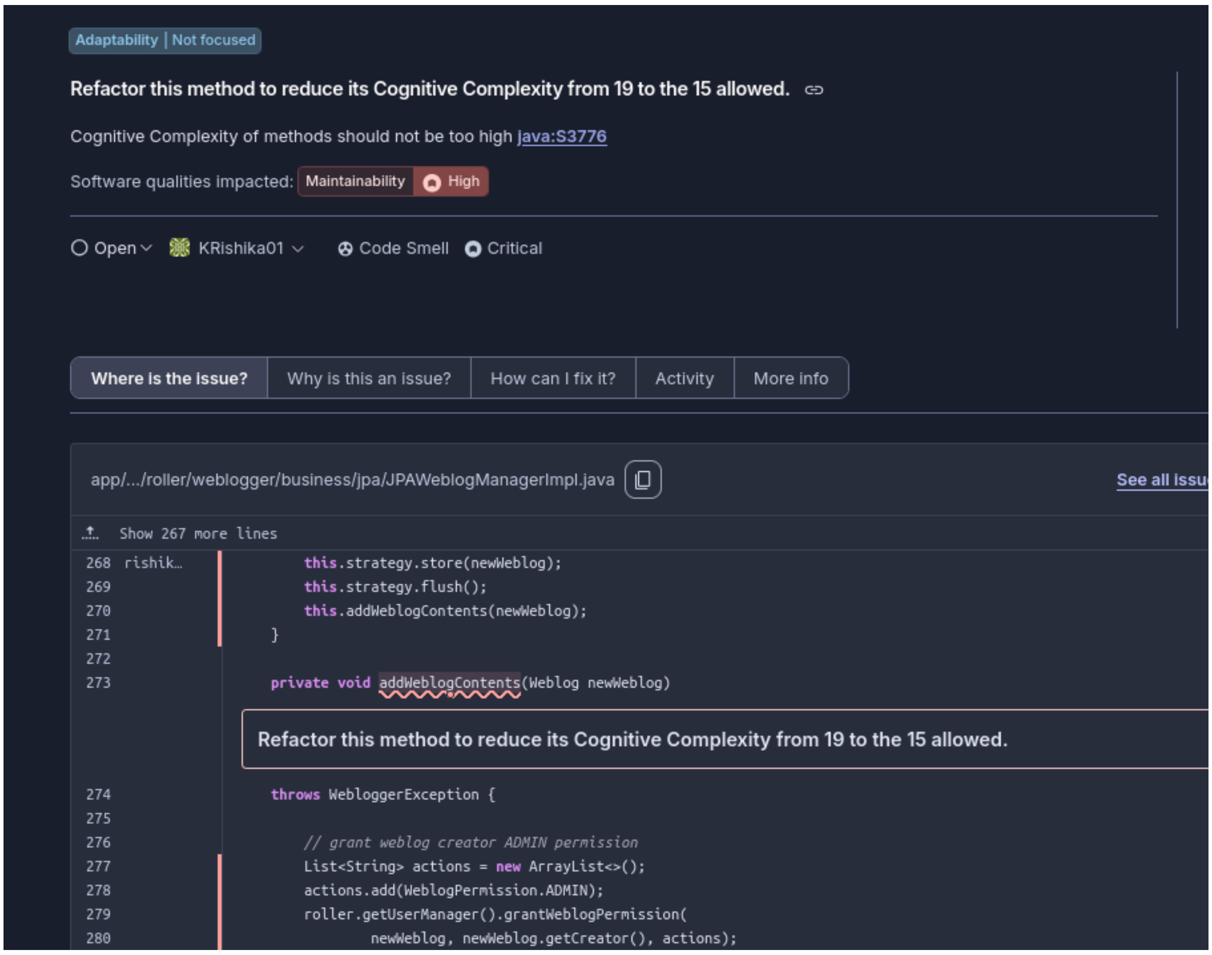
1. **JPAWeblogManagerImpl**
   Design Smell - **Feature Envy**

   A method has **Feature Envy** when it uses another class's data and methods more than its own class's data.

   SonarQube Reference:


Adaptability | Not focused

Refactor this method to reduce its Cognitive Complexity from 19 to the 15 allowed.

Cognitive Complexity of methods should not be too high java:S3776

Software qualities impacted: Maintainability High

Open KRishika01 Code Smell Critical

Where is the issue? | Why is this an issue? | How can I fix it? | Activity | More info

app/.../roller/weblogger/business/jpa/JPAWeblogManagerImpl.java

See all issu

Show 267 more lines

```
rishik…         this.strategy.store(newWeblog);
                this.strategy.flush();
                this.addWeblogContents(newWeblog);
            }

            private void addWeblogContents(Weblog newWeblog)
```

Refactor this method to reduce its Cognitive Complexity from 19 to the 15 allowed.

```
            throws WebloggerException {

                // grant weblog creator ADMIN permission
                List<String> actions = new ArrayList<>();
                actions.add(WeblogPermission.ADMIN);
                roller.getUserManager().grantWeblogPermission(
                        newWeblog, newWeblog.getCreator(), actions);
```

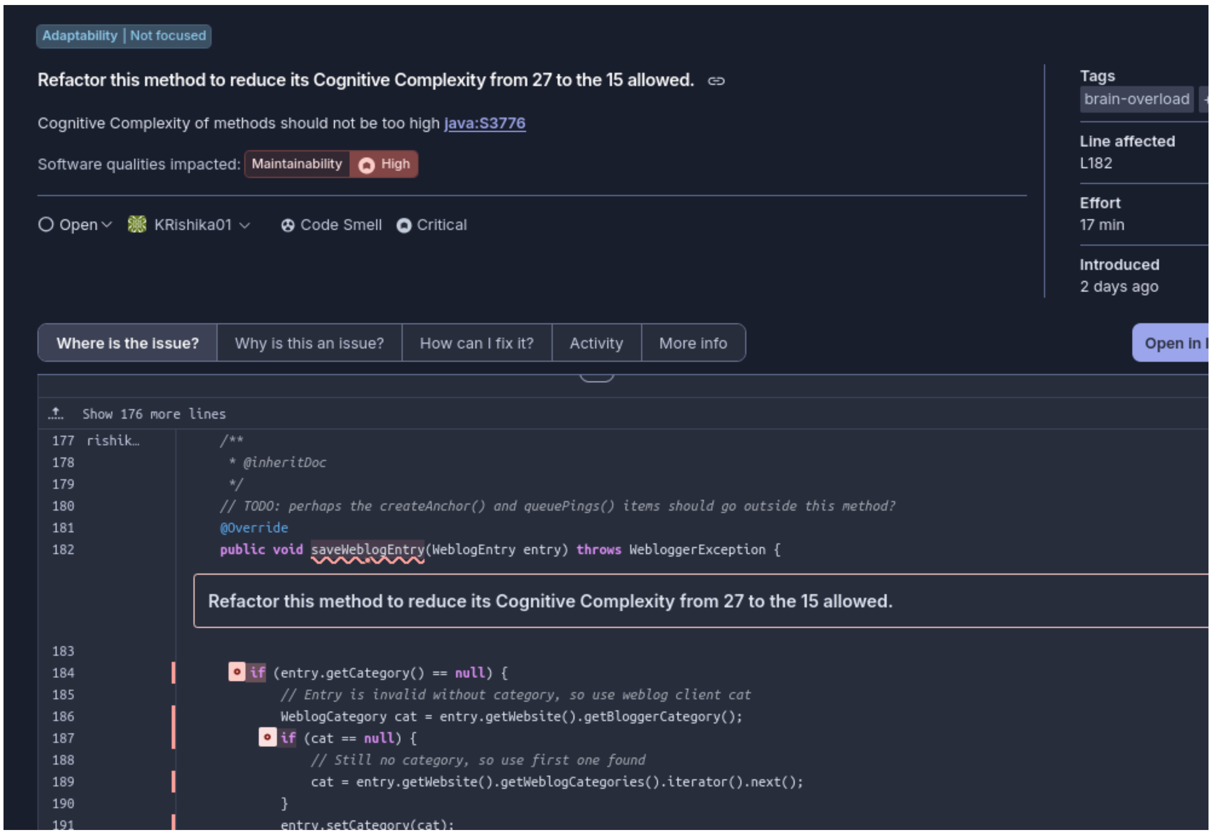
High Cognitive Complexity means

- Handles too many cases
- Coordinates many external objects
- Knows too much about another class

Designite Java Reference:
The tool detected an instance of this smell because addWeblogContents is more interested in members of the type: Weblogger.

2. **JPAWeblogEntryManagerImpl**
Design Smell - **Insufficient Modularization**

SonarQube Reference


Adaptability | Not focused

Refactor this method to reduce its Cognitive Complexity from 27 to the 15 allowed.

Cognitive Complexity of methods should not be too high java:S3776

Software qualities impacted: Maintainability High

Open KRishika01 Code Smell Critical

Tags: brain-overload
Line affected: L182
Effort: 17 min
Introduced: 2 days ago

Where is the issue? | Why is this an issue? | How can I fix it? | Activity | More info | Open in

```
Show 176 more lines
177 rishik…    /**
178             * @inheritDoc
179             */
180            // TODO: perhaps the createAnchor() and queuePings() items should go outside this method?
181            @Override
182            public void saveWeblogEntry(WeblogEntry entry) throws WebloggerException {
```

Refactor this method to reduce its Cognitive Complexity from 27 to the 15 allowed.

```
183
184                if (entry.getCategory() == null) {
185                    // Entry is invalid without category, so use weblog client cat
186                    WeblogCategory cat = entry.getWebsite().getBloggerCategory();
187                    if (cat == null) {
188                        // Still no category, so use first one found
189                        cat = entry.getWebsite().getWeblogCategories().iterator().next();
190                    }
191                    entry.setCategory(cat);
```

- The method has too much logic in one place
- Many branches, loops, or nested conditions
- The method is handling multiple responsibilities

Designite Java Reference:

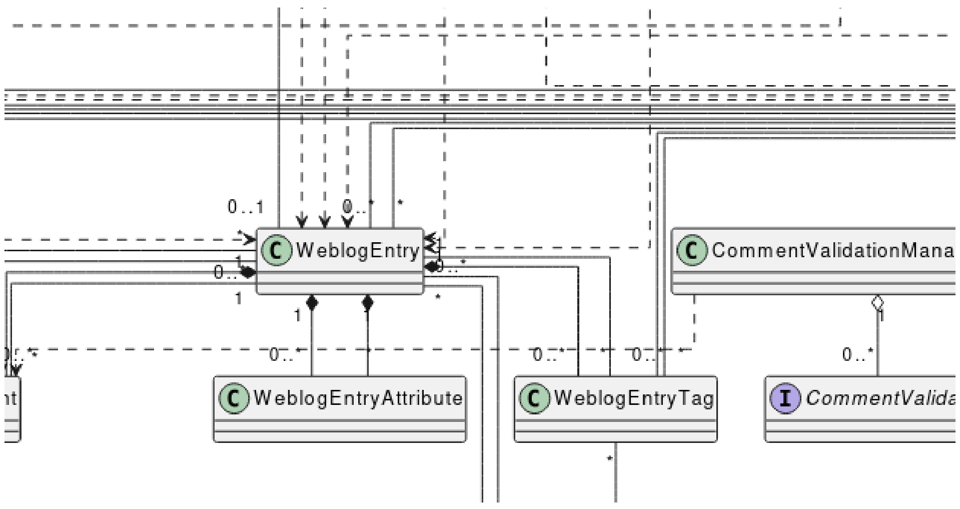
The tool detected the smell in this class because the class has a bloated interface (a large number of public methods). Total public methods in the class: 44 public methods

3. **WeblogEntry**
Design smell - **hub-like modularisation**

Designite reference:
The tool detected the smell in this class because this class has high number of incoming as well as outgoing dependencies. Incoming dependencies: EntryEdit; Entries; EntryRemove; Comments; EntryBean; WeblogEntriesPermalinkPager; WeblogEntriesListPager;

WeblogEntriesPreviewPager; WeblogEntriesLatestPager; WeblogEntriesMonthPager; WeblogEntriesDayPager; WeblogTrackbackRequest; WeblogCommentRequest; WeblogPreviewRequest; WeblogPageRequest; SiteWideCache; URLModel; TrackbackServlet; PageServlet; CommentServlet; WeblogCalendarModel; BigWeblogCalendarModel; WeblogEntryManager; PluginManagerImpl; PluginManager; EncodePreTagsPlugin; WeblogEntryPlugin; ObfuscateEmailPlugin; ConvertLineBreaksPlugin; SmileysPlugin; IndexOperation; AddEntryOperation; LuceneIndexManager; RemoveEntryOperation; ReIndexEntryOperation; IndexManager; AutoPingManager; JPAWeblogEntryManagerImpl; JPAAutoPingManagerImpl; Weblog; WeblogEntryComment; WeblogEntryTag; CommentSearchCriteria; WeblogEntryAttribute; WeblogEntryWrapper; CacheManager; CacheHandler; MailUtil; Trackback; RollerAtomHandler; EntryCollection; BloggerAPIHandler; MetaWeblogAPIHandler. Outgoing dependencies: WeblogEntry.PubStatus; Weblog; WeblogCategory; User; WebloggerFactory; WeblogEntryAttribute; WebloggerException; WeblogEntryTag; Utilities; WebloggerRuntimeConfig; WeblogEntryManager; CommentSearchCriteria; DateUtil; GlobalPermission; WeblogPermission; UserManager; WeblogEntryPlugin; HTMLSanitizer; WeblogEntryComment; RollerConstants

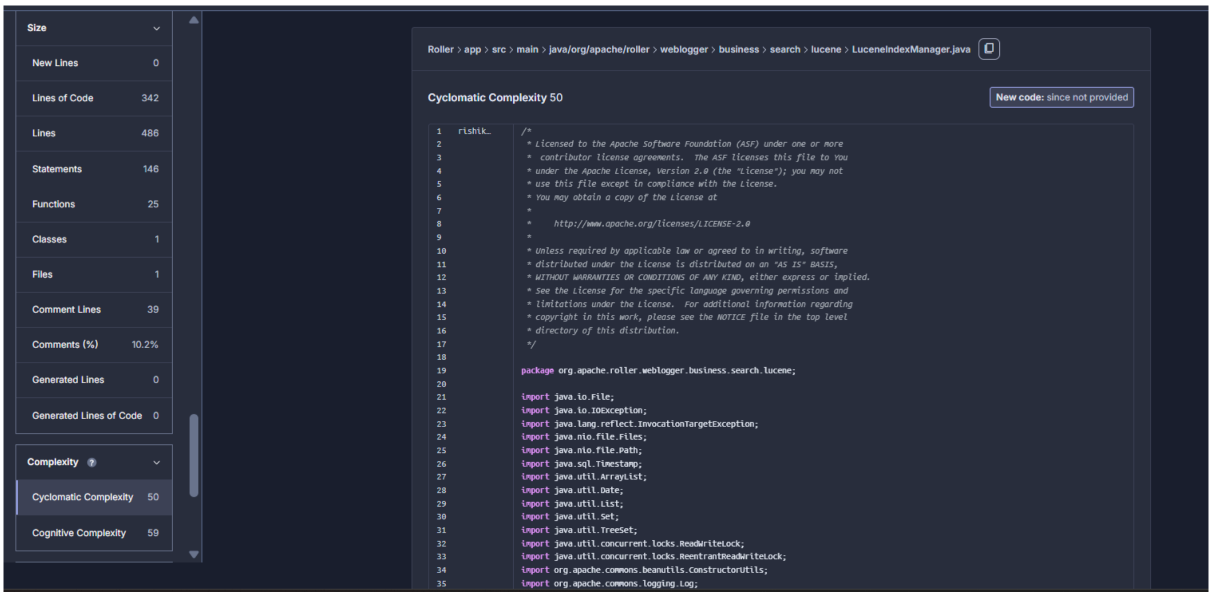
Uml reference:



- Acts as a **dependency hub** with very high fan-in and fan-out
- **Tightly coupled** to many unrelated modules

- Violates **single responsibility principle**
- Changes in this class can **impact many other classes**
- Difficult to **test, maintain, and evolve** independently
- Reduces overall **modularity and system stability**

4. **LuceneIndexManager**
   Design smell: Spaghetti Code
   Spaghetti code refers to source code with a convoluted, tangled, and unstructured control flow, making it difficult to understand, maintain, or extend.
   SonarQube Reference:



Size	
New Lines	0
Lines of Code	342
Lines	486
Statements	146
Functions	25
Classes	1
Files	1
Comment Lines	39
Comments (%)	10.2%
Generated Lines	0
Generated Lines of Code	0

Complexity	
Cyclomatic Complexity	50
Cognitive Complexity	59

Roller > app > src > main > java/org/apache/roller > weblogger > business > search > lucene > LuceneIndexManager.java

Cyclomatic Complexity 50

New code: since not provided

```
rishik…  /*
          * Licensed to the Apache Software Foundation (ASF) under one or more
          *  contributor license agreements.  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.  For additional information regarding
          * copyright in this work, please see the NOTICE file in the top level
          * directory of this distribution.
          */

         package org.apache.roller.weblogger.business.search.lucene;

         import java.io.File;
         import java.io.IOException;
         import java.lang.reflect.InvocationTargetException;
         import java.nio.file.Files;
         import java.nio.file.Path;
         import java.sql.Timestamp;
         import java.util.ArrayList;
         import java.util.Date;
         import java.util.List;
         import java.util.Set;
         import java.util.TreeSet;
         import java.util.concurrent.locks.ReadWriteLock;
         import java.util.concurrent.locks.ReentrantReadWriteLock;
         import org.apache.commons.beanutils.ConstructorUtils;
         import org.apache.commons.logging.Log;
```

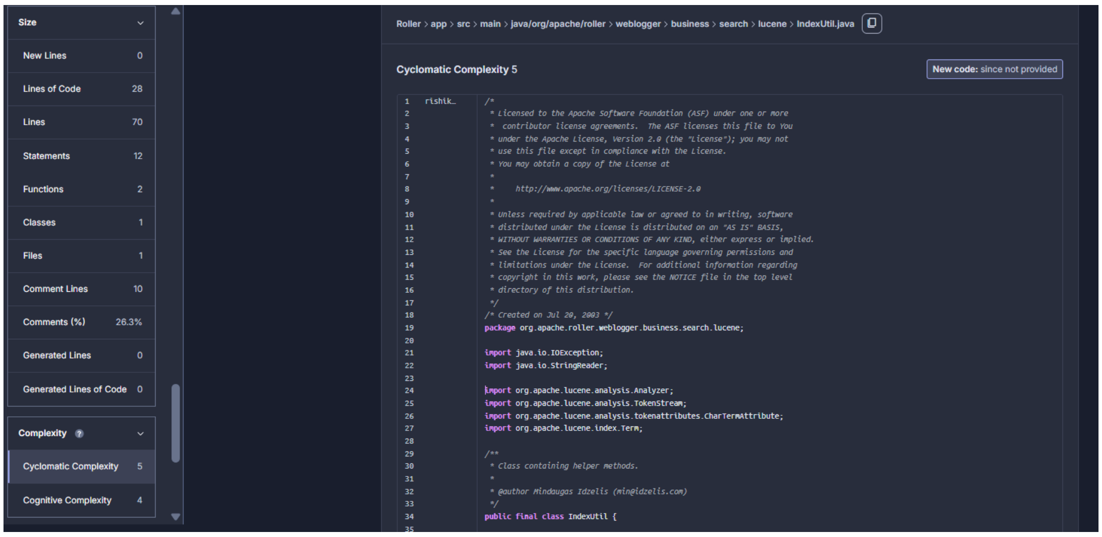
   From the picture we can see that,
   LOC: 342
   Cognitive complexity: 59
   Cyclomatic complexity: 50
   Indicators of Spaghetti code:
   1. Very high cyclomatic complexity (50)
      - Many branching paths hence tangled flow
   2. Very high cognitive complexity (59)
      - Nested conditionals mixed concerns, jumps in logic
   3. High LOC in a single class/method
      - 342 LOC → suggests long, monolithic logic blocks

5. **IndexUtil**

   Design smell: **Lazy class**

It only has one static method that wraps Lucene's Term creation, which is minimal functionality that doesn't justify a separate class

SonarQube reference:



Size	
New Lines	0
Lines of Code	28
Lines	70
Statements	12
Functions	2
Classes	1
Files	1
Comment Lines	10
Comments (%)	26.3%
Generated Lines	0
Generated Lines of Code	0

Complexity	
Cyclomatic Complexity	5
Cognitive Complexity	4

Roller > app > src > main > java/org/apache/roller > weblogger > business > search > lucene > IndexUtil.java

Cyclomatic Complexity 5

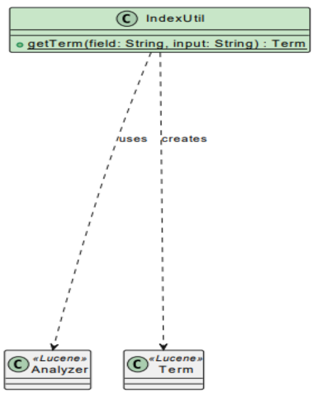
New code: since not provided

```
rishik…  /*
          * Licensed to the Apache Software Foundation (ASF) under one or more
          *  contributor license agreements.  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.  For additional information regarding
          * copyright in this work, please see the NOTICE file in the top level
          * directory of this distribution.
          */
         /* Created on Jul 20, 2003 */
         package org.apache.roller.weblogger.business.search.lucene;

         import java.io.IOException;
         import java.io.StringReader;

         import org.apache.lucene.analysis.Analyzer;
         import org.apache.lucene.analysis.TokenStream;
         import org.apache.lucene.analysis.tokenattributes.CharTermAttribute;
         import org.apache.lucene.index.Term;

         /**
          * Class containing helper methods.
          *
          * @author Mindaugas Idzelis (min@idzelis.com)
          */
         public final class IndexUtil {
```

Justification:

1. **Single method** - It only has one public static method `getTerm()`
2. **Minimal responsibility** - Just wraps Lucene token analysis into Term creation
3. **No meaningful abstraction** - It's just a "helper" class with no real identity
4. **Unnecessary indirection** - The ~20 lines of logic could easily live in the classes that actually use it

UML justification:



6. User Manager

Design Smell - God Service

Evidence from Designite Java:

Insufficient Modularization

The tool detected the smell in this class because the class contains a large number of methods. Total methods in the class: 31 methods.

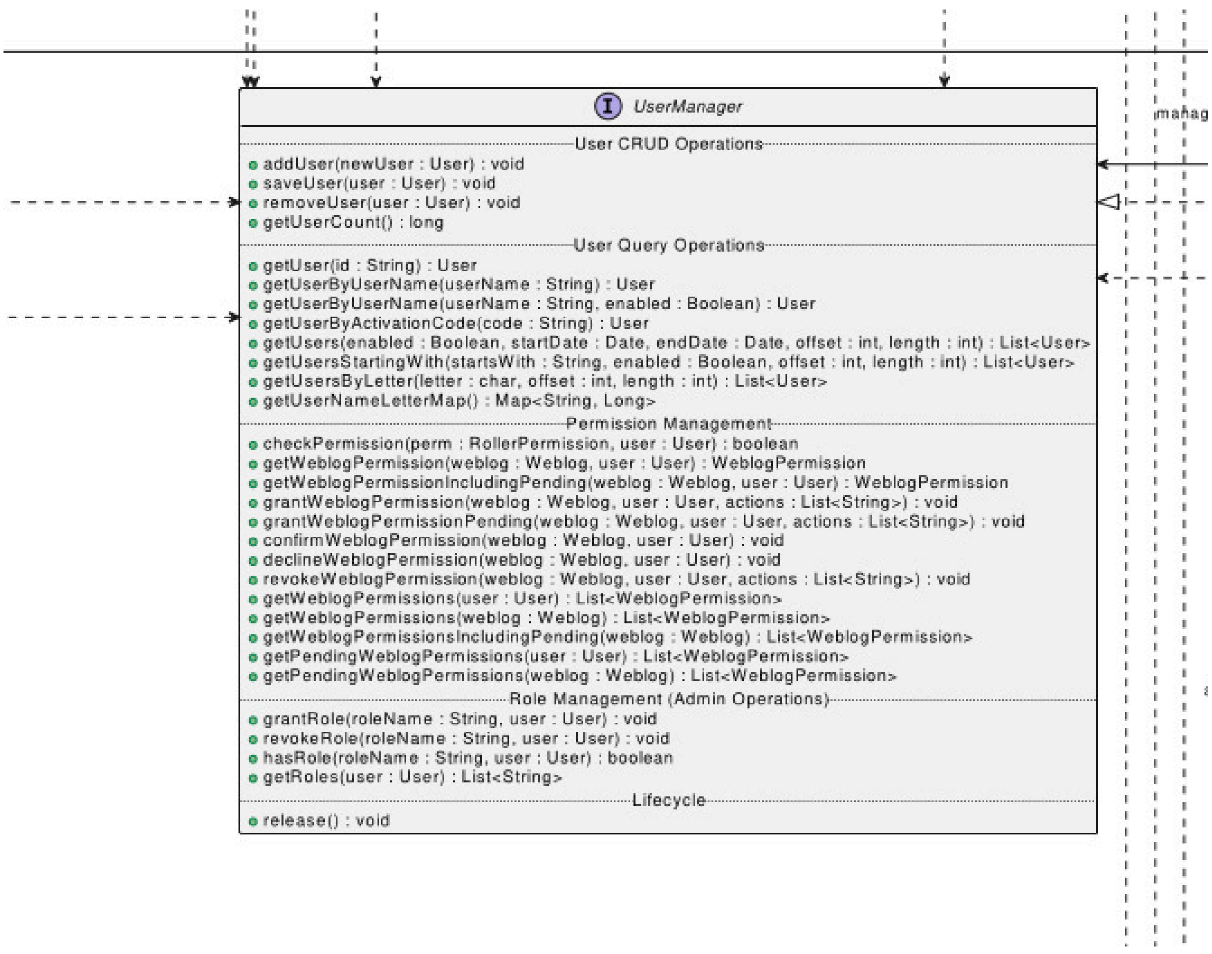
Justification:

UserManager mixes multiple business concerns in a single abstraction: user CRUD, user search, weblog permissions and admin role management. This violates the Single Responsibility Principle (SRP) and leads to a God Service smell. The concrete implementation `JPAUserManagerImpl` becomes a central "do‑everything" class that knows about user persistence, permission rules, role administration, and even user invitation/activation workflows indirectly.

Such concentration of responsibilities:

- Makes the class hard to understand and change (any change in permissions or roles risks breaking unrelated user CRUD or queries).
- Increases the chance of bugs, because behavior is cross‑cutting and interleaved.
- Reduces testability, as unit tests must set up many collaborators and scenarios.

A more cohesive design would separate UserRepository/UserQueryService, PermissionService, and RoleAdminService, with `UserManager` coordinating instead of owning all details.



```
UserManager
User CRUD Operations
addUser(newUser : User) : void
saveUser(user : User) : void
removeUser(user : User) : void
getUserCount() : long
User Query Operations
getUser(id : String) : User
getUserByUserName(userName : String) : User
getUserByUserName(userName : String, enabled : Boolean) : User
getUserByActivationCode(code : String) : User
getUsers(enabled : Boolean, startDate : Date, endDate : Date, offset : int, length : int) : List<User>
getUsersStartingWith(startsWith : String, enabled : Boolean, offset : int, length : int) : List<User>
getUsersByLetter(letter : char, offset : int, length : int) : List<User>
getUserNameLetterMap() : Map<String, Long>
Permission Management
checkPermission(perm : RollerPermission, user : User) : boolean
getWeblogPermission(weblog : Weblog, user : User) : WeblogPermission
getWeblogPermissionIncludingPending(weblog : Weblog, user : User) : WeblogPermission
grantWeblogPermission(weblog : Weblog, user : User, actions : List<String>) : void
grantWeblogPermissionPending(weblog : Weblog, user : User, actions : List<String>) : void
confirmWeblogPermission(weblog : Weblog, user : User) : void
declineWeblogPermission(weblog : Weblog, user : User) : void
revokeWeblogPermission(weblog : Weblog, user : User, actions : List<String>) : void
getWeblogPermissions(user : User) : List<WeblogPermission>
getWeblogPermissions(weblog : Weblog) : List<WeblogPermission>
getWeblogPermissionsIncludingPending(weblog : Weblog) : List<WeblogPermission>
getPendingWeblogPermissions(user : User) : List<WeblogPermission>
getPendingWeblogPermissions(weblog : Weblog) : List<WeblogPermission>
Role Management (Admin Operations)
grantRole(roleName : String, user : User) : void
revokeRole(roleName : String, user : User) : void
hasRole(roleName : String, user : User) : boolean
getRoles(user : User) : List<String>
Lifecycle
release() : void
```