

Thank you for your interest in joining the TruSTAR engineering team! We invite you to complete this screener. Please return your completed questionnaire before the end of the week so we can grade it ASAP.

Please submit code as if you intended to ship it to production. The details matter. Documentation and tests are expected, as is well written, simple, idiomatic code. Feel free to use an IDE and different libraries to avoid writing code from scratch. Your code should be ready to package and deploy if needed.

We found that creating a brief github project is the best way to deliver the code (grant access permission to your hiring manager).

### Notes

1. Please place your code under `com.trustar.interview.[question]` package so it's easy to figure out where answers are placed (don't worry if you end up with duplicated code in each package)
2. Use your preferred testing, packaging, dependency injection frameworks if needed be. Internally we use Spock / Junit, Gradle(w) and Spring.
3. `Readme.MD` are great ways to document general intent.

### Questions

1- Write a java function that, given a list of `java.util.regex.Pattern` and a string, it extracts and returns all the matches for the patterns found in the string. For example:

```
List<Pattern> patterns = ImmutableList.of(
    Pattern.compile("(the)"),
    Pattern.compile("(fox|f.nce)");
f(patterns, "The fox jumped over the fence");
```

Returns the list `["the", "fox", "fence"]`

2- Modify the function so when a pattern matches, the substring that produced the match can't be reused by any subsequent patterns.

```
List<Pattern> patterns = ImmutableList.of(
    Pattern.compile("th."),
    Pattern.compile("he"),
    Pattern.compile("fox"));
f(patterns, "The fox jumped over the fence");
```

Returns the list `["the", "he", "fox"]` (the pattern `he` appears twice in the original string, but it's already "consumed" when it matches the pattern `th.`)

3- Modify the function to add a third parameter known as “Blacklist pattern”. The strings the function returns can’t match the blacklist pattern.

```
List<Pattern> patterns = ImmutableList.of(
    Pattern.compile("(the)"),
    Pattern.compile("(fox|fence)"),
    Pattern.compile("(abc)"));
f(patterns, "The fox jumped over the fence", Pattern.compile("f.x"));
```

Returns the list ["the", "fence"] ("fox" is ignored because it matches the blacklist pattern)

4- The Mitre organization (<http://www.mitre.org/>) hosts a Cyber Threat Intelligence Repository at <https://github.com/mitre/cti/>. We are interested in finding information about **Advanced Persistent Threats or APTs** - malicious state sponsored actors that try to gain access to computer networks for their own (or their country) gain.

Write an application that connects to the Mitre CTI repository, and extract all the URLs and names of APTs that can be found on any of the files in the path  
/enterprise-attack/intrusion-set.

Keep in mind the following:

- APTs are named APT<sub>nn</sub>, where nn is a number (for example, in the file intrusion-set--899ce53f-13a0-479b-a0e4-67d46e241542.json you will find “APT29”. APT29 is also known as “Cozy Bear”, a group of Russian hackers made famous during the US 2016 Presidential Election)
- Do not focus on perfecting a URL regex. For this program, URLs are anything that start with http:// or https://, and end with a space or a double quote " (because the content of the files is simply JSON).
- Currently we aren’t interested in extracting URLs from the domains: symantec.com, cybereason.com.

5- Reflecting on our solution, can you think of any issues we might run into if we want to use these functions in production, where we expect thousands of extractions to be requested per second? Please describe an architecture you believe will support that volume of calls.