Stream API

Gesamtpunktzahl 20/24

E-Mail-Adresse *

ewebotah@gmail.com

✗ Что такое функциональный интерфейс в Java? 0/1

- ○ a) Интерфейс с множеством абстрактных методов.
- ◉ b) Интерфейс с одним абстрактным методом. ✗
- ○ c) Интерфейс с одним абстрактным методом и может иметь множество методов с реализацией по умолчанию.
- ○ d) Интерфейс с одним статическим методом.

Richtige Antwort

◉ c) Интерфейс с одним абстрактным методом и может иметь множество методов с реализацией по умолчанию.

Feedback

Функциональный интерфейс в Java - это интерфейс, который имеет ровно один абстрактный метод, и может иметь множество методов с реализацией по умолчанию или статических методов.

✓ Какой метод используется для фильтрации элементов в Stream API? 1/1

- a) map
- b) filter ✓
- c) reduce
- d) collect

Feedback

Метод filter используется для фильтрации элементов стрима на основе заданного условия.

✓ Какой метод Stream API в Java возвращает количество элементов в стриме? 1/1

- a) map
- b) filter
- c) count ✓
- d) collect

Feedback

Пояснение: Метод count возвращает количество элементов в стриме.

✓ Что такое метод-ссылка в Java? 1/1

- a) Способ создания нового метода.
- b) Способ передать ссылку на метод или конструктор. ✓
- c) Способ создать новый объект.
- d) Способ вызвать метод.

Feedback

Метод-ссылка в Java позволяет ссылаться на методы или конструкторы без вызова их.

✓ Какой предопределенный функциональный интерфейс используется для представления операций, которые принимают два аргумента и возвращают результат? 1/1

- a) UnaryOperator
- b) Predicate
- c) BiFunction ✓
- d) Consumer

Feedback

Пояснение: Интерфейс BiFunction представляет операции, которые принимают два аргумента и возвращают результат.

✓ Как создать стрим из коллекции в Java? 1/1

- a) Stream.of(collection)
- b) Stream.create(collection)
- c) collection.stream() ✓
- d) new Stream<>(collection)

Feedback

Пояснение: Для создания стрима из коллекции используется метод collection.stream().

✓ Какой метод является терминальным в Stream API? 1/1

- a) map
- b) filter
- c) flatMap
- d) collect ✓

Feedback

Метод collect является терминальным методом, который трансформирует стрим в другую форму, например, в коллекцию.

✓ Какой из перечисленных методов является конвейерным в Stream API? 1/1

- a) forEach
- b) map ✓
- c) collect
- d) count

Feedback

Метод map является конвейерным методом, который преобразует каждый элемент стрима.

✗ Какой класс используется для чтения байтов из файла в Java? 0/1

- a) FileWriter
- b) PrintWriter
- c) FileInputStream
- d) FileOutput ✗

Richtige Antwort

- c) FileInputStream

Feedback

Пояснение: Класс FileInputStream используется для чтения байтов из файла.

✓ Какой класс используется для записи текста в файл в Java? 1/1

- a) FileInputStream
- b) FileWriter ✓
- c) FileReader
- d) InputStream

Feedback

Пояснение: Класс FileWriter используется для записи текста в файл в Java.

✓ Что будет выведено на экран после выполнения следующего кода? 1/1

```
Function<Integer, String> func = (num) -> "Number: " + num;
System.out.println(func.apply(5));
```

- a) Number
- b) 5
- c) Number: 5 ✓
- d) Ошибка компиляции

Feedback

Пояснение: Функция func принимает целое число и возвращает строку, представляющую это число с префиксом "Number: ".

✗ Что будет выведено на экран после выполнения следующего кода? 0/1

```
List<String> words = Arrays.asList("apple", "banana", "cherry");
long count = words.stream().filter(word -> word.length() > 5).count();
System.out.println(count);
```

- a) 2
- b) 3 ✗
- c) 0
- d) 1

Richtige Antwort

- a) 2

Feedback

Пояснение: Фильтр убирает слова с длиной меньше или равной 5, оставляя только "banana" и "cherry". Счетчик возвращает 2.

✓ Что будет выведено на экран после выполнения следующего кода? 1/1

```
List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5);
List<Integer> evenNumbers = numbers.stream().filter(n -> n % 2 ==
0).collect(Collectors.toList());
System.out.println(evenNumbers);
```

- a) [1, 2, 3, 4, 5]
- b) [1, 3, 5]
- c) [2, 4] ✓
- d) [1, 2, 3]

Feedback

Пояснение: Фильтр убирает нечетные числа, оставляя только четные числа 2 и 4.

✓ Что будет выведено на экран после выполнения следующего кода? 1/1

```
List<String> words = Arrays.asList("apple", "banana", "cherry");
String result =
words.stream().map(String::toUpperCase).collect(Collectors.joining(", "));
System.out.println(result);
```

- a) apple, banana, cherry
- b) APPLE, BANANA, CHERRY ✓
- c) APPLE; BANANA; CHERRY
- d) Ошибка компиляции

Feedback

Пояснение: Метод map преобразует каждое слово в верхний регистр, а collect соединяет их в одну строку, разделенную запятыми и пробелами.

✓ Что будет выведено на экран после выполнения следующего кода? 1/1

```
List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5); int sum =
numbers.stream().reduce(0, (a, b) -> a + b); System.out.println(sum);
```

- a) 0
- b) 15 ✓
- c) 10
- d) Ошибка компиляции

Feedback

Пояснение: Метод reduce суммирует все числа в списке, начиная с начального значения 0, результатом является 15.

✓ Что будет выведено на экран после выполнения следующего кода? 1/1

```
Predicate<String> predicate = s -> s.length() > 3;
System.out.println(predicate.test("Hello"));
```

- a) false
- b) true ✓
- c) Ошибка компиляции
- d) Ничего

Feedback

Пояснение: Предикат тестирует, больше ли длина строки 3, и возвращает true, так как длина "Hello" - 5.

✓ Что будет выведено на экран после выполнения следующего кода? 1/1

```
Stream.of("one", "two", "three").forEach(System.out::println);
```

- a) onetwothree
- b) Ошибка компиляции
- c) one two three ✓
- d) Ничего

Feedback

Пояснение: Метод forEach проходит по каждому элементу стрима и вызывает System.out.println для каждого элемента.

✓ Что будет выведено на экран после выполнения следующего кода? 1/1

```
String text = "apple";
Consumer<String> consumer = System.out::println;
consumer.accept(text);
```

- a) text
- b) apple ✓
- c) Ошибка компиляции
- d) Ничего

Feedback

Пояснение: Потребитель consumer принимает строку и печатает ее на экран.

✓ Что будет выведено на экран после выполнения следующего кода? 1/1

```
String result = Stream.of("a", "b", "c").reduce("", (s1, s2) -> s1 + s2);
System.out.println(result);
```

- a) a
- b) abc ✓
- c) a b c
- d) Ошибка компиляции

Feedback

Пояснение: Метод reduce объединяет все строки в стриме в одну строку "abc".

✓ Что будет выведено на экран после выполнения следующего кода? 1/1

```
List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5);
List<Integer> squared = numbers.stream().map(n -> n *
n).collect(Collectors.toList());
System.out.println(squared);
```

- a) [1, 2, 3, 4, 5]
- b) [0, 1, 2, 3, 4]
- c) [1, 4, 9, 16, 25] ✓
- d) [2, 4, 6, 8, 10]

Feedback

Пояснение: Метод map возводит каждое число в квадрат, и collect собирает результат в новый список.

✗ Перепишите следующий цикл с использованием Stream API: 0/1

```
List<String> list = Arrays.asList("a", "b", "c");
for (String s : list) {
    if (s.equals("a")) {
        System.out.println(s.toUpperCase());
    }
}
```

- a) list.stream().filter(s -> s.equals("a")).forEach(s -> System.out.println(s.toUpperCase()));
- b) list.stream().map(String::toUpperCase).filter(s -> s.equals("a")).forEach(System.out::println);
- c) list.stream().forEach(s -> System.out.println(s.toUpperCase()));
- d) list.stream().filter(s -> s.equals("a")).map(String::toUpperCase).forEach(System.out::println); ✗

Richtige Antwort

- a) list.stream().filter(s -> s.equals("a")).forEach(s -> System.out.println(s.toUpperCase()));

Feedback

Пояснение: Опция a правильно переписывает данный цикл, фильтруя список так, чтобы оставить только строку "a", а затем печатает ее в верхнем регистре.

✓ Перепишите следующий цикл с использованием Stream API: 1/1

```
List<Integer> list = Arrays.asList(1, 2, 3, 4, 5);
List<Integer> evenNumbers = new ArrayList<>();
for (Integer i : list) {
    if (i % 2 == 0) {
        evenNumbers.add(i);
    }
}
```

- (●) a) `List<Integer> evenNumbers = list.stream().filter(i -> i % 2 == 0).collect(Collectors.toList());` ✓
- ( ) b) `List<Integer> evenNumbers = list.stream().map(i -> i % 2 == 0).collect(Collectors.toList());`
- ( ) c) `List<Integer> evenNumbers = list.stream().forEach(i -> i % 2 == 0).collect(Collectors.toList());`
- ( ) d) `List<Integer> evenNumbers = list.stream().filter(i -> i % 2 == 1).collect(Collectors.toList());`

Feedback

Пояснение: Опция a правильно переписывает данный цикл, фильтруя список так, чтобы оставить только четные числа.

✓ Перепишите следующий цикл с использованием Stream API: 1/1

```
List<Integer> list = Arrays.asList(1, 2, 3, 4, 5);
int sum = 0;
for (Integer i : list) {
    sum += i;
}
```

- a) int sum = list.stream().mapToInt(i -> i).sum();
- b) int sum = list.stream().reduce(0, (a, b) -> a + b); ✓
- c) int sum = list.stream().forEach(i -> i + i);
- d) int sum = list.stream().map(i -> i + i).sum();

Feedback

Опция b правильно переписывает данный цикл, используя метод reduce для суммирования всех чисел в списке.

✓ Перепишите следующий цикл с использованием Stream API: 1/1

```
List<String> list = Arrays.asList("a", "b", "c", "d");
List<String> selected = new ArrayList<>();
for (String s : list) {
    if (s.compareTo("b") > 0) {
        selected.add(s);
    }
}
```

- (●) a) `List<String> selected = list.stream().filter(s -> s.compareTo("b") > 0).collect(Collectors.toList());` ✓
- ( ) b) `List<String> selected = list.stream().map(s -> s.compareTo("b") > 0).collect(Collectors.toList());`
- ( ) c) `List<String> selected = list.stream().filter(s -> s.compareTo("b") < 0).collect(Collectors.toList());`
- ( ) d) `List<String> selected = list.stream().forEach(s -> s.compareTo("b") > 0).collect(Collectors.toList());`

Feedback

Пояснение: Опция a правильно переписывает данный цикл, используя метод filter для фильтрации строк, которые больше строки "