Code Smells

Bellow are the 3 code smells I identified as was delegated to during the team meetings.

The first Code Smell Identified was of type: “Duplicate Code”.

```
299                 if ((i + 1) < layoutEntries.size()) {
300                     if (layoutEntries.get(i + 1).doLayout(bibtex, database).trim().isEmpty()) {
301                         i++;
302                         previousSkipped = true;
303                         continue;
304                     }
```

This code smell is located in the following path: jabref -> src -> main -> java -> org. jabref -> logic -> layout -> format -> LayoutEntry.java (lines 299 through 304).

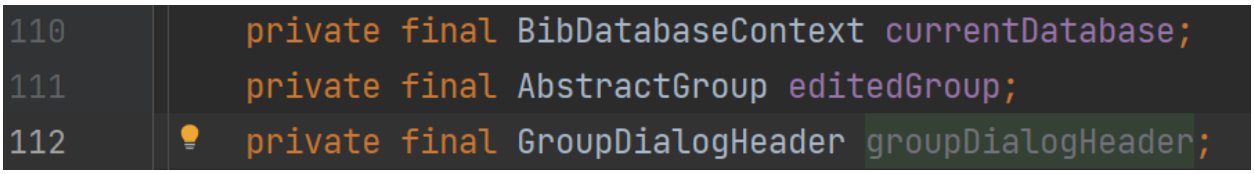
In this case the collapsible ***if*** statements in line 299 and 300 could be merged to increase readability.

Refactoring suggestion:

```
if (fieldText == null) {
    if ( ((i + 1) < layoutEntries.size()) && (layoutEntries.get(i + 1).doLayout(bibtex, database).trim().isEmpty()) ){
            i++;
            previousSkipped = true;
            continue;
    }
```

Code Smells

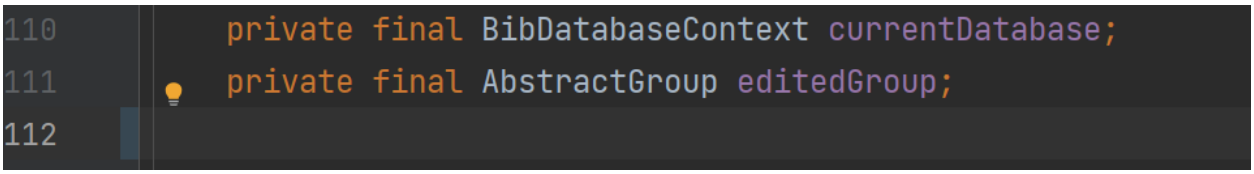
The second Code Smell Identified was of type: “Dead Code”.



This code smell is located in the following path: jabref -> src -> main -> java -> org. jabref -> gui -> groups -> GroupDialogViewModel.java (lines 110 through 112).

If a private field is declared but not used in the program, it can be considered dead code and should therefore be removed. This will improve maintainability because developers will not wonder what the variable is used for.

Refactoring suggestion:



Code Smells

The third Code Smell Identified was of type: “Dispensable Comments”.

```
/**
 * This is an immutable class representing information of either <CODE>author</CODE> or <CODE>editor</CODE> field in
 * <p>
 * Constructor performs parsing of raw field text and stores preformatted data. Various accessor methods return autho
 * <p>
 * Parsing algorithm is designed to satisfy two requirements: (a) when author's name is typed correctly, the result s
 * <ol>
 * <li> 'author field' is a sequence of tokens;
 * <ul>
 * <li> tokens are separated by sequences of whitespaces (<CODE>Character.isWhitespace(c)==true</CODE>),
 * commas (,), dashes (-), and tildas (~);
 * <li> every comma separates tokens, while sequences of other separators are
 * equivalent to a single separator; for example: "a - b" consists of 2 tokens
 * ("a" and "b"), while "a,-,b" consists of 3 tokens ("a", "", and "b")
 * <li> anything enclosed in braces belongs to a single token; for example:
 * "abc x{a,b,-~ c}x" consists of 2 tokens, while "abc xa,b,-~ cx" consists of 4
 * tokens ("abc", "xa","b", and "cx");
 * <li> a token followed immediately by a dash is "dash-terminated" token, and
 * all other tokens are "space-terminated" tokens; for example: in "a-b- c - d"
 * tokens "a" and "b" are dash-terminated and "c" and "d" are space-terminated;
 * <li> for the purposes of splitting of 'author name' into parts and
 * construction of abbreviation of first name, one needs definitions of first
 * letter of a token, case of a token, and abbreviation of a token:
 * <ul>
 * <li> 'first letter' of a token is the first letter character (<CODE>Character.isLetter(c)==true</CODE>)
 * that does not belong to a sequence of letters that immediately follows "\"
 * character, with one exception: if "\" is followed by "aa", "AA", "ae", "AE",
 * "l", "L", "o", "O", "oe", "OE", "i", or "j" followed by non-letter, the
 * 'first letter' of a token is a letter that follows "\"; for example: in
```

This code smell is located in the following path: jabref -> src -> main -> java -> org. jabref -> model -> entry -> types -> AuthorList.java (lines 13 through 114).

A dispensable is something pointless and unneeded whose absence would make the code cleaner, more efficient and easier to understand, in this case there’s a huge block of comments that could be completely removed.

Refactoring suggestion: Delete the comment block.