Derek Aguirre

Contents

Introduction

The goal of this assignment is to analyze codebases using PMD. The tool will help to find specific issues in the regarding design or codestyle. ModelMine will be used as a tool for searching for various codebases that meet a few criteria:

- Majority of code must be written in Java
- Projects within a 5 year timeline
- No Academic Projects
- Nontrivial Projects
    - (4-5 Contributors, 100 commits, 2 GitHub Stars, Non-fork)

The five codebases I will be going over (along with the git links) are provided below.

Methodology

This section will be split into two parts: ModelMine, and PMD Code Analyzation & Refactoring Process.

ModelMine

I conducted my research through ModelMine to find the following codebases:

- Google Cloud Dataflow Template Pipelines
    - https://github.com/GoogleCloudPlatform/DataflowTemplates
- Android ChatBot powered by IBM Watson
    - https://github.com/IBM-Cloud/chatbot-watson-android
- GoogleAuth
    - https://github.com/wstrange/GoogleAuth
- Spring Cloud Function
    - https://github.com/spring-cloud/spring-cloud-function
- Spring PetClinic Microservices
    - https://github.com/spring-petclinic/spring-petclinic-microservices

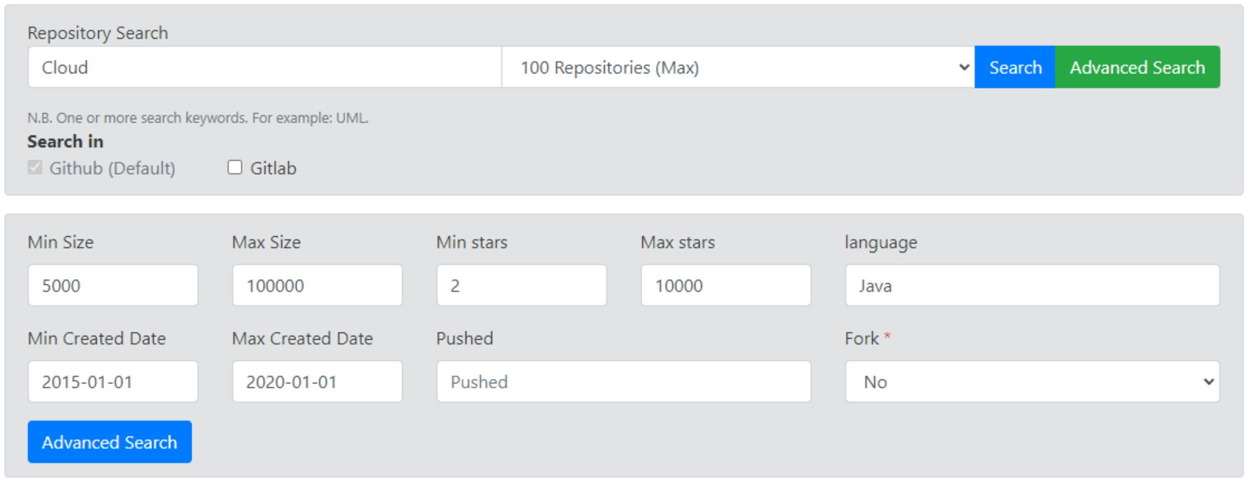
Here's a quick summary of how I utilized the ModelMine tool:

I used the advanced search feature to yield a specific subset of codebases. I made sure that the codebases had at least:

- Min and Max Sizes of 5,000-100,000 respectively

- Min and Max Stars of 2 – 10,000 respectively
- Predominately Java-Based
- Within a 5 year period of Jan 2015 – Jan 2020
- Non-forked code base

This is how I inputted the constraints into the tool:



Repository Search

Cloud | 100 Repositories (Max) | Search | Advanced Search

N.B. One or more search keywords. For example: UML.

Search in

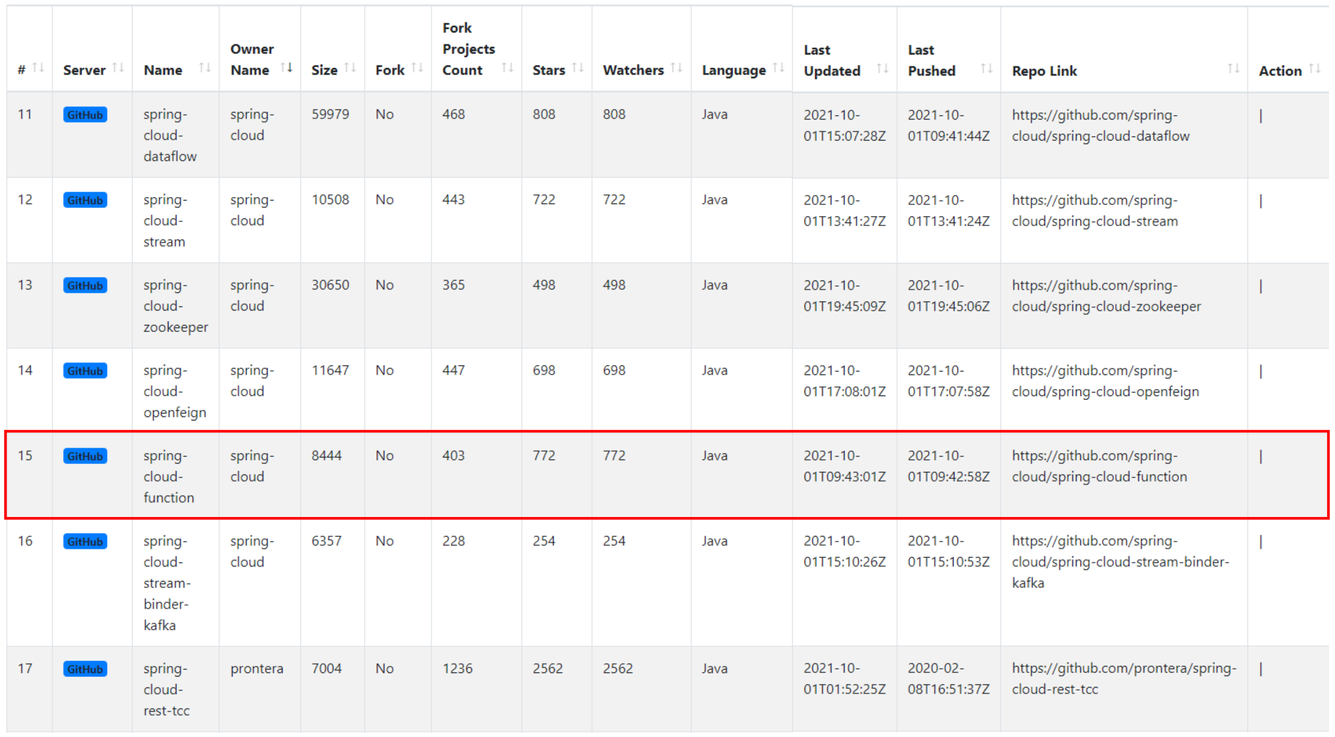
[✓ Github (Default)] Gitlab

Min Size	Max Size	Min stars	Max stars	language
5000	100000	2	10000	Java

Min Created Date	Max Created Date	Pushed	Fork *
2015-01-01	2020-01-01	Pushed	No

Advanced Search

With the search now complete, it allows me to select any of the codebases from a selection that meet the criteria I provided. Entry '15' in this list is one of the codebases I will be looking at for this project



#	Server	Name	Owner Name	Size	Fork	Fork Projects Count	Stars	Watchers	Language	Last Updated	Last Pushed	Repo Link	Action
11	GitHub	spring-cloud-dataflow	spring-cloud	59979	No	468	808	808	Java	2021-10-01T15:07:28Z	2021-10-01T09:41:44Z	https://github.com/spring-cloud/spring-cloud-dataflow	|
12	GitHub	spring-cloud-stream	spring-cloud	10508	No	443	722	722	Java	2021-10-01T13:41:27Z	2021-10-01T13:41:24Z	https://github.com/spring-cloud/spring-cloud-stream	|
13	GitHub	spring-cloud-zookeeper	spring-cloud	30650	No	365	498	498	Java	2021-10-01T19:45:09Z	2021-10-01T19:45:06Z	https://github.com/spring-cloud/spring-cloud-zookeeper	|
14	GitHub	spring-cloud-openfeign	spring-cloud	11647	No	447	698	698	Java	2021-10-01T17:08:01Z	2021-10-01T17:07:58Z	https://github.com/spring-cloud/spring-cloud-openfeign	|
15	GitHub	spring-cloud-function	spring-cloud	8444	No	403	772	772	Java	2021-10-01T09:43:01Z	2021-10-01T09:42:58Z	https://github.com/spring-cloud/spring-cloud-function	|
16	GitHub	spring-cloud-stream-binder-kafka	spring-cloud	6357	No	228	254	254	Java	2021-10-01T15:10:26Z	2021-10-01T15:10:53Z	https://github.com/spring-cloud/spring-cloud-stream-binder-kafka	|
17	GitHub	spring-cloud-rest-tcc	prontera	7004	No	1236	2562	2562	Java	2021-10-01T01:52:25Z	2020-02-08T16:51:37Z	https://github.com/prontera/spring-cloud-rest-tcc	|

PMD Code Analyzation & Refactoring

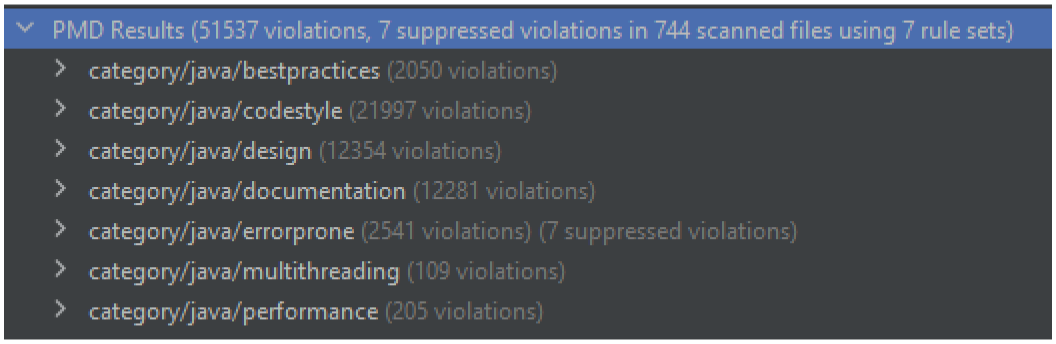
I have selected 5 codebases to analyze with PMD to see if they violate any code smell violations, and if so, I will show my approach to fixing a selected issue. Specifically I will look for two design violations, and one code style violation.

Google Cloud Dataflow Template Pipelines

Initial Requirements…

- 99.5% Java
- Updated in 2021
- 48 Contributors
- 576 Commits
- 730 Stars

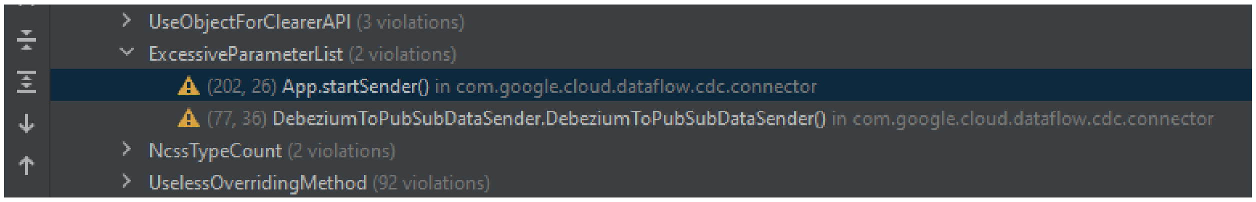
According to PMD, there are 51,537 code smell violations in this codebase.



```
PMD Results (51537 violations, 7 suppressed violations in 744 scanned files using 7 rule sets)
    category/java/bestpractices (2050 violations)
    category/java/codestyle (21997 violations)
    category/java/design (12354 violations)
    category/java/documentation (12281 violations)
    category/java/errorprone (2541 violations) (7 suppressed violations)
    category/java/multithreading (109 violations)
    category/java/performance (205 violations)
```

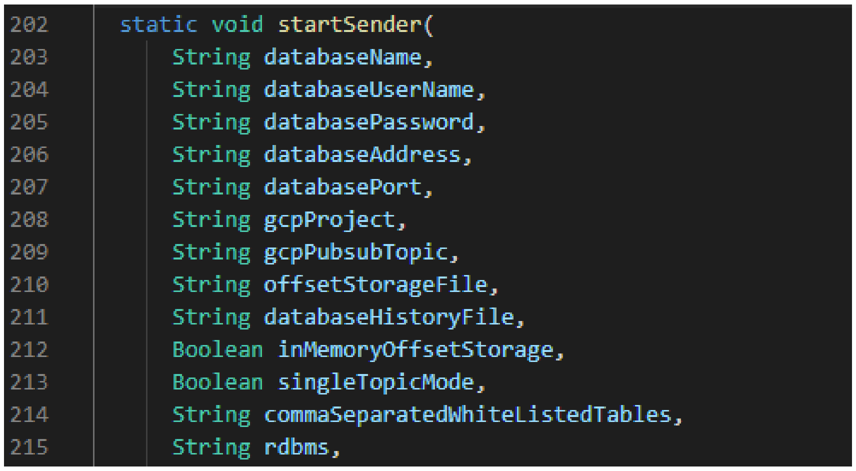
Design Error 1: Excessive Parameter List

There is a method that has too many parameters and thus will be much more difficult to work when trying to use it later on.



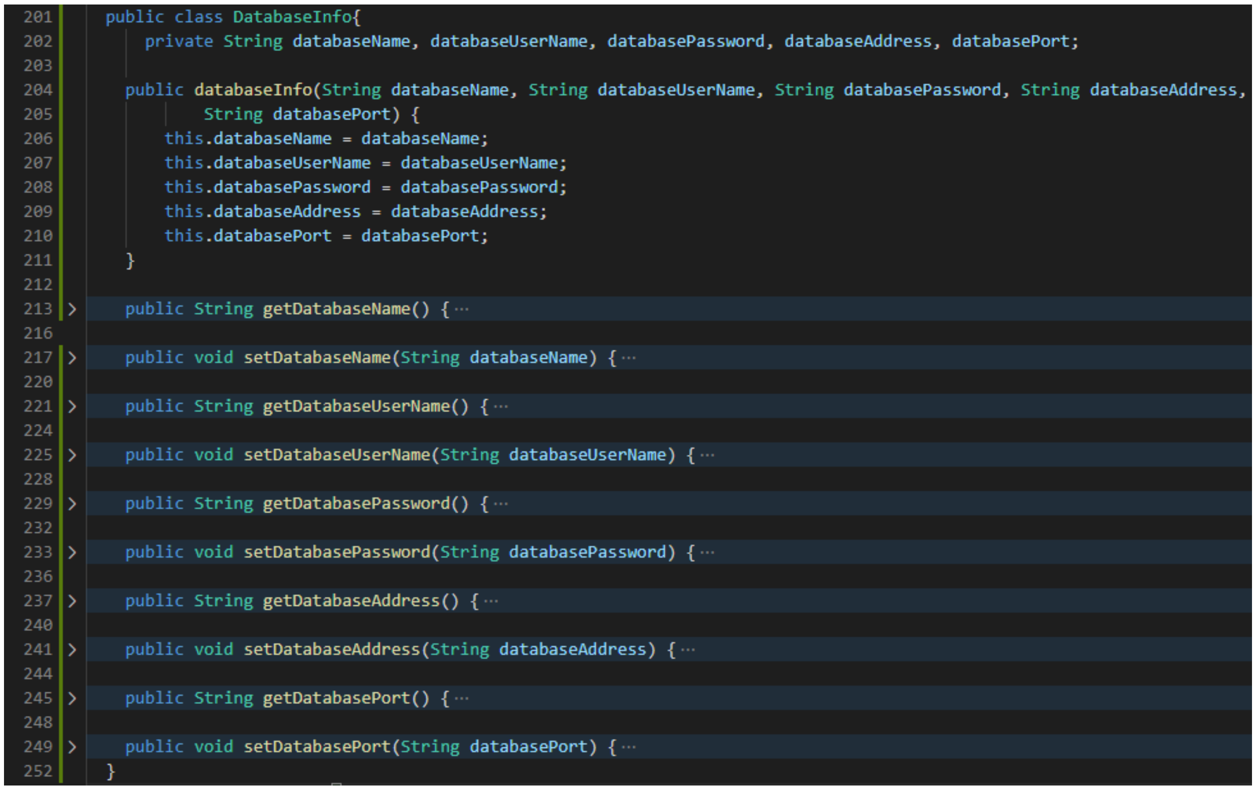
```
    UseObjectForClearerAPI (3 violations)
    ExcessiveParameterList (2 violations)
        (202, 26) App.startSender() in com.google.cloud.dataflow.cdc.connector
        (77, 36) DebeziumToPubSubDataSender.DebeziumToPubSubDataSender() in com.google.cloud.dataflow.cdc.connector
    NcssTypeCount (2 violations)
    UselessOverridingMethod (92 violations)
```

My approach to refactoring this issue will be to migrate most fields of the method into an object to help alleviate the clutter.



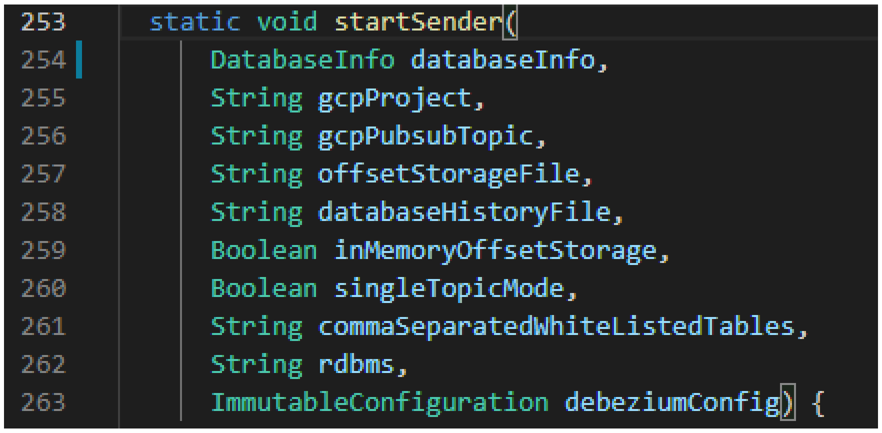
```
static void startSender(
    String databaseName,
    String databaseUserName,
    String databasePassword,
    String databaseAddress,
    String databasePort,
    String gcpProject,
    String gcpPubsubTopic,
    String offsetStorageFile,
    String databaseHistoryFile,
    Boolean inMemoryOffsetStorage,
    Boolean singleTopicMode,
    String commaSeparatedWhiteListedTables,
    String rdbms,
```

This is my proposed solution by taking out all the database* fields and migrating them to an object



```
public class DatabaseInfo{
    private String databaseName, databaseUserName, databasePassword, databaseAddress, databasePort;

  public databaseInfo(String databaseName, String databaseUserName, String databasePassword, String databaseAddress,
        String databasePort) {
      this.databaseName = databaseName;
      this.databaseUserName = databaseUserName;
      this.databasePassword = databasePassword;
      this.databaseAddress = databaseAddress;
      this.databasePort = databasePort;
  }

  public String getDatabaseName() {···

  public void setDatabaseName(String databaseName) {···

  public String getDatabaseUserName() {···

  public void setDatabaseUserName(String databaseUserName) {···

  public String getDatabasePassword() {···

  public void setDatabasePassword(String databasePassword) {···

  public String getDatabaseAddress() {···

  public void setDatabaseAddress(String databaseAddress) {···

  public String getDatabasePort() {···

  public void setDatabasePort(String databasePort) {···
}
```

This is how the object would look as a parameter, alleviating some of the clutter.

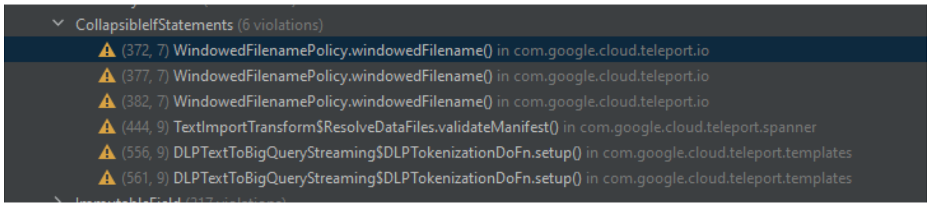


```java
static void startSender(
    DatabaseInfo databaseInfo,
    String gcpProject,
    String gcpPubsubTopic,
    String offsetStorageFile,
    String databaseHistoryFile,
    Boolean inMemoryOffsetStorage,
    Boolean singleTopicMode,
    String commaSeparatedWhiteListedTables,
    String rdbms,
    ImmutableConfiguration debeziumConfig) {
```

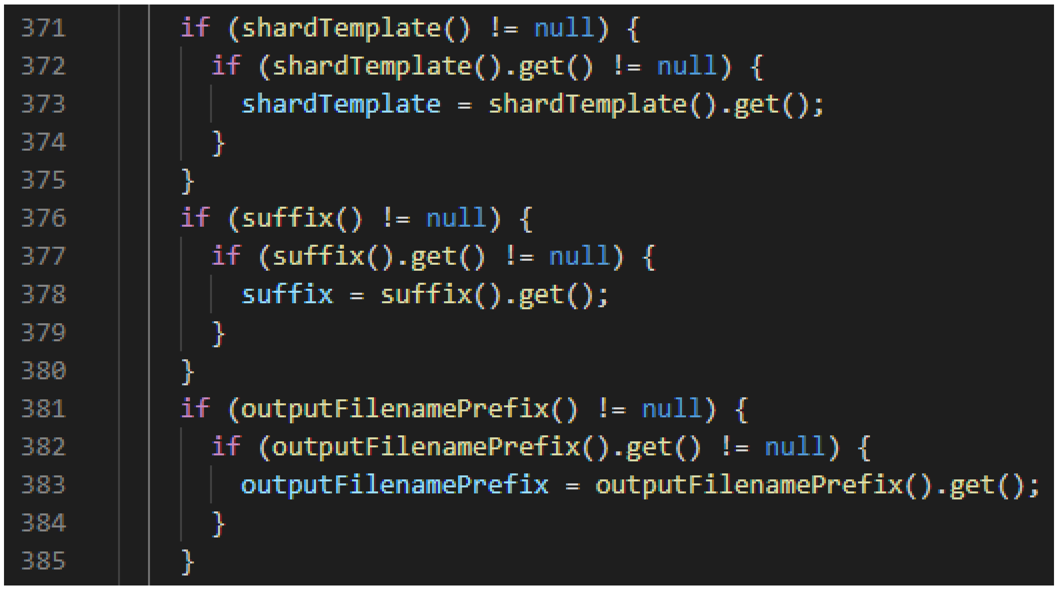
Design Error 2: Collapsible If Statements

This issue arises when an if statement is unnecessarily nested and can be collapsed by separating the conditions with a boolean short-circuit operator.

This is what the issue looks like in code:

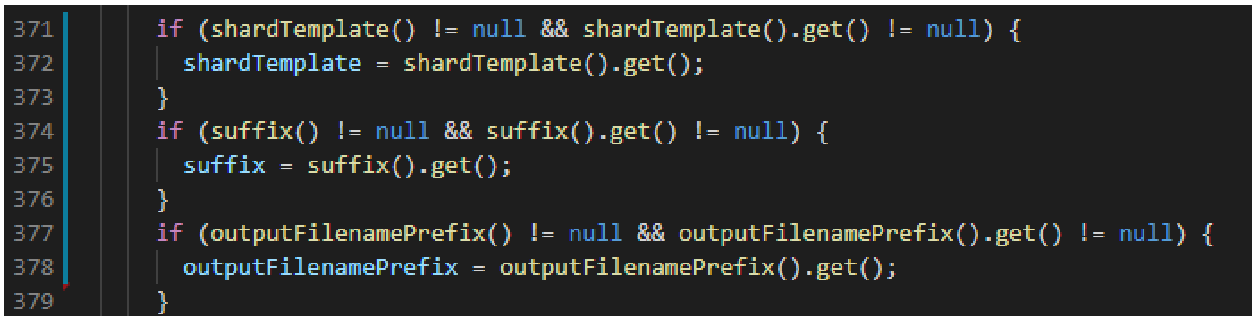


```
⌄ CollapsibleIfStatements (6 violations)
    ⚠ (372, 7) WindowedFilenamePolicy.windowedFilename() in com.google.cloud.teleport.io
    ⚠ (377, 7) WindowedFilenamePolicy.windowedFilename() in com.google.cloud.teleport.io
    ⚠ (382, 7) WindowedFilenamePolicy.windowedFilename() in com.google.cloud.teleport.io
    ⚠ (444, 9) TextImportTransform$ResolveDataFiles.validateManifest() in com.google.cloud.teleport.spanner
    ⚠ (556, 9) DLPTextToBigQueryStreaming$DLPTokenizationDoFn.setup() in com.google.cloud.teleport.templates
    ⚠ (561, 9) DLPTextToBigQueryStreaming$DLPTokenizationDoFn.setup() in com.google.cloud.teleport.templates
```



```java
371  if (shardTemplate() != null) {
372    if (shardTemplate().get() != null) {
373      shardTemplate = shardTemplate().get();
374    }
375  }
376  if (suffix() != null) {
377    if (suffix().get() != null) {
378      suffix = suffix().get();
379    }
380  }
381  if (outputFilenamePrefix() != null) {
382    if (outputFilenamePrefix().get() != null) {
383      outputFilenamePrefix = outputFilenamePrefix().get();
384    }
385  }
```

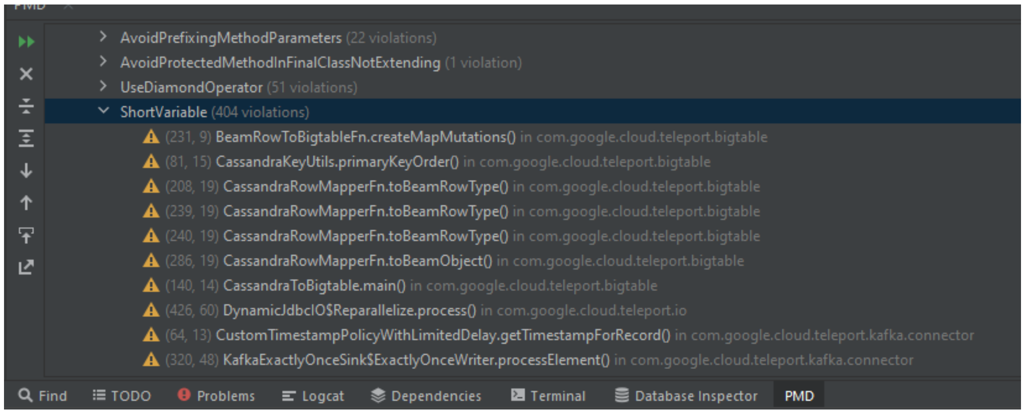
The screenshot below is my approach to fixing it.



```
if (shardTemplate() != null && shardTemplate().get() != null) {
  shardTemplate = shardTemplate().get();
}
if (suffix() != null && suffix().get() != null) {
  suffix = suffix().get();
}
if (outputFilenamePrefix() != null && outputFilenamePrefix().get() != null) {
  outputFilenamePrefix = outputFilenamePrefix().get();
}
```

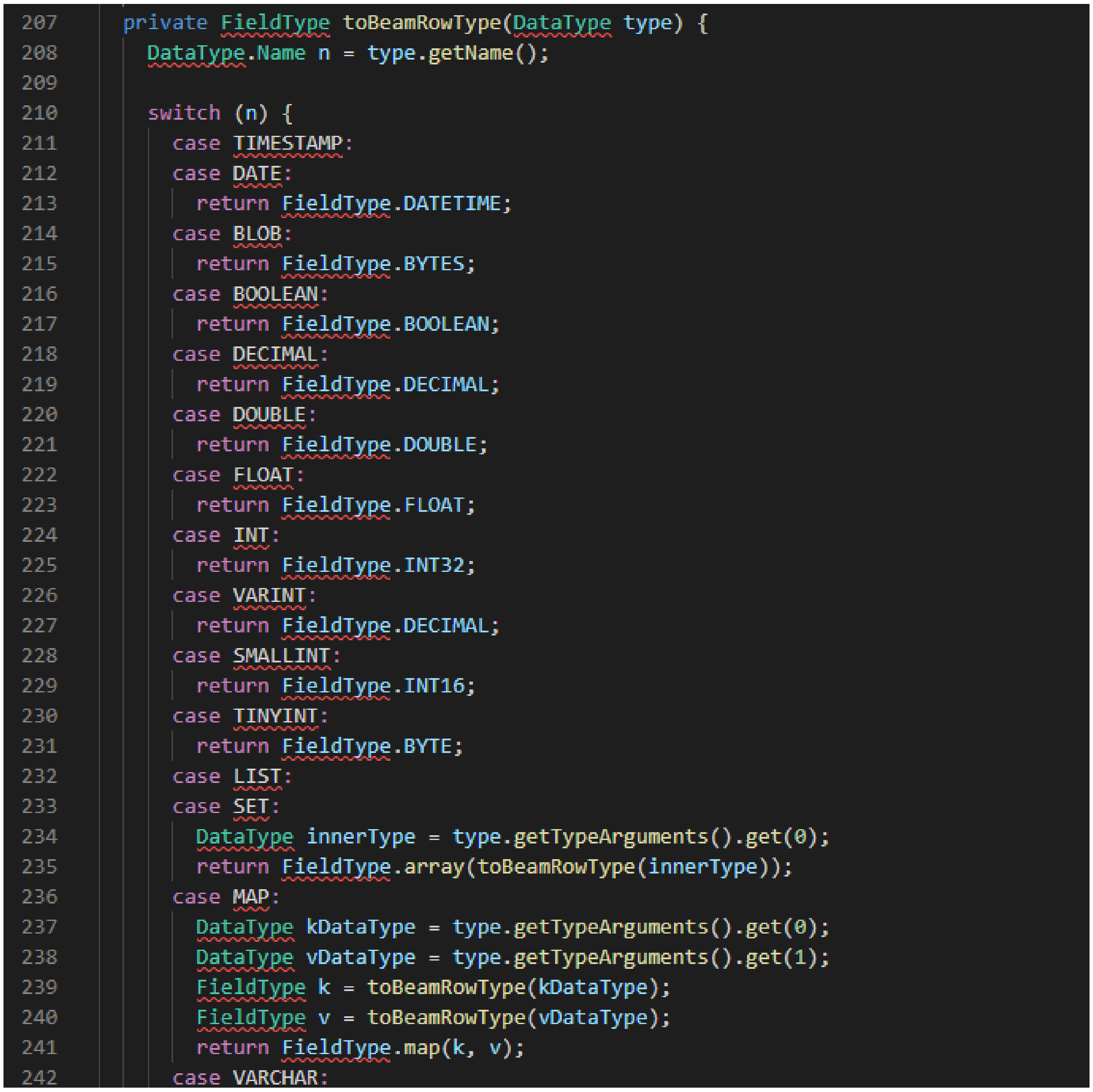
Code Style Error: Short Variable Names

This issue is a simple fix but it can get tricky the more times someone uses this variable name, especially when dealing with objects and bigger functions.



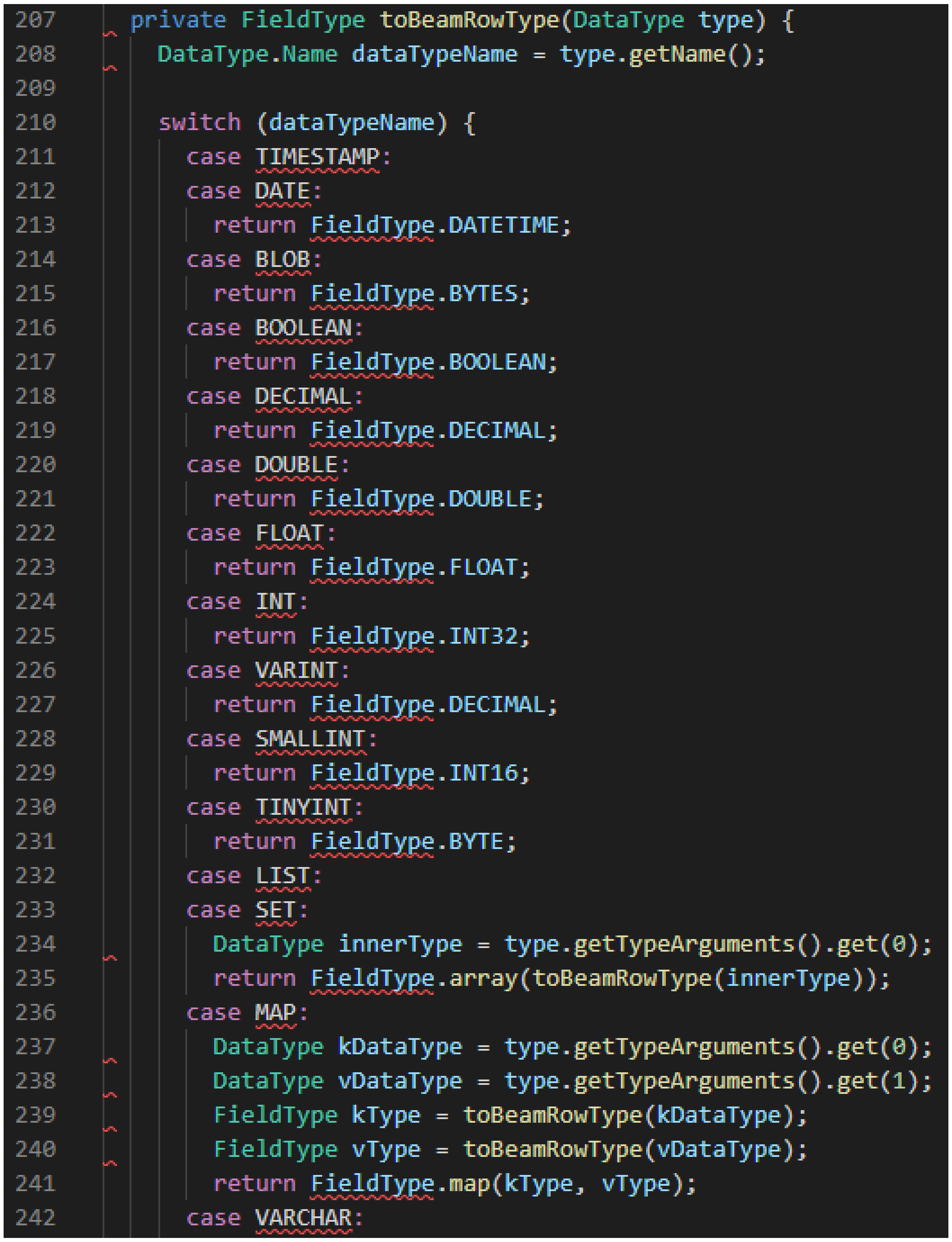
```
> AvoidPrefixingMethodParameters (22 violations)
> AvoidProtectedMethodInFinalClassNotExtending (1 violation)
> UseDiamondOperator (51 violations)
v ShortVariable (404 violations)
    (231, 9) BeamRowToBigtableFn.createMapMutations() in com.google.cloud.teleport.bigtable
    (81, 15) CassandraKeyUtils.primaryKeyOrder() in com.google.cloud.teleport.bigtable
    (208, 19) CassandraRowMapperFn.toBeamRowType() in com.google.cloud.teleport.bigtable
    (239, 19) CassandraRowMapperFn.toBeamRowType() in com.google.cloud.teleport.bigtable
    (240, 19) CassandraRowMapperFn.toBeamRowType() in com.google.cloud.teleport.bigtable
    (286, 19) CassandraRowMapperFn.toBeamObject() in com.google.cloud.teleport.bigtable
    (140, 14) CassandraToBigtable.main() in com.google.cloud.teleport.bigtable
    (426, 60) DynamicJdbcIO$Reparallelize.process() in com.google.cloud.teleport.io
    (64, 13) CustomTimestampPolicyWithLimitedDelay.getTimestampForRecord() in com.google.cloud.teleport.kafka.connector
    (320, 48) KafkaExactlyOnceSink$ExactlyOnceWriter.processElement() in com.google.cloud.teleport.kafka.connector
```

This is how the issue looks inside a file in the codebase:



```
  private FieldType toBeamRowType(DataType type) {
    DataType.Name n = type.getName();

    switch (n) {
      case TIMESTAMP:
      case DATE:
        return FieldType.DATETIME;
      case BLOB:
        return FieldType.BYTES;
      case BOOLEAN:
        return FieldType.BOOLEAN;
      case DECIMAL:
        return FieldType.DECIMAL;
      case DOUBLE:
        return FieldType.DOUBLE;
      case FLOAT:
        return FieldType.FLOAT;
      case INT:
        return FieldType.INT32;
      case VARINT:
        return FieldType.DECIMAL;
      case SMALLINT:
        return FieldType.INT16;
      case TINYINT:
        return FieldType.BYTE;
      case LIST:
      case SET:
        DataType innerType = type.getTypeArguments().get(0);
        return FieldType.array(toBeamRowType(innerType));
      case MAP:
        DataType kDataType = type.getTypeArguments().get(0);
        DataType vDataType = type.getTypeArguments().get(1);
        FieldType k = toBeamRowType(kDataType);
        FieldType v = toBeamRowType(vDataType);
        return FieldType.map(k, v);
      case VARCHAR:
```

My fix for the variable names is as follows: n -> dataTypeName, k -> kType, v -> vType along with some other fixes throughout the file, those changes can be viewed on my github.



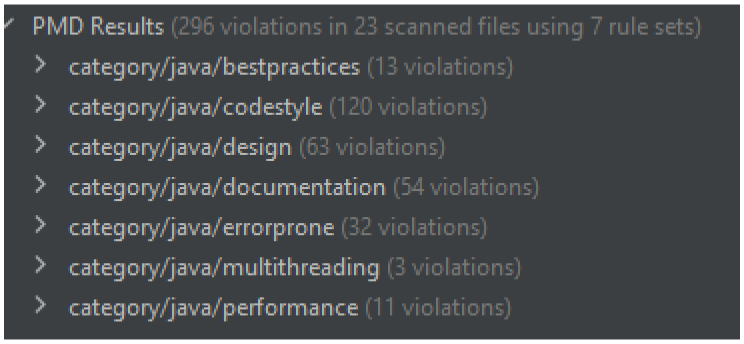
```java
private FieldType toBeamRowType(DataType type) {
  DataType.Name dataTypeName = type.getName();

  switch (dataTypeName) {
    case TIMESTAMP:
    case DATE:
      return FieldType.DATETIME;
    case BLOB:
      return FieldType.BYTES;
    case BOOLEAN:
      return FieldType.BOOLEAN;
    case DECIMAL:
      return FieldType.DECIMAL;
    case DOUBLE:
      return FieldType.DOUBLE;
    case FLOAT:
      return FieldType.FLOAT;
    case INT:
      return FieldType.INT32;
    case VARINT:
      return FieldType.DECIMAL;
    case SMALLINT:
      return FieldType.INT16;
    case TINYINT:
      return FieldType.BYTE;
    case LIST:
    case SET:
      DataType innerType = type.getTypeArguments().get(0);
      return FieldType.array(toBeamRowType(innerType));
    case MAP:
      DataType kDataType = type.getTypeArguments().get(0);
      DataType vDataType = type.getTypeArguments().get(1);
      FieldType kType = toBeamRowType(kDataType);
      FieldType vType = toBeamRowType(vDataType);
      return FieldType.map(kType, vType);
    case VARCHAR:
```

Android Chatbot Powered by IBM Watson

Initial Requirements…

- 100% Java
- Updated in 2020
- 4 Contributors
- 149 Commits
- 180 Stars

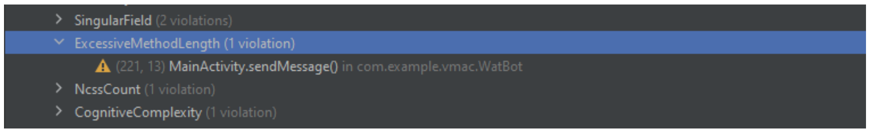
According to PMD, there are <u>296</u> code smell violations in this codebase.



```
PMD Results (296 violations in 23 scanned files using 7 rule sets)
  > category/java/bestpractices (13 violations)
  > category/java/codestyle (120 violations)
  > category/java/design (63 violations)
  > category/java/documentation (54 violations)
  > category/java/errorprone (32 violations)
  > category/java/multithreading (3 violations)
  > category/java/performance (11 violations)
```

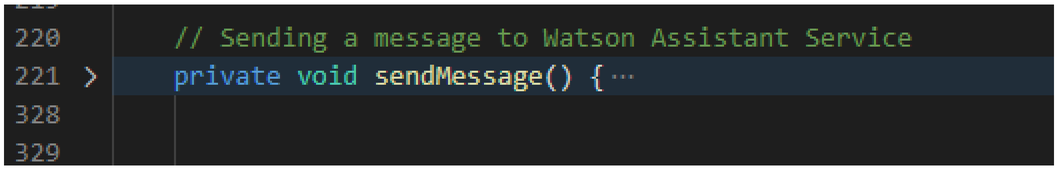
Design Error 1: Excessive Method Length

A method in this file is too long and can become difficult to read/understand later.



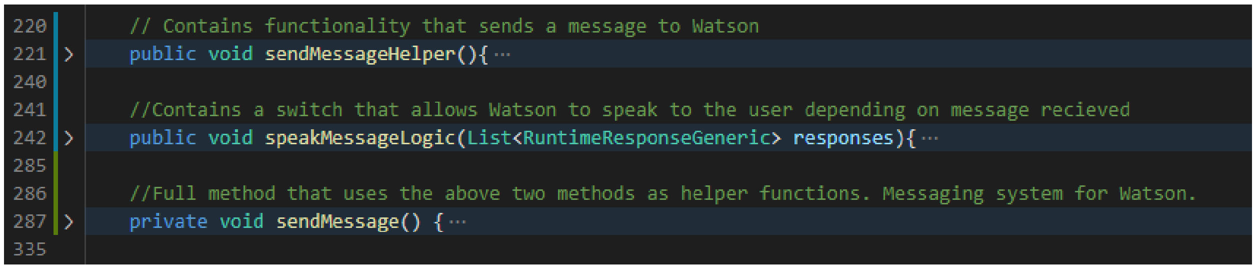
```
  > SingularField (2 violations)
  v ExcessiveMethodLength (1 violation)
      (221, 13) MainActivity.sendMessage() in com.example.vmac.WatBot
  > NcssCount (1 violation)
  > CognitiveComplexity (1 violation)
```

This is the offending method, which spans lines 221-328 which is a total of 107 lines.



```
220     // Sending a message to Watson Assistant Service
221     private void sendMessage() {…
328
329
```

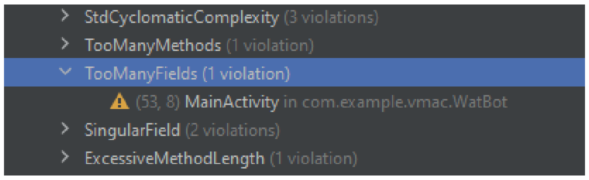
A potential fix can be adding helper functions to assist with the computation of the method. With the addition of a couple helper functions, the sendMessage() function now spans a total of 48 lines of code.



```
220     // Contains functionality that sends a message to Watson
221     public void sendMessageHelper(){…
240
241     //Contains a switch that allows Watson to speak to the user depending on message recieved
242     public void speakMessageLogic(List<RuntimeResponseGeneric> responses){…
285
286     //Full method that uses the above two methods as helper functions. Messaging system for Watson.
287     private void sendMessage() {…
335
```

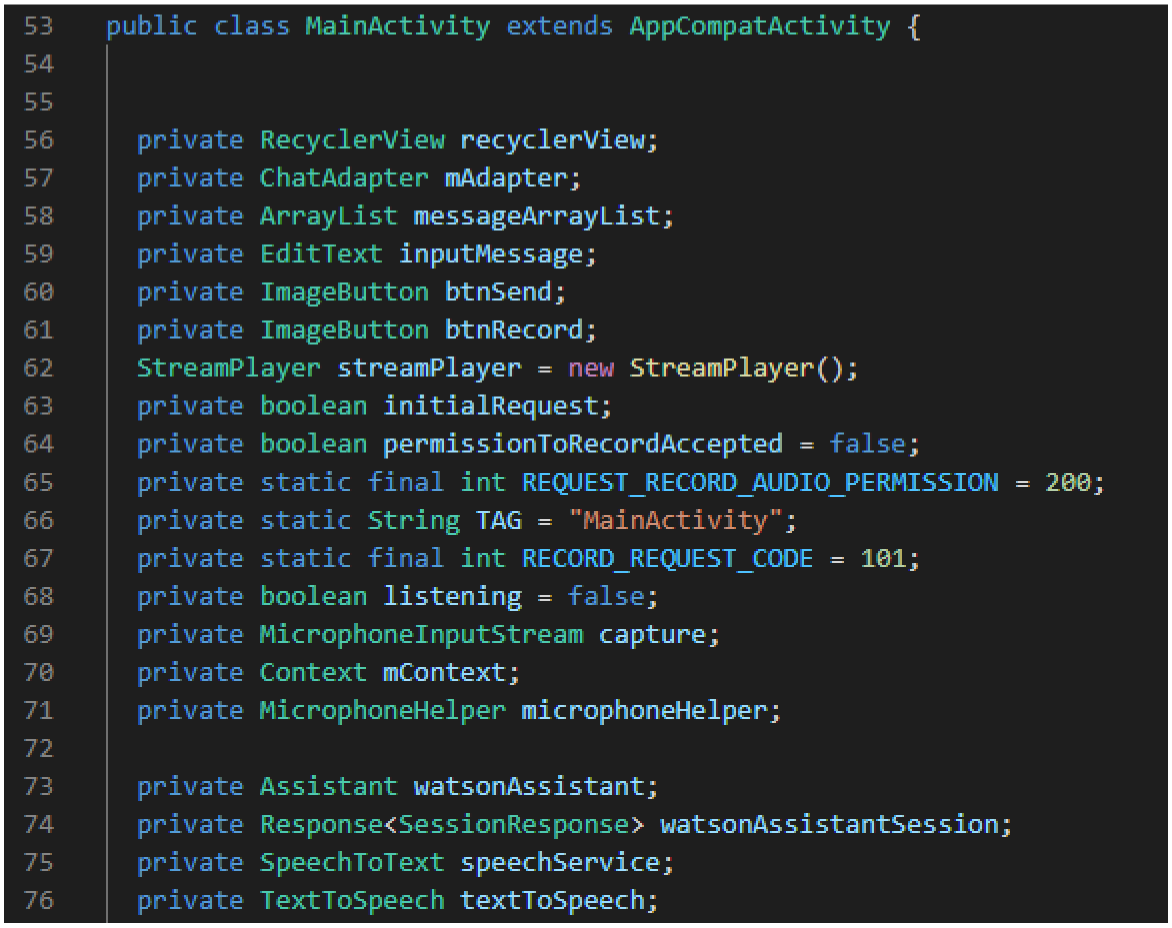
Design Error 2: Too Many Fields

The class is cluttered with too many attributes. Migrating related attributes can fix the issue.



```
> StdCyclomaticComplexity (3 violations)
> TooManyMethods (1 violation)
v TooManyFields (1 violation)
    ⚠ (53, 8) MainActivity in com.example.vmac.WatBot
> SingularField (2 violations)
> ExcessiveMethodLength (1 violation)
```

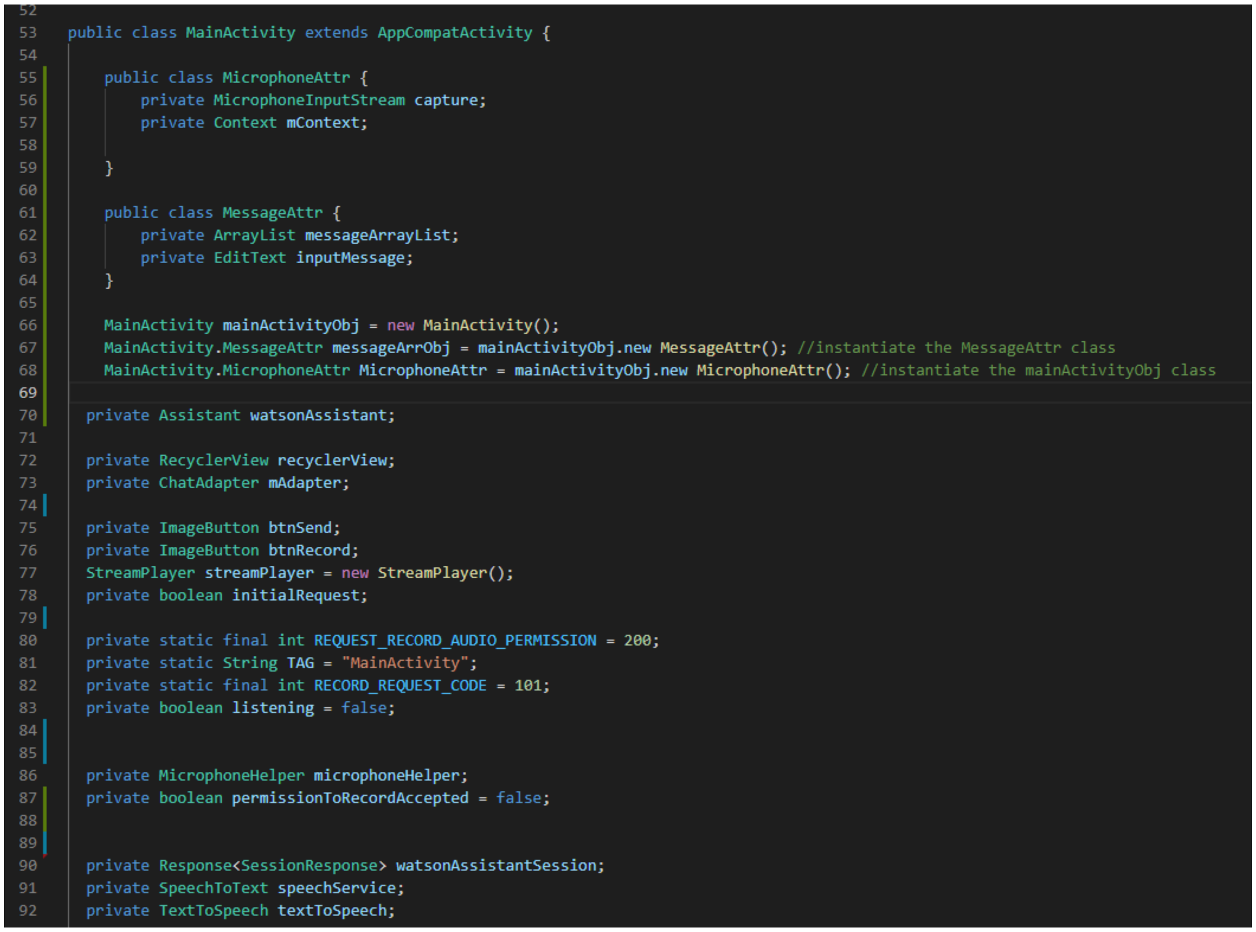
This is what the issue looks like:



```java
public class MainActivity extends AppCompatActivity {


    private RecyclerView recyclerView;
    private ChatAdapter mAdapter;
    private ArrayList messageArrayList;
    private EditText inputMessage;
    private ImageButton btnSend;
    private ImageButton btnRecord;
    StreamPlayer streamPlayer = new StreamPlayer();
    private boolean initialRequest;
    private boolean permissionToRecordAccepted = false;
    private static final int REQUEST_RECORD_AUDIO_PERMISSION = 200;
    private static String TAG = "MainActivity";
    private static final int RECORD_REQUEST_CODE = 101;
    private boolean listening = false;
    private MicrophoneInputStream capture;
    private Context mContext;
    private MicrophoneHelper microphoneHelper;

    private Assistant watsonAssistant;
    private Response<SessionResponse> watsonAssistantSession;
    private SpeechToText speechService;
    private TextToSpeech textToSpeech;
```

This is my attempt at fixing the issue:

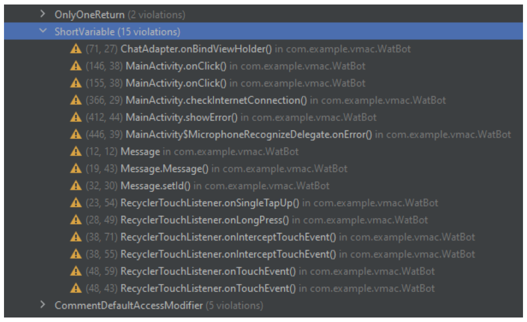
I put related attributes inside of their own classes so that they can be used as objects. Lines 66-68 are there for instantiation of the inner classes.



```
52
53  public class MainActivity extends AppCompatActivity {
54
55      public class MicrophoneAttr {
56          private MicrophoneInputStream capture;
57          private Context mContext;
58
59      }
60
61      public class MessageAttr {
62          private ArrayList messageArrayList;
63          private EditText inputMessage;
64      }
65
66      MainActivity mainActivityObj = new MainActivity();
67      MainActivity.MessageAttr messageArrObj = mainActivityObj.new MessageAttr(); //instantiate the MessageAttr class
68      MainActivity.MicrophoneAttr MicrophoneAttr = mainActivityObj.new MicrophoneAttr(); //instantiate the mainActivityObj class
69
70    private Assistant watsonAssistant;
71
72    private RecyclerView recyclerView;
73    private ChatAdapter mAdapter;
74
75    private ImageButton btnSend;
76    private ImageButton btnRecord;
77    StreamPlayer streamPlayer = new StreamPlayer();
78    private boolean initialRequest;
79
80    private static final int REQUEST_RECORD_AUDIO_PERMISSION = 200;
81    private static String TAG = "MainActivity";
82    private static final int RECORD_REQUEST_CODE = 101;
83    private boolean listening = false;
84
85
86    private MicrophoneHelper microphoneHelper;
87    private boolean permissionToRecordAccepted = false;
88
89
90    private Response<SessionResponse> watsonAssistantSession;
91    private SpeechToText speechService;
92    private TextToSpeech textToSpeech;
```

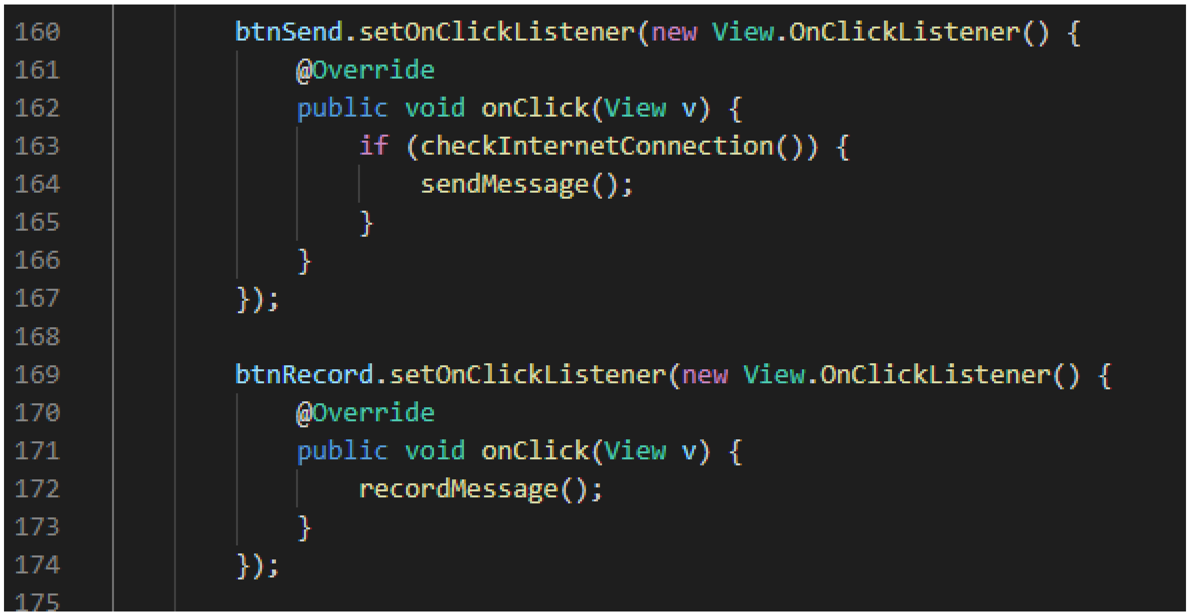
Code Style Error: Short Variables

Another issue where programmers are using short variables that are not descriptive enough to be useful when referring back to them later on.



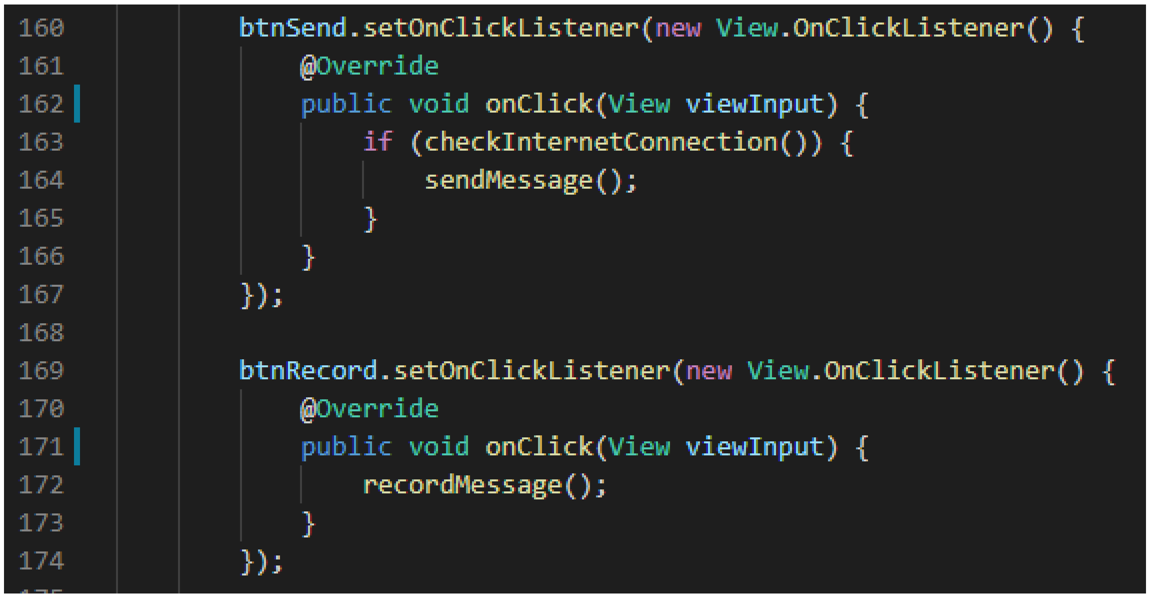
```
  > OnlyOneReturn (2 violations)
  v ShortVariable (15 violations)
      ⚠ (71, 27) ChatAdapter.onBindViewHolder() in com.example.vmac.WatBot
      ⚠ (146, 38) MainActivity.onClick() in com.example.vmac.WatBot
      ⚠ (155, 38) MainActivity.onClick() in com.example.vmac.WatBot
      ⚠ (366, 29) MainActivity.checkInternetConnection() in com.example.vmac.WatBot
      ⚠ (412, 44) MainActivity.showError() in com.example.vmac.WatBot
      ⚠ (446, 39) MainActivity$MicrophoneRecognizeDelegate.onError() in com.example.vmac.WatBot
      ⚠ (12, 12) Message in com.example.vmac.WatBot
      ⚠ (19, 43) Message.Message() in com.example.vmac.WatBot
      ⚠ (32, 30) Message.setId() in com.example.vmac.WatBot
      ⚠ (23, 54) RecyclerTouchListener.onSingleTapUp() in com.example.vmac.WatBot
      ⚠ (28, 49) RecyclerTouchListener.onLongPress() in com.example.vmac.WatBot
      ⚠ (38, 71) RecyclerTouchListener.onInterceptTouchEvent() in com.example.vmac.WatBot
      ⚠ (38, 55) RecyclerTouchListener.onInterceptTouchEvent() in com.example.vmac.WatBot
      ⚠ (48, 59) RecyclerTouchListener.onTouchEvent() in com.example.vmac.WatBot
      ⚠ (48, 43) RecyclerTouchListener.onTouchEvent() in com.example.vmac.WatBot
  > CommentDefaultAccessModifier (5 violations)
```

This is what the issue looks like inside the code:



```java
        btnSend.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (checkInternetConnection()) {
                    sendMessage();
                }
            }
        });

        btnRecord.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                recordMessage();
            }
        });
```

My suggested fix for the issue below:



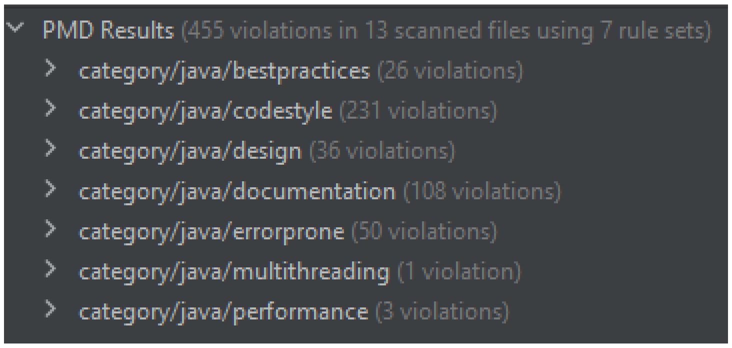
```java
        btnSend.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View viewInput) {
                if (checkInternetConnection()) {
                    sendMessage();
                }
            }
        });

        btnRecord.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View viewInput) {
                recordMessage();
            }
        });
```

GoogleAuth

Initial Requirements…

- 96.6% Java
- Updated in 2021
- 10 Contributors
- 272 Commits
- 890 Stars

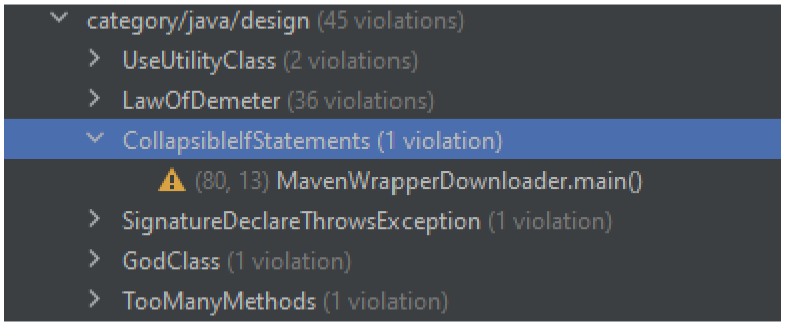
According to PMD, there are 455 code smell violations in this codebase.



```
PMD Results (455 violations in 13 scanned files using 7 rule sets)
  category/java/bestpractices (26 violations)
  category/java/codestyle (231 violations)
  category/java/design (36 violations)
  category/java/documentation (108 violations)
  category/java/errorprone (50 violations)
  category/java/multithreading (1 violation)
  category/java/performance (3 violations)
```

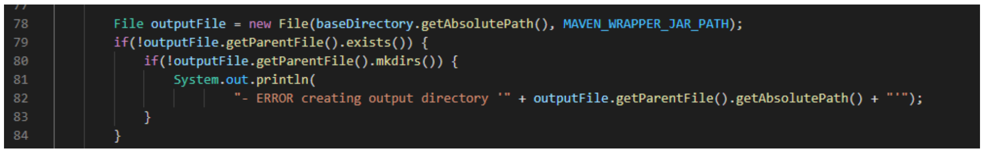
Design Error 1: Collapsible If Statements

The same issue shows here where an if statement is nested but can be collapsed by separating the conditions with a boolean short-circuit operator.



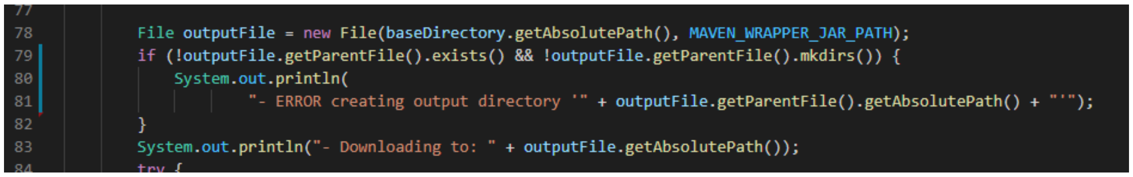
```
category/java/design (45 violations)
  UseUtilityClass (2 violations)
  LawOfDemeter (36 violations)
  CollapsibleIfStatements (1 violation)
    (80, 13) MavenWrapperDownloader.main()
  SignatureDeclareThrowsException (1 violation)
  GodClass (1 violation)
  TooManyMethods (1 violation)
```

This is the statement in question:



```
78        File outputFile = new File(baseDirectory.getAbsolutePath(), MAVEN_WRAPPER_JAR_PATH);
79        if(!outputFile.getParentFile().exists()) {
80            if(!outputFile.getParentFile().mkdirs()) {
81                System.out.println(
82                        "- ERROR creating output directory '" + outputFile.getParentFile().getAbsolutePath() + "'");
83            }
84        }
```

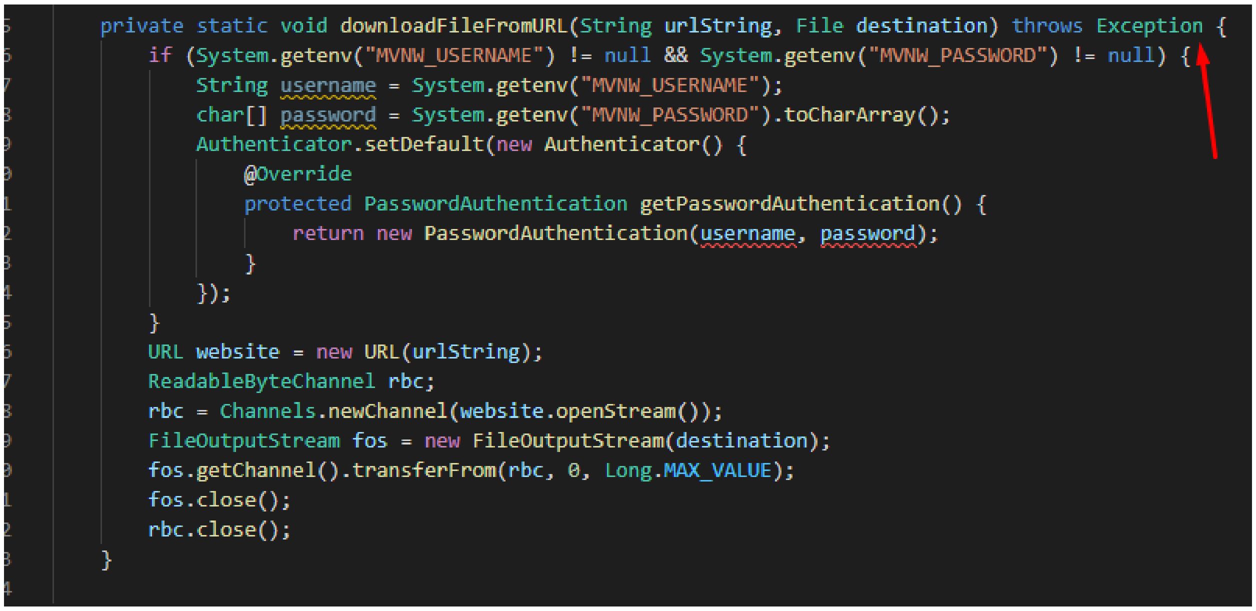
This is my proposed fix by combining the conditions into one if statement:



```
78        File outputFile = new File(baseDirectory.getAbsolutePath(), MAVEN_WRAPPER_JAR_PATH);
79        if (!outputFile.getParentFile().exists() && !outputFile.getParentFile().mkdirs()) {
80            System.out.println(
81                    "- ERROR creating output directory '" + outputFile.getParentFile().getAbsolutePath() + "'");
82        }
83        System.out.println("- Downloading to: " + outputFile.getAbsolutePath());
84        try {
```

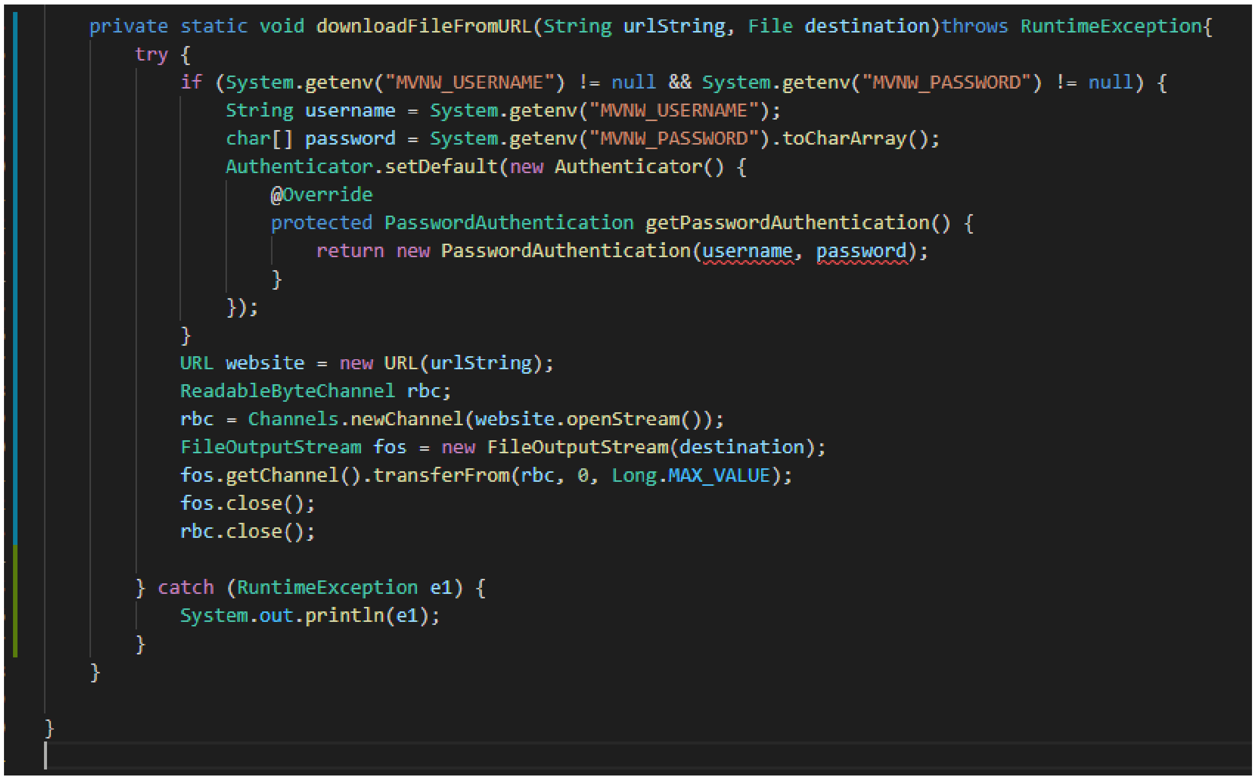
Design Error 2: Signature Declare Throws Exception

A method/constructor should not explicitly throw the generic Exception since it is unclear which exceptions can be thrown from the method.



```
private static void downloadFileFromURL(String urlString, File destination) throws Exception {
    if (System.getenv("MVNW_USERNAME") != null && System.getenv("MVNW_PASSWORD") != null) {
        String username = System.getenv("MVNW_USERNAME");
        char[] password = System.getenv("MVNW_PASSWORD").toCharArray();
        Authenticator.setDefault(new Authenticator() {
            @Override
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(username, password);
            }
        });
    }
    URL website = new URL(urlString);
    ReadableByteChannel rbc;
    rbc = Channels.newChannel(website.openStream());
    FileOutputStream fos = new FileOutputStream(destination);
    fos.getChannel().transferFrom(rbc, 0, Long.MAX_VALUE);
    fos.close();
    rbc.close();
}
```

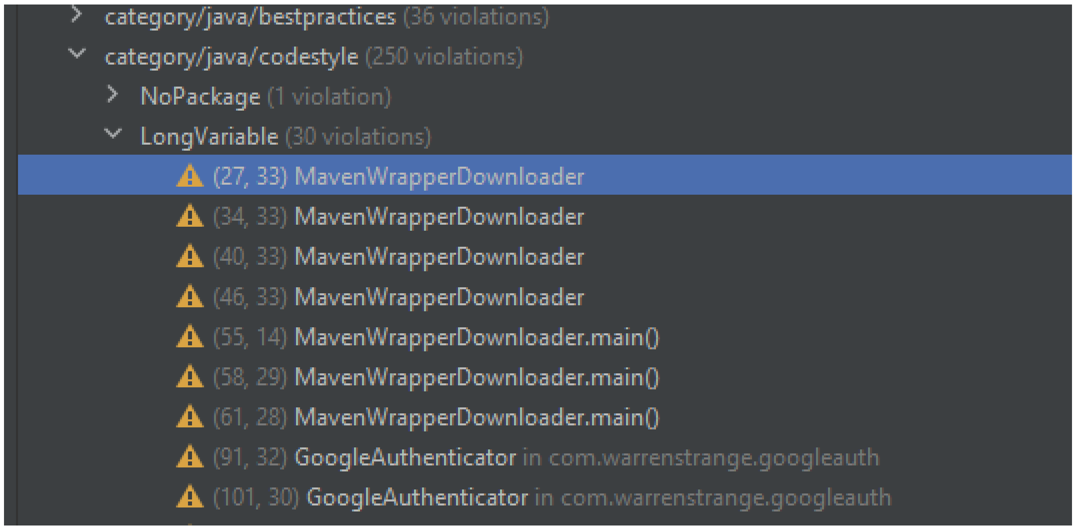
I will fix this by putting the code inside of a try-catch and throwing a RuntimeException instead. This approach makes it much easier to document which exceptions are being caught through this method and makes debugging easier.



```
private static void downloadFileFromURL(String urlString, File destination)throws RuntimeException{
    try {
        if (System.getenv("MVNW_USERNAME") != null && System.getenv("MVNW_PASSWORD") != null) {
            String username = System.getenv("MVNW_USERNAME");
            char[] password = System.getenv("MVNW_PASSWORD").toCharArray();
            Authenticator.setDefault(new Authenticator() {
                @Override
                protected PasswordAuthentication getPasswordAuthentication() {
                    return new PasswordAuthentication(username, password);
                }
            });
        }
        URL website = new URL(urlString);
        ReadableByteChannel rbc;
        rbc = Channels.newChannel(website.openStream());
        FileOutputStream fos = new FileOutputStream(destination);
        fos.getChannel().transferFrom(rbc, 0, Long.MAX_VALUE);
        fos.close();
        rbc.close();

    } catch (RuntimeException e1) {
        System.out.println(e1);
    }
}


}
```

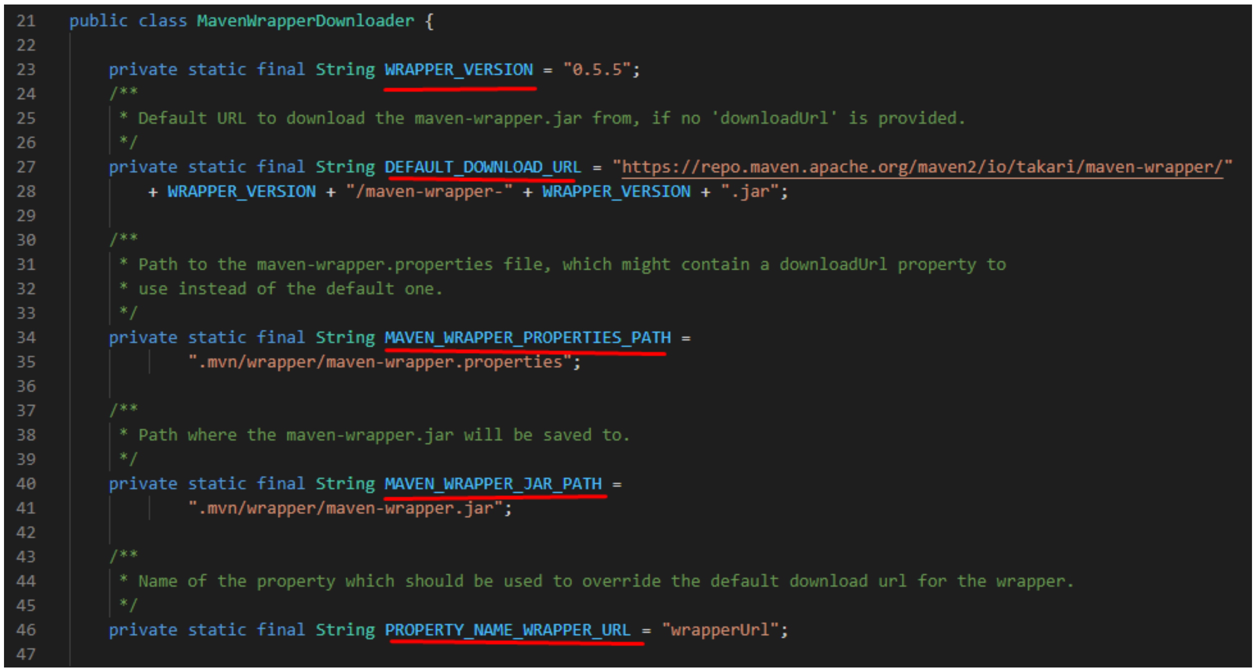
Code Style Error: Long Variable Names

This issue, unlike the short variable name code style error is that sometimes variables can be *too* descriptive and thus clutter the screen with unnecessary information.



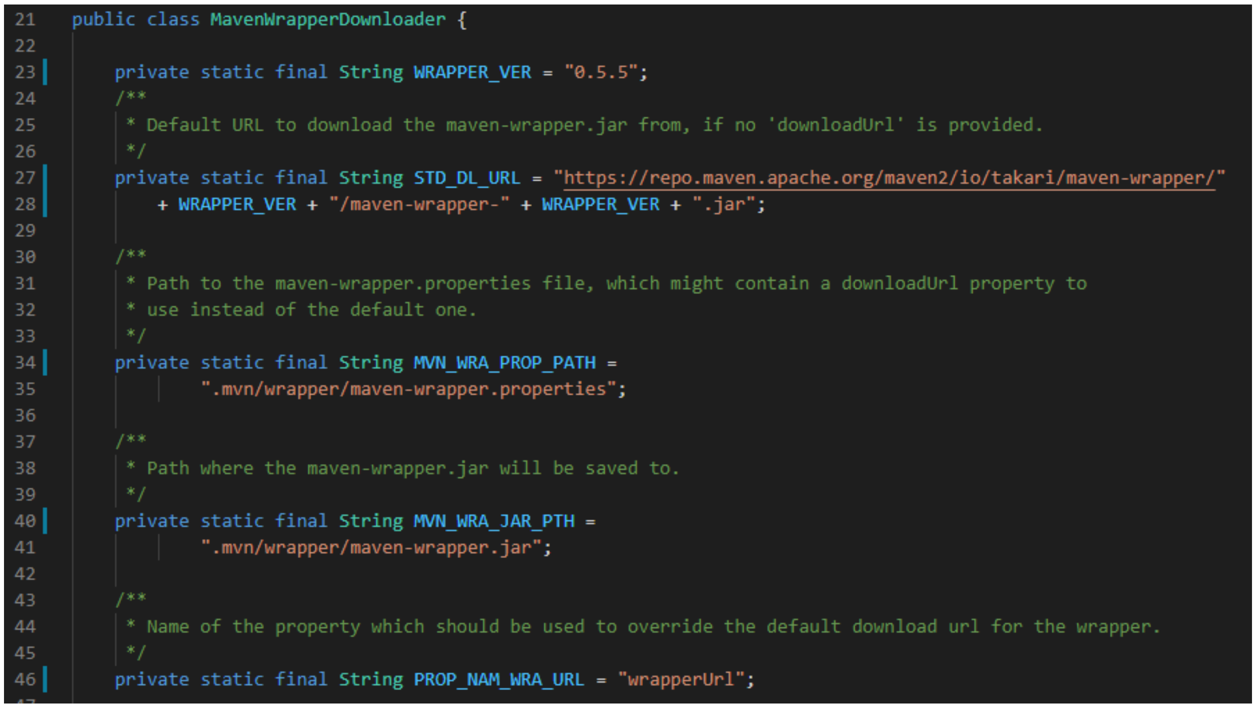
```
> category/java/bestpractices (36 violations)
v category/java/codestyle (250 violations)
    > NoPackage (1 violation)
    v LongVariable (30 violations)
        (27, 33) MavenWrapperDownloader
        (34, 33) MavenWrapperDownloader
        (40, 33) MavenWrapperDownloader
        (46, 33) MavenWrapperDownloader
        (55, 14) MavenWrapperDownloader.main()
        (58, 29) MavenWrapperDownloader.main()
        (61, 28) MavenWrapperDownloader.main()
        (91, 32) GoogleAuthenticator in com.warrenstrange.googleauth
        (101, 30) GoogleAuthenticator in com.warrenstrange.googleauth
```

This is what PMD highlighted as being potentially problematic in the future.



```java
public class MavenWrapperDownloader {

    private static final String WRAPPER_VERSION = "0.5.5";
    /**
     * Default URL to download the maven-wrapper.jar from, if no 'downloadUrl' is provided.
     */
    private static final String DEFAULT_DOWNLOAD_URL = "https://repo.maven.apache.org/maven2/io/takari/maven-wrapper/"
        + WRAPPER_VERSION + "/maven-wrapper-" + WRAPPER_VERSION + ".jar";

    /**
     * Path to the maven-wrapper.properties file, which might contain a downloadUrl property to
     * use instead of the default one.
     */
    private static final String MAVEN_WRAPPER_PROPERTIES_PATH =
            ".mvn/wrapper/maven-wrapper.properties";

    /**
     * Path where the maven-wrapper.jar will be saved to.
     */
    private static final String MAVEN_WRAPPER_JAR_PATH =
            ".mvn/wrapper/maven-wrapper.jar";

    /**
     * Name of the property which should be used to override the default download url for the wrapper.
     */
    private static final String PROPERTY_NAME_WRAPPER_URL = "wrapperUrl";

```

My fix for this (while maintaining their naming convention) is as follows:



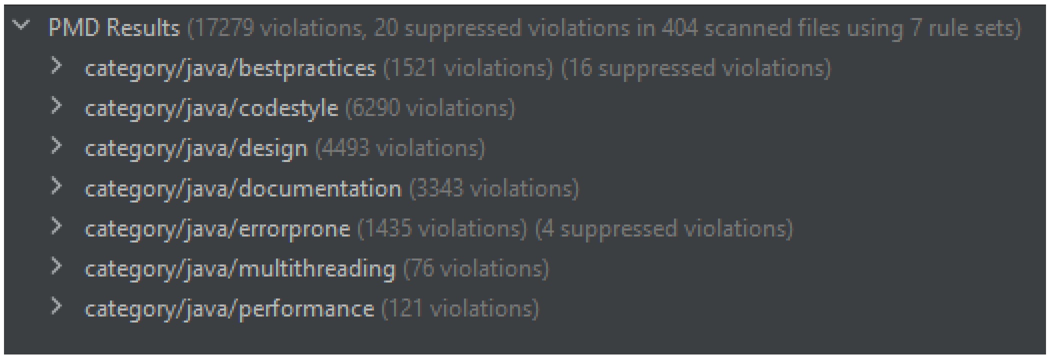
```java
public class MavenWrapperDownloader {

    private static final String WRAPPER_VER = "0.5.5";
    /**
     * Default URL to download the maven-wrapper.jar from, if no 'downloadUrl' is provided.
     */
    private static final String STD_DL_URL = "https://repo.maven.apache.org/maven2/io/takari/maven-wrapper/"
        + WRAPPER_VER + "/maven-wrapper-" + WRAPPER_VER + ".jar";

    /**
     * Path to the maven-wrapper.properties file, which might contain a downloadUrl property to
     * use instead of the default one.
     */
    private static final String MVN_WRA_PROP_PATH =
            ".mvn/wrapper/maven-wrapper.properties";

    /**
     * Path where the maven-wrapper.jar will be saved to.
     */
    private static final String MVN_WRA_JAR_PTH =
            ".mvn/wrapper/maven-wrapper.jar";

    /**
     * Name of the property which should be used to override the default download url for the wrapper.
     */
    private static final String PROP_NAM_WRA_URL = "wrapperUrl";
```

Spring Cloud Function

Initial Requirements…

- 98.5% Java
- Updated in 2021
- 68 Contributors
- 1398 Commits
- 772 Stars

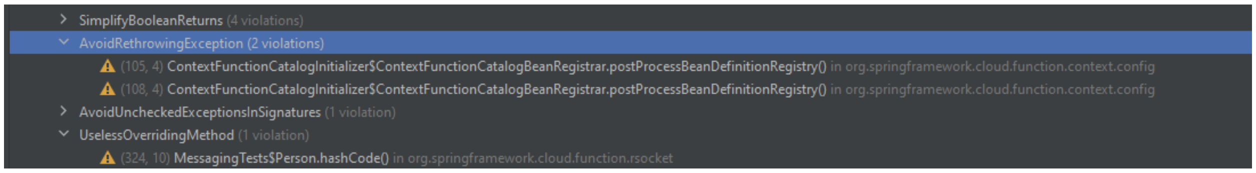
According to PMD, there are <u>17,279</u> code smell violations in this codebase.



```
PMD Results (17279 violations, 20 suppressed violations in 404 scanned files using 7 rule sets)
    category/java/bestpractices (1521 violations) (16 suppressed violations)
    category/java/codestyle (6290 violations)
    category/java/design (4493 violations)
    category/java/documentation (3343 violations)
    category/java/errorprone (1435 violations) (4 suppressed violations)
    category/java/multithreading (76 violations)
    category/java/performance (121 violations)
```

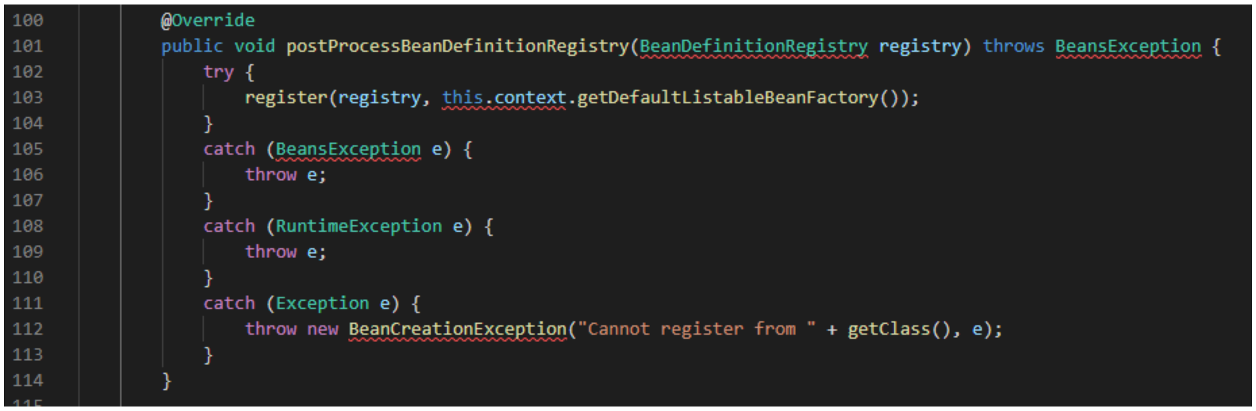
Design Error 1: Avoid Rethrowing Exceptions

Catch blocks that rethrow a caught exception only add to code size and runtime complexity



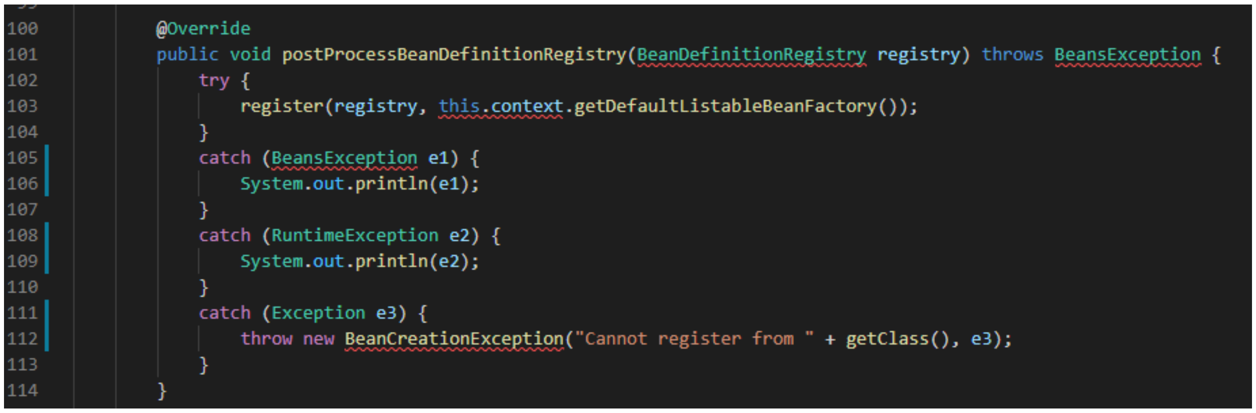
```
    SimplifyBooleanReturns (4 violations)
    AvoidRethrowingException (2 violations)
        (105, 4) ContextFunctionCatalogInitializer$ContextFunctionCatalogBeanRegistrar.postProcessBeanDefinitionRegistry() in org.springframework.cloud.function.context.config
        (108, 4) ContextFunctionCatalogInitializer$ContextFunctionCatalogBeanRegistrar.postProcessBeanDefinitionRegistry() in org.springframework.cloud.function.context.config
    AvoidUncheckedExceptionsInSignatures (1 violation)
    UselessOverridingMethod (1 violation)
        (324, 10) MessagingTests$Person.hashCode() in org.springframework.cloud.function.rsocket
```

This is what the PMD highlights as being a code smell violation:



```java
100         @Override
101         public void postProcessBeanDefinitionRegistry(BeanDefinitionRegistry registry) throws BeansException {
102             try {
103                 register(registry, this.context.getDefaultListableBeanFactory());
104             }
105             catch (BeansException e) {
106                 throw e;
107             }
108             catch (RuntimeException e) {
109                 throw e;
110             }
111             catch (Exception e) {
112                 throw new BeanCreationException("Cannot register from " + getClass(), e);
113             }
114         }
```

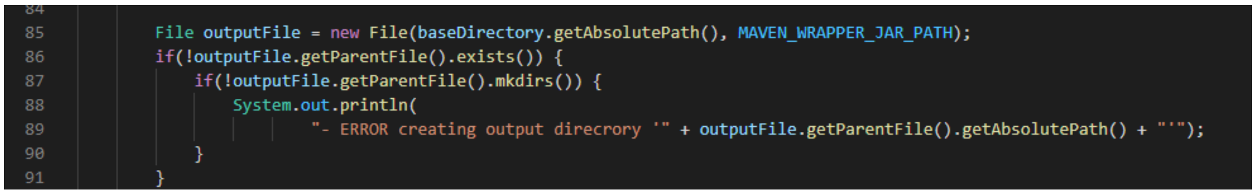
My attempted fix is to do anything else than throw the exception since throwing it is wasting it and we want to know when an exception is caught otherwise there would not be a catch in the first place.



```
    @Override
    public void postProcessBeanDefinitionRegistry(BeanDefinitionRegistry registry) throws BeansException {
        try {
            register(registry, this.context.getDefaultListableBeanFactory());
        }
        catch (BeansException e1) {
            System.out.println(e1);
        }
        catch (RuntimeException e2) {
            System.out.println(e2);
        }
        catch (Exception e3) {
            throw new BeanCreationException("Cannot register from " + getClass(), e3);
        }
    }
```

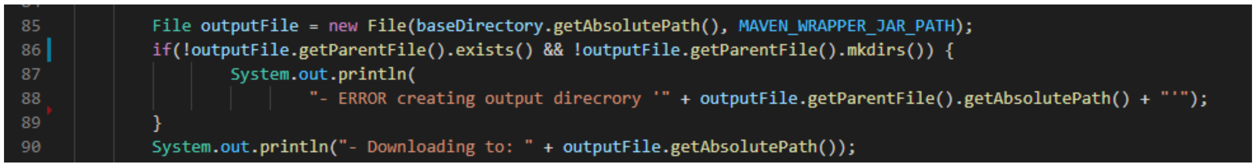
Design Error 2: Collapsible If Statements

Another instance of if statements that can be consolidated



```
        File outputFile = new File(baseDirectory.getAbsolutePath(), MAVEN_WRAPPER_JAR_PATH);
        if(!outputFile.getParentFile().exists()) {
            if(!outputFile.getParentFile().mkdirs()) {
                System.out.println(
                        "- ERROR creating output direcrory '" + outputFile.getParentFile().getAbsolutePath() + "'");
            }
        }
```

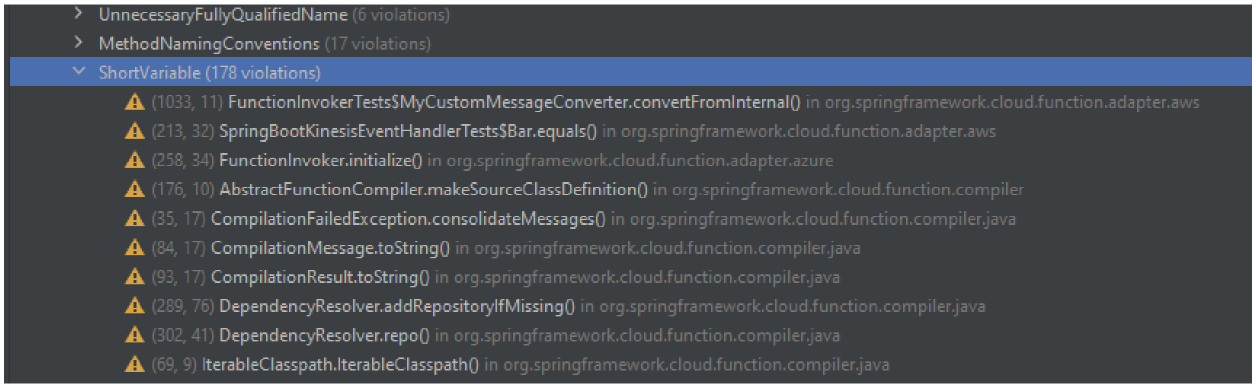
Here is my fix:



```
        File outputFile = new File(baseDirectory.getAbsolutePath(), MAVEN_WRAPPER_JAR_PATH);
        if(!outputFile.getParentFile().exists() && !outputFile.getParentFile().mkdirs()) {
                System.out.println(
                        "- ERROR creating output direcrory '" + outputFile.getParentFile().getAbsolutePath() + "'");
        }
        System.out.println("- Downloading to: " + outputFile.getAbsolutePath());
```

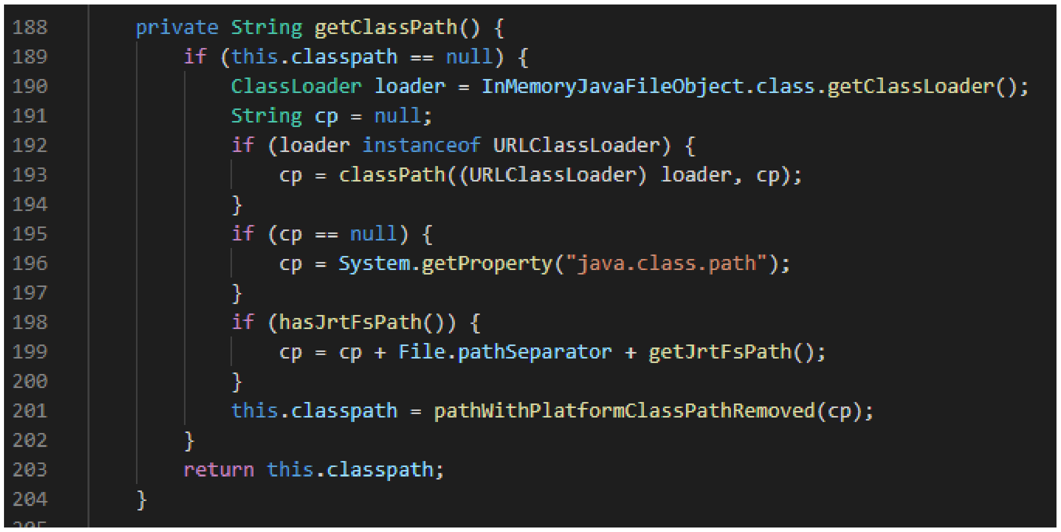
Code Style Error: Short Variable Names

Another code smell violation of naming variable names too short



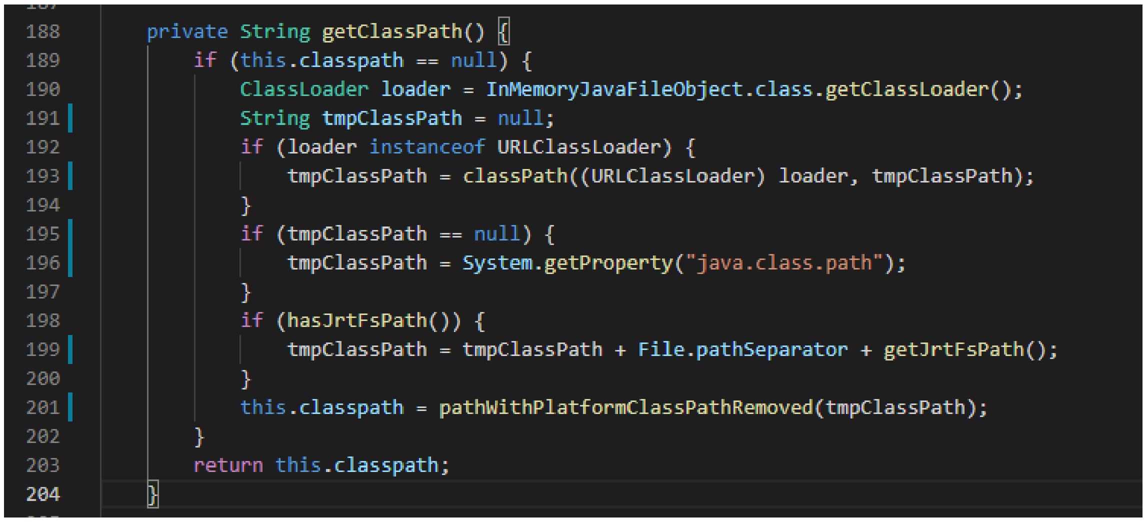
```
> UnnecessaryFullyQualifiedName (6 violations)
> MethodNamingConventions (17 violations)
v ShortVariable (178 violations)
    (1033, 11) FunctionInvokerTests$MyCustomMessageConverter.convertFromInternal() in org.springframework.cloud.function.adapter.aws
    (213, 32) SpringBootKinesisEventHandlerTests$Bar.equals() in org.springframework.cloud.function.adapter.aws
    (258, 34) FunctionInvoker.initialize() in org.springframework.cloud.function.adapter.azure
    (176, 10) AbstractFunctionCompiler.makeSourceClassDefinition() in org.springframework.cloud.function.compiler
    (35, 17) CompilationFailedException.consolidateMessages() in org.springframework.cloud.function.compiler.java
    (84, 17) CompilationMessage.toString() in org.springframework.cloud.function.compiler.java
    (93, 17) CompilationResult.toString() in org.springframework.cloud.function.compiler.java
    (289, 76) DependencyResolver.addRepositoryIfMissing() in org.springframework.cloud.function.compiler.java
    (302, 41) DependencyResolver.repo() in org.springframework.cloud.function.compiler.java
    (69, 9) IterableClasspath.IterableClasspath() in org.springframework.cloud.function.compiler.java
```

The short variable for classpath can be changed to something more suitable for its use



```java
    private String getClassPath() {
        if (this.classpath == null) {
            ClassLoader loader = InMemoryJavaFileObject.class.getClassLoader();
            String cp = null;
            if (loader instanceof URLClassLoader) {
                cp = classPath((URLClassLoader) loader, cp);
            }
            if (cp == null) {
                cp = System.getProperty("java.class.path");
            }
            if (hasJrtFsPath()) {
                cp = cp + File.pathSeparator + getJrtFsPath();
            }
            this.classpath = pathWithPlatformClassPathRemoved(cp);
        }
        return this.classpath;
    }
```

This is my proposed fix:



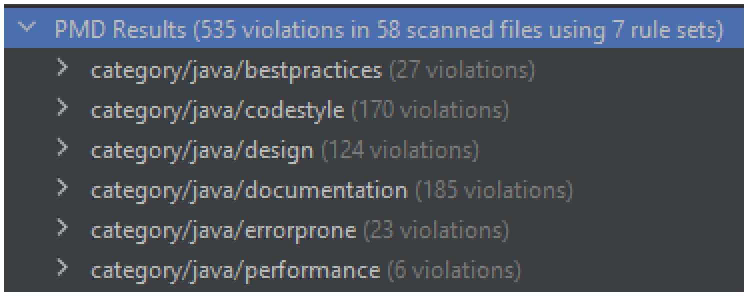
```java
    private String getClassPath() {
        if (this.classpath == null) {
            ClassLoader loader = InMemoryJavaFileObject.class.getClassLoader();
            String tmpClassPath = null;
            if (loader instanceof URLClassLoader) {
                tmpClassPath = classPath((URLClassLoader) loader, tmpClassPath);
            }
            if (tmpClassPath == null) {
                tmpClassPath = System.getProperty("java.class.path");
            }
            if (hasJrtFsPath()) {
                tmpClassPath = tmpClassPath + File.pathSeparator + getJrtFsPath();
            }
            this.classpath = pathWithPlatformClassPathRemoved(tmpClassPath);
        }
        return this.classpath;
    }
```

Spring Pet Clinic Microservices

Initial Requirements…

- 67.4% Java
- Updated in 2021
- 31 Contributors
- 703 Commits
- 980 Stars

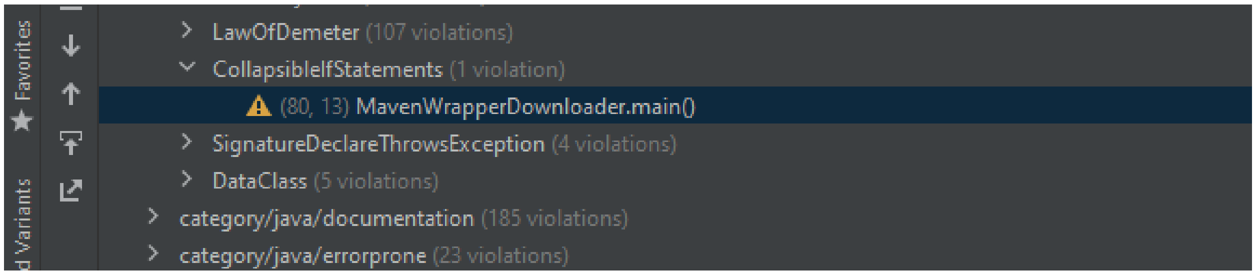
According to PMD, there are <u>535</u> code smell violations in this codebase.



```
PMD Results (535 violations in 58 scanned files using 7 rule sets)
  category/java/bestpractices (27 violations)
  category/java/codestyle (170 violations)
  category/java/design (124 violations)
  category/java/documentation (185 violations)
  category/java/errorprone (23 violations)
  category/java/performance (6 violations)
```

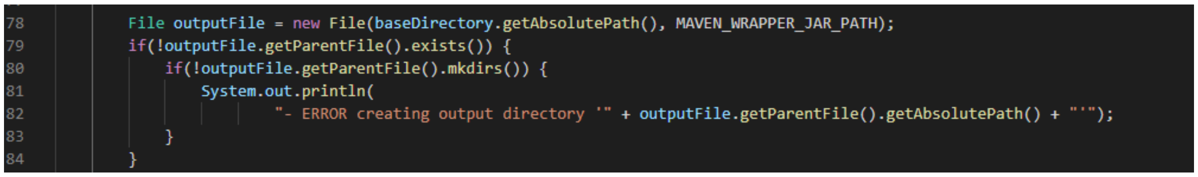
Design Error 1: Collapsible If Statements

Another instance of a code smell violation that uses too many if statements than necessary



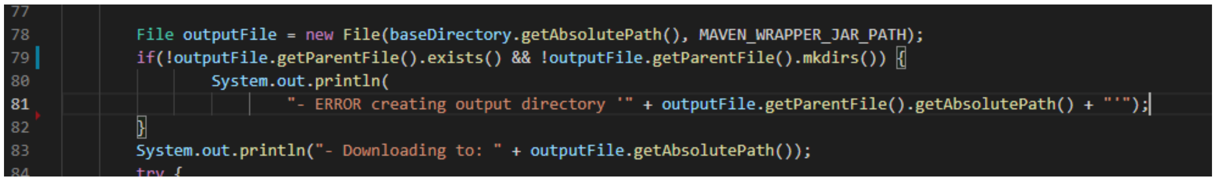
```
    LawOfDemeter (107 violations)
    CollapsibleIfStatements (1 violation)
      (80, 13) MavenWrapperDownloader.main()
    SignatureDeclareThrowsException (4 violations)
    DataClass (5 violations)
  category/java/documentation (185 violations)
  category/java/errorprone (23 violations)
```

The if statement that violates the code smell conventions:



```
        File outputFile = new File(baseDirectory.getAbsolutePath(), MAVEN_WRAPPER_JAR_PATH);
        if(!outputFile.getParentFile().exists()) {
            if(!outputFile.getParentFile().mkdirs()) {
                System.out.println(
                        "- ERROR creating output directory '" + outputFile.getParentFile().getAbsolutePath() + "'");
            }
        }
```

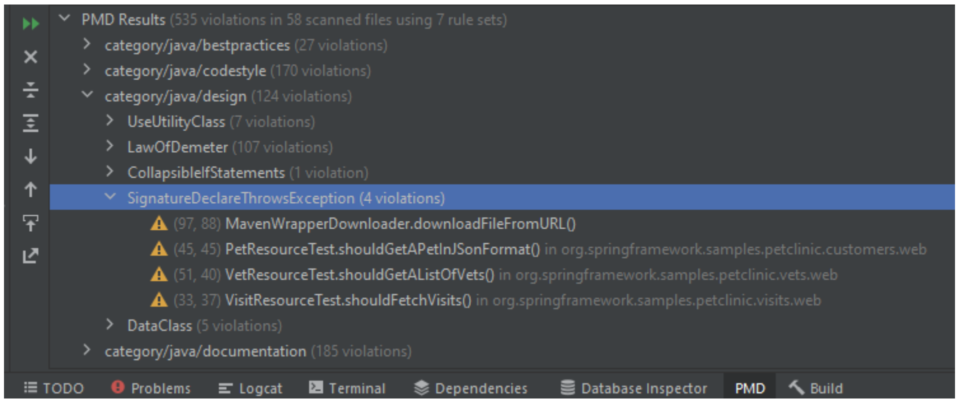
My fix to the violation:



```
        File outputFile = new File(baseDirectory.getAbsolutePath(), MAVEN_WRAPPER_JAR_PATH);
        if(!outputFile.getParentFile().exists() && !outputFile.getParentFile().mkdirs()) {
                System.out.println(
                        "- ERROR creating output directory '" + outputFile.getParentFile().getAbsolutePath() + "'");
        }
        System.out.println("- Downloading to: " + outputFile.getAbsolutePath());
        try {
```

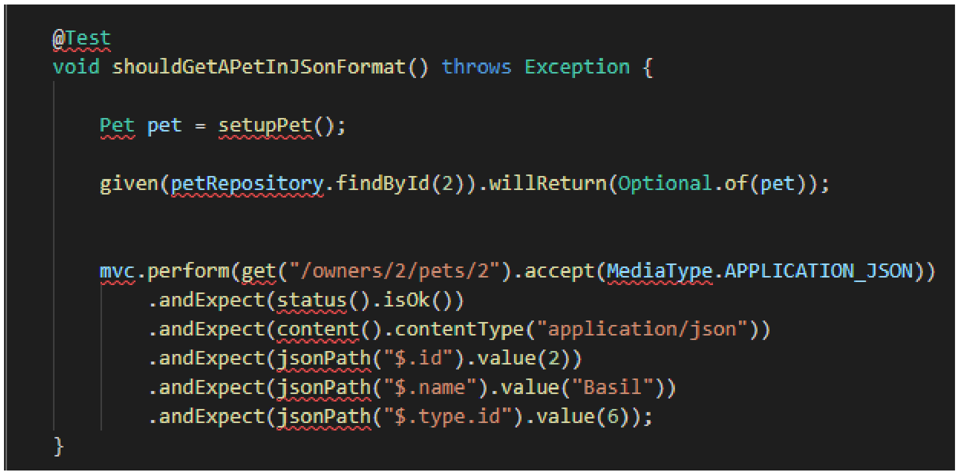
Design Error 2: Signature Declare Throws Exception

An instance where the method itself throws a generic Exception. For the sake of testing/debugging capability, I will put this in a try-catch



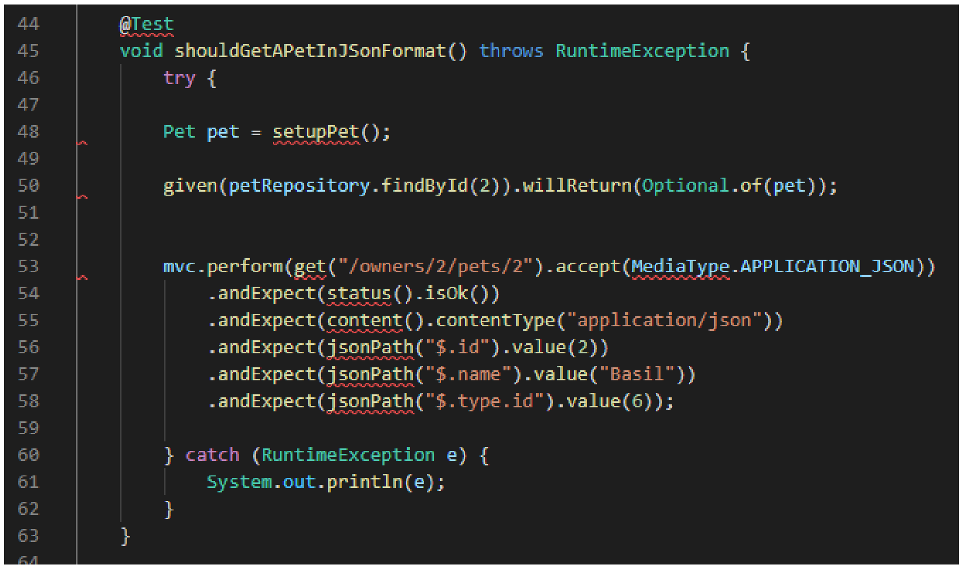
```
PMD Results (335 violations in 58 scanned files using 7 rule sets)
  category/java/bestpractices (27 violations)
  category/java/codestyle (170 violations)
  category/java/design (124 violations)
    UseUtilityClass (7 violations)
    LawOfDemeter (107 violations)
    CollapsibleIfStatements (1 violation)
    SignatureDeclareThrowsException (4 violations)
        (97, 88) MavenWrapperDownloader.downloadFileFromURL()
        (45, 45) PetResourceTest.shouldGetAPetInJSonFormat() in org.springframework.samples.petclinic.customers.web
        (51, 40) VetResourceTest.shouldGetAListOfVets() in org.springframework.samples.petclinic.vets.web
        (33, 37) VisitResourceTest.shouldFetchVisits() in org.springframework.samples.petclinic.visits.web
    DataClass (5 violations)
  category/java/documentation (185 violations)
```

The method that has a code smell violation:



```java
@Test
void shouldGetAPetInJSonFormat() throws Exception {

    Pet pet = setupPet();

    given(petRepository.findById(2)).willReturn(Optional.of(pet));


    mvc.perform(get("/owners/2/pets/2").accept(MediaType.APPLICATION_JSON))
        .andExpect(status().isOk())
        .andExpect(content().contentType("application/json"))
        .andExpect(jsonPath("$.id").value(2))
        .andExpect(jsonPath("$.name").value("Basil"))
        .andExpect(jsonPath("$.type.id").value(6));
}
```

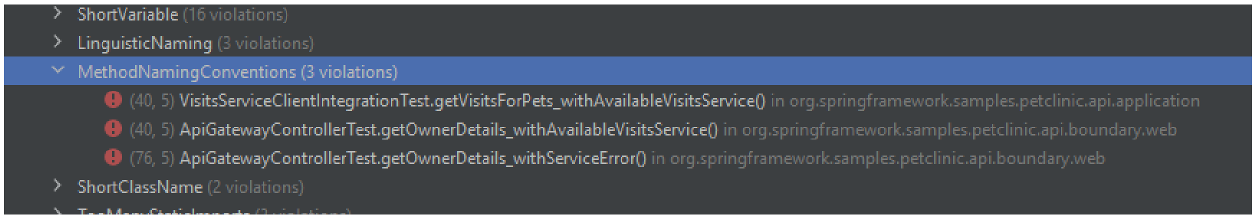
My proposed fix is provided below:



```java
    @Test
    void shouldGetAPetInJSonFormat() throws RuntimeException {
        try {

        Pet pet = setupPet();

        given(petRepository.findById(2)).willReturn(Optional.of(pet));


        mvc.perform(get("/owners/2/pets/2").accept(MediaType.APPLICATION_JSON))
            .andExpect(status().isOk())
            .andExpect(content().contentType("application/json"))
            .andExpect(jsonPath("$.id").value(2))
            .andExpect(jsonPath("$.name").value("Basil"))
            .andExpect(jsonPath("$.type.id").value(6));

        } catch (RuntimeException e) {
            System.out.println(e);
        }
    }
```

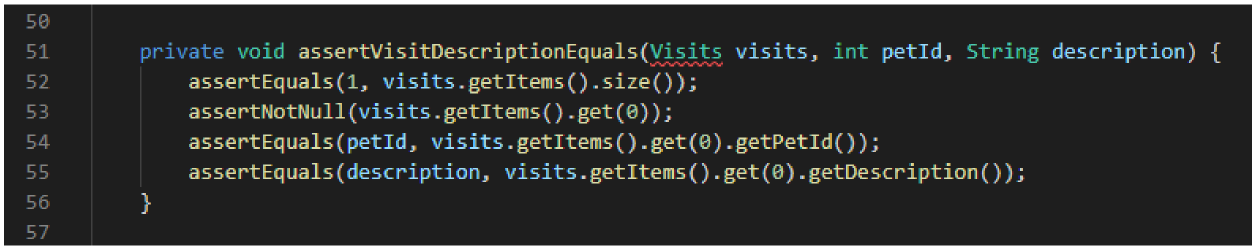
Code Style Error: Inconsistent Naming Convention

When there are variable names that do not have a uniform naming convention



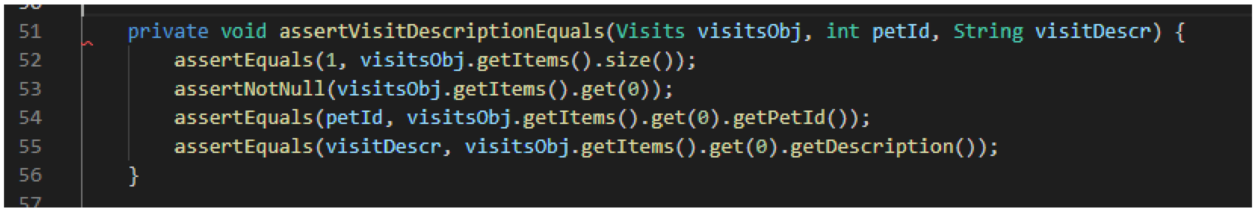
```
> ShortVariable (16 violations)
> LinguisticNaming (3 violations)
v MethodNamingConventions (3 violations)
    (40, 5) VisitsServiceClientIntegrationTest.getVisitsForPets_withAvailableVisitsService() in org.springframework.samples.petclinic.api.application
    (40, 5) ApiGatewayControllerTest.getOwnerDetails_withAvailableVisitsService() in org.springframework.samples.petclinic.api.boundary.web
    (76, 5) ApiGatewayControllerTest.getOwnerDetails_withServiceError() in org.springframework.samples.petclinic.api.boundary.web
> ShortClassName (2 violations)
```

PMD marked this as a potential issue since one variable is camelCase and the others are not. A simple fix is to make them all camelCase or non camelCase.



```java

    private void assertVisitDescriptionEquals(Visits visits, int petId, String description) {
        assertEquals(1, visits.getItems().size());
        assertNotNull(visits.getItems().get(0));
        assertEquals(petId, visits.getItems().get(0).getPetId());
        assertEquals(description, visits.getItems().get(0).getDescription());
    }

```

I will opt to change them all in to camelCase:



```java
    private void assertVisitDescriptionEquals(Visits visitsObj, int petId, String visitDescr) {
        assertEquals(1, visitsObj.getItems().size());
        assertNotNull(visitsObj.getItems().get(0));
        assertEquals(petId, visitsObj.getItems().get(0).getPetId());
        assertEquals(visitDescr, visitsObj.getItems().get(0).getDescription());
    }
```

Results

The detailed changes for my codebase analyzations can be found on my github repository:

https://github.com/derekaguirre/Codebase-Analysis

I have also provided the links to each fix for ease of navigation.

Google Cloud Dataflow Template Pipelines

Design Error 1: Excessive Parameter List

https://github.com/derekaguirre/Codebase-Analysis/commit/2657bd7422b00fe5b694e34aee2ef445a0e0e90b

Design Error 2: Collapsible If Statements

https://github.com/derekaguirre/Codebase-Analysis/commit/a90a6a981cefc88978a90f7a6617fb93ba3ebb64

Code Style Error: Short Variable Names

https://github.com/derekaguirre/Codebase-Analysis/commit/8a0937fd8a0a4d1299bbce227a14f540b0d7ab1c

Android Chatbot Powered by IBM Watson:

Design Error 1: Excessive Method Length

https://github.com/derekaguirre/Codebase-Analysis/commit/07d7447edc39d49cb8ccabb30d9c2f3f7bcb909c

Design Error 2: Too Many Fields

https://github.com/derekaguirre/Codebase-Analysis/commit/efa312a05f8ad57e5264d59d21c8944b32d9caff

Code Style Error: Short Variables

https://github.com/derekaguirre/Codebase-Analysis/commit/438dbbf9db4ebc0fb1bd5c1da6d54f07dbf90a23

GoogleAuth:

Design Error 1: Collapsible If Statements
https://github.com/derekaguirre/Codebase-Analysis/commit/9c30b99c1873d5565a0f04af63f31919be8f19f5

Design Error 2: Signature Declare Throws Exception
https://github.com/derekaguirre/Codebase-Analysis/commit/07c0ba9b431d64c18721c0bf3016fab038a5eb0d

Code Style Error: Long Variable Names
https://github.com/derekaguirre/Codebase-Analysis/commit/4e22a5237a615ef7eba1d4aed453c4482734a0f5

Spring Cloud Function

Design Error 1: Avoid Rethrowing Exceptions
https://github.com/derekaguirre/Codebase-Analysis/commit/c04e26e6f7fbab88152d957237e0cb04f4e31e30

Design Error 2: Collapsible If Statements
https://github.com/derekaguirre/Codebase-Analysis/commit/167fd2b1b39e51d105bd05813f577d4fa2a3ab6b

Code Style Error: Short Variable Names
https://github.com/derekaguirre/Codebase-Analysis/commit/7ad2a12d7bccc2074c506818a0f3e1b844c2c245

Spring Pet Clinic Microservices

Design Error 1: Collapsible If Statements
https://github.com/derekaguirre/Codebase-Analysis/commit/96f2e7ed7bf3883475a0c122a53e27520763b6d9

Design Error 2: Signature Declare Throws Exception
https://github.com/derekaguirre/Codebase-Analysis/commit/fa90312d0f9219b8f500a6a8afe83479f4b717bb

Code Style Error: Short Variable Names
https://github.com/derekaguirre/Codebase-Analysis/commit/612245664d8d6877b853614a153197ab8fbf2730

Analysis

This project has shown how detailed and serious code smell violations can be. It is important for businesses to check code for code smell violations before pushing any changes to their master branch. A majority of the code smell violations I analyzed were trivial problems but they still took a long time to finish due to the size and complexity of the codebases I was working with. It is important to hash out code smell violations before a codebase grows too large which may transform a trivial refactoring problem into a full codebase refactoring project.

Conclusion

Overall, the assignment was a good introduction into analyzing various codebases and gaining experience with refactoring other people's code. The main takeaway from this project is that a part of a codebase may have a code smell violation that starts off small, and by the time the issue needs to be addressed, the technical debt has grown in complexity exponentially. PMD is an important tool to learn how to use because it can be useful for analyzing even personal projects which will help highlight any violations. The more I became acquainted with the violations that PMD highlighted, the easier it was to recognize the same patterns in my own programs.