**Design Patterns:**

# Singleton (One example):

Code:  
// The only instance of this class:  
private static JabRefPreferences *singleton***;**

Location:

src/main/java/org/jabref/preferences/JabRefPreferences.java

Reason:

// The only instance of this class:

# Builder (Two examples):

Code:

public static class Builder {  
  
 private boolean showPreviewAsExtraTab**;** private List<PreviewLayout> previewCycle**;** private int previewCyclePosition**;** private Number previewPanelDividerPosition**;** private String previewStyle**;** private final String previewStyleDefault**;** public Builder(PreviewPreferences previewPreferences) {  
 this.previewCycle = previewPreferences.getPreviewCycle()**;** this.previewCyclePosition = previewPreferences.getPreviewCyclePosition()**;** this.previewPanelDividerPosition = previewPreferences.getPreviewPanelDividerPosition()**;** this.previewStyle = previewPreferences.getPreviewStyle()**;** this.previewStyleDefault = previewPreferences.getDefaultPreviewStyle()**;** this.showPreviewAsExtraTab = previewPreferences.showPreviewAsExtraTab()**;** }  
  
 public Builder withShowAsExtraTab(boolean showAsExtraTab) {  
 this.showPreviewAsExtraTab = showAsExtraTab**;** return this**;** }  
  
 public Builder withPreviewCycle(List<PreviewLayout> previewCycle) {  
 this.previewCycle = previewCycle**;** return withPreviewCyclePosition(previewCyclePosition)**;** }  
  
 public Builder withPreviewCyclePosition(int position) {  
 if (previewCycle.isEmpty()) {  
 previewCyclePosition = **0;** } else {  
 previewCyclePosition = position**;** while (previewCyclePosition < **0**) {  
 previewCyclePosition += previewCycle.size()**;** }  
 previewCyclePosition %= previewCycle.size()**;** }  
 return this**;** }  
  
 public Builder withPreviewPanelDividerPosition(Number previewPanelDividerPosition) {  
 this.previewPanelDividerPosition = previewPanelDividerPosition**;** return this**;** }  
  
 public Builder withPreviewStyle(String previewStyle) {  
 this.previewStyle = previewStyle**;** return this**;** }  
  
 public PreviewPreferences build() {  
 return new PreviewPreferences(previewCycle**,** previewCyclePosition**,** previewPanelDividerPosition**,** previewStyle**,** previewStyleDefault**,** showPreviewAsExtraTab)**;** }  
}

Location:

src/main/java/org/jabref/preferences/PreviewPreferences.java

Reason:

Multiple construction methods under same method.

Code:

private ComplexSearchQueryBuilder() {  
}  
  
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**;**}  
  
public ComplexSearchQueryBuilder fromYearAndToYear(Integer fromYear**,** Integer toYear) {  
 if (Objects.*nonNull*(singleYear)) {  
 throw new IllegalArgumentException("You can not use single year and year range search.")**;** }  
 this.fromYear = Objects.*requireNonNull*(fromYear)**;** this.toYear = Objects.*requireNonNull*(toYear)**;** return this**;**}  
  
public ComplexSearchQueryBuilder singleYear(Integer singleYear) {  
 if (Objects.*nonNull*(fromYear) || Objects.*nonNull*(toYear)) {  
 throw new IllegalArgumentException("You can not use single year and year range search.")**;** }  
 this.singleYear = Objects.*requireNonNull*(singleYear)**;** return this**;**}  
  
public ComplexSearchQueryBuilder journal(String journal) {  
 if (Objects.*requireNonNull*(journal).isBlank()) {  
 throw new IllegalArgumentException("Parameter must not be blank")**;** }  
 this.journal = String.*format*("\"%s\""**,** journal.replace("\""**,** ""))**;** return this**;**}  
  
public ComplexSearchQueryBuilder DOI(String doi) {  
 if (Objects.*requireNonNull*(doi).isBlank()) {  
 throw new IllegalArgumentException("Parameter must not be blank")**;** }  
 this.doi = doi.replace("\""**,** "")**;** return this**;**}  
  
public ComplexSearchQueryBuilder terms(Collection<Term> terms) {  
 terms.forEach(term -> {  
 String termText = term.text()**;** switch (term.field().toLowerCase()) {  
 case "author" -> this.author(termText)**;** case "title" -> this.titlePhrase(termText)**;** case "abstract" -> this.abstractPhrase(termText)**;** case "journal" -> this.journal(termText)**;** case "doi" -> this.DOI(termText)**;** case "year" -> this.singleYear(Integer.*valueOf*(termText))**;** case "year-range" -> this.parseYearRange(termText)**;** case "default" -> this.defaultFieldPhrase(termText)**;** }  
 })**;** return this**;**}  
  
*/\*\*  
 \* Instantiates the AdvancesSearchConfig from the provided Builder parameters  
 \* If all text fields are empty an empty optional is returned  
 \*  
 \** ***@return*** *ComplexSearchQuery instance with the fields set to the values defined in the building instance.  
 \** ***@throws*** *IllegalStateException An IllegalStateException is thrown in case all text search fields are empty.  
 \* See: https://softwareengineering.stackexchange.com/questions/241309/builder-pattern-when-to-fail/241320#241320  
 \*/*public ComplexSearchQuery build() throws IllegalStateException {  
 if (textSearchFieldsAndYearFieldsAreEmpty()) {  
 throw new IllegalStateException("At least one text field has to be set")**;** }  
 return new ComplexSearchQuery(defaultFieldPhrases**,** authors**,** titlePhrases**,** abstractPhrases**,** fromYear**,** toYear**,** singleYear**,** journal**,** doi)**;**}

Location:

src/main/java/org/jabref/logic/importer/fetcher/ComplexSearchQuery.java

Reason:

Multiple construction methods under same method.