Politecnico di Milano

Software engineering 2

Code inspection v1.0

Alessandro Caprarelli	874206
Roberta Iero	873513
Giorgio De Luca	875598

February 3, 2017

Contents

Chapter 1

Introduction

1.1 Purpose

The purpose of this document is to recap all the results deriving from the analysis of the code. This analysis is performed taking into account the main inspection techniques that check if the code is 'well-written' or not. The term 'well-written' means that the code has to be written following a certain set of rules. These are summarized in the following checklist:

- Naming Conventions
- Indention
- Braces
- File Organization
- Wrapping Lines
- Comments
- Java Source Files
- Package and Import Statements
- Class and Interface Declarations
- Initialization and Declarations
- Method Calls
- Arrays

- Object Comparison
- Output Format
- Computation, Comparisons and Assignments
- Exceptions
- Flow of Control
- Files

1.2 Scope

The main scope of this document is to give developers a list of mistakes to repair in order to make the code more robust and of quality. In this way if the developers write the code following the same conventions, it will be also more readable.

1.3 Definition, acronyms, abbrevations

1.3.1 Definition

1.3.2 Acronyms

- **CI:** Code inspection

1.3.3 Abbrevetations

1.4 Reference documents

- Code inspection assignment document

1.5 Document overview

This document is composed of five sections:

- **Introduction:** this section contains the description of the document, of its purpose and some general information.
- **Assigned classes:** this section contains the list of the classes that will be inspected in section 4.
- **Functional role:** this part describes what the classes, that are going to be inspected in section 4, do.
- **List of Issues:** in this last section are listed all the issues found during the inspection of the code of the previously described classes. In particular for each class is specified which kind of rule is violated and what would be the solution.

Chapter 2

Assigned classes

The assigned classes that we are going to describe and inspect are:

- `EbayStoreCategoryFacade.java`
- `ControllerViewArtifactInfo.java`

2.1 EbayStoreCategoryFacade.java

The class is declared as follows:

```
57  public class EbayStoreCategoryFacade {
58      ....
345 }
```

Listing 2.1: EbayStoreCategoryFacade declaration

This class resides in a package declared at the beginning of the file:

```
19  package org.apache.ofbiz.ebaystore;
```

Listing 2.2: Package declaration

This package is inside a module called `EbayStore` and its complete pathname is:

```
/apache-ofbiz-16.11.01/specialpurpose/ebaystore/src/main/java/org/
    ↪ apache/ofbiz/ebaystore/EbayStoreCategoryFacade.java
```

2.2 ControllerViewArtifactInfo.java

The class is declared as follows:

```
35 public class ControllerViewArtifactInfo extends ArtifactInfoBase
   ↪ {
36     ....
129 }
```

Listing 2.3: ControllerViewArtifactInfo declaration

This class resides in a package declared at the beginning of the file:

```
19 package org.apache.ofbiz.webtools.artifactinfo;
```

Listing 2.4: Package declaration

This package is inside a module called `WebTools` and its complete pathname is:

```
/apache-ofbiz-16.11.01/framework/webtools/src/main/java/org/apache/
   ↪ ofbiz/webtools/artifactinfo/ControllerViewArtifactInfo.java
```

Chapter 3

Functional roles

3.1 EbayStoreCategoryFacade.java

This class is implemented by means of the Facade pattern. As the name suggests this pattern is used to create an architectural Facade. In fact `EbayStoreCategoryFacade` has the scope of providing a simple interface for a larger and more complex portion of code

This class is used by the following class:

- `EbayEvents`
- `EbayStoreHelper`
- `EbayStoreOptions`

3.2 ControllerViewArtifactInfo.java

This class is a subclass of `ArtifactInfoBase` so it inherits all the field and methods of it. The inherited methods are:

```
@Override
public String getDisplayName() {...}

@Override
public String getDisplayType() {...}

@Override
```

```
public String getType() {...}

@Override
public String getUniqueId() {...}

@Override
public URL getLocationURL() throws MalformedURLException {...}

@Override
public boolean equals(Object obj) {...}
```

Listing 3.1: Inherited methods by ControllerViewArtifactInfo.java

`ControllerViewArtifactInfo` also has other methods, in addition to those inherited by `ArtifactInfoBase`. These methods are:

```
public ControllerViewArtifactInfo(URL controllerXmlUrl, String
    ↪ viewUri, ArtifactInfoFactory aif) throws GeneralException
    ↪ {...}

public URL getControllerXmlUrl() {...}

public String getViewUri() {...}

public Set<ControllerRequestArtifactInfo>
    ↪ getRequestsThatThisViewIsResponseTo() {...}

public ScreenWidgetArtifactInfo getScreenCalledByThisView() {...}
```

Listing 3.2: Other methods of ControllerViewArtifactInfo.java

An Artifact is a product of a software development process.

In this case `ControllerViewArtifactInfo` is a subclass of the Artifact class `ArtifactInfoBase`. As it's possible to notice from its methods listed above, the only actions that `ControllerViewArtifactInfo` makes available are those of finding information about a Controller View class type.

This class is used by the following classes:

- `ArtifactInfoFactory`
- `ControllerRequestArtifactInfo`
- `ScreenWidgetArtifactInfo`

Chapter 4

Issues related to ControllerViewArtifactInfo class

4.1 Naming conventions

4.1.1 Wrong naming conventions

The class attribute `module` is declared at line 36 as static final and therefore its name should be in uppercase.

```
36 public static final String module =
   ↪ ControllerViewArtifactInfo.class.getName();
```

Listing 4.1: Issue

```
36 public static final String MODULE =
   ↪ ControllerViewArtifactInfo.class.getName();
```

Listing 4.2: Possible solution

4.1.2 Meaningless names

The name of the variable `that`, declared at line 114, is meaningless.

```
114 ControllerViewArtifactInfo that = (ControllerViewArtifactInfo)
    ↪ obj;
```

Listing 4.3: Issue

```
114 ControllerViewArtifactInfo cvai = (ControllerViewArtifactInfo)
    ↪ obj;
```

Listing 4.4: Possible solution

4.2 File organization

4.2.1 Lines length more than 80 characters

The following lines exceed 80 characters but are still acceptable:

```
36 public static final String module =
    ↪ ControllerViewArtifactInfo.class.getName();
```

Listing 4.5: Line 36 acceptable violation of the rule

```
59 // populate screenCalledByThisView and reverse in
    ↪ aif.allViewInfosReferringToScreen
```

Listing 4.6: Line 59 acceptable violation of the rule

```
84 String location =
    ↪ UtilURL.getOfbizHomeRelativeLocation(this.controllerXmlUrl);
```

Listing 4.7: Line 84 acceptable violation of the rule

```
115 return UtilObject.equalsHelper(this.controllerXmlUrl,
    ↪ that.controllerXmlUrl) &&
```

Listing 4.8: Line 115 acceptable violation of the rule

```
122 public Set<ControllerRequestArtifactInfo>
    ↪ getRequestsThatThisViewIsResponseTo() {
```

Listing 4.9: Line 122 acceptable violation of the rule

The following lines exceed 80 characters and may be reformatted in a better way:

```
60 if ("screen".equals(this.viewInfoMap.type) ||
   ↪ "screenfop".equals(this.viewInfoMap.type) ||
61                  "screentext".equals(this.viewInfoMap.type) ||
                      ↪ "screenxml".equals(this.viewInfoMap.type))
                      ↪ {
```

Listing 4.10: Line 60 violation of the rule

```
60 if ("screen".equals(this.viewInfoMap.type) ||
61     "screenfop".equals(this.viewInfoMap.type) ||
62     "screentext".equals(this.viewInfoMap.type) ||
63     "screenxml".equals(this.viewInfoMap.type)) {
```

Listing 4.11: Line 60 possible solution

4.2.2 Lines length more than 120 characters

The following lines exceed 120 characters and must be reformatted in a better way:

```
45 public ControllerViewArtifactInfo(URL controllerXmlUrl, String
   ↪ viewUri, ArtifactInfoFactory aif) throws GeneralException {
```

Listing 4.12: Line 45 violation of the rule

```
45 public ControllerViewArtifactInfo(URL controllerXmlUrl,
46                                    String viewUri,
47                                    ArtifactInfoFactory aif)
48     throws GeneralException {
```

Listing 4.13: Line 45 possible solution

```
53 throw new GeneralException("Could not find Controller View [" +
   ↪ viewUri + "] at URL [" + controllerXmlUrl.toExternalForm()
   ↪ + "]");
```

Listing 4.14: Line 53 violation of the rule

```
53 throw new GeneralException("Could not find Controller View [" +
   ↪ viewUri + "]" +
54 " at URL [" + controllerXmlUrl.toExternalForm() + "]");
```

Listing 4.15: Line 53 possible solution

```
57 throw new GeneralException("Controller view with name [" +
   ↪ viewUri + "] is not defined in controller file [" +
   ↪ controllerXmlUrl + "].");
```

Listing 4.16: Line 57 violation of the rule

```
57 throw new GeneralException("Controller view with name [" +
   ↪ viewUri + "]" +
58 " is not defined in controller file [" + controllerXmlUrl +
   ↪ "].");
```

Listing 4.17: Line 57 possible solution

```
65 this.screenCalledByThisView =
   ↪ this.aif.getScreenWidgetArtifactInfo(fullScreenName.substring(poundIndex-
   ↪ fullScreenName.substring(0, poundIndex));
```

Listing 4.18: Line 65 violation of the rule

```
65 this.screenCalledByThisView =
   ↪ this.aif.getScreenWidgetArtifactInfo
66  (fullScreenName.substring(poundIndex+1),
      ↪ fullScreenName.substring(0, poundIndex));
```

Listing 4.19: Line 65 possible solution

```
68 UtilMisc.addToSortedSetInMap(this,
   ↪ aif.allViewInfosReferringToScreen,
   ↪ this.screenCalledByThisView.getUniqueId());
```

Listing 4.20: Line 68 violation of the rule

```
68 UtilMisc.addToSortedSetInMap(this,
69                              aif.allViewInfosReferringToScreen,
70                              this.screenCalledByThisView.getUniqueId());
```

Listing 4.21: Line 68 possible solution

4.2.3 Unuseful blank lines to remove

The following blank lines are not useful to separate declarations of variables:

```
40 protected URL controllerXmlUrl;
41 protected String viewUri;
42
43 protected ConfigXMLReader.ViewMap viewInfoMap;
44
45 protected ScreenWidgetArtifactInfo screenCalledByThisView = null;
```

Listing 4.22: Lines 40-45 violation of the rule

```
47 this.controllerXmlUrl = controllerXmlUrl;
48 this.viewUri = viewUri;
49
50 this.viewInfoMap = aif.getControllerViewMap(controllerXmlUrl,
   ↪ viewUri);
```

Listing 4.23: Lines 47-50 violation of the rule

4.2.4 Useful blank lines to add

The following lines need a blank line to separate sections of code:

```
18 */
19 package org.apache.ofbiz.webtools.artifactinfo;
```

Listing 4.24: Lines 18-19 violation of the rule

```
18 */
19
20 package org.apache.ofbiz.webtools.artifactinfo;
```

Listing 4.25: Lines 18-19 possible solution

4.3 Package and import statements

The following declaration could be improved modifying the import statement:

```
41 protected ConfigXMLReader.ViewMap viewInfoMap;
```

Listing 4.26: Line 41 issue

```
30 import org.apache.ofbiz.webapp.control.ConfigXMLReader.ViewMap;
31
32 /**
```

```
33  *
34  */
35 public class ControllerViewArtifactInfo extends ArtifactInfoBase
     ↪ {
36     public static final String module =
         ↪ ControllerViewArtifactInfo.class.getName();
37
38     protected URL controllerXmlUrl;
39     protected String viewUri;
40
41     protected ViewMap viewInfoMap;
```

Listing 4.27: Line 41 possible solution

4.4 Comments

4.4.1 Classes, interfaces, methods not described

The following classes and methods should be properly described using Javadoc comments:

```
32 /**
33  *
34  */
35 public class ControllerViewArtifactInfo extends ArtifactInfoBase
     ↪ {
```

Listing 4.28: ControllerViewArtifactInfo class description missing

```
44 public ControllerViewArtifactInfo(URL controllerXmlUrl, String
     ↪ viewUri, ArtifactInfoFactory aif) throws GeneralException {
```

Listing 4.29: Constructor description missing

4.5 Java source files

Javadoc descriptions are missing for all the classes and methods.

4.6 Class and interface declarations

4.6.1 Methods grouping by functionality

The following method is in the middle of a group of getter methods, grouped by a functionality:

```
113 @Override
114 public boolean equals(Object obj) {
115     if (obj instanceof ControllerViewArtifactInfo) {
116         ControllerViewArtifactInfo that =
              ↪ (ControllerViewArtifactInfo) obj;
117         return UtilObject.equalsHelper(this.controllerXmlUrl,
              ↪ that.controllerXmlUrl) &&
118             UtilObject.equalsHelper(this.viewUri, that.viewUri);
119     } else {
120         return false;
121     }
122 }
```

Listing 4.30: equals method in the middle of getters

4.7 Initialization and declarations

The following string should be declared as `privatestaticfinal` variable and used instead of the plain text.

```
113 if (location.endsWith("/WEB-INF/controller.xml")) {
```

Listing 4.31: Constant is missing

,

```
112 private static final WEB-INF-CONTROLLER =
      ↪ "/WEB-INF/controller.xml";
113 if (location.endsWith(WEB-INF-CONTROLLER)) {
```

Listing 4.32: Possible solution

4.8 Method calls

4.8.1 Hard-coded values

The following method should have as `endIndex` a parameter related to the string constant proposed in the listing 4.32 and not an hard-coded value.

```
88 location = location.substring(0, location.length() - 23);
```

Listing 4.33: substring invocation

```
86 private static final WEB-INF-CONTROLLER =
     ↪ "/WEB-INF/controller.xml";
87 if (location.endsWith("/WEB-INF/controller.xml")) {
88       location = location.substring(0, location.length() -
           ↪ WEB-INF-CONTROLLER.lenght);
89 }
```

Listing 4.34: substring invocation possible solution

4.8.2 Method invocations on posssible null objects

The following methods are invoked on objects that may be null therefore is needed a check:

```
50 this.viewInfoMap = aif.getControllerViewMap(controllerXmlUrl,
     ↪ viewUri);
```

Listing 4.35: getControllerViewMap invocation

```
50 if(aif!=null){
51      this.viewInfoMap =
           ↪ aif.getControllerViewMap(controllerXmlUrl, viewUri);
52 }
```

Listing 4.36: getControllerViewMap invocation possible solution

---

```
87 if (location.endsWith("/WEB-INF/controller.xml")) {
```

Listing 4.37: getControllerViewMap invocation

```
87 if(location!=null){
88     if (location.endsWith("/WEB-INF/controller.xml")) {
89 }
```

Listing 4.38: getControllerViewMap invocation possible solution

4.9 Other errors

The following second `if` statement is useless. It will never be executed.

```
52 if (this.viewInfoMap == null) {
53     throw new GeneralException("Could not find Controller View
       ↪ [" + viewUri + "] at URL [" +
       ↪ controllerXmlUrl.toExternalForm() + "]");
54 }
55 if (this.viewInfoMap == null) {
56     throw new GeneralException("Controller view with name [" +
       ↪ viewUri + "] is not defined in controller file [" +
       ↪ controllerXmlUrl + "].");
57 }
```

Listing 4.39: Exeption would block the second if

Chapter 5

Issues related to EbayStoreCategoryFacade class

5.1 Naming conventions

5.1.1 Wrong naming conventions

The class attribute `module` is declared at line 58 as static final and therefore its name should be in uppercase.

```
58 public static final String module =
   ↪ EbayStoreCategoryFacade.class.getName();
```

Listing 5.1: Issue

```
58 public static final String Module =
   ↪ EbayStoreCategoryFacade.class.getName();
```

Listing 5.2: Possible solution

The method `AttributesEnabled()` at line 330, should start with a lower case character in order to match the camel cased pattern.

```
330 public boolean AttributesEnabled() {...}
```

Listing 5.3: Issue

```
36 public boolean attributesEnabled() {...}
```

Listing 5.4: Possible solution

5.2 Braces

5.2.1 Curly braces missing

The following `if` blocks should have curly braces to surround the only one statement.

```
105 if (recommendationsArray == null || recommendationsArray.length
    ↪ == 0)
106         return;
```

Listing 5.5: Issue

```
105 if (recommendationsArray == null || recommendationsArray.length
    ↪ == 0){
106     return;
107 }
```

Listing 5.6: Possible solution

```
271 if (theme != null) temGroup.put("TemplateGroupName",
    ↪ theme.getGroupName());
```

Listing 5.7: Issue

```
271 if (theme != null) {
272         temGroup.put("TemplateGroupName", theme.getGroupName());
273 }
```

Listing 5.8: Possible solution

```
274 if (theme != null) temGroup.put("TemplateGroupName", "_NA_");
```

Listing 5.9: Issue

```
274 if (theme != null) {
275         temGroup.put("TemplateGroupName", "_NA_");
276 }
```

Listing 5.10: Possible solution

5.3 File organization

5.3.1 Lines length more than 80 characters

The following lines exceed 80 characters but are still acceptable:

```
71 private StoreOwnerExtendedListingDurationsType
   ↪ storeOwnerExtendedListingDuration = null;
```

Listing 5.11: Line 71 acceptable violation of the rule

```
93 AttributeSet[] itemSpecAttrSets =
   ↪ attrMaster.getItemSpecificAttributeSetsForCategories(ids);
```

Listing 5.12: Line 93 acceptable violation of the rule

```
94 AttributeSet[] siteWideAttrSets =
   ↪ attrMaster.getSiteWideAttributeSetsForCategories(ids);
```

Listing 5.13: Line 94 acceptable violation of the rule

```
100 GetCategorySpecificsCall getCatSpe = new
    ↪ GetCategorySpecificsCall(apiContext);
```

Listing 5.14: Line 100 acceptable violation of the rule

```
102 DetailLevelCodeType[] detailLevels = new DetailLevelCodeType[]
    ↪ {DetailLevelCodeType.RETURN_ALL};
```

Listing 5.15: Line 102 acceptable violation of the rule

```
104 RecommendationsType[] recommendationsArray =
    ↪ getCatSpe.getCategorySpecifics();
```

Listing 5.16: Line 104 acceptable violation of the rule

```
115 SiteDefaultsType siteDefaults = this.siteFacade.
    ↪ getSiteFeatureDefaultMap().get(apiContext.getSite());
```

Listing 5.17: Line 115acceptable violation of the rule

```
130 ListingDurationDefinitionsType listDuration =
    ↪ featureDefinition.getListingDurations();
```

Listing 5.18: Line 130 acceptable violation of the rule

```
131 ListingDurationDefinitionType[] durationArray =
    ↪ listDuration.getListingDuration();
```

Listing 5.19: Line 131 acceptable violation of the rule

```
134 listingDurationMap.put(durationArray[i].getDurationSetID(),
    ↪ durationArray[i].getDuration());
```

Listing 5.20: Line 134 acceptable violation of the rule

```
144 listingDurationReferenceMap.put(listingDuration[i].getType().value(),
    ↪ listingDuration[i].getValue());
```

Listing 5.21: Line 144 acceptable violation of the rule

```
156 storeOwnerExtendedListingDuration =
    ↪ siteDefaults.getStoreOwnerExtendedListingDurations();
```

Listing 5.22: Line 156 acceptable violation of the rule

```
162 private static BuyerPaymentMethodCodeType[]
    ↪ fiterPaymentMethod(BuyerPaymentMethodCodeType[]
    ↪ paymentMethods) {
```

Listing 5.23: Line 162 acceptable violation of the rule

```
163 List<BuyerPaymentMethodCodeType> al = new
    ↪ ArrayList<BuyerPaymentMethodCodeType>();
```

Listing 5.24: Line 163 acceptable violation of the rule

```
203 // invoke the method specified by methodName and return the
    ↪ corresponding return value
```

Listing 5.25: Line 203 acceptable violation of the rule

```
208 private Object invokeMethodByName(CategoryFeatureType cf, String
    ↪ methodName) {...}
```

Listing 5.26: Line 208 acceptable violation of the rule

```
224 List<Map<String,Object>> temGroupList = new
    ↪ LinkedList<Map<String,Object>>();
```

Listing 5.27: Line 224 acceptable violation of the rule

```
226 GetDescriptionTemplatesCall call = new
    ↪ GetDescriptionTemplatesCall(this.apiContext);
```

Listing 5.28: Line 226 acceptable violation of the rule

```
231 DescriptionTemplateType[] descriptionTemplateTypeList =
    ↪ resp.getDescriptionTemplate();
```

Listing 5.29: Line 231 acceptable violation of the rule

```
232 Debug.logInfo("layout of category "+ this.catId +":"+
    ↪ resp.getLayoutTotal(), module);
```

Listing 5.30: Line 232 acceptable violation of the rule

```
233 for (DescriptionTemplateType descTemplateType :
    ↪ descriptionTemplateTypeList) {...}
```

Listing 5.31: Line 233 acceptable violation of the rule

```
236 if ("THEME".equals(String.valueOf(descTemplateType.getType())))
    ↪ {...}
```

Listing 5.32: Line 236 acceptable violation of the rule

```
238 template.put("TemplateId",
    ↪ String.valueOf(descTemplateType.getID()));
```

Listing 5.33: Line 238 acceptable violation of the rule

```
239 template.put("TemplateImageURL", descTemplateType.getImageURL());
```

Listing 5.34: Line 239 acceptable violation of the rule

```
245 if (temGroup.get("TemplateGroupId").
    ↪ equals(descTemplateType.getGroupID().toString())) {...}
```

Listing 5.35: Line 245 acceptable violation of the rule

```
253 templateGroup.put("TemplateGroupId",
    ↪ descTemplateType.getGroupID().toString());
```

Listing 5.36: Line 253 acceptable violation of the rule

```
259 templateList =
    ↪ UtilGenerics.checkList(templateGroup.get("Templates"));
```

Listing 5.37: Line 259 acceptable violation of the rule

```
263 else if
    ↪ ("Layout".equals(String.valueOf(descTemplateType.getType())))
    ↪ {...}
```

Listing 5.38: Line 263 acceptable violation of the rule

```
270 if (theme.getGroupID() ==
    ↪ Integer.parseInt(temGroup.get("TemplateGroupId").toString()))
    ↪ {...}
```

Listing 5.39: Line 270 acceptable violation of the rule

```
334 public StoreOwnerExtendedListingDurationsType
    ↪ getStoreOwnerExtendedListingDuration() {...}
```

Listing 5.40: Line 334 acceptable violation of the rule

The following lines exceed 80 characters and may be reformatted in a better way:

```
 99 private void syncNameRecommendationTypes() throws ApiException,
    ↪ SdkException, Exception {...}
```

Listing 5.41: Line 99 violation of the rule

```
 99 private void syncNameRecommendationTypes()
100         throws ApiException, SdkException, Exception {...}
```

Listing 5.42: Line 99 possible solution

```
181 private Object getInheritProperty(String catId,String methodName,
182             Map<String, CategoryType> categoriesCacheMap,
                ↪ Map<String, CategoryFeatureType> cfsMap)
                ↪ throws Exception {...}
```

Listing 5.43: Line 181 violation of the rule

```
181 private Object getInheritProperty(String catId,
182         String methodName,
183         Map<String, CategoryType> categoriesCacheMap,
184         Map<String, CategoryFeatureType> cfsMap)
185     throws Exception {
```

Listing 5.44: Line 181 possible solution

```
199 return getInheritProperty(cat.getCategoryParentID(0),
    ↪ methodName, categoriesCacheMap, cfsMap);
```

Listing 5.45: Line 199 violation of the rule

```
199 return getInheritProperty ( cat.getCategoryParentID(0),
200     methodName,
201     categoriesCacheMap,
202     cfsMap);
```

Listing 5.46: Line 199 possible solution

```
221 public List<Map<String,Object>> syncAdItemTemplates() throws
    ↪ ApiException, SdkSoapException, SdkException {...}
```

Listing 5.47: Line 221 violation of the rule

```
221 public List<Map<String,Object>> syncAdItemTemplates()
222         throws ApiException, SdkSoapException, SdkException {
```

Listing 5.48: Line 221 possible solution

5.3.2 Lines length more than 120 characters

The following lines exceed 120 characters and must be reformatted in a better way:

```
75 public EbayStoreCategoryFacade(String catId, ApiContext
   ↪ apiContext, IAttributesMaster attrMaster,
   ↪ EbayStoreSiteFacade siteFacade) throws SdkException,
   ↪ Exception {...}
```

Listing 5.49: Line 75 violation of the rule

```
75 public EbayStoreCategoryFacade(String catId,
76         ApiContext apiContext,
77         IAttributesMaster attrMaster,
78         EbayStoreSiteFacade siteFacade)
79     throws SdkException, Exception {
```

Listing 5.50: Line 75 possible solution

```
112 Map<String, CategoryType> categoriesCacheMap =
    ↪ this.siteFacade.getSiteCategoriesMap().get(apiContext.getSite());
```

Listing 5.51: Line 112 violation of the rule

```
95 AttributeSet[] joinedAttrSets =
    ↪ attrMaster.joinItemSpecificAndSiteWideAttributeSets(itemSpecAttrSets,
    ↪ siteWideAttrSets);
```

Listing 5.52: Line 95 violation of the rule

```
114 Map<String, CategoryFeatureType> cfsMap =
    ↪ this.siteFacade.getSiteCategoriesFeaturesMap().get(apiContext.getSite());
```

Listing 5.53: Line 114 violation of the rule

```
116 FeatureDefinitionsType featureDefinition =
    ↪ this.siteFacade.getSiteFeatureDefinitionsMap().get(apiContext.getSite());
```

Listing 5.54: Line 116 violation of the rule

```
119 itemSpecificEnabled =
    ↪ (ItemSpecificsEnabledCodeType)getInheritProperty(catId,
    ↪ "getItemSpecificsEnabled", categoriesCacheMap, cfsMap);
```

Listing 5.55: Line 119 violation of the rule

```
119 itemSpecificEnabled =
    ↪ (ItemSpecificsEnabledCodeType)getInheritProperty(catId,
120         "getItemSpecificsEnabled",
121         categoriesCacheMap,
122         cfsMap);
```

Listing 5.56: Line 119 possible solution

```
124 retPolicyEnabled = (Boolean)getInheritProperty(catId,
    ↪ "isReturnPolicyEnabled", categoriesCacheMap, cfsMap);
```

Listing 5.57: Line 124 violation of the rule

```
124 retPolicyEnabled = (Boolean)getInheritProperty(catId,
125         "isReturnPolicyEnabled",
126         categoriesCacheMap,
127         cfsMap);
```

Listing 5.58: Line 124 possible solution

```
138 ListingDurationReferenceType[] listingDuration =
    ↪ (ListingDurationReferenceType[])getInheritProperty(catId,
    ↪ "getListingDuration", categoriesCacheMap, cfsMap);
```

Listing 5.59: Line 138 violation of the rule

```
138 ListingDurationReferenceType[] listingDuration =
    ↪ (ListingDurationReferenceType[])getInheritProperty(catId,
139         "getListingDuration",
140         categoriesCacheMap,
141         cfsMap);
```

Listing 5.60: Line 138 possible solution

```
138 paymentMethods =
    ↪ (BuyerPaymentMethodCodeType[])getInheritProperty(catId,
    ↪ "getPaymentMethod", categoriesCacheMap, cfsMap);
```

Listing 5.61: Line 138 violation of the rule

```
138 paymentMethods =
    ↪ (BuyerPaymentMethodCodeType[])getInheritProperty(catId,
139         "getPaymentMethod",
140         categoriesCacheMap,
141         cfsMap);
```

Listing 5.62: Line 138 possible solution

5.3.3 Useless blank lines to remove

The following blank lines are not useful to separate declarations of variables:

```
63 private EbayStoreSiteFacade siteFacade = null;
64
65 private AttributeSet[] joinedAttrSets = null;
```

Listing 5.63: Lines 63 violation of the rule

```
113 Map<String, CategoryType> categoriesCacheMap =
    ↪ this.siteFacade.getSiteCategoriesMap().get(apiContext.getSite());
114
115 Map<String, CategoryFeatureType> cfsMap =
    ↪ this.siteFacade.getSiteCategoriesFeaturesMap().get(apiContext.getSite());
```

Listing 5.64: Lines 113 violation of the rule

```
155 paymentMethods = fiterPaymentMethod(paymentMethods);
156
157 storeOwnerExtendedListingDuration =
    ↪ siteDefaults.getStoreOwnerExtendedListingDurations();
```

Listing 5.65: Lines 155 violation of the rule

```
155 storeOwnerExtendedListingDuration =
    ↪ siteDefaults.getStoreOwnerExtendedListingDurations();
156
157 bestOfferEnabled = featureDefinition.getBestOfferEnabled();
```

Listing 5.66: Lines 155 violation of the rule

5.3.4 Useful blank lines to add

The following lines need a blank line to separate sections of code:

```
18 */
19 package org.apache.ofbiz.ebaystore;
```

Listing 5.67: Lines 18-19 violation of the rule

```
18 */
19
20 package org.apache.ofbiz.ebaystore;
```

Listing 5.68: Lines 18-19 possible solution

```
37 import com.ebay.sdk.call.GetDescriptionTemplatesCall;
38 import
    ↪ com.ebay.soap.eBLBaseComponents.BestOfferEnabledDefinitionType;
```

Listing 5.69: Lines 37-38 violation of the rule

```
37 import com.ebay.sdk.call.GetDescriptionTemplatesCall;
38
39 import
   ↪ com.ebay.soap.eBLBaseComponents.BestOfferEnabledDefinitionType;
```

Listing 5.70: Lines 37-38 possible solution

```
58 public static final String module =
   ↪ EbayStoreCategoryFacade.class.getName();
59 private ApiContext apiContext = null;
```

Listing 5.71: Lines 58-59 violation of the rule

```
58 public static final String module =
   ↪ EbayStoreCategoryFacade.class.getName();
59
60 private ApiContext apiContext = null;
```

Listing 5.72: Lines 58-59 possible solution

```
122 }
123 //get returnPolicyEnabled feature
```

Listing 5.73: Lines 122-123 violation of the rule

```
122 }
123
124 //get returnPolicyEnabled feature
```

Listing 5.74: Lines 122-123 possible solution

```
184 CategoryFeatureType cf = cfsMap.get(catId);
185 // invoke the method indicated by methodName
```

Listing 5.75: Lines 184-185 violation of the rule

```
184 CategoryFeatureType cf = cfsMap.get(catId);
185
186 // invoke the method indicated by methodName
```

Listing 5.76: Lines 184-185 possible solution

```
229  resp = (GetDescriptionTemplatesResponseType) call.execute(req);
230  if (resp != null && "SUCCESS".equals(resp.getAck().toString()))
     ↪ {
```

Listing 5.77: Lines 229-230 violation of the rule

```
229  resp = (GetDescriptionTemplatesResponseType) call.execute(req);
230
231  if (resp != null && "SUCCESS".equals(resp.getAck().toString()))
     ↪ {
```

Listing 5.78: Lines 229-230 possible solution

```
284  List<Map<String,Object>> themes = new
     ↪ LinkedList<Map<String,Object>>();
285  for (Map<String,Object> temp : this.adItemTemplates) {
```

Listing 5.79: Lines 284-285 violation of the rule

```
284  List<Map<String,Object>> themes = new
     ↪ LinkedList<Map<String,Object>>();
285
286  for (Map<String,Object> temp : this.adItemTemplates) {
```

Listing 5.80: Lines 284-285 possible solution

5.4 Comments

5.4.1 Classes, interfaces, methods not described

The following classes and methods should be properly described using Javadoc comments:

```
57  public class EbayStoreCategoryFacade {...}
```

Listing 5.81: EbayStoreCategoryFacade class Javadoc missing

```
74  public EbayStoreCategoryFacade(String catId, ApiContext
    ↪ apiContext,
75          IAttributesMaster attrMaster,
76          EbayStoreSiteFacade siteFacade)
77      throws SdkException, Exception {...}
```

Listing 5.82: Constructor description missing

```
82 private void syncCategoryMetaData() throws SdkException,
   ↪ Exception {...}
```

Listing 5.83: Method description missing

```
89 private void syncJoinedAttrSets() throws SdkException, Exception
   ↪ {...}
```

Listing 5.84: Method description missing

```
98 private void syncNameRecommendationTypes() throws ApiException,
   ↪ SdkException, Exception {...}
```

Listing 5.85: Method description missing

```
110 public void syncCategoryFeatures() throws Exception {...}
```

Listing 5.86: Method description missing

```
220 public List<Map<String,Object>> syncAdItemTemplates()
221         throws ApiException, SdkSoapException, SdkException {...}
```

Listing 5.87: Method description missing

```
329 public boolean AttributesEnabled() {...}
```

Listing 5.88: Method description missing

```
161 private static BuyerPaymentMethodCodeType[]
   ↪ fiterPaymentMethod(BuyerPaymentMethodCodeType[]
   ↪ paymentMethods) {...}
```

Listing 5.89: method Javadoc missing

5.4.2 Javadoc tags not described

The following Javadoc tags should be more clear, specifying when a null value may be returned or an excpetion thrown.

```
177 /**
178 * @return generic Object
179 * @throws Exception
180 */
```

Listing 5.90: tags not described

```
205 /**
206 * @return generic object
207 */
```

Listing 5.91: tags not described

5.4.3 Complex statement not described

The following complex statements should be better documented to explain what they do.

```
233 for (DescriptionTemplateType descTemplateType :
    ↪ descriptionTemplateTypeList) {
234     List<Map<String,Object>> templateList = null;
235     Map<String,Object> templateGroup = null;
236     if
        ↪ ("THEME".equals(String.valueOf(descTemplateType.getType())))
        ↪ {
237         Map<String,Object> template = new HashMap<String,
            ↪ Object>();
238         template.put("TemplateId",
            ↪ String.valueOf(descTemplateType.getID()));
239         template.put("TemplateImageURL",
            ↪ descTemplateType.getImageURL());
240         template.put("TemplateName", descTemplateType.getName());
241         template.put("TemplateType", descTemplateType.getType());
242
243         // check group template by groupId
244         for (Map<String,Object> temGroup : temGroupList) {
245             if
                ↪ (temGroup.get("TemplateGroupId").equals(descTemplateType.getG
                ↪ {
246                 templateGroup = temGroup;
247                 break;
248             }
249         }
250         if (templateGroup == null) {
251             templateGroup = new HashMap<String, Object>();
252             templateList = new LinkedList<Map<String,Object>>();
253             templateGroup.put("TemplateGroupId",
                ↪ descTemplateType.getGroupID().toString());
254             templateList.add(template);
255             templateGroup.put("Templates", templateList);
```

```
256                 temGroupList.add(templateGroup);
257             } else {
258                 if (templateGroup.get("Templates") != null) {
259                     templateList =
                        ↪ UtilGenerics.checkList(templateGroup.get("Templates"));
260                     templateList.add(template);
261                 }
262             }
263         } else if
            ↪ ("Layout".equals(String.valueOf(descTemplateType.getType())))
            ↪ {
264         }
265     }
```

Listing 5.92: complex for not documented

5.5 Java source files

Javadoc descriptions are missing for all the classes and methods.

5.6 Class and interface declarations

5.6.1 Methods grouping by functionality

The following method is in the middle of a group of getter methods, grouped by a functionality:

```
330 public boolean AttributesEnabled() {
331     return this.joinedAttrSets != null &&
            ↪ this.joinedAttrSets.length > 0;
332 }
```

Listing 5.93: AttributesEnabled method in the middle of getters

5.7 Method calls

The following methods are invoked on objects that may be null therefore is needed a check:

```
93   AttributeSet[] itemSpecAttrSets =
       ↪ attrMaster.getItemSpecificAttributeSetsForCategories(ids);
```

Listing 5.94: getItemSpecificAttributeSetsForCategories invocation

```
93 AttributeSet[] itemSpecAttrSets = null;
94 if(attrMaster != null){
95       itemSpecAttrSets =
           ↪ attrMaster.getItemSpecificAttributeSetsForCategories(ids);
96 }
```

Listing 5.95: getItemSpecificAttributeSetsForCategories invocation possible solution

---

```
94 AttributeSet[] siteWideAttrSets =
     ↪ attrMaster.getSiteWideAttributeSetsForCategories(ids);
```

Listing 5.96: getSiteWideAttributeSetsForCategories invocation

```
94 AttributeSet[] siteWideAttrSets = null;
95 if(attrMaster != null){
96      siteWideAttrSets =
          ↪ attrMaster.getSiteWideAttributeSetsForCategories(ids);
97 }
```

Listing 5.97: getSiteWideAttributeSetsForCategories invocation possible solution

---

```
95 AttributeSet[] joinedAttrSets =
     ↪ attrMaster.joinItemSpecificAndSiteWideAttributeSets(itemSpecAttrSets,
     ↪ siteWideAttrSets);
```

Listing 5.98: joinItemSpecificAndSiteWideAttributeSets invocation

```
95 AttributeSet[] joinedAttrSets = null;
96 if(aif != null){
97      joinedAttrSets =
          ↪ attrMaster.joinItemSpecificAndSiteWideAttributeSets
          ↪ (itemSpecAttrSets, siteWideAttrSets);
98 }
```

Listing 5.99: joinItemSpecificAndSiteWideAttributeSets invocation possible solution

---

```
108 this.nameRecommendationTypes =
    ↪ recommendations.getNameRecommendation();
```

Listing 5.100: getNameRecommendation invocation

```
108 if(recommendations != null){
109     this.nameRecommendationTypes =
        ↪ recommendations.getNameRecommendation();
110 }
```

Listing 5.101: getNameRecommendation invocation possible solution

---

```
112 Map<String, CategoryType> categoriesCacheMap =
    ↪ this.siteFacade.getSiteCategoriesMap().get(apiContext.getSite());
```

Listing 5.102: get invocation

```
112 Map<String, CategoryType> categoriesCacheMap = null;
113 if(this.siteFacade.getSiteCategoriesMap() != null){
114     categoriesCacheMap =
        ↪ this.siteFacade.getSiteCategoriesMap().get(apiContext.getSite());
115 }
```

Listing 5.103: get invocation possible solution

```
114 Map<String, CategoryFeatureType> cfsMap =
    ↪ this.siteFacade.getSiteCategoriesFeaturesMap().
    ↪ get(apiContext.getSite());
```

Listing 5.104: get invocation

```
114 Map<String, CategoryFeatureType> cfsMap = null;
115 if(this.siteFacade.getSiteCategoriesFeaturesMap() != null){
116     cfsMap = this.siteFacade.getSiteCategoriesFeaturesMap().
        ↪ get(apiContext.getSite());
117 }
```

Listing 5.105: get invocation possible solution

```
115 SiteDefaultsType siteDefaults =
    ↪ this.siteFacade.getSiteFeatureDefaultMap().get(apiContext.getSite());
```

Listing 5.106: get invocation

```
115 SiteDefaultsType siteDefaults = null;
116 if(this.siteFacade.getSiteFeatureDefaultMap() != null){
117     siteDefaults =
        ↪ this.siteFacade.getSiteFeatureDefaultMap().get(apiContext.getSite());
118 }
```

Listing 5.107: get invocation possible solution

```
116 FeatureDefinitionsType featureDefinition =
    ↪ this.siteFacade.getSiteFeatureDefinitionsMap().get(apiContext.getSite());
```

Listing 5.108: get invocation

```
116 FeatureDefinitionsType featureDefinition = null;
117 if(this.siteFacade.getSiteFeatureDefinitionsMap() != null){
118     featureDefinition =
        ↪ this.siteFacade.getSiteFeatureDefinitionsMap().
        ↪ get(apiContext.getSite());
119 }
```

Listing 5.109: get invocation possible solution

```
121 itemSpecificEnabled = siteDefaults.getItemSpecificsEnabled();
```

Listing 5.110: getItemSpecificsEnabled invocation

```
121 if(siteDefaults != null){
122     itemSpecificEnabled = siteDefaults.getItemSpecificsEnabled();
123 }
```

Listing 5.111: getItemSpecificsEnabled invocation possible solution

---

```
126 retPolicyEnabled = siteDefaults.isReturnPolicyEnabled();
```

Listing 5.112: isReturnPolicyEnabled invocation

```
126 if(siteDefaults != null){
127     retPolicyEnabled = siteDefaults.isReturnPolicyEnabled();
128 }
```

Listing 5.113: isReturnPolicyEnabled invocation possible solution

---

```
130 ListingDurationDefinitionsType listDuration =
    ↪ featureDefinition.getListingDurations();
```

Listing 5.114: getListingDurations invocation

```
130 ListingDurationDefinitionsType listDuration = null;
131 if(featureDefinition != null){
132     listDuration = featureDefinition.getListingDurations();
133 }
```

Listing 5.115: getListingDurations invocation possible solution

---

```
131 ListingDurationDefinitionType[] durationArray =
    ↪ listDuration.getListingDuration();
```

Listing 5.116: getListingDuration invocation

```
131 ListingDurationDefinitionType[] durationArray;
132 if(featureDefinition != null){
133     durationArray = listDuration.getListingDuration();
134 }
```

Listing 5.117: getListingDuration invocation possible solution

---

```
134 listingDurationMap.put(durationArray[i].getDurationSetID(),
    ↪ durationArray[i].getDuration());
```

Listing 5.118: getDuration invocation

```
134 if(durationArray[i] != null){
135     listingDurationMap.put(durationArray[i].getDurationSetID(),
        ↪ durationArray[i].getDuration());
136 }
```

Listing 5.119: getDuration invocation possible solution

```
134 listingDurationMap.put(durationArray[i].getDurationSetID(),
    ↪ durationArray[i].getDuration());
```

Listing 5.120: getDurationSetID invocation

```
134 if(durationArray[i] != null){
135     listingDurationMap.put(durationArray[i].getDurationSetID(),
        ↪ durationArray[i].getDuration());
136 }
```

Listing 5.121: getDurationSetID invocation possible solution

```
134 listingDurationMap.put(durationArray[i].getDurationSetID(),
    ↪ durationArray[i].getDuration());
```

Listing 5.122: getDurationSetID invocation

```
134 if(durationArray[i] != null){
135     listingDurationMap.put(durationArray[i].getDurationSetID(),
        ↪ durationArray[i].getDuration());
136 }
```

Listing 5.123: getDurationSetID invocation possible solution

```
144 listingDurationReferenceMap.put(listingDuration[i]
    ↪ .getType().value(),listingDuration[i].getValue());
```

Listing 5.124: getType invocation

```
144 if(listingDuration[i] != null){
145     listingDurationReferenceMap.put(listingDuration[i]
        ↪ .getType().value(),listingDuration[i].getValue());
146 }
```

Listing 5.125: getType invocation possible solution

```
144 listingDurationReferenceMap.put(listingDuration[i]
    ↪ .getType().value(),listingDuration[i].getValue());
```

Listing 5.126: getValue invocation

```
144 if(listingDuration[i] != null){
145     listingDurationReferenceMap.put(listingDuration[i]
        ↪ .getType().value(),listingDuration[i].getValue());
146 }
```

Listing 5.127: getValue invocation possible solution

```
192 CategoryType cat = categoriesCacheMap.get(catId);
```

Listing 5.128: get invocation

```
192 CategoryType cat = null;
193 if(categoriesCacheMap != null){
194     cat = categoriesCacheMap.get(catId);
195 }
```

Listing 5.129: get invocation possible solution

```
194 if (cat.getCategoryLevel() == 1)
```

Listing 5.130: getCategoryLevel invocation

```
194 if(cat != null && cat.getCategoryLevel() == 1)
```

Listing 5.131: getCategoryLevel invocation possible solution

```
199 return getInheritProperty(cat.getCategoryParentID(0),
    ↪ methodName, categoriesCacheMap, cfsMap);
```

Listing 5.132: getCategoryParentID invocation

```
199 return cat != null ?
    ↪ getInheritProperty(cat.getCategoryParentID(0), methodName,
    ↪ categoriesCacheMap, cfsMap) : null;
```

Listing 5.133: getCategoryParentID invocation possible solution

```
245 if (temGroup.get("TemplateGroupId")
    ↪ .equals(descTemplateType.getGroupID().toString()))
```

Listing 5.134: get invocation

```
245 if (temGroup != null && temGroup.get("TemplateGroupId")
    ↪ .equals(descTemplateType.getGroupID().toString()))
```

Listing 5.135: get invocation possible solution

```
286 if (temp.get("TemplateGroupId").equals(temGroupId))
```

Listing 5.136: get invocation

```
286 if (temp != null &&
    ↪ temp.get("TemplateGroupId").equals(temGroupId))
```

Listing 5.137: get invocation possible solution

5.8 Other errors

The following else if statement should have at least a correct statement to execute.

```
263 } else if
    ↪ ("Layout".equals(String.valueOf(descTemplateType.getType())))
    ↪ {
264 }
```

Listing 5.138: statement does nothing

The following declarations of data structures such as Map and HashMap should have a whitespace after the comma:

```
67 private Map<Integer,String[]> listingDurationMap = null;
```

Listing 5.139: withespace needed after comma

```
68 private Map<String,Integer> listingDurationReferenceMap = null;
```

Listing 5.140: withespace needed after comma

```
73 private List<Map<String,Object>> adItemTemplates = null;
```

Listing 5.141: withespace needed after comma

```
73 private List<Map<String,Object>> adItemTemplates = null;
```

Listing 5.142: withespace needed after comma

```
221 public List<Map<String,Object>> syncAdItemTemplates() throws
    ↪ ApiException, SdkSoapException, SdkException {
```

Listing 5.143: withespace needed after comma

```
224 List<Map<String,Object>> temGroupList = new
    ↪ LinkedList<Map<String,Object>>();
```

Listing 5.144: withespace needed after comma

```
234 List<Map<String,Object>> templateList = null;
```

Listing 5.145: withespace needed after comma

```
235 Map<String,Object> templateGroup = null;
236
237 Map<String,Object> template = new HashMap<String, Object>();
```

Listing 5.146: withespace needed after comma

```
244 for (Map<String,Object> temGroup : temGroupList) {
```

Listing 5.147: withespace needed after comma

```
252 templateList = new LinkedList<Map<String,Object>>();
```

Listing 5.148: withespace needed after comma

```
252 for (Map<String,Object> temGroup : temGroupList) {
```

Listing 5.149: withespace needed after comma

```
283 public List<Map<String,Object>> getAdItemTemplates(String
    ↪ temGroupId) {
284   List<Map<String,Object>> themes = new
        ↪ LinkedList<Map<String,Object>>();
```

```
285     for (Map<String,Object> temp : this.adItemTemplates) {
```

Listing 5.150: withespace needed after comma

```
342 public List<Map<String,Object>> getAdItemTemplates() {
```

Listing 5.151: withespace needed after comma

The string concatenation should have white spaces before and after the ‘+‘ sign.

```
232 Debug.logInfo("layout of category "+ this.catId +":"+
    ↪ resp.getLayoutTotal(), module);
```

Listing 5.152: withespace needed after comma
