

CODE SMELLS   
WHY

unecessary comments just clumping up the code and making it harder to read

where src/main/java/org/jabref/gui/importer/ImportAction.java

LONG METHOD

over 200 lines of code, should be seperated into auxilary methods to improve readability

src/main/java/org/jabref/logic/importer/fileformat/RisImporter.java  
RisImporter

public ParserResult importDatabase(BufferedReader reader) throws IOException {  
 List<BibEntry> bibitems = new ArrayList<>();  
  
 // use optional here, so that no exception will be thrown if the file is empty  
 String linesAsString = reader.lines().reduce((line, nextline) -> line + "\n" + nextline).orElse("");  
  
 String[] entries = linesAsString.replace("\u2013", "-").replace("\u2014", "--").replace("\u2015", "--")  
 .split("ER -.\*(\\n)\*");  
  
 // stores all the date tags from highest to lowest priority  
 List<String> dateTags = Arrays.*asList*("Y1", "PY", "DA", "Y2");  
  
 for (String entry1 : entries) {  
  
 String dateTag = "";  
 String dateValue = "";  
 int datePriority = dateTags.size();  
 int tagPriority;  
  
 EntryType type = StandardEntryType.*Misc*;  
 String author = "";  
 String editor = "";  
 String startPage = "";  
 String endPage = "";  
 String comment = "";  
 Optional<Month> month = Optional.*empty*();  
 Map<Field, String> fields = new HashMap<>();  
  
 String[] lines = entry1.split("\n");  
  
 for (int j = 0; j < lines.length; j++) {  
 StringBuilder current = new StringBuilder(lines[j]);  
 boolean done = false;  
 while (!done && (j < (lines.length - 1))) {  
 if ((lines[j + 1].length() >= 6) && !" - ".equals(lines[j + 1].substring(2, 6))) {  
 if ((current.length() > 0) && !Character.*isWhitespace*(current.charAt(current.length() - 1))  
 && !Character.*isWhitespace*(lines[j + 1].charAt(0))) {  
 current.append(' ');  
 }  
 current.append(lines[j + 1]);  
 j++;  
 } else {  
 done = true;  
 }  
 }  
 String entry = current.toString();  
 if (entry.length() < 6) {  
 continue;  
 } else {  
 String tag = entry.substring(0, 2);  
 String value = entry.substring(6).trim();  
 if ("TY".equals(tag)) {  
 if ("BOOK".equals(value)) {  
 type = StandardEntryType.*Book*;  
 } else if ("JOUR".equals(value) || "MGZN".equals(value)) {  
 type = StandardEntryType.*Article*;  
 } else if ("THES".equals(value)) {  
 type = StandardEntryType.*PhdThesis*;  
 } else if ("UNPB".equals(value)) {  
 type = StandardEntryType.*Unpublished*;  
 } else if ("RPRT".equals(value)) {  
 type = StandardEntryType.*TechReport*;  
 } else if ("CONF".equals(value)) {  
 type = StandardEntryType.*InProceedings*;  
 } else if ("CHAP".equals(value)) {  
 type = StandardEntryType.*InCollection*;  
 } else if ("PAT".equals(value)) {  
 type = IEEETranEntryType.*Patent*;  
 } else {  
 type = StandardEntryType.*Misc*;  
 }  
 } else if ("T1".equals(tag) || "TI".equals(tag)) {  
 String oldVal = fields.get(StandardField.*TITLE*);  
 if (oldVal == null) {  
 fields.put(StandardField.*TITLE*, value);  
 } else {  
 if (oldVal.endsWith(":") || oldVal.endsWith(".") || oldVal.endsWith("?")) {  
 fields.put(StandardField.*TITLE*, oldVal + " " + value);  
 } else {  
 fields.put(StandardField.*TITLE*, oldVal + ": " + value);  
 }  
 }  
 fields.put(StandardField.*TITLE*, fields.get(StandardField.*TITLE*).replaceAll("\\s+", " ")); // Normalize whitespaces  
 } else if ("BT".equals(tag)) {  
 fields.put(StandardField.*BOOKTITLE*, value);  
 } else if (("T2".equals(tag) || "J2".equals(tag) || "JA".equals(tag)) && ((fields.get(StandardField.*JOURNAL*) == null) || "".equals(fields.get(StandardField.*JOURNAL*)))) {  
 // if there is no journal title, then put second title as journal title  
 fields.put(StandardField.*JOURNAL*, value);  
 } else if ("JO".equals(tag) || "J1".equals(tag) || "JF".equals(tag)) {  
 // if this field appears then this should be the journal title  
 fields.put(StandardField.*JOURNAL*, value);  
 } else if ("T3".equals(tag)) {  
 fields.put(StandardField.*SERIES*, value);  
 } else if ("AU".equals(tag) || "A1".equals(tag) || "A2".equals(tag) || "A3".equals(tag) || "A4".equals(tag)) {  
 if ("".equals(author)) {  
 author = value;  
 } else {  
 author += " and " + value;  
 }  
 } else if ("ED".equals(tag)) {  
 if (editor.isEmpty()) {  
 editor = value;  
 } else {  
 editor += " and " + value;  
 }  
 } else if ("JA".equals(tag) || "JF".equals(tag)) {  
 if (type.equals(StandardEntryType.*InProceedings*)) {  
 fields.put(StandardField.*BOOKTITLE*, value);  
 } else {  
 fields.put(StandardField.*JOURNAL*, value);  
 }  
 } else if ("LA".equals(tag)) {  
 fields.put(StandardField.*LANGUAGE*, value);  
 } else if ("CA".equals(tag)) {  
 fields.put(new UnknownField("caption"), value);  
 } else if ("DB".equals(tag)) {  
 fields.put(new UnknownField("database"), value);  
 } else if ("IS".equals(tag) || "AN".equals(tag) || "C7".equals(tag) || "M1".equals(tag)) {  
 fields.put(StandardField.*NUMBER*, value);  
 } else if ("SP".equals(tag)) {  
 startPage = value;  
 } else if ("PB".equals(tag)) {  
 if (type.equals(StandardEntryType.*PhdThesis*)) {  
 fields.put(StandardField.*SCHOOL*, value);  
 } else {  
 fields.put(StandardField.*PUBLISHER*, value);  
 }  
 } else if ("AD".equals(tag) || "CY".equals(tag) || "PP".equals(tag)) {  
 fields.put(StandardField.*ADDRESS*, value);  
 } else if ("EP".equals(tag)) {  
 endPage = value;  
 if (!endPage.isEmpty()) {  
 endPage = "--" + endPage;  
 }  
 } else if ("ET".equals(tag)) {  
 fields.put(StandardField.*EDITION*, value);  
 } else if ("SN".equals(tag)) {  
 fields.put(StandardField.*ISSN*, value);  
 } else if ("VL".equals(tag)) {  
 fields.put(StandardField.*VOLUME*, value);  
 } else if ("N2".equals(tag) || "AB".equals(tag)) {  
 String oldAb = fields.get(StandardField.*ABSTRACT*);  
 if (oldAb == null) {  
 fields.put(StandardField.*ABSTRACT*, value);  
 } else if (!oldAb.equals(value) && !value.isEmpty()) {  
 fields.put(StandardField.*ABSTRACT*, oldAb + OS.*NEWLINE* + value);  
 }  
 } else if ("UR".equals(tag) || "L2".equals(tag) || "LK".equals(tag)) {  
 fields.put(StandardField.*URL*, value);  
 } else if (((tagPriority = dateTags.indexOf(tag)) != -1) && (value.length() >= 4)) {  
  
 if (tagPriority < datePriority) {  
 String year = value.substring(0, 4);  
  
 try {  
 Year.*parse*(year, *formatter*);  
 // if the year is parsebale we have found a higher priority date  
 dateTag = tag;  
 dateValue = value;  
 datePriority = tagPriority;  
 } catch (DateTimeParseException ex) {  
 // We can't parse the year, we ignore it  
 }  
 }  
 } else if ("KW".equals(tag)) {  
 if (fields.containsKey(StandardField.*KEYWORDS*)) {  
 String kw = fields.get(StandardField.*KEYWORDS*);  
 fields.put(StandardField.*KEYWORDS*, kw + ", " + value);  
 } else {  
 fields.put(StandardField.*KEYWORDS*, value);  
 }  
 } else if ("U1".equals(tag) || "U2".equals(tag) || "N1".equals(tag)) {  
 if (!comment.isEmpty()) {  
 comment = comment + OS.*NEWLINE*;  
 }  
 comment = comment + value;  
 } else if ("M3".equals(tag) || "DO".equals(tag)) {  
 addDoi(fields, value);  
 } else if ("C3".equals(tag)) {  
 fields.put(StandardField.*EVENTTITLE*, value);  
 } else if ("N1".equals(tag) || "RN".equals(tag)) {  
 fields.put(StandardField.*NOTE*, value);  
 } else if ("ST".equals(tag)) {  
 fields.put(StandardField.*SHORTTITLE*, value);  
 } else if ("C2".equals(tag)) {  
 fields.put(StandardField.*EPRINT*, value);  
 fields.put(StandardField.*EPRINTTYPE*, "pubmed");  
 } else if ("TA".equals(tag)) {  
 fields.put(StandardField.*TRANSLATOR*, value);  
  
 // fields for which there is no direct mapping in the bibtext standard  
 } else if ("AV".equals(tag)) {  
 fields.put(new UnknownField("archive\_location"), value);  
 } else if ("CN".equals(tag) || "VO".equals(tag)) {  
 fields.put(new UnknownField("call-number"), value);  
 } else if ("DB".equals(tag)) {  
 fields.put(new UnknownField("archive"), value);  
 } else if ("NV".equals(tag)) {  
 fields.put(new UnknownField("number-of-volumes"), value);  
 } else if ("OP".equals(tag)) {  
 fields.put(new UnknownField("original-title"), value);  
 } else if ("RI".equals(tag)) {  
 fields.put(new UnknownField("reviewed-title"), value);  
 } else if ("RP".equals(tag)) {  
 fields.put(new UnknownField("status"), value);  
 } else if ("SE".equals(tag)) {  
 fields.put(new UnknownField("section"), value);  
 } else if ("ID".equals(tag)) {  
 fields.put(new UnknownField("refid"), value);  
 }  
 }  
 // fix authors  
 if (!author.isEmpty()) {  
 author = AuthorList.*fixAuthorLastNameFirst*(author);  
 fields.put(StandardField.*AUTHOR*, author);  
 }  
 if (!editor.isEmpty()) {  
 editor = AuthorList.*fixAuthorLastNameFirst*(editor);  
 fields.put(StandardField.*EDITOR*, editor);  
 }  
 if (!comment.isEmpty()) {  
 fields.put(StandardField.*COMMENT*, comment);  
 }  
  
 fields.put(StandardField.*PAGES*, startPage + endPage);  
 }  
  
 // if we found a date  
 if (dateTag.length() > 0) {  
 fields.put(StandardField.*YEAR*, dateValue.substring(0, 4));  
  
 String[] parts = dateValue.split("/");  
 if ((parts.length > 1) && !parts[1].isEmpty()) {  
 try {  
 int monthNumber = Integer.*parseInt*(parts[1]);  
 month = Month.*getMonthByNumber*(monthNumber);  
 } catch (NumberFormatException ex) {  
 // The month part is unparseable, so we ignore it.  
 }  
 }  
 }  
  
 // Remove empty fields:  
 fields.entrySet().removeIf(key -> (key.getValue() == null) || key.getValue().trim().isEmpty());  
  
 // create one here  
 // type is set in the loop above  
 BibEntry entry = new BibEntry(type);  
 entry.setField(fields);  
 // month has a special treatment as we use the separate method "setMonth" of BibEntry instead of directly setting the value  
 month.ifPresent(entry::setMonth);  
 bibitems.add(entry);  
 }  
 return new ParserResult(bibitems);  
}

Long method

why» too many parameters (8)

src/main/java/org/jabref/gui/maintable/RightClickMenu.java

should try to have global variables so it's shorter to read and has a smaller chance of messing up

