrays.*stream*(arr1)  
 .filter(x -> Arrays.*stream*(arr2)  
 .anyMatch(y -> y == x))  
 .toArray();  
 }  
}

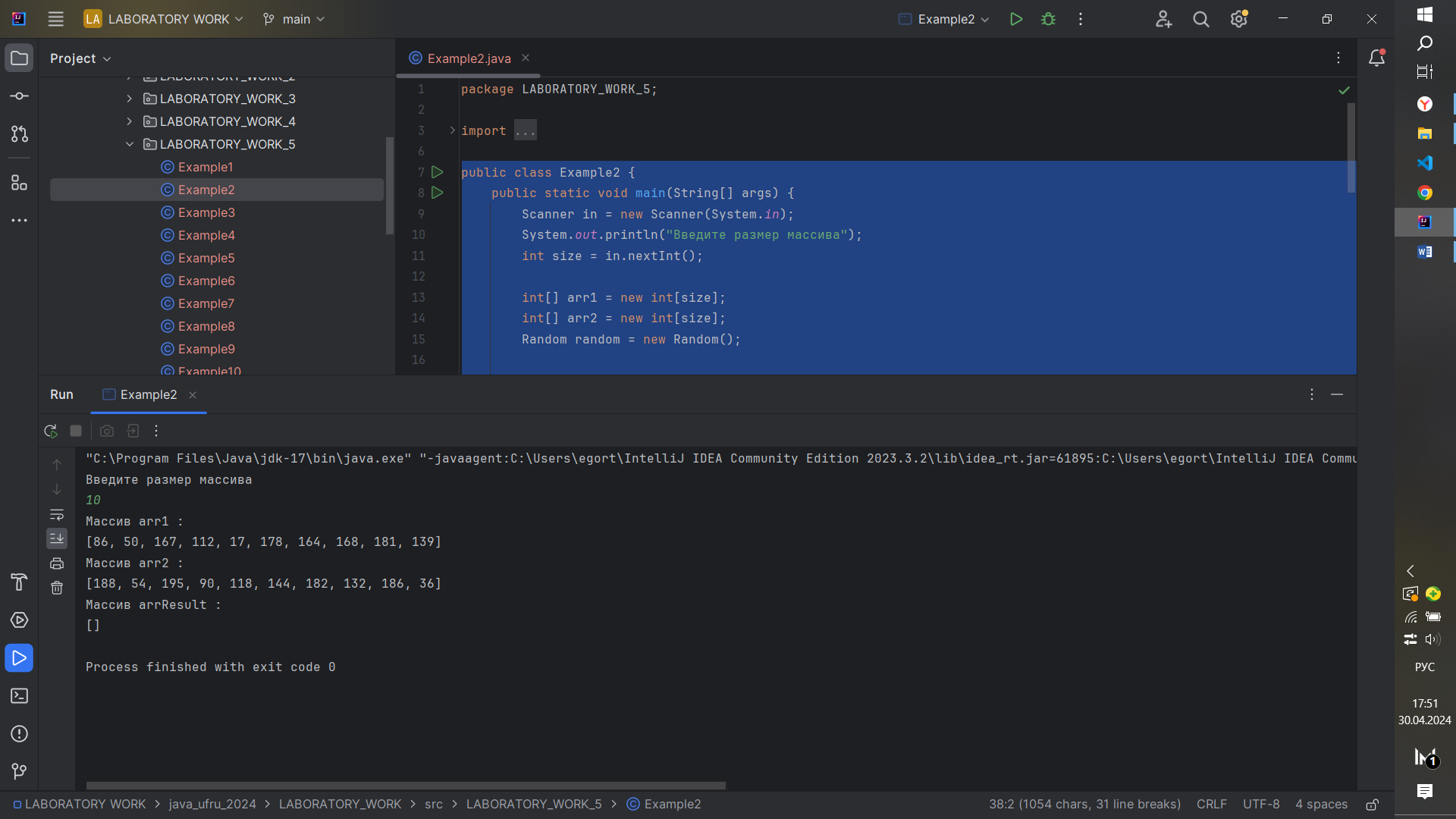
Код создает два массива случайных целых чисел заданного размера, находит общие элементы между этими массивами и выводит все три массива на консоль.

В методе main:

* Пользователю предлагается ввести размер массивов.
* Создаются два массива arr1 и arr2 с размером, заданным пользователем.
* Заполняются оба массива случайными целыми числами от 0 до 199 с помощью Random.
* Выводятся на консоль исходные массивы arr1 и arr2.
* Вызывается метод findCommonElements для получения массива с общими элементами из arr1 и arr2.
* Выводится массив arrResult с общими элементами на консоль.

Метод findCommonElements:

* Принимает два массива целых чисел.
* Возвращает новый массив, содержащий только те элементы, которые присутствуют в обоих исходных массивах, используя java.util.stream для фильтрации и преобразования в новый массив.



Example3

public class Example3 {  
 public static void main(String[] args) {  
 String string = "Напишите функцию, которая принимает на входи список" +  
 "строк и возвращает новый список, Содержащий только те строки, " +  
 "который Начинаются с большой буквы.";  
  
 List<String> strings = List.*of*(string.split(" "));  
 System.*out*.println("\n" + "Строка после сплитования" + "\n");  
 for (String e : strings) {  
 System.*out*.println(e);  
 }  
  
 List<String> stringsAfter = *filterCapitalizedStrings*(strings);  
  
 System.*out*.println("\n" + "Строка после преобразования : " + "\n");  
 for (String e : stringsAfter) {  
 System.*out*.println(e);  
 }  
 }  
  
 private static List<String> filterCapitalizedStrings(List<String> list) {  
 return list.stream()  
 .filter(s -> Character.*isUpperCase*(s.charAt(0)))  
 .collect(Collectors.*toList*());  
 }  
}

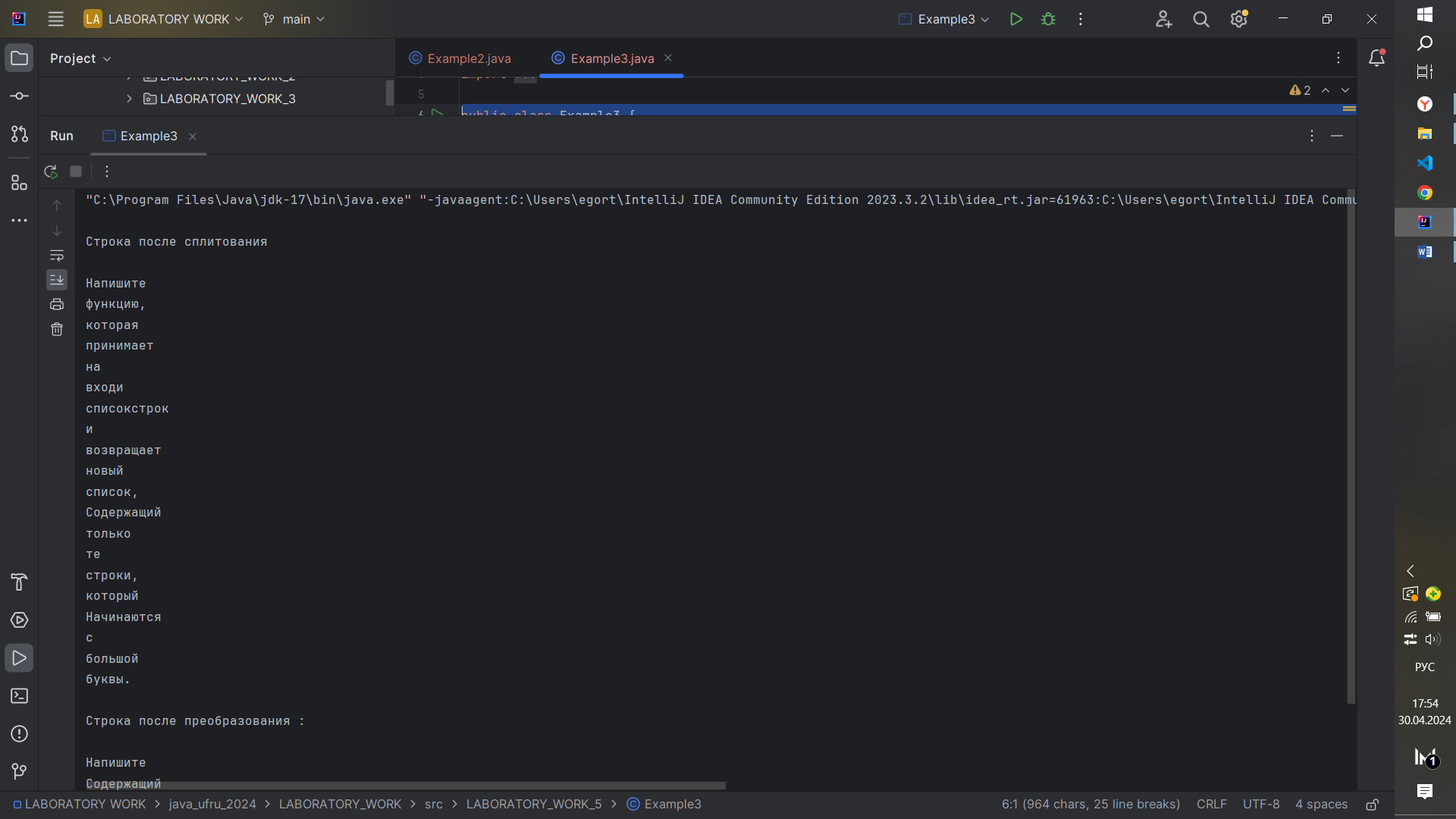
Код разбивает строку на слова, создает список из этих слов, фильтрует список, оставляя только те слова, которые начинаются с заглавной буквы, и выводит оба списка на консоль.

В методе main:

* Определяется строка, содержащая текст.
* Строка разбивается на слова с помощью split(" "), и результат преобразуется в список strings.
* Выводится на консоль список strings после сплитования.
* Вызывается метод filterCapitalizedStrings для получения списка слов, начинающихся с заглавной буквы.
* Выводится на консоль список stringsAfter после фильтрации.

Метод filterCapitalizedStrings:

* Принимает список строк.
* Возвращает новый список, содержащий только те строки, которые начинаются с заглавной буквы, используя java.util.stream для фильтрации и сбора результатов в новый список.



Example4

public class Example4 {  
 public static void main(String[] args) {  
 List<Integer> integers = new ArrayList<>();  
 Random random = new Random();  
  
 System.*out*.println("\n" + "Список до: " + "\n");  
  
 for (int i = 0; i < 10; i++){  
 integers.add(random.nextInt(1000));  
 System.*out*.println(integers.get(i));  
 }  
  
 List<Integer> integersAfter = *squareList*(integers);  
  
 System.*out*.println("\n" + "Спсиок после возведения в квадрат: " + "\n");  
  
 for (Integer i : integersAfter){  
 System.*out*.println(i);  
 }  
 }  
  
 private static List<Integer> squareList(List<Integer> list) {  
 return list.stream().map(x -> x \* x).collect(Collectors.*toList*());  
 }  
}

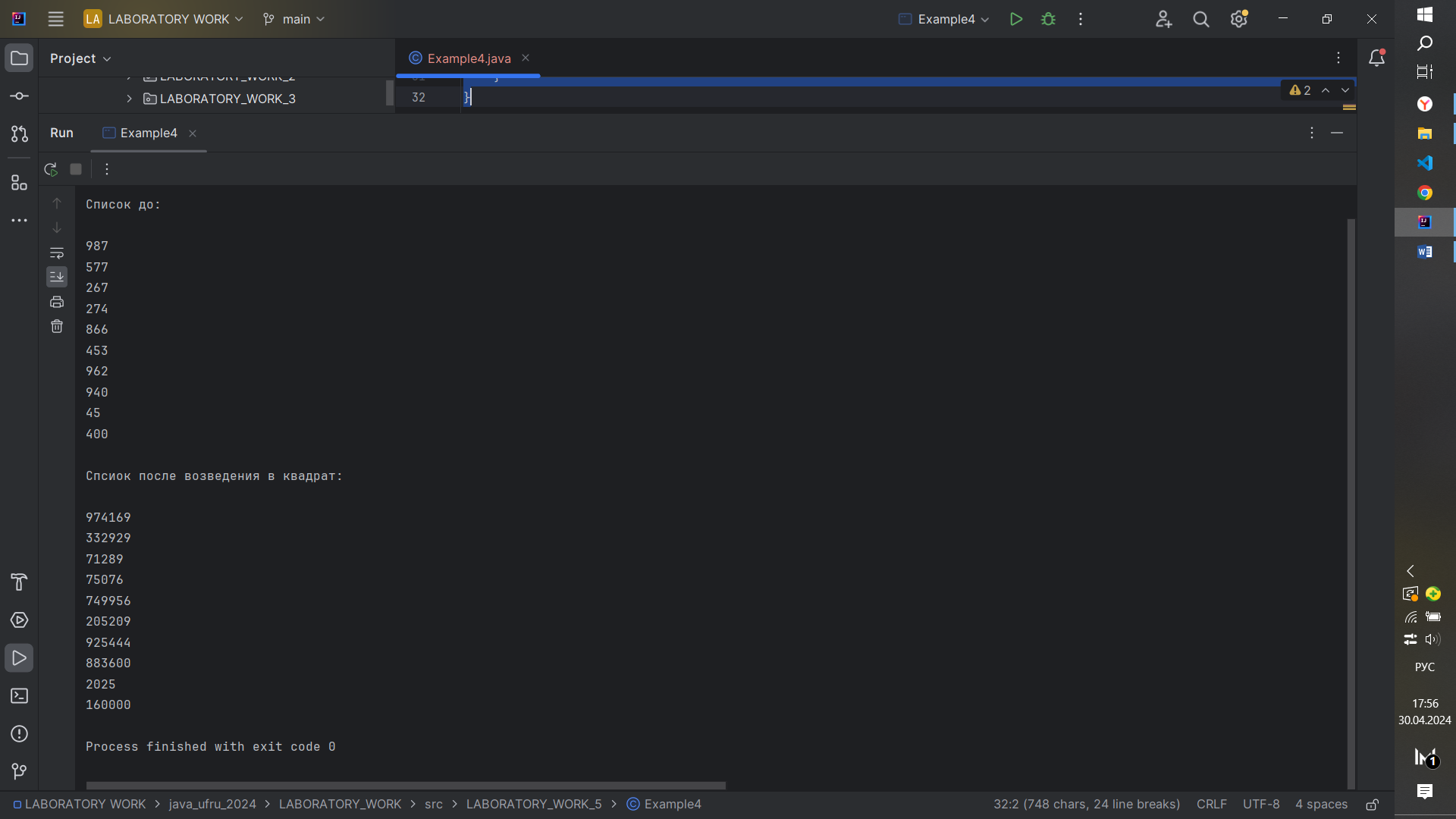
Код создает список случайных целых чисел, возводит каждое число в квадрат, создает новый список с квадратами чисел и выводит оба списка на консоль.

В методе main:

* Создается пустой список integers.
* Генерируются 10 случайных целых чисел от 0 до 999 и добавляются в список integers.
* Выводится на консоль исходный список integers.
* Вызывается метод squareList для получения списка с квадратами чисел из integers.
* Выводится на консоль список integersAfter с квадратами чисел.

Метод squareList:

* Принимает список целых чисел.
* Возвращает новый список, содержащий квадраты каждого числа из исходного списка, используя java.util.stream для преобразования и сбора результатов в новый список.



Example5

public class Example5 {  
 public static void main(String[] args) {  
 int[] numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  
 int[] evenNumbers = *filterEvenNumbers*(numbers);  
  
 System.*out*.println("Исходный массив: " + Arrays.*toString*(numbers));  
 System.*out*.println("Массив с четными числами: " + Arrays.*toString*(evenNumbers));  
 }  
  
  
 private static int[] filterEvenNumbers(int[] numbers) {  
 return Arrays.*stream*(numbers)  
 .filter(x -> x % 2 == 0)  
 .toArray();  
 }  
}

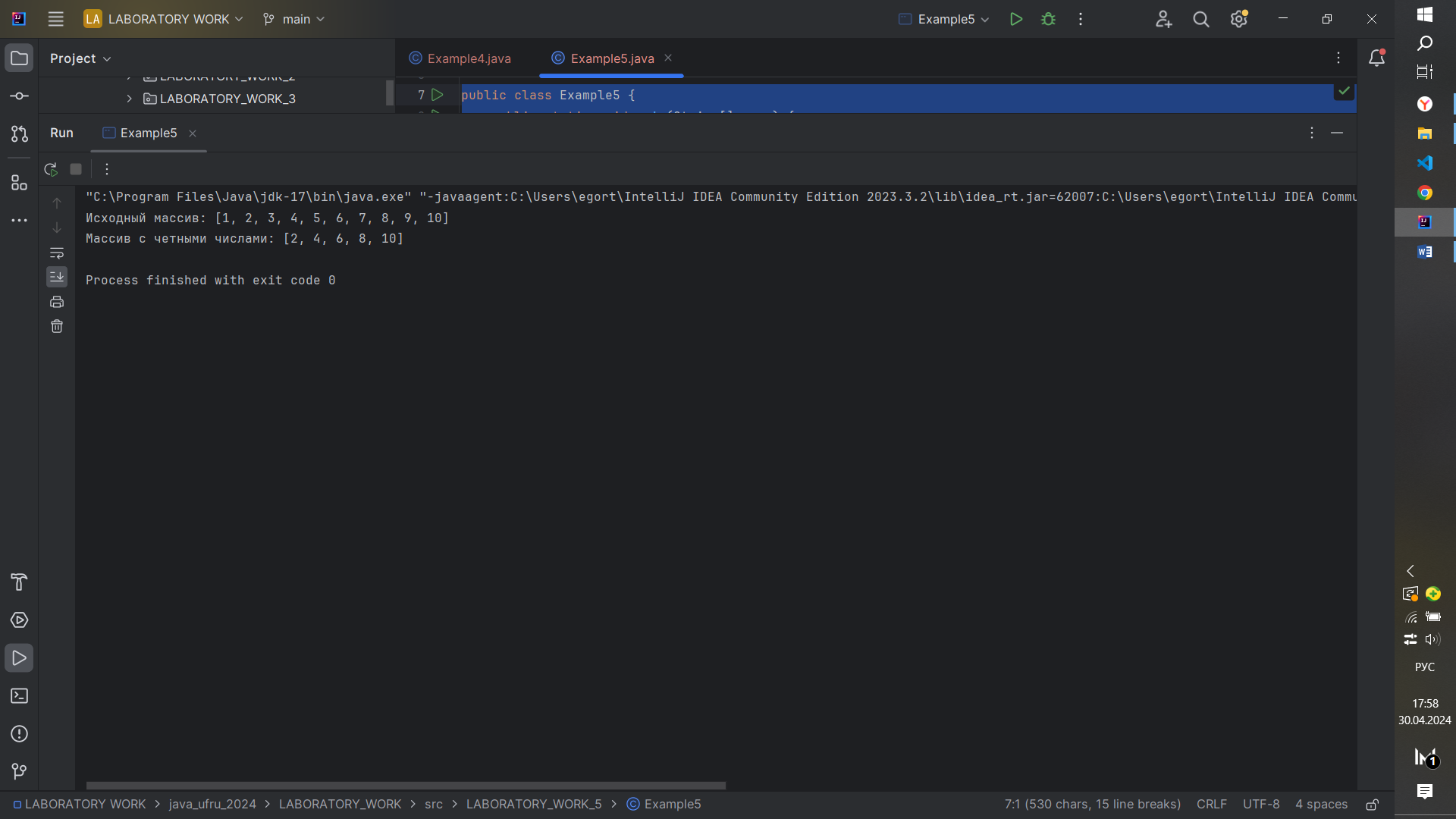
Код создает массив целых чисел от 1 до 10, фильтрует этот массив, оставляя только четные числа, и выводит оба массива на консоль.

В методе main:

* Определяется массив numbers с числами от 1 до 10.
* Вызывается метод filterEvenNumbers для получения массива с четными числами из numbers.
* Выводится на консоль исходный массив numbers и массив evenNumbers с четными числами.

Метод filterEvenNumbers:

* Принимает массив целых чисел.
* Возвращает новый массив, содержащий только четные числа из исходного массива, используя java.util.stream для фильтрации и преобразования в новый массив.



Example6

public class Example6 {  
 public static void main(String[] args) {  
 int[] array1 = {1, 2, 3, 4, 5};  
 int[] array2 = {3, 4, 5, 6, 7};  
 int[] commonElements = *findCommonElements*(array1, array2);  
  
 System.*out*.println("Первый массив: " + Arrays.*toString*(array1));  
 System.*out*.println("Второй массив: " + Arrays.*toString*(array2));  
 System.*out*.println("Общие элементы: " + Arrays.*toString*(commonElements));  
 }  
  
 private static int[] findCommonElements(int[] array1, int[] array2) {  
 Set<Integer> set1 = Arrays.*stream*(array1).boxed().collect(Collectors.*toSet*());  
 Set<Integer> set2 = Arrays.*stream*(array2).boxed().collect(Collectors.*toSet*());  
 Set<Integer> commonSet = new HashSet<>(set1);  
 commonSet.retainAll(set2);  
 return commonSet.stream().mapToInt(Integer::intValue).toArray();  
 }  
}

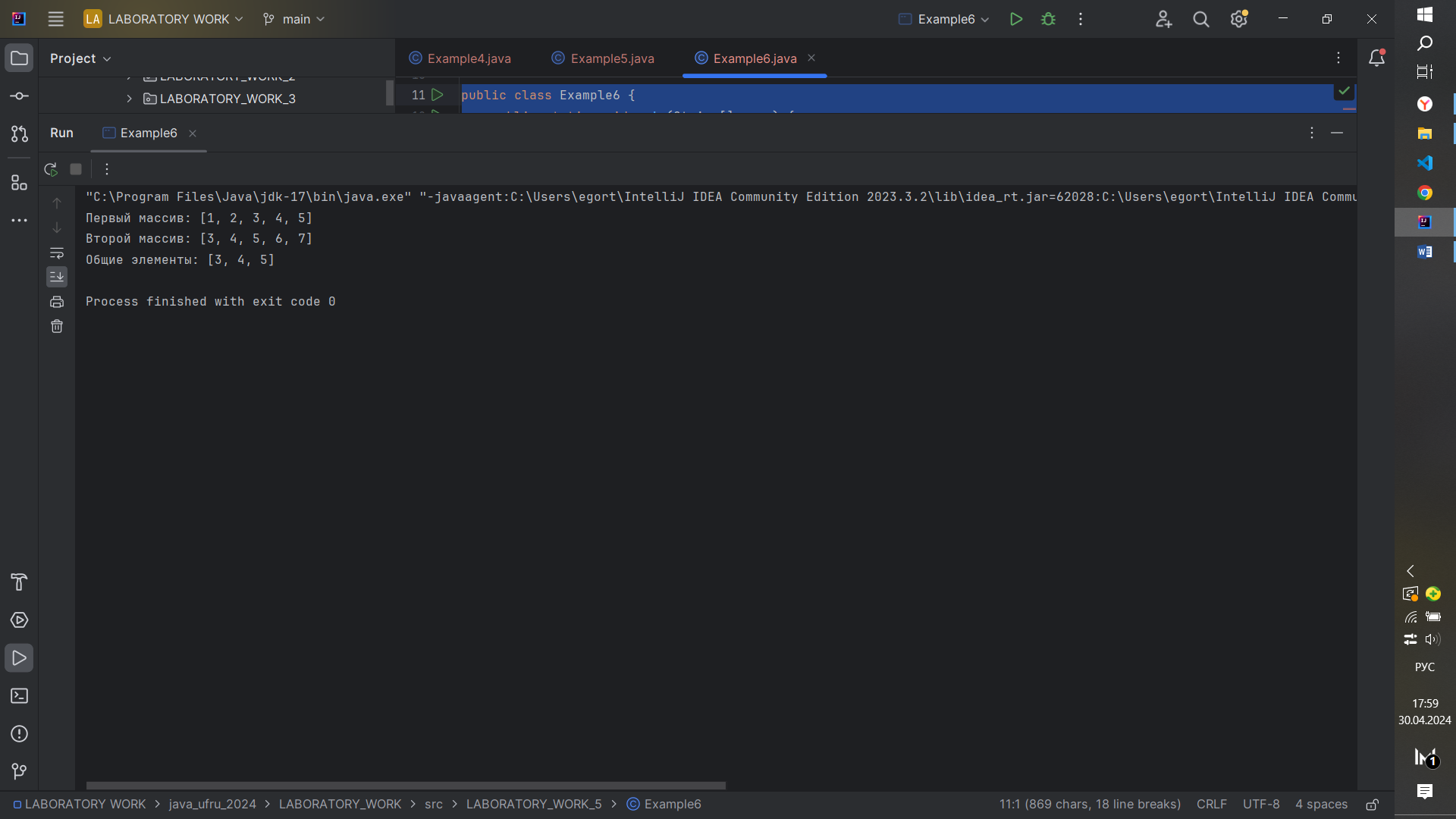
Код создает два массива целых чисел, находит общие элементы между этими массивами, создает новый массив с общими элементами и выводит все три массива на консоль.

В методе main:

* Определяются два массива array1 и array2 с числами.
* Вызывается метод findCommonElements для получения массива с общими элементами из array1 и array2.
* Выводится на консоль первый массив array1, второй массив array2 и массив commonElements с общими элементами.

Метод findCommonElements:

* Принимает два массива целых чисел.
* Преобразует каждый массив в множество с помощью java.util.stream.
* Создает третье множество commonSet из первого множества и выполняет операцию retainAll для нахождения общих элементов с помощью второго множества.
* Преобразует множество commonSet обратно в поток, а затем в массив с помощью mapToInt и toArray, и возвращает этот массив.



Example7

public class Example7 {  
 public static void main(String[] args) {  
 List<String> strings = Arrays.*asList*("Привет", "hello", "VR", "vK", "JAVA");  
  
 List<String> capitalizedStrings = *filterCapitalized*(strings);  
  
 System.*out*.println("Исходный список: " + strings);  
 System.*out*.println("Отфильтрованные строки (c большой буквы): " + capitalizedStrings);  
 }  
  
 private static List<String> filterCapitalized(List<String> strings) {  
 return strings.stream()  
 .filter(s -> Character.*isUpperCase*(s.charAt(0)))  
 .collect(Collectors.*toList*());  
 }  
}

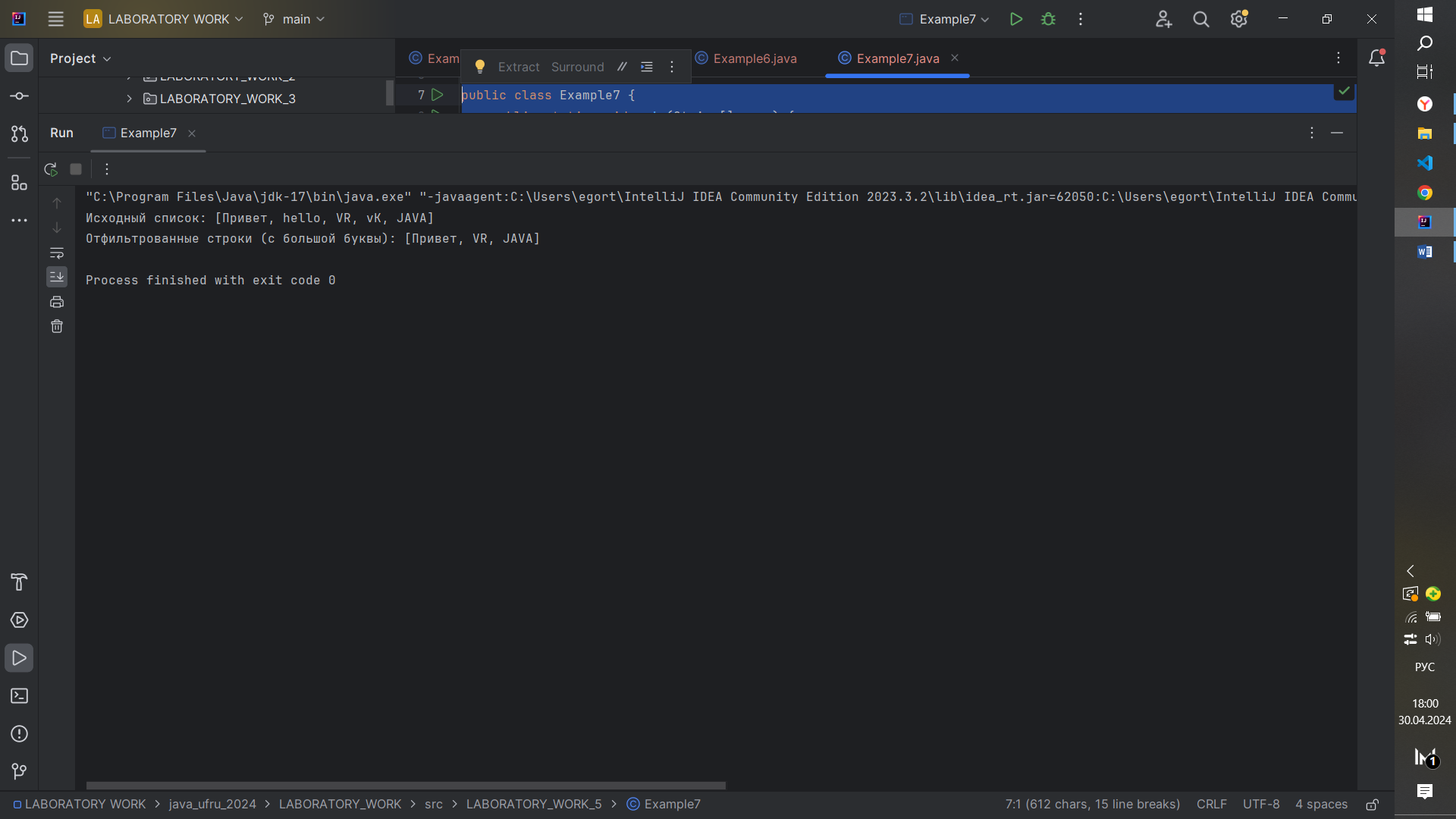
Код создает список строк, фильтрует этот список, оставляя только те строки, которые начинаются с заглавной буквы, и выводит оба списка на консоль.

В методе main:

* Определяется список strings с различными строками.
* Вызывается метод filterCapitalized для получения списка с строками, начинающимися с заглавной буквы.
* Выводится на консоль исходный список strings и список capitalizedStrings с отфильтрованными строками.

Метод filterCapitalized:

* Принимает список строк.
* Возвращает новый список, содержащий только те строки, которые начинаются с заглавной буквы, используя java.util.stream для фильтрации и сбора результатов в новый список.



Example8

public class Example8 {  
 public static void main(String[] args) {  
 List<Integer> numbers = Arrays.*asList*(11, 22, 13, 34, 55);  
 List<Integer> squares = *calculateSquares*(numbers);  
  
 System.*out*.println("Исходный список: " + numbers);  
 System.*out*.println("Список квадратов чисел: " + squares);  
 }  
  
 private static List<Integer> calculateSquares(List<Integer> numbers) {  
 return numbers.stream()  
 .map(x -> x \* x)  
 .collect(Collectors.*toList*());  
 }  
}

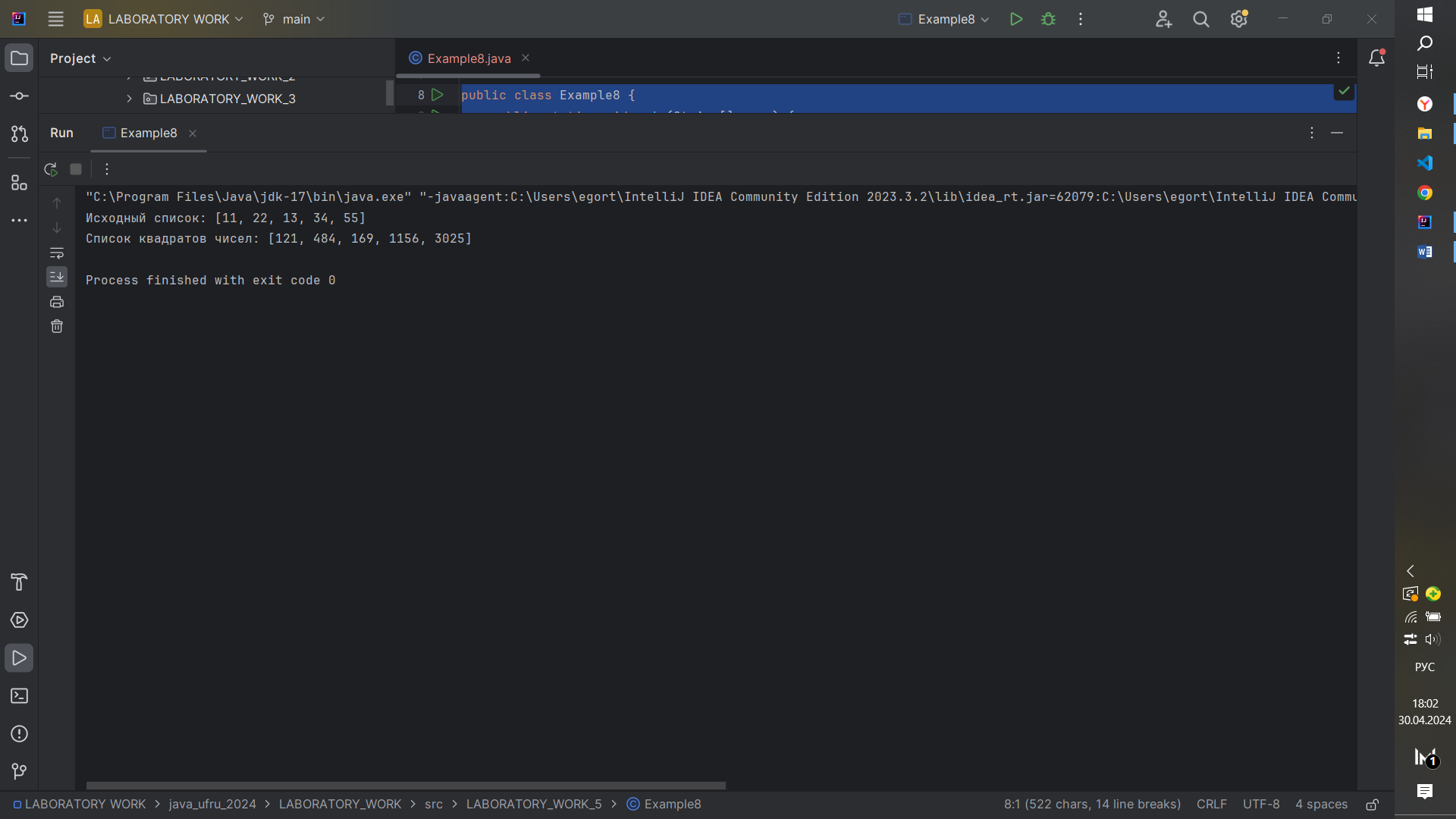
Код создает список целых чисел, вычисляет квадраты каждого числа, создает новый список с квадратами чисел и выводит оба списка на консоль.

В методе main:

* Определяется список numbers с целыми числами.
* Вызывается метод calculateSquares для получения списка с квадратами чисел из numbers.
* Выводится на консоль исходный список numbers и список squares с квадратами чисел.

Метод calculateSquares:

* Принимает список целых чисел.
* Возвращает новый список, содержащий квадраты каждого числа из исходного списка, используя java.util.stream для преобразования и сбора результатов в новый список.



Example9

public class Example9 {  
 public static void main(String[] args) {  
 List<String> strings = Arrays.*asList*("cat", "dog", "elephant", "lion", "tiger");  
 String substring = "ele";  
 List<String> filteredStrings = *filterStringsContainingSubstring*(strings, substring);  
  
 System.*out*.println("Исходный список: " + strings);  
 System.*out*.println("Отфильтрованные строки: " + filteredStrings);  
 }  
  
 public static List<String> filterStringsContainingSubstring(List<String> strings, String substring) {  
 return strings.stream()  
 .filter(s -> s.contains(substring))  
 .collect(Collectors.*toList*());  
 }  
}

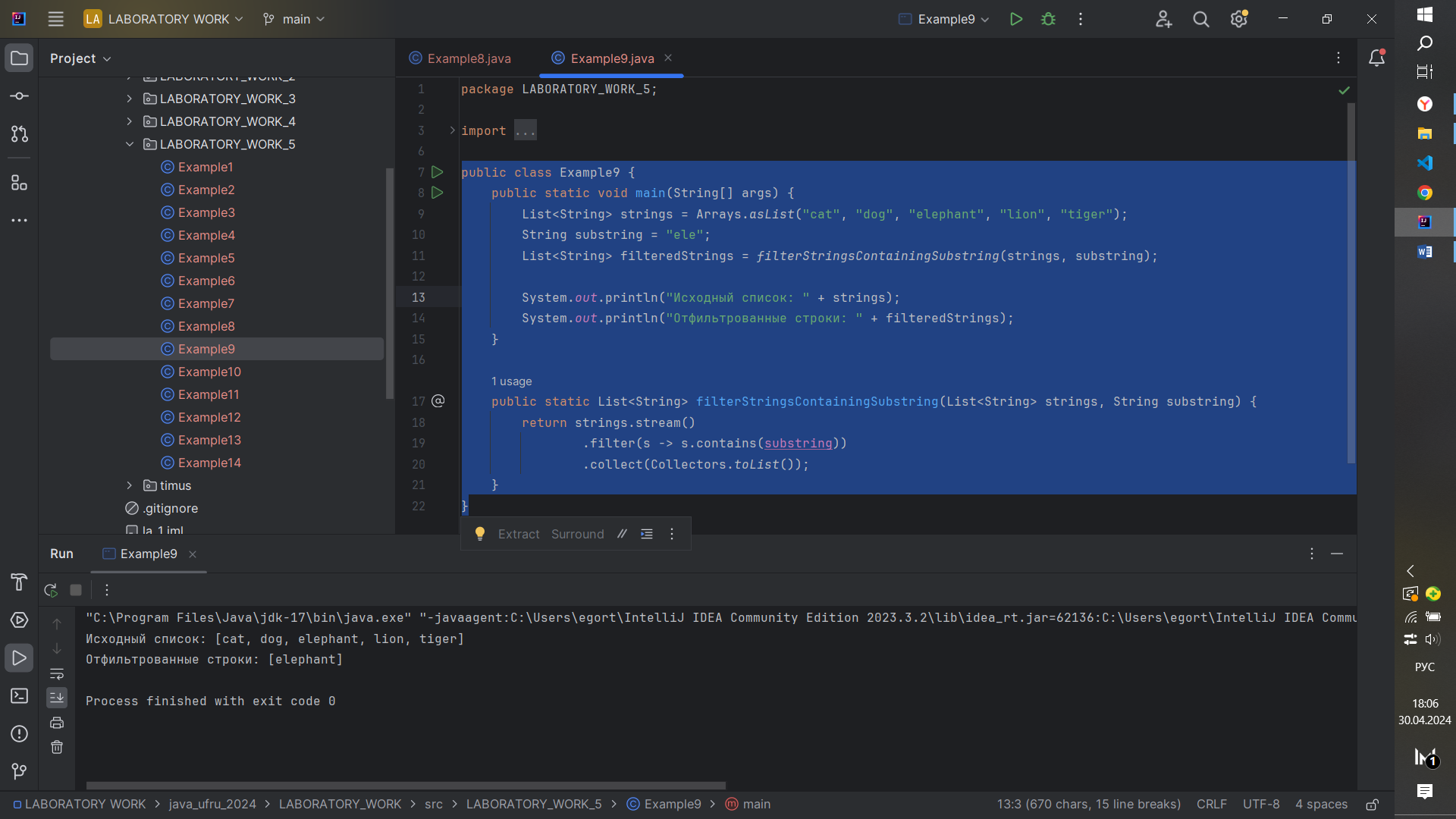
Код создает список строк, фильтрует этот список, оставляя только те строки, которые содержат заданную подстроку, и выводит оба списка на консоль.

В методе main:

* Определяется список strings со строками.
* Определяется подстрока substring.
* Вызывается метод filterStringsContainingSubstring для получения списка с строками, содержащими заданную подстроку.
* Выводится на консоль исходный список strings и список filteredStrings с отфильтрованными строками.

Метод filterStringsContainingSubstring:

* Принимает список строк и подстроку.
* Возвращает новый список, содержащий только те строки, которые содержат заданную подстроку, используя java.util.stream для фильтрации и сбора результатов в новый список.



Example10

public class Example10 {  
 public static void main(String[] args) {  
 List<Integer> numbers = Arrays.*asList*(15, 21, 32, 48, 55, 66, 77, 84, 99, 105);  
 int divisor = 7;  
 List<Integer> divisibleNumbers = *filterDivisibleNumbers*(numbers, divisor);  
  
 System.*out*.println("Исходный список чисел: " + numbers);  
 System.*out*.println("Числа, делящиеся на " + divisor + " без остатка: " + divisibleNumbers);  
 }  
  
 public static List<Integer> filterDivisibleNumbers(List<Integer> numbers, int divisor) {  
 return numbers.stream()  
 .filter(n -> n % divisor == 0)  
 .collect(Collectors.*toList*());  
 }  
}

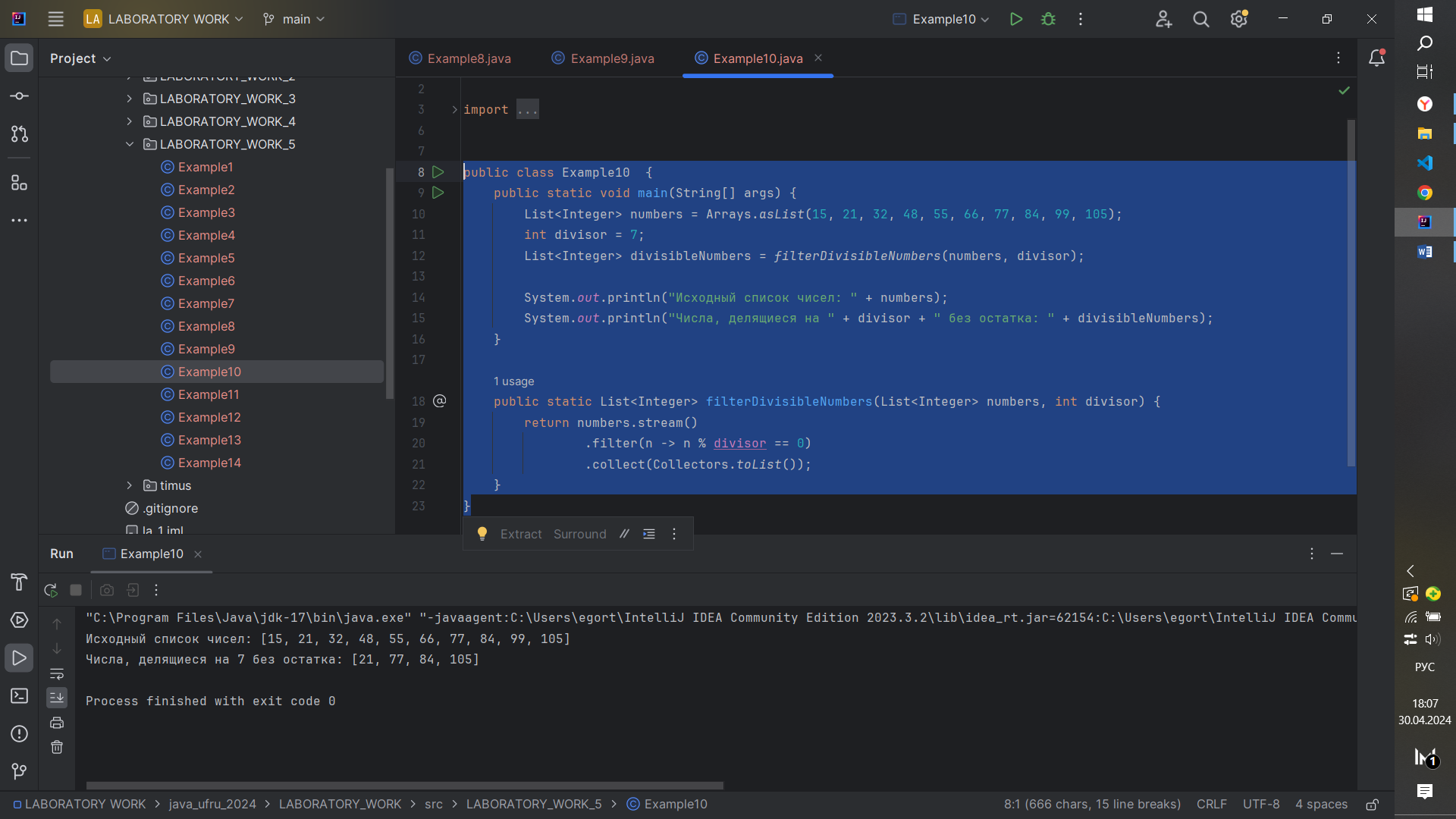
Код создает список целых чисел, фильтрует этот список, оставляя только те числа, которые делятся на заданное число без остатка, и выводит оба списка на консоль.

В методе main:

* Определяется список numbers с целыми числами.
* Определяется число divisor, на которое будут проверяться делимость чисел.
* Вызывается метод filterDivisibleNumbers для получения списка с числами, делящимися на divisor без остатка.
* Выводится на консоль исходный список numbers и список divisibleNumbers с числами, делящимися на divisor без остатка.

Метод filterDivisibleNumbers:

* Принимает список целых чисел и число divisor.
* Возвращает новый список, содержащий только те числа, которые делятся на divisor без остатка, используя java.util.stream для фильтрации и сбора результатов в новый список.



Example11

public class Example11 {  
 public static void main(String[] args) {  
 List<String> strings = Arrays.*asList*("cat", "elephant", "giraffe", "kangaroo", "lion", "rhinoceros", "tiger", "whale", "zebra");  
 int minLength = 7;  
 List<String> longStrings = *filterLongStrings*(strings, minLength);  
  
 System.*out*.println("Исходный список: " + strings);  
 System.*out*.println("Строки длиной больше " + minLength + ": " + longStrings);  
 }  
  
 public static List<String> filterLongStrings(List<String> strings, int minLength) {  
 return strings.stream()  
 .filter(s -> s.length() > minLength)  
 .collect(Collectors.*toList*());  
 }  
}

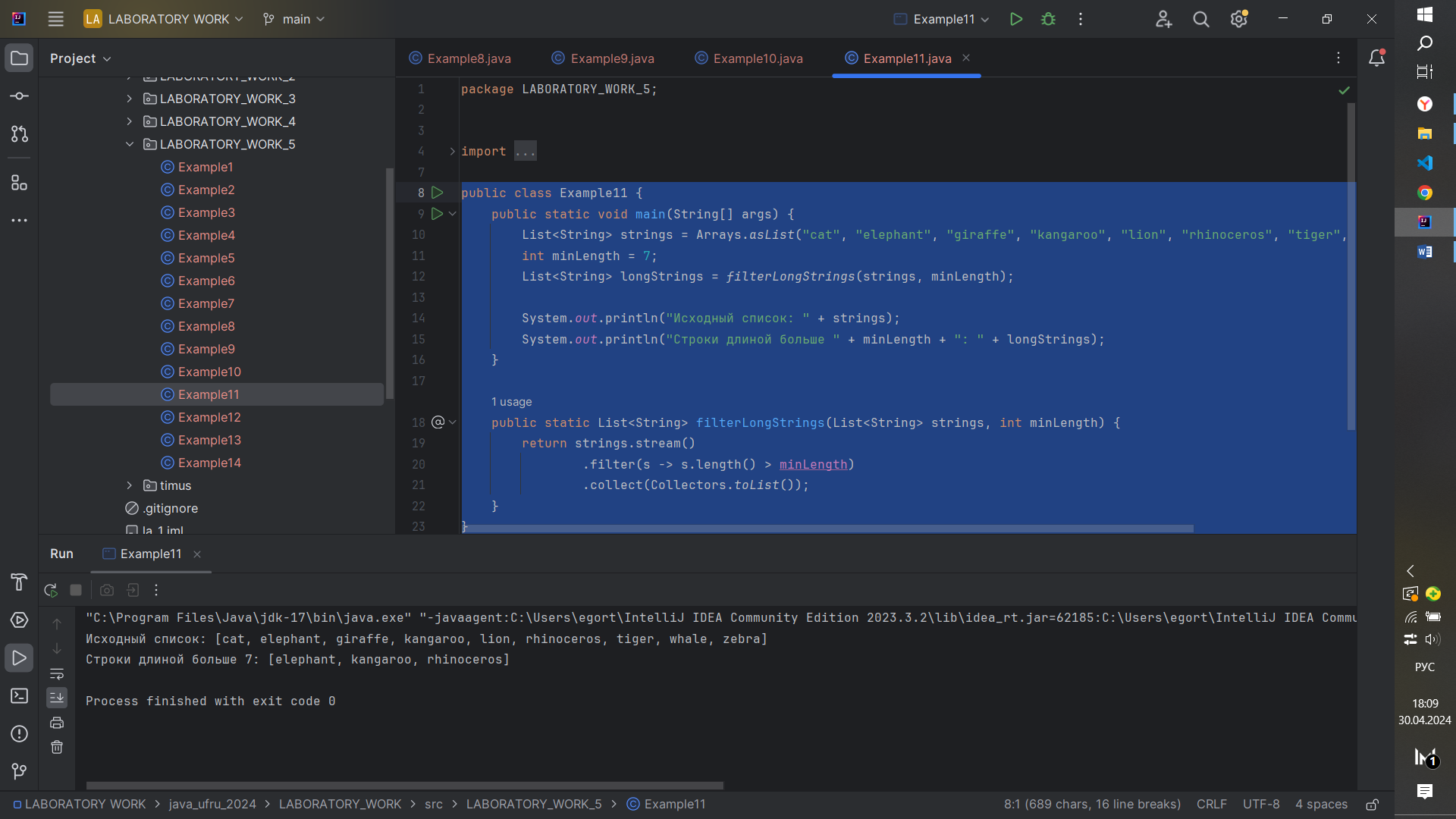
Код создает список строк, фильтрует этот список, оставляя только те строки, которые имеют длину больше заданного значения, и выводит оба списка на консоль.

В методе main:

* Определяется список strings со строками.
* Определяется минимальная длина minLength, которая будет использоваться для фильтрации строк.
* Вызывается метод filterLongStrings для получения списка с длинными строками, длина которых больше minLength.
* Выводится на консоль исходный список strings и список longStrings с длинными строками.

Метод filterLongStrings:

* Принимает список строк и минимальную длину minLength.
* Возвращает новый список, содержащий только те строки, которые имеют длину больше minLength, используя java.util.stream для фильтрации и сбора результатов в новый список.



Example12

public class Example12 {  
 public static void main(String[] args) {  
 List<Integer> numbers = Arrays.*asList*(10, 25, 30, 47, 55, 60, 72, 88, 99, 105);  
 int threshold = 50;  
 List<Integer> greaterNumbers = *filterGreaterNumbers*(numbers, threshold);  
  
 System.*out*.println("Исходный список: " + numbers);  
 System.*out*.println("Числа больше " + threshold + ": " + greaterNumbers);  
 }  
  
 public static List<Integer> filterGreaterNumbers(List<Integer> numbers, int threshold) {  
 return numbers.stream()  
 .filter(n -> n > threshold)  
 .collect(Collectors.*toList*());  
 }  
}

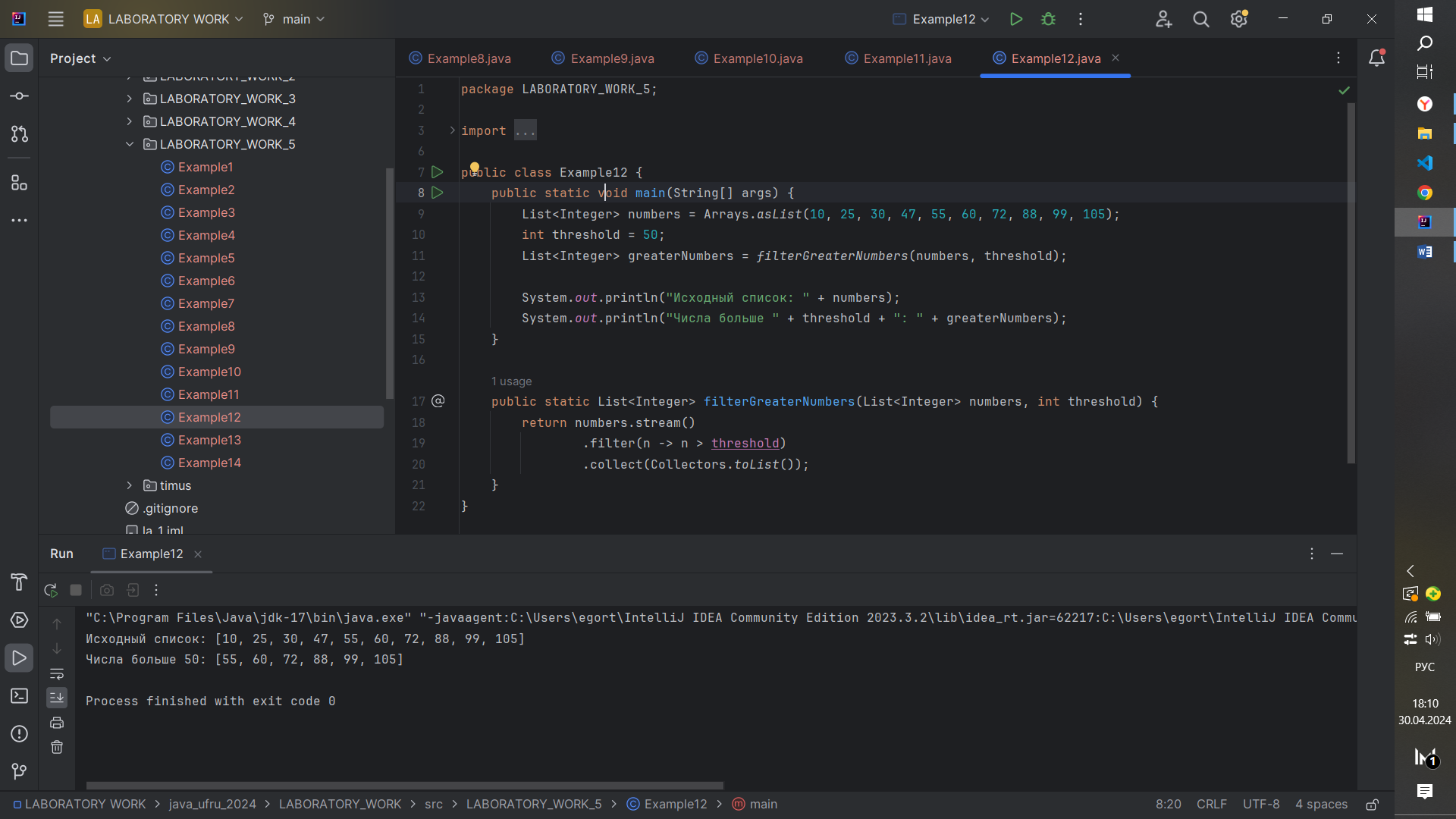
Код в классе Example12 создает список целых чисел, фильтрует этот список, оставляя только те числа, которые больше заданного значения, и выводит оба списка на консоль.

В методе main:

* Определяется список numbers с целыми числами.
* Определяется пороговое значение threshold.
* Вызывается метод filterGreaterNumbers для получения списка с числами, которые больше threshold.
* Выводится на консоль исходный список numbers и список greaterNumbers с числами, большими threshold.

Метод filterGreaterNumbers:

* Принимает список целых чисел и пороговое значение threshold.
* Возвращает новый список, содержащий только те числа, которые больше threshold, используя java.util.stream для фильтрации и сбора результатов в новый список.



Example13

public class Example13 {  
 public static void main(String[] args) {  
 List<String> strings = Arrays.*asList*("apple", "banana123", "cherry", "date!", "elderberry", "fig", "grape5", "honey", "igloo", "jungle", "213");  
 List<String> letterOnlyStrings = *filterLetterOnlyStrings*(strings);  
  
 System.*out*.println("Исходный список: " + strings);  
 System.*out*.println("Строки, содержащие только буквы: " + letterOnlyStrings);  
 }  
  
 public static List<String> filterLetterOnlyStrings(List<String> strings) {  
 return strings.stream()  
 .filter(s -> s.matches("[a-zA-Z]+"))  
 .collect(Collectors.*toList*());  
 }  
}

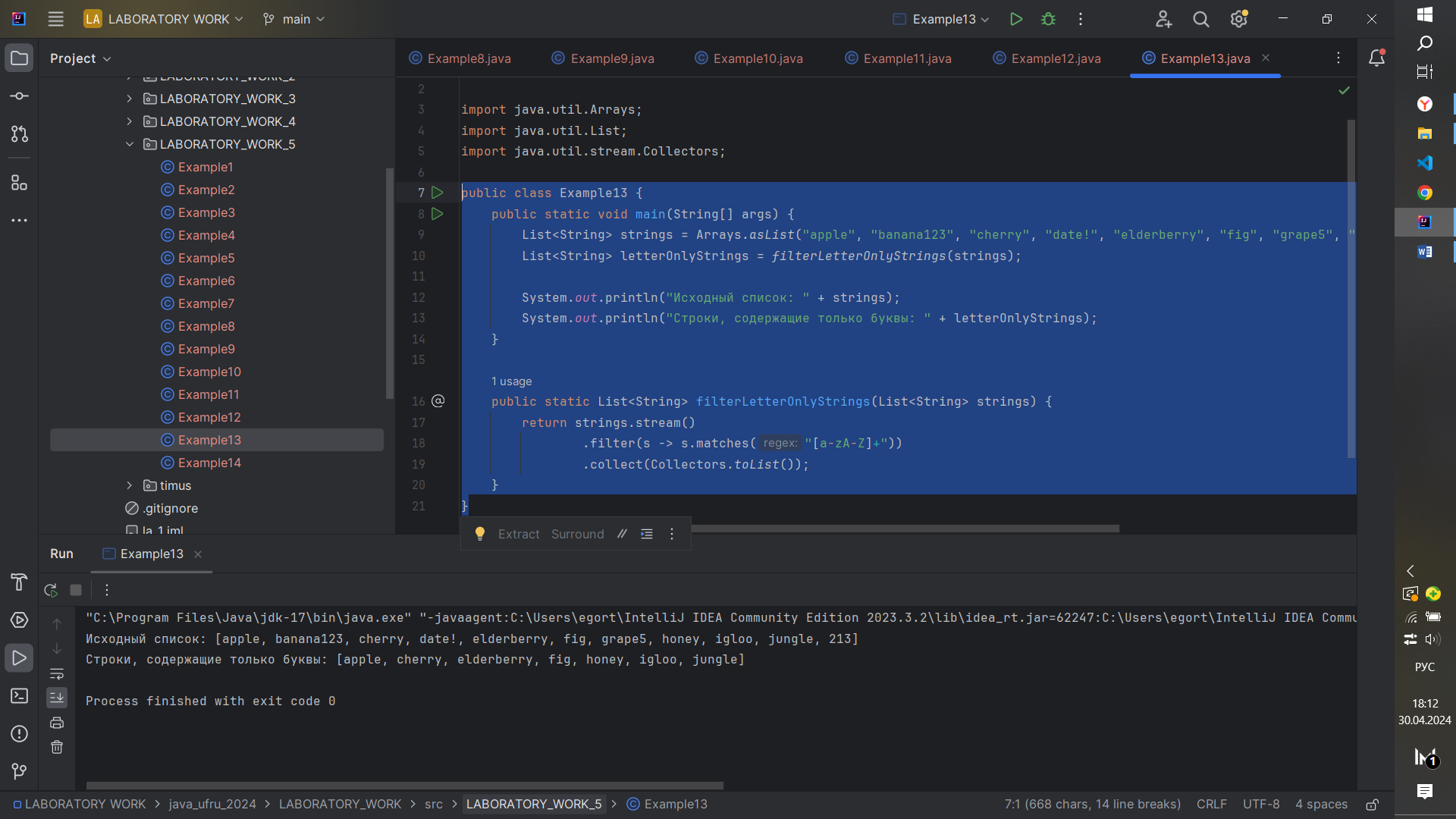
Код в классе Example13 создает список строк, фильтрует этот список, оставляя только те строки, которые содержат только буквы (без цифр и специальных символов), и выводит оба списка на консоль.

В методе main:

* Определяется список strings со строками, включающими буквы, цифры и специальные символы.
* Вызывается метод filterLetterOnlyStrings для получения списка с строками, содержащими только буквы.
* Выводится на консоль исходный список strings и список letterOnlyStrings с строками, состоящими только из букв.

Метод filterLetterOnlyStrings:

* Принимает список строк.
* Возвращает новый список, содержащий только те строки, которые состоят только из букв, используя регулярное выражение [a-zA-Z]+ для фильтрации и сбора результатов в новый список.



Example14

public class Example14 {  
 public static void main(String[] args) {  
 List<Integer> numbers = Arrays.*asList*(10, 25, 30, 47, 55, 60, 72, 88, 99, 105);  
 int threshold = 50;  
 List<Integer> lesserNumbers = *filterLesserNumbers*(numbers, threshold);  
  
 System.*out*.println("Исходный список: " + numbers);  
 System.*out*.println("Числа меньше " + threshold + ": " + lesserNumbers);  
 }  
  
 public static List<Integer> filterLesserNumbers(List<Integer> numbers, int threshold) {  
 return numbers.stream()  
 .filter(n -> n < threshold)  
 .collect(Collectors.*toList*());  
 }  
}

Код создает список целых чисел, фильтрует этот список, оставляя только те числа, которые меньше заданного значения, и выводит оба списка на консоль.

В методе main:

* Определяется список numbers с целыми числами.
* Определяется пороговое значение threshold.
* Вызывается метод filterLesserNumbers для получения списка с числами, которые меньше threshold.
* Выводится на консоль исходный список numbers и список lesserNumbers с числами, меньшими threshold.

Метод filterLesserNumbers:

* Принимает список целых чисел и пороговое значение threshold.
* Возвращает новый список, содержащий только те числа, которые меньше threshold, используя java.util.stream для фильтрации и сбора результатов в новый список.

