Java 8 Stream and Lambda Expressions – Parsing File **Example** 🎼 (/users/183661/eyalgo.html) by Eyal Golan (/users/183661/eyalgo.html) 🤻 MVB · Jan. 12, 15 · Java Zone (/java-jdk-development-9,347 Views 心 Like (0) ● Comment (0) ☆ Save ♥ Tweet The Java Zone is brought to you in partnership with JetBrains (/qo? i=106823&u=http%3A%2F%2Fblog.jetbrains.com%2Fkotlin%2F2O15%2F11%2Fthe-kotlin-language-1-O-beta-is-information and the state of the sthere%2F%3Futm_source%3Ddzonejava%26utm_medium%3Dpromolink%26utm_content%3Dlanguage_matters%26utm_campaign%3Dkotlin). Learn more about Kotlin (/go? $i=106823\&u=http\%3A\%2F\%2Fblog\ jetbrains.com\%2Fkotlin\%2F2015\%2F11\%2Fthe-kotlin-language-1-o-beta-is-here\%2F\%3Futm_source\%3Ddzonejava\%26utm_medium\%3Dpromo$ link%26utm content%3Dlanguage matters%26utm campaign%3Dkotlin), a new programming language designed by JetBrains to solve problems that software developers face every day. Recently I wanted to extract certain data from an output log. Here's part of the log file: 2015-01-06 11:33:03 b.s.d.task [INFO] Emitting: eVentToRequestsBolt _ack_ack [-6722594615019711369 -1335723027906100557] 2015-01-06 11:33:03 c.s.p.d.FackagesForvider [INFO] ===---> Londed package com.foo.bar 2015-01-06 11:33:04 b.s.d.executor [INFO] Processing received message source: eventToManageBolt:2, stream: _ack_ack, id: {}, [-6722594615019: 11360 -1335723027906100557] 2015-01-06 11:33:04 c.s.p.d.PackagesForvider [INFO] ===--> Londed package co.il.boo 2015-01-06 11:33:04 c.s.p.d.PackagesForvider [INFO] ===--> Londed package dot.org.biz I decided to do it using the Java8 Stream and Lambda Expression features Read DZONE โปลงชะ 25กาย (ฟุลงส์ะ เป็น เป็นจับยังยากายการ tutorials tools news) Stream<String> lines = Files.lines(Paths.get(args[1])); Over a million developers have joined DZone. Sign In / Join Filter relevant lines (GUIDES (GUIDES) ZONES ((PORTALS) | AGILE ((AGILE-METHODOLOGY-TRAINING-TOOLS-NEWS) I paged to got the page ages pages and write them into another file. Not all lines contained the data I need, hence filter only relevant ones. lines.filter(line -> line.contains("===---> Loaded package")) Parsing the relevant lines Then, I needed to parse the relevant lines. I did it by first splitting each line to an array of Strings and then taking the last element in that array. In other words, I did a double mapping. First a line to an array and then an array to a String. Writing to output file The last part was taking each string and write it to a file. That was the terminal operation. .forEach(package -> writeToFile(fw, package)); writeToFile is a method I created. The reason is that Java File System throws IOException. You can't use checked exceptions in lambda expressions Here's a full example (note, I don't check input) import java.io.FileWriter; import java.io.IOException; import java.nio.file.Files; import java.nio.file.Paths; import java.util.Arrays; import java.util.List; import java.util.stream.Stream; public class App { public static void main(String[] args) throws IOException { StreamString> lines = null; if (args.length = 2) { lines = Files.lines(Paths.get(args[1])); } } 00557]"; String s2 = "2015-01-06 11:33:03 b.s.d.executor [INFO] Processing received message source: eventToHanageBolt:2, stream: _ack_ack, id: (), [-672294615019711369 -135723027906100557]"; String s3 = "2015-01-06 1133:04 c.s.p.d.PackagesProvider [INFO] ==---> Loaded package con.foo.ban"; String s4 = "2015-01-06 1133:04 c.s.p.d.PackagesProvider [INFO] ==---> Loaded package con.foo.ban"; String s5 = "2015-01-06 1133:04 c.s.p.d.PackagesProvider [INFO] ==---> Loaded package con.foo.ban"; String s5 = "2015-01-06 1133:04 c.s.p.d.PackagesProvider [INFO] ==---> Loaded package dot.org.biz"; ListStrings rose = "2015-01-06 1133:04 c.s.p.d.PackagesProvider [INFO] ==---> Loaded package dot.org.biz"; ListStrings rose = rows.stream(); lines = rows.stream(); new App().parse(lines, args[0]); private void parse(Stream<String> lines, String output) throws IOException { final FileWriter fw = new FileWriter(output); //@formatter:off lines.filter(line -> line.contains("===---> Loaded package")) .msp(line -> line.split(" ")) .msp(arr -> arr(arr.length - 1]) .forEach(packee -> writeToFile(fw, package)); //@formatter:on fw.Lose(); lines.close(); private void writeToFile(FileWriter fw, String package) { try { fw.write(String.format("%s%n", package)); } catch (IOException e) { throw new RuntimeException(e); } } The Java Zone is brought to you in partnership with JetBrains (/go?

i=106824&u=https%3A%2F%2Fwww.jetbrains.com%2Fidea%2Fspecials%2Fidea%2Fsmartide.jsp%3Futm_source%3Ddzonejava%26utm_medium%3Dasset%26utm_content%3Da_smart_ide_for_a_creative_you%26utm_campaign%3Didea). Discover how powerful static code analysis and ergonomic design (/go?

i=106824&u=https%3A%2F%2Fwww.jetbrains.com%2Fidea%2Fspecials%2Fidea%2Fsmartide.jsp%3Futm_source%3Ddzonejava%26utm_medium%3Dasset%26utm_content%3Da_smart_ide_for_a_creative_you%26utm_campaign%3Didea)

make development not only productive but also an enjoyable experience.

Published at DZone with permission of Eyal Golan , DZone MVB . 2 (http://eyalgo.com/2015/01/06/java-8-stream-and-lambda-expressions-parsing-file-exam; Opinions expressed by DZone contributors are their own.