CARIAD
A VOLKSWAGEN GROUP COMPANY

Java/Kotlin Test Task

Description

Write an HTTP service that exposes an endpoint "/strings". This endpoint receives a list of URLs through the "GET" query parameter. The parameter in use is called "u". It can appear more than once.

```
http://localhost:8080/strings?u=http://example.com/primes&u=http://foobar.com/fibo
```

When the "/strings" endpoint is called, your service shall retrieve each of these URLs if they turn out to be syntactically valid URLs. Each URL will return a JSON data structure that looks like this:

```
{ "strings": [ "one", "two", "three", "five", "eight", "thirteen" ] }
```

The JSON data structure will contain an object with a key named "strings", and a value that is a list of strings. After retrieving each of these URLs, the service shall merge the strings coming from all URLs, sort them in ascending order and make sure that each string only appears once in the result. The endpoint shall then return a JSON data structure like in the example above with the result as the list of strings.

The endpoint needs to return the result as quickly as possible, but always within 500 milliseconds. It needs to be able to deal with error conditions when retrieving the URLs. If a URL takes too long to respond, it must be ignored. It is valid to return an empty list as a result only if all URLs returned errors or took too long to respond.

Example

The service receives an HTTP request:

```
GET /strings?u=http://example.com/primes&u=http://foobar.com/fibo HTTP/1.0
```

It then retrieves the URLs specified as parameters.

The first URL returns this response:

```
>>> GET /primes HTTP/1.0
>>> Host: example.com
>>>
<<< HTTP/1.0 200 OK
<<< Content-Type: application/json
<<< Content-Length: 34
<<<
<<< { "strings": [ "two", "three", "five", "seven", "eleven", "thirteen" ] }
```

The second URL returns this response:

```
>>> GET /fibo HTTP/1.0
>>> Host: foobar.com
>>>
<<< HTTP/1.0 200 OK
<<< Content-Type: application/json
<<< Content-Length: 40
<<<
<<< { "strings": [ "one", "one", "two", "three", "five", "eight", "thirteen", "twenty one" ] }
```

The service then calculates the result and returns it.

```
<<< HTTP/1.0 200 OK
<<< Content-Type: application/json
<<< Content-Length: 44
<<<
<<< { "strings": [ "eight", "eleven", "five", "one", "seven", "thirteen", "three", "twenty one", "two" ]}
```

Additional requirements/information

Solve the task described above using Java/Kotlin programming language. Feel free to use Spring/Boot if you like.

Document source code, both using comments and in a separate text file that describes the intentions and rationale behind your solution. Also, write down any ambiguities that you see in the task description, and describe to you how you interpreted them and why. Write unit and integration (if applicable) tests for your code.

CARIAD
A VOLKSWAGEN GROUP COMPANY

Test server

For testing purposes, there is an example server that, listens on port 8090 and provides the endpoints /primes, /fibo, /odd, and /rand.

Please, run the example server by using this command

```
docker run --detach --publish 8090:8090 digitalcrab/coding-challenge-test-server:latest
```

And try out some calls before you start (please note that the test server simulates timeouts and responses longer then 500 milliseconds):

```
curl http://127.0.0.1:8090/primes
```

Have fun!