Separate URL creation to enable proper logging

Context and Problem Statement

Fetchers are failing. The reason why they are failing needs to be investigated.

- Claim 1: Knowing the URL which was used to query the fetcher eases debugging
- Claim 2: Somehow logging the URL eases debugging (instead of showing it in the debugger only)

How to properly log the URL used for fetching?

Decision Drivers

- Code should be easy to read
- Include URL in the exception instead of logging in case an exception is thrown already (see https://howtodoinjava.com/best-practices/java-exception-handling-best-practices/#6)

Considered Options

- Separate URL creation
- Create URL when logging the URL
- Include URL creation as statement before the stream creation in the try-with-resources block

Decision Outcome

Chosen option: “Separate URL creation”, because comes out best (see below).

Pros and Cons of the Options

Separate URL creation

```
URL urlForQuery;
try {
    urlForQuery = getURLForQuery(query);
} catch (URISyntaxException | MalformedURLException | FetcherException e) {
```

```
        throw new FetcherException(String.format("Search URI %s is malformed", query), e);
    }
    try (InputStream stream = getUrlDownload(complexQueryURL).asInputStream()) {
        ...
    } catch (IOException e) {
        throw new FetcherException("A network error occurred while fetching from " + urlForQuery.toStrin
    } catch (ParseException e) {
        throw new FetcherException("An internal parser error occurred while fetching from " + urlForQuer
    }
```

- Good, because exceptions thrown at method are directly caught
- Good, because exceptions in different statements belong to different catch blocks
- Good, because code to determine URL is written once
- OK, because “Java by Comparison” does not state anything about it
- Bad, because multiple try/catch statements are required
- Bad, because this style seems to be uncommon to Java coders

Create URL when logging the URL

The “logging” is done when throwing the exception.

Example code:

```
    try (InputStream stream = getUrlDownload(getURLForQuery(query)).asInputStream()) {
        ...
    } catch (URISyntaxException | MalformedURLException | FetcherException e) {
        throw new FetcherException(String.format("Search URI %s is malformed", query), e);
    } catch (IOException e) {
        try {
            throw new FetcherException("A network error occurred while fetching from " + getURLForQuery(
        } catch (URISyntaxException | MalformedURLException uriSyntaxException) {
            // does not happen
            throw new FetcherException("A network error occurred", e);
        }
    } catch (ParseException e) {
        try {
            throw new FetcherException("An internal parser error occurred while fetching from " + getURL
        } catch (URISyntaxException | MalformedURLException uriSyntaxException) {
            // does not happen
            throw new FetcherException("An internal parser error occurred", e);
        }
    }
```

- Good, because code inside the `try` statement stays the same

- OK, because “Java by Comparison” does not state anything about it
- Bad, because an additional try/catch-block is added to each catch statement
- Bad, because needs a `throw` statement in the `URISyntaxException` catch block (even though at this point the exception cannot be thrown), because Java otherwise misses a `return` statement.

Include URL creation as statement before the stream creation in the try-with-resources block

```
try (URL urlForQuery = getURLForQuery(query); InputStream stream = urlForQuery.asInputStream()) {
    ...
} catch (URISyntaxException | MalformedURLException | FetcherException e) {
    throw new FetcherException(String.format("Search URI %s is malformed", query), e);
} catch (IOException e) {
    throw new FetcherException("A network error occurred while fetching from " + urlForQuery.toStrin
} catch (ParseException e) {
    throw new FetcherException("An internal parser error occurred while fetching from " + urlForQuer
}
```

- Good, because the single try/catch-block can be kept
- Good, because logical flow is kept
- Bad, because does not compile (because URL is not an `AutoClosable`)
