**Code Smells**

Below I describe the three “code smells” I found, as proposed.

**The first “code smell”: Shotgun surgery**

public ComplexSearchQueryBuilder defaultFieldPhrase(String defaultFieldPhrase) {  
 if (Objects.*requireNonNull*(defaultFieldPhrase).isBlank()) {  
 throw new IllegalArgumentException("Parameter must not be blank");  
 }  
 // Strip all quotes before wrapping  
 this.defaultFieldPhrases.add(String.*format*("\"%s\"", defaultFieldPhrase.replace("\"", "")));  
 return this;  
}  
  
*/\*\*  
 \* Adds author and wraps it in quotes  
 \*/*public ComplexSearchQueryBuilder author(String author) {  
 if (Objects.*requireNonNull*(author).isBlank()) {  
 throw new IllegalArgumentException("Parameter must not be blank");  
 }  
 // Strip all quotes before wrapping  
 this.authors.add(String.*format*("\"%s\"", author.replace("\"", "")));  
 return this;  
}  
  
*/\*\*  
 \* Adds title phrase and wraps it in quotes  
 \*/*public ComplexSearchQueryBuilder titlePhrase(String titlePhrase) {  
 if (Objects.*requireNonNull*(titlePhrase).isBlank()) {  
 throw new IllegalArgumentException("Parameter must not be blank");  
 }  
 // Strip all quotes before wrapping  
 this.titlePhrases.add(String.*format*("\"%s\"", titlePhrase.replace("\"", "")));  
 return this;  
}  
  
*/\*\*  
 \* Adds abstract phrase and wraps it in quotes  
 \*/*public ComplexSearchQueryBuilder abstractPhrase(String abstractPhrase) {  
 if (Objects.*requireNonNull*(abstractPhrase).isBlank()) {  
 throw new IllegalArgumentException("Parameter must not be blank");  
 }  
 // Strip all quotes before wrapping  
 this.titlePhrases.add(String.*format*("\"%s\"", abstractPhrase.replace("\"", "")));  
 return this;  
}

This long parameter list can be found in **jabref > logic >importer > fetcher > transformers > ComplexSearchQuery.**

In this case, there are blocks of code very similar, present in many places of the code. A new method, containing the recurrent code that deals with the small changes between blocks to prevent such repetition, is recommended. Another observation would be creating a new constant containing the phrase "Parameter must not be blank".

if (Objects.*requireNonNull*(abstractPhrase).isBlank()) {  
 throw new IllegalArgumentException("Parameter must not be blank");  
}

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

**The second “code smell”: long parameter list**

private ComplexSearchQuery(List<String> defaultField, List<String> authors, List<String> titlePhrases, List<String> abstractPhrases, Integer fromYear, Integer toYear, Integer singleYear, String journal, String doi) {  
 this.defaultField = defaultField;  
 this.authors = authors;  
 this.titlePhrases = titlePhrases;  
 this.abstractPhrases = abstractPhrases;  
 this.fromYear = fromYear;  
 // Some APIs do not support, or not fully support, year based search. In these cases, the non applicable parameters are ignored.  
 this.toYear = toYear;  
 this.journal = journal;  
 this.singleYear = singleYear;  
 this.doi = doi;  
}

This long parameter list can be found in **jabref > logic >importer > fetcher > transformers > ComplexSearchQuery.**

Here, the method has nine parameters which I believe is too much. The following parameters seem to be related and could be stored in a specific object - Integer fromYear, Integer toYear, and Integer singleYear.

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

**The third “code smell”: long method**

@Override  
public Optional<BibEntry> performSearchById(String identifier) throws FetcherException {  
 Optional<DOI> doi = DOI.*parse*(identifier);  
  
 try {  
 if (doi.isPresent()) {  
 Optional<BibEntry> fetchedEntry;  
  
 // mEDRA does not return a parsable bibtex string  
 if (getAgency(doi.get()).isPresent() && "medra".equalsIgnoreCase(getAgency(doi.get()).get())) {  
 return new Medra().performSearchById(identifier);  
 }  
 URL doiURL = new URL(doi.get().getURIAsASCIIString());  
  
 // BibTeX data  
 URLDownload download = getUrlDownload(doiURL);  
 download.addHeader("Accept", MediaTypes.*APPLICATION\_BIBTEX*);  
 String bibtexString;  
 try {  
 bibtexString = download.asString();  
 } catch (IOException e) {  
 // an IOException will be thrown if download is unable to download from the doiURL  
 throw new FetcherException(Localization.*lang*("No DOI data exists"), e);  
 }  
  
 // BibTeX entry  
 fetchedEntry = BibtexParser.*singleFromString*(bibtexString, preferences, new DummyFileUpdateMonitor());  
 fetchedEntry.ifPresent(this::doPostCleanup);  
  
 // Check if the entry is an APS journal and add the article id as the page count if page field is missing  
 if (fetchedEntry.isPresent() && fetchedEntry.get().hasField(StandardField.*DOI*)) {  
 BibEntry entry = fetchedEntry.get();  
 if (isAPSJournal(entry, entry.getField(StandardField.*DOI*).get()) && !entry.hasField(StandardField.*PAGES*)) {  
 setPageCountToArticleId(entry, entry.getField(StandardField.*DOI*).get());  
 }  
 }  
  
 return fetchedEntry;  
 } else {  
 throw new FetcherException(Localization.*lang*("Invalid DOI: '%0'.", identifier));  
 }  
 } catch (IOException e) {  
 throw new FetcherException(Localization.*lang*("Connection error"), e);  
 } catch (ParseException e) {  
 throw new FetcherException("Could not parse BibTeX entry", e);  
 } catch (JSONException e) {  
 throw new FetcherException("Could not retrieve Registration Agency", e);  
 }  
}

This long method can be found in **jabref > logic >importer > fetcher > transformers > DoiFetcher.**

The method above is hard to read. It could be divided into more methods to reduce its complexity and allow a clearer way to analyze it. One suggestion would be extracting the following block and inserting it into a new method to improve the general method's readability:

try {  
 bibtexString = download.asString();  
} catch (IOException e) {  
 // an IOException will be thrown if download is unable to download from the doiURL  
 throw new FetcherException(Localization.*lang*("No DOI data exists"), e);  
}

Francisco Abreu Freire, nº 58667