Item 67
git_comments:

1. * * A base TermsEnum that adds default implementations for * * {@link #attributes()} * {@link #termState()} * {@link #seekExact(BytesRef)} * {@link #seekExact(BytesRef, TermState)} * * * In some cases, the default implementation may be slow and consume huge memory, so subclass SHOULD have its own * implementation if possible.
2. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
3. * Sole constructor. (For invocation by subclass * constructors, typically implicit.)

git_commits:

1. **summary:** LUCENE-8292: Make TermsEnum fully abstract (#574)
   message: LUCENE-8292: Make TermsEnum fully abstract (#574)

github_issues:

github_issues_comments:

github_pulls:

1. **title:** LUCENE-8292: Make TermsEnum fully abstract
   body:

github_pulls_comments:

github_pulls_reviews:

1. move to its own file?

jira_issues:

1. **summary:** Fix FilterLeafReader.FilterTermsEnum to delegate all seekExact methods
   description: FilterLeafReader#FilterTermsEnum wraps another TermsEnum and delegates many methods. It misses some seekExact() methods, thus it is not possible to the delegate to override these methods to have specific behavior (unlike the TermsEnum API which allows that). The fix is straightforward: simply override these seekExact() methods and delegate.

jira_issues_comments:

1. **body:** This may be a bit trappy: if someone writes a FilterTermsEnum that hides some terms, now they have 2 more methods to override for their impl to be correct. I don't understand why you are saying that it is not possible to override the behavior of these methods without your change, can you clarify?
   label: code-design
2. **body:** 1- "Not possible to override": I was not clear. It is still possible for a delegate TermsEnum to override the seekExact() method. But it will never be called since the FilterTermsEnum above always calls seekCeil(). 2- "Two more methods to override": You're right. Although normally the same code should be reusable, it should not be tedious. I see the trappy point.
   label: code-design
3. **body:** Indeed these methods need to be overridden explicitly if you want them to be used. In general, we do not delegate methods that have a default implementation because the default implementation is correct regardless of what the wrapper class does. Overriding these methods in FilterTermsEnum to delegate to

the wrapped instance would make room for bugs by requiring more methods to be overridden for the wrapper to be correct.
label: code-design

4. **body:** When looking at TermsEnum API, what I understand is that seekExact() defaults to calling seekCeil(), but if needed (not for correctness but for performance consideration) we can override it to have a specialized seek that searches only the exact term and does not have to position to the next term if not found. This may have an impact for some TermsEnum extensions (a really noticeable impact in my case, that's why I noticed this issue). To me the current behavior of FilterTermsEnum is not correct with regard to TermsEnum API. (And I noticed that AssertingLeafReader overrides seekExact()). Adding these two methods in FilterTermsEnum fixes correctness, even if I agree it makes more room for bugs.
label: code-design
5. Another option would be to modify the TermsEnum.seekExact() method and make it final, or have the javadoc be explicit that it should not be overridden. (though I don't like this option)
6. _Any_ use of a delegating wrapper is arguably "trappy" in the sense that you need to be mindful of what you should and should not override to do whatever it is you are doing. So I think we might as well delegate everything – at least at the time of creating the subclass you can look at the FilterTermsEnum and observe the methods to potentially override yourself. Today you need to know there are some "hidden" ones further below in the hierarchy. Sidenote: if we were all using Kotlin, we probably would not bother to have such Filter/delegate classes in Lucene because Kotlin [makes it trivial to auto-delegate all members|https://kotlinlang.org/docs/reference/delegation.html]. You still need to be mindful of what you need to override to do whatever it is you need to do.
7. I just realized that the current no-default-override behavior is actually enforced by a test TestFilterLeafReader.testOverrideMethods. I still think all methods should be overridden, but I understand that this may not be the expected behavior currently.
8. Maybe it would be helpful if we simply add a _comment_ to FilterTermsEnum of the methods it does not override but could still be by subclasses that wish to? e.g.: {code:java} //inherited methods with default impls that could still be overridden: // public AttributeSource attributes() // public boolean seekExact(BytesRef text) throws IOException // public void seekExact(BytesRef term, TermState state) throws IOException // public TermState termState() throws IOException {code} The main down-side is that it could be easy to forget to maintain this comment as changes happen over time. WDYT [~jpountz]?
9. [~dsmiley], if I create a subclass of FilterTermsEnum to override seekExact, how can I make other classes in Lucene create this subclass instead of FilterTermsEnum? Would I have to also override other classes or other factories?
10. Yes, you'd need to modify the other classes to override to get whatever behavior you want.
11. **body:** Actually there is also another related issue with this FilterLeafReader#FilterTermsEnum delegate pattern. It does not delegate termState() nor seekExact(ByteRef, TermState) methods. Which means the termState is never used, so the term queries repeat twice the same seek (seekCeil) instead of using the termState to improve performance (normally the termState is kept by TermContext#build()). Practical example: When one configures a timeout for queries, internally an ExitableDirectoryReader is created. And its ExitableTermsEnum, which extends FilterTermsEnum, makes all term queries repeat twice the same seekCeil().
label: code-design
12. Great example! I was playing around with TestExitableDirectoryReader and there's definitely a loss of passing the term state. I set a breakpoint here [https://github.com/apache/lucene-solr/blob/master/lucene/core/src/java/org/apache/lucene/search/TermQuery.java#L136] then ran in a debugger (after increasing the test timeout and removing the Ignore annotation) then stepped in to the default implementation of seekExact(term,state) for TestTermsEnum – which doesn't delegate. I manually added delegation of this method there. Then _again_ ran into the default implementation for ExtiableDirectoryReader's ExitableTermsEnum. In this one little adventure, I his this thing twice. _There's certainly a bug here_. Either FilterTermsEnum should delegate everything, or these two subclasses of TermsEnum mentioned above ought to delegate these methods but I bet there are more out there if we look closer. I appreciate modifying the delegation policy of FilterTermsEnum is not a decision to be taken lightly and would probably not happen until a major release.
13. Adrien last said: {quote}In general, we do not delegate methods that have a default implementation because the default implementation is correct regardless of what the wrapper class does. Overriding these methods in FilterTermsEnum to delegate to the wrapped instance would make room for bugs by requiring more methods to be overridden for the wrapper to be correct. {quote} CC [~simonw] curious about your thoughts too _In general_ I can see this. For the case of termState() and seekExact(termState) in particular, I don't. Hypothetically what could go wrong if FilterTermsEnum delegated? When I think of

filtering a TermsEnum, I think of something that might match a subset of terms from the underlying TermsEnum. The TermsEnum must be positioned to something that matches when termState is called. If seekExact(termState) in the underlying TermsEnum receives some termState impl it doesn't identify (isn't of a class it knows), then you get the default functional behavior, which is safe. I'm looking at the 6 subclasses of FilterTermsEnum we have in Lucene and I don't see an issue. (Interestingly, 3 of them are in the UnifiedHighlighter). I checked that delegating these two methods doesn't result in test failures too, aside from TestFilterLeafReader.testOverrideMethods which expressly tests our policy.

14. **body:** I do see both points here. [~dsmiley] I hate how trappy this is and [~jpountz] I completely agree with you. My suggestions here would be to add an additional class TermsEnum that has all methods abstract and BaseTermsEnum that can add default impls. FilterTermsEnum then subclasses TermsEnum and does the right thing. Other classes that don't need to override stuff like seekExact and seek(BytesRef, TermState) / TermState termState() can simply subclass BaseTermsEnum and we don't have to duplicate code all over the place. I don't think we need to do this in other places were we have the same pattern but in this case the traps are significant and we can fix it with a simple class in-between?
    label: code-design
15. +1 to your proposal Simon. Such a proposal could even result in BaseTermsEnum implementing seekExact.
16. **body:** I don't think there is a perfect fix here, the current approach in master is less prone to correctness bugs but more to performance bugs while Simon's proposal is less prone to performance bugs but more prone to correctness bugs, in the case that you forget to override one of the TermsEnum methods? I'm fine either way.
    label: code-design
17. I don't think I ever saw a "nice" solution to the problem of subclassing and delegating filters. Java's built-in classes suffer from the same problem (FilterInputStream for example).
18. the solution is, don't use delegators except over interfaces (or all-abstract).
19. I opened a PR here https://github.com/apache/lucene-solr/pull/574
20. Commit 4a513fa99f638cb65e0cae59bfdf7af410c0327a in lucene-solr's branch refs/heads/master from Simon Willnauer [https://gitbox.apache.org/repos/asf?p=lucene-solr.git;h=4a513fa] LUCENE-8292: Make TermsEnum fully abstract (#574)
21. Commit fd1fc2637071347201df3ecede659c47a0414d9d in lucene-solr's branch refs/heads/branch_8_0 from Simon Willnauer [https://gitbox.apache.org/repos/asf?p=lucene-solr.git;h=fd1fc26] LUCENE-8292: Make TermsEnum fully abstract (#574)
22. Commit 8dfbbec892d2db40e5e855b11ff8288e3524bac4 in lucene-solr's branch refs/heads/branch_8x from Simon Willnauer [https://gitbox.apache.org/repos/asf?p=lucene-solr.git;h=8dfbbec] LUCENE-8292: Make TermsEnum fully abstract (#574)
23. Thanks Simon. I didn't think this could get in to 8.x at the last second or I would have volunteered. FYI [~romseygeek] so you're aware.
24. [~dsmiley] I coordinated this with [~romseygeek] given that we had to respin for https://issues.apache.org/jira/browse/SOLR-13126 anyhow.