STREAMS – СТРИМЫ

**Stream API** это новый (с Java8) способ работать со структурами данных в функциональном стиле. Чаще всего с помощью stream в Java работают с коллекциями, но на самом деле этот механизм может использоваться для самых различных данных.

**Способы создания Stream:**

1. *Классический: Создание стрима из коллекции*

Collection<String> collection = Arrays.asList("a1", "a2", "a3");

Stream<String> streamFromCollection = collection.stream();

*// Создание стрима из коллекции*

Collection<String> collection = Arrays.asList("a1", "a2", "a3");

Stream<String> streamFromCollection = collection.stream();

System.out.println("streamFromCollection = " + streamFromCollection.collect(Collectors.toList()));

*// напечатает streamFromCollection = [a1, a2, a3]*

1. *Создание стрима из значений*

Stream<String> streamFromValues = Stream.of("a1", "a2", "a3");

*// Создание стрима из значений*

Stream<String> streamFromValues = Stream.of("a1", "a2", "a3");

System.out.println("streamFromValues = " + streamFromValues.collect(Collectors.toList()));

*// напечатает streamFromValues = [a1, a2, a3]*

1. *Создание стрима из массива*

String[] array = {"a1","a2","a3"};

Stream<String> streamFromArrays = Arrays.stream(array);

*// Создание стрима из массива*

String[] array = {"a1","a2","a3"};

Stream<String> streamFromArrays = Arrays.stream(array);

System.out.println("streamFromArrays = " + streamFromArrays.collect(Collectors.toList()));

*// напечатает streamFromArrays = [a1, a2, a3]*

или

Stream<String> streamFromArrays1 = Stream.of(array);

System.out.println("streamFromArrays1 = " + streamFromArrays1.collect(Collectors.toList()));

*// напечатает streamFromArrays = [a1, a2, a3]*

1. *Создание стрима из файла (каждая строка в файле будет отдельным элементом в стриме)*

Stream<String> streamFromFiles = Files.lines(Paths.get("file.txt"));

*// Создание стрима из файла (каждая запись в файле будет отдельной строкой в стриме)*

File file = new File("1.tmp");

file.deleteOnExit();

PrintWriter out = new PrintWriter(file);

out.println("a1");

out.println("a2");

out.println("a3");

out.close();

Stream<String> streamFromFiles = Files.lines(Paths.get(file.getAbsolutePath())); System.out.println("streamFromFiles = " + streamFromFiles.collect(Collectors.toList())); *// напечатает streamFromFiles = [a1, a2, a3]*

1. *Создание стрима из строки*

IntStream streamFromString = "123".chars();

*// Создание числового стрима из строки*

IntStream streamFromString = "123".chars();

System.out.print("streamFromString = ");

streamFromString.forEach((e)->System.out.print(e + " , "));

*// напечатает streamFromString = 49 , 50 , 51 ,* - соответствует

1. *С помощью Stream.builder*

Stream<Object> stream = Stream.builder().add("a1").add("a2").add("a3").build();

*// С помощью Stream.builder*

Stream.Builder<String> builder = Stream.builder();

Stream<String> streamFromBuilder = builder.add("a1").add("a2").add("a3").build(); System.out.println("streamFromBuilder = " + streamFromBuilder.collect((Collectors.toList())));

*// напечатает streamFromFiles = [a1, a2, a3]*

1. *Создание бесконечного стрима с помощью Stream.iterate* (*начальное условие*, *выражение генерации*)

Stream<Integer> streamFromIterate = Stream.iterate(1, n -> n + 1);

*// С помощью Stream.iterate*

Stream<Integer> streamFromIterate = Stream.iterate(1, n -> n + 2);

System.out.println("streamFromIterate = " + streamFromIterate.limit(3).collect(Collectors.toList()));

*// напечатает streamFromIterate = [1, 3, 5]*

1. *Создание бесконечного стрима с помощью Stream.generate (выражение генерации)*

Stream<String> streamFromGenerate = Stream.generate(() -> "a1");

*// С помощью Stream.generate*

Stream<String> streamFromGenerate = Stream.generate(() -> "a1"); System.out.println("streamFromGenerate = " + streamFromGenerate.limit(3).collect(Collectors.toList()));

*// напечатает streamFromGenerate = [a1, a1, a1]*

1. *Создание параллельного стрима*

Stream<String> stream = collection.parallelStream();

*// Создать параллельный стрим из коллекции*

Stream<String> parallelStream = collection.parallelStream();

System.out.println("parallelStream = " + parallelStream.collect(Collectors.toList()));

*// напечатает parallelStream = [a1, a2, a3]*

*+10 - // Создать пустой стрим*

Stream<String> streamEmpty = Stream.empty();

System.out.println("streamEmpty = " + streamEmpty.collect(Collectors.toList()));

*// напечатает streamEmpty = []*