

ESOF

**ElasticSearch**

****

**Project of ESOF 2018 -- MIEIC**

**Head of CU: João Carlos Pascoal Faria**

**Class 3**

**Group 1**

* João Pedro Viveiros Franco --- up201604828 --- up201604828@fe.up.pt
* Tiago Pinho Cardoso --- up201605762 --- up201605762@fe.up.pt
* Tomás Nuno Fernandes Novo --- up201604503 --- up201604503@fe.up.pt

**Supervisor:** Filipe Alexandre Pais de Figueiredo Correia

November, 2018

**Índex**

1. ElasticSearch – The Open Source Project
   1. How alive is ElasticSearch?………..……...............................................3
   2. How important is ElasticSearch?……………………………………….4
   3. What is ElasticSearch good for?........................................................4
   4. What are the technologies involved?..................................................5
2. ElasticSearch Issues…………………………...........................................6

**2.1.** Issue #34127……………………………...........................................7

**2.1.1.** Description of the issue……………………………………………..7

**2.1.2.** Requirements……………………………………………………….7

**2.1.3.** Source Code Files……….………………………………………….7

**2.1.4.** System Architecture………………………………………………...8

**2.1.5.** Design of the fix………….………………...................................9

**2.1.6.** Fix Source Code…………………………………………………...10

**2.1.5.** Validate the fix…………………………………………………….13

**2.1.6.** Submission of the fix………………………………………………13

**2.1.6.** Wrap-up……………………………………………………………14

**2.2.** Issue #34609…………………………………..................................14

**2.2.1.** Description of the issue……………………………………………14

**2.2.2.** Requirements………………………………………………………14

**2.2.3.** Source Code Files……….…………………………………………15

**2.2.4.** System Architecture……………………………………………….16

**2.2.5.** Design of the fix………….……………………………………….16

**2.2.6.** Fix Source Code………….……………………………………….17

**2.2.7.** Validate the fix………….…………………………………………19

**2.2.8.** Submission of the fix………….………………………………….19

**2.2.9.** Wrap-up……….…………………………………………………..19

1. **ElasticSearch - The Open Source Project**
   1. **How alive is ElasticSearch?**

**ElasticSearch** was born in 2004 and since then it has been groing, specially the comunity that works on it. This community is very active. There are **1,103 contributors** and **1,508 open issues**. Until now, there are already **16.138 closed issues**, confirming that ElasticSearch has been one of the most interesting projects to work on.

The search engine versions have been changing since ElasticSearch was created. It has **different versions**, with **version 6.x** being in **active development** with new features, and **version 5.x is still under support**. Older versions are inactive for around one year.

The **version 5.x is feature-freeze** and it's **only getting bug fixes** at the moment. Although the development for this version is not very active, the latest release, 5.6.12, was released at 19th September, which is recent. Moreover, some work on the next minor version, 5.6.13, is ongoing.

The development of the current public release, 6.4.x, is very active with almost daily commits. The next planned release, 6.5, is even more active, which means the **project contributors** are **actively working to improve the product**, either by tweaking existing functionality or by adding new features.

Overall, there are **over 100 commits weekly** on average.

Although the list of collaborators (core development team) is not known for this particular repository, **the Elastic organization has 129 people**. They have several repositories, thus this number is not a sufficient indicator for ElasticSearch in particular. However, considering the number of merged pull request and closed issues, **we can say there are enough people managing the project**.

Anyone interested in giving their contribution can easily do it. The project has a guideline for **contribution** with project's information and the steps to follow.

Summarizing, ElasticSearch is really alive and its life is sustained thanks to the numerous contributors who keep working on this amazing project, so we can expect an amazing lifetime for it.

* 1. **How importante is ElasticSearch?**

Elastic search is a highly important tool for most companies, as it is a distributed, RESTful search and analytics engine capable of solving a growing number of use cases.

It is particularly relevant because it lets you perform and combine many types of searches — structured, unstructured, geo, metric — any way you want, allowing you to query data in various ways to obtain several important analytics.

For example, with this software you could search for all posts (social media and the likes) containing mentions of your product, then you could take those results and aggregate them by geography; you could then narrow it down to a specific country, even sort things out by date or break it down by the three main states of that region.

Thus, it ends up playing a somewhat vital role in things like datamining, data analysis and many subjects in that general area.

* 1. **What is ElasticSearch good for?**

Elasticsearch can be used to perform a lot of different use cases: "classical" full text search, analytics store, auto completer, spell checker, alerting engine, and as a general purpose document store.

It is especially good for analytics, because it integrates well with the ELK stack, and its interface is good for the REST/JSON driven configuration.

It provides a scalable search, since it scales horizontally to handle many events per secondmwhile automatically managing how indices and queries are distributed across many operations.

It also has a near real-time search, as a result of the implementation of inverted indices with finite state transducers for full-text querying, BKD tees for storing numeric and geo data and a column store for analytics.

Thus, it’s good for those seeking an engine that’s effective at quickly and dynamically searching through large amounts of data, as is the case for a search-intensive application.

* 1. **What are the technologies involved?**

**Type:** Software Service. Interaction with a REST API or Java API, with multiple Clients available: Java, Groovy, .Net, PHP, Perl, Python, Ruby.

**Languages:**

* **Java** – Main language
* **Groovy** – Used in buildSrc. There’s an issue open to replace Groovy with Java
* **AsciiDoc** – Documentation
* **YAML, JSON** – Mainly used on the REST API Specification

1. **ElasticSearch Issues**

* **Issue #34127 – Include origin information in deprecation logging**
* **Issue #34609 – SQL: add tests for RLIKE**

**2.1. Issue #34127** - <https://github.com/elastic/elasticsearch/issues/34127>

1. **Description of the Issue**

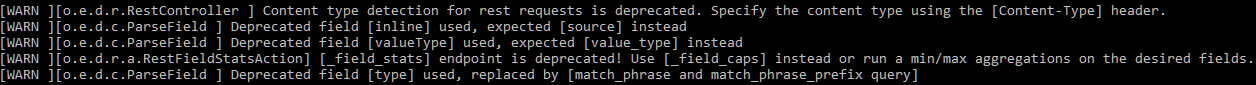
The ElasticSearch’s issue #34127 is the result of the lack of information on the deprecation logging. What happens is that the deprecation logging only logs the timestamp and the message of deprecation. The suggestion to correct it is to add more information to the DeprecationLogger of the project.

1. **Requirements**

Aiming to obtain the solution for this issue, it was required to add more information on the deprecation message, like the hostname or IP address. We also had to know the basics of Java programming.

To solve this issue we improved the deprecation message by adding some useful information besides the information that is given.

An example of a deprecation message is:



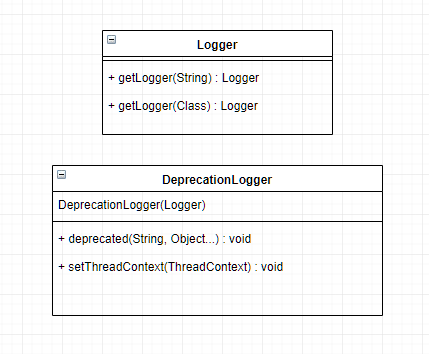
1. **Source code files**

The files that are directly related to **#34127** issue are:

* **elasticsearch/server/src/main/java/org/elasticsearch/common/logging/DeprecationLogger.java –** File of the Deprecation Logger where the issue was.
* **elasticsearch/server/src/test/java/org/elasticsearch/common/logging/DeprecationLoggerTests.java –** File where the Deprecation Logger is tested.
* **elasticsearch/server/src/main/java/org/elasticsearch/trnasposrt/TransportService.java –** File that has the responsibility to send the port to the Deprecation Logger

1. **System Architecture**

After a lot of research on the open source project ElasticSearch, we achieved some knowledge of how this project works. Specifically, in this issue, the architecture has the following UML:

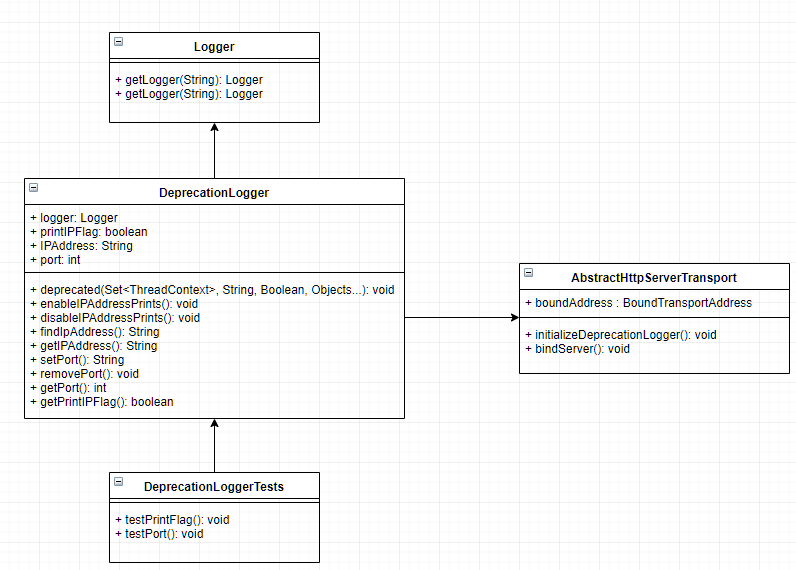
****

The DeprecationLogger allows various Elasticsearch classes to warn the user when they use a method that is going to be removed from the project soon. The Logger class allows other classes to easily log actions to the user when deemed necessary, without having to deal with things like identifying the class calling the logger and allows for different types of logging(Error, Information Warning).

1. **Design of the fix**

The solution was to add more information on the deprecation message, like the hostname or IP address. To add this information about the user we used the NetworkInterface and the InetAddress class of the java.net library.

In practice, the fix that we implemented consists to show this information on the Deprecation Logger.



For example, this deprecation message changes from this:

https://scontent.fopo2-2.fna.fbcdn.net/v/t1.15752-9/48418100_787510974922025_4279621253297340416_n.png?_nc_cat=104&_nc_ht=scontent.fopo2-2.fna&oh=bcecd199241b7ced042aa9361c767747&oe=5C90A4C6

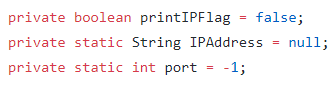
To this:

1. **Fix Source Code**

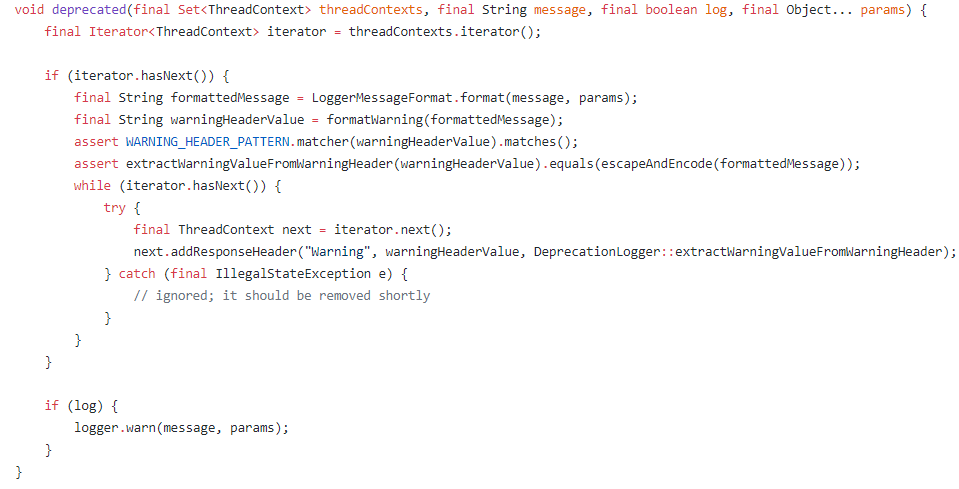
The code alterations we implemented are the following, ordered by files:

* server/src/main/java/org/elasticsearch/common/logging/DeprecationLogger.java

1. First we created some new variables

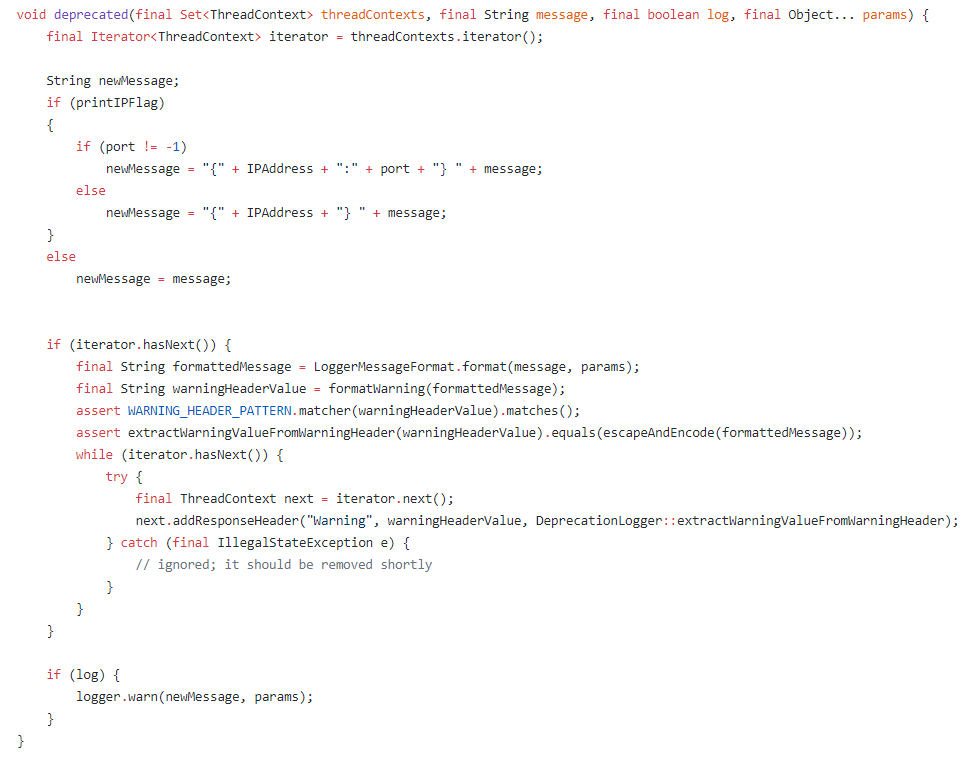


1. Then we changed this function



Antes

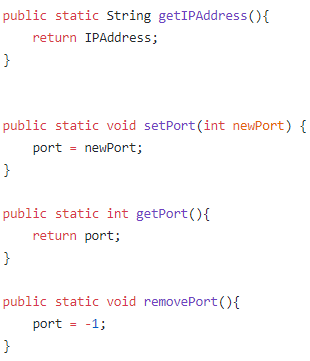
-------------------------------------------------------------------------------------------------



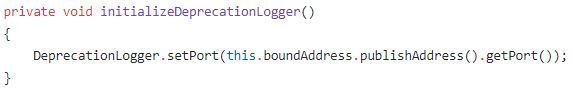
Depois

1. Then we added some functions: the first two are relatable to prints, the first enables them and the second disables them. The third function finds the user IP address and the four is its get function. The functions **setPort(), getPort()** and **removePort()** are related to the port. The last one, **getPrintIPFlag()** returns the Boolean value of the variable we created on step 1.



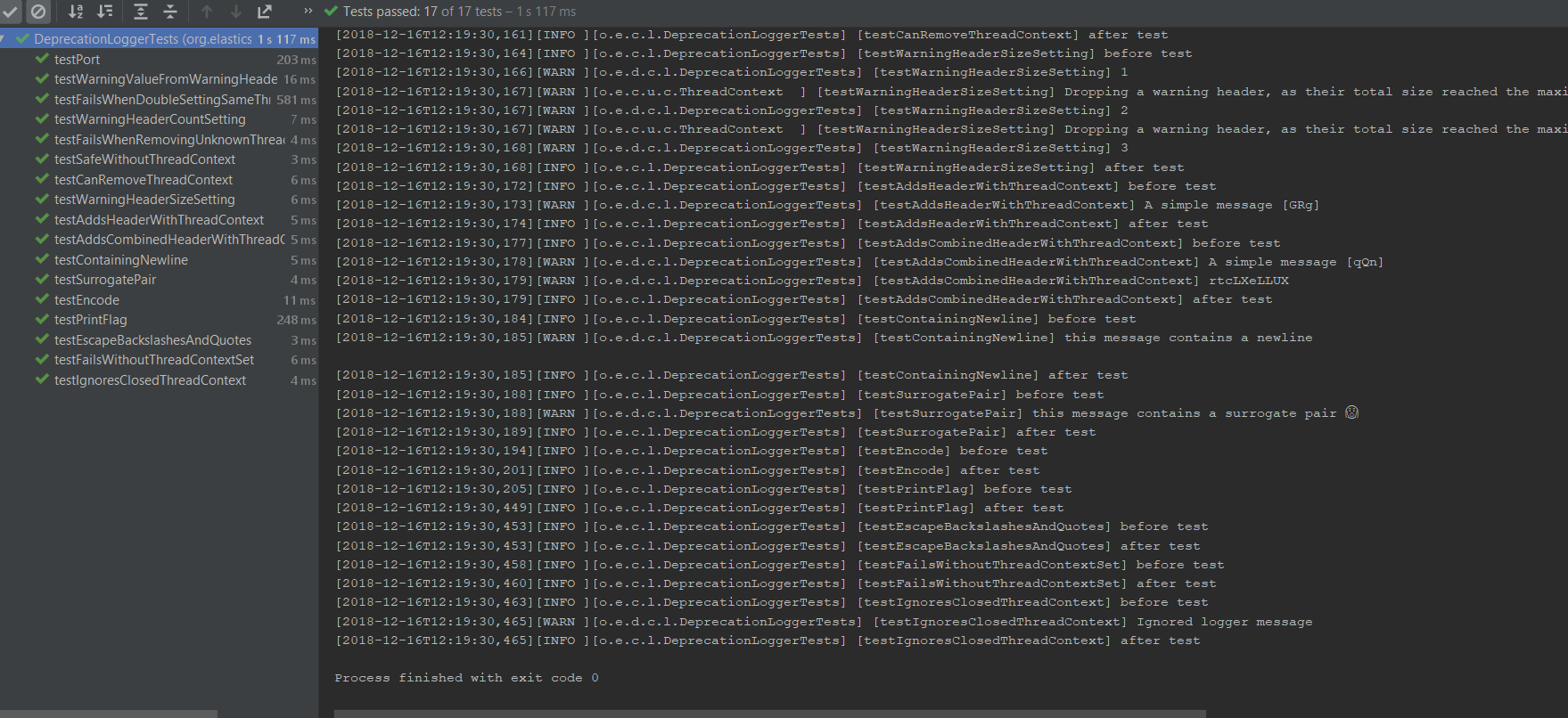


* server/src/main/java/org/elasticsearch/http/AbstractHttpServerTransport.java
  1. In this file we created a function that initializes the Deprecation Logger



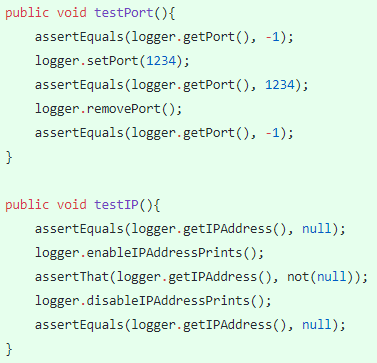
* 1. After that, we changed the function **bindServer()**, which has the purpose to bind and start to accept incoming connections, adding the function we created in step 1. of this file.

The following print screen shows that our implemented fix works.



1. **Validate the fix**

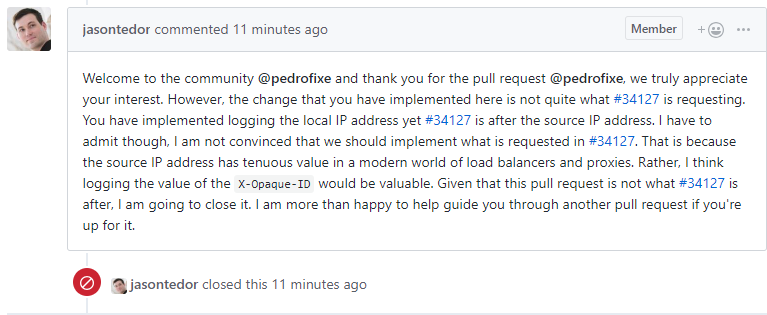
To test our changes, we created the following tests on the file **server/src/test/java/org/elasticsearch/common/logging/DeprecationLoggerTests.java**



These tests check validate the initial state of the PrintIPFlag and the Port and the IPAddress of the user.

1. **Submission of the fix**

We tested and submitted our fix of this issue by making a pull request. Unfortunately, it was turned down.



Here is the link to the pull request: <https://github.com/elastic/elasticsearch/pull/36687>

1. **Wrap-up**

To solve the issue we had to divide our work. We consider that the following is a fair work division of each student:

* João Pedro – 50%
* Tomás Novo – 35%
* Tiago Cardoso – 15%
  1. **Issue #34609** - <https://github.com/elastic/elasticsearch/issues/34609>

1. **Description of the Issue**

The #34609 ElasticSearch’s issue related to SQL language. In this issue, there is a class called **RLike** that has no tests for and it would be very positive if someone could make tests to ensure that everything runs perfectly in the meantime.

1. **Requirements**

To solve this issue we had to know the basics of SQL and how tests work in Java.

This issue required the implementation of some tests for **RLike** class.

1. **Source Code Files**

Files directly related to the issue:

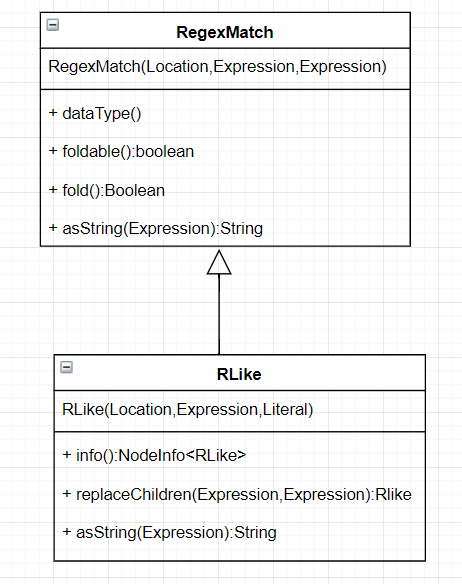
* **elasticsearch/x-pack/plugin/sql/src/main/java/org/elasticsearch/xpack/sql/expression/predicate/regex/RLike.java –** Where RLike is implemented



* **elasticsearch/x-pack/plugin/sql/src/test/java/org/elasticsearch/xpack/sql/expression/predicate/regex/RLikeTests.java –** Newfile where we implemented the tests

1. **System Architecture**

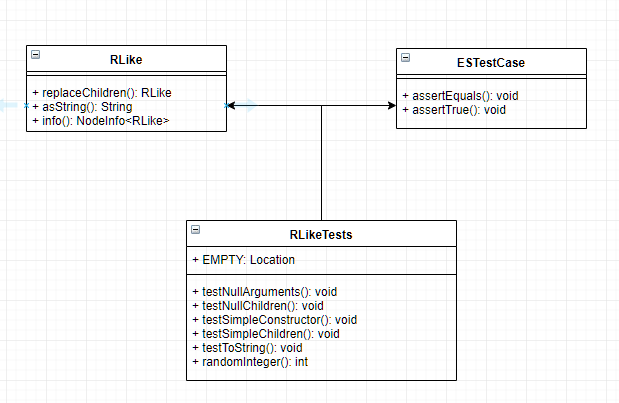
Researching on the open source project ElasticSearch gave us the knowledge of how this issue happens. This issue’s architecture has the following UML:



The RLike class is the equivalent to RLike SQL operator that performs a pattern match of a string expression against a pattern.

1. **Design of the fix**

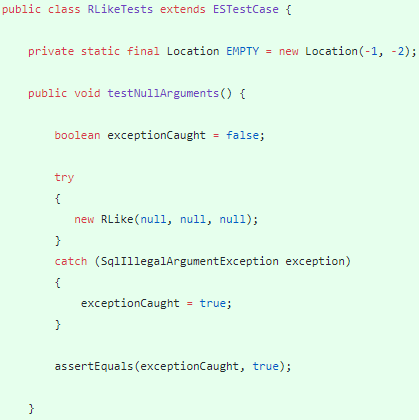
Our plan to fix this issue is to create some tests for the class RLike.

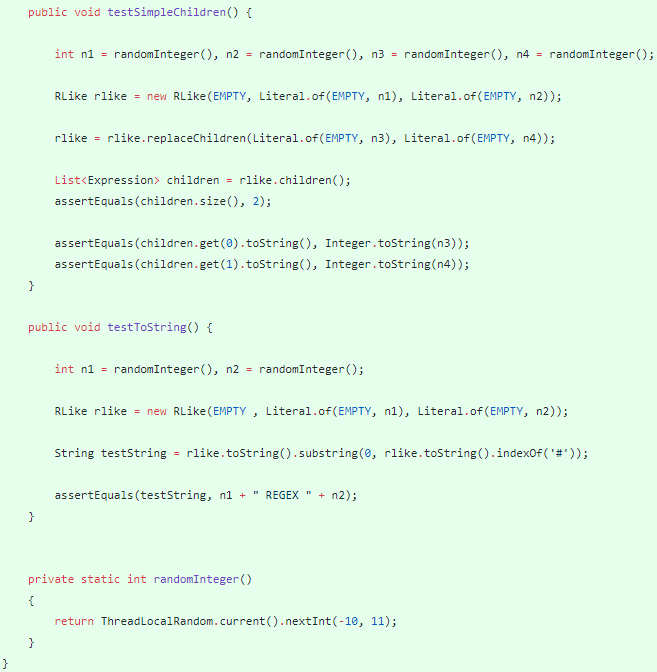


The ESTestCase is a generic class to implement tests in Elasticsearch.

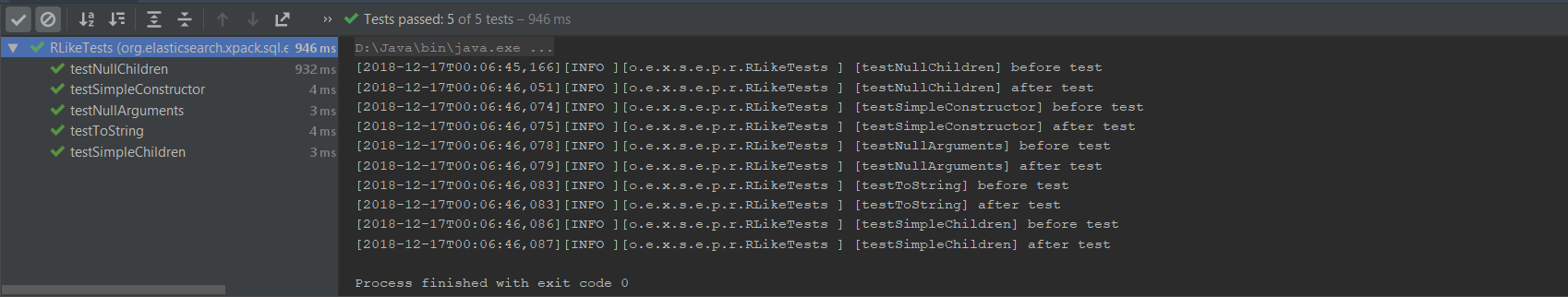
1. **Fix Source Code**

With the objective to implement the tests for RLike class, we created the following code:





The following print shows that our implemented fix works.



1. **Validate the fix**

The issue itself was to implement tests about this class. Therefore, we tested against null arguments and children, with simple arguments and children and with the toString function.

1. **Submission of the fix**

Like the first issue we worked on, we submitted the fix of this issue by making a pull request and we’re currently waiting for answer. The pull request can be accessed by the following link[: https://github.com/elastic/elasticsearch/pull/36688](file:///C:\Users\Utilizador\Downloads\:%20%20https:\github.com\elastic\elasticsearch\pull\36688)

1. **Wrap-up**

To solve the issue we had to divide our work. We consider that the following is a fair work division of each student:

* Tomás Novo – 50 %
* João Franco – 35 %
* Tiago Cardoso – 15 %