Understanding Documentation Presentation Smells

Dear developer, thank you for being a part of this study!

APIs (Application Programming Interfaces) are interfaces to reusable software libraries and frameworks. The proper learning of APIs is paramount to support modern day rapid software development. To achieve this goal, APIs typically are supported by official documentation. An API documentation is a product itself, which warrants the creation and maintenance principles similar to any existing software product.

A good documentation can facilitate the proper usage of an API, while a bad documentation can severely harm its adoption. Unfortunately, research shows that API official documentation can be often incomplete, incorrect, and outdated.

This survey is used to evaluate a research that is focused on a special kind of documentation problem that is Documentation Presentation Smell i.e. presentation related issues in documentation.

The survey will take 15-20 minutes for you to complete. All the responses will be anonymized and we will not share your personal information with anyone else.

Thank you very much for your participation!
*Required

Information About You

In this section, we ask you questions to know your software development experience, experience in object-oriented programming (especially Java).

1. Current Profession *

Mark only one oval.

- ◯ Software Developer
- ◯ Technical lead
- ◯ Research engineer
- ◯ Faculty member
- ◯ Other: _____

2. Year(s) of experience in software development *

Mark only one oval.

- 0 - 2
- 3 - 5
- 6 - 8
- 9 - 11
- 12 - 14
- 15 or more

Documentation Presentation Smell

In this work, we define a new type of problem that might occur in various API documentations. We are calling it "Documentation Presentation Smell".

Similar to code smells i.e. recurrent patterns of bad programming practice, smells can occur in documentation as well. A documentation presentation smell (i.e. bad design, presentation issue in documentation) does not make the documentation incorrect, but it makes it difficult to properly understand and use the underlying documentation unit. A documentation unit can be the documentation of a method, a type (e.g., a class), a package, and an API overall. Following some previous works in API documentation, we consider documentation unit as the documentation of a method.

We have identified 5 types of documentation presentation smells. They are:

1. Lazy

2. Bloated

3. Tangled

4. Excessive Structural Info

5. Fragmented

This survey is focused on collecting your thoughts on this classification which will assist our further analysis.

1. Lazy Documentation

Definition: Documentations that contain very small information to convey to the readers. Does not contain any extra information except what can be perceived directly from the function name.

Example-1

Method Prototype:
public void setBaseURI(String baseURI)

Documentation:
Sets the base URI.

Specified by:
getBaseURI in interface XMLCryptoContext
Returns:
the base URI, or null if not specified

3. Do you think the documentation mentioned above (Example-1) is Lazy? *

Mark only one oval.

- Strongly agree
- Agree
- Neutral
- Disagree
- Strongly disagree

1. Lazy Documentation

Definition: Documentations that contain very small information to convey to the readers. Does not contain any extra information except what can be perceived directly from the function name.

Example-2

Method Prototype:
public MouseMotionListener getMouseMotionListener()

Documentation:
Implementation of ComboPopup.getMouseMotionListener().
Specified by:
`getMouseMotionListener` in interface `ComboPopup`
Returns:
a `MouseMotionListener` or null

4. Do you think the documentation mentioned above (Example-2) is Lazy? *

Mark only one oval.

- Strongly agree
- Agree
- Neutral
- Disagree
- Strongly disagree

1. Lazy Documentation

Definition: Documentations that contain very small information to convey to the readers. Does not contain any extra information except what can be perceived directly from the function name.

5. Based on your experience of the last three months, how frequently did you observe the lazy documentation? *

Mark only one oval.

- Never
- Once or twice
- Occasionally
- Frequently
- No opinion

6. How severe was the lazy documentation problem to complete your development task, when you last observed it? *

Mark only one oval.

- ◯ Not a Problem
- ◯ Moderate (kind of irritating)
- ◯ Severe (I wasted a lot of time on this but figured it out)
- ◯ Blocker (I could not get past it and picked another api)
- ◯ No opinion

7. Any comment about your experience with the lazy documentation problems?

_____

_____

_____

_____

_____

2. Bloated Documentation

Definition: Documentations that are excessively lengthy and verbose. Hence, boring to read and understand.

Example-1

Method Prototype:
public DateTimeFormatterBuilder appendPattern(String pattern)

Documentation:
Appends the elements defined by the specified pattern to the builder.
All letters 'A' to 'Z' and 'a' to 'z' are reserved as pattern letters. The characters '#', '{' and '}' are reserved for future use. The characters '[' and ']' indicate optional patterns. The following pattern letters are defined:

Symbol	Meaning	Presentation	Examples
G	era	text	AD; Anno Domini; A
u	year	year	2004; 04
y	year-of-era	year	2004; 04
D	day-of-year	number	189
M/L	month-of-year	number/text	7; 07; Jul; July; J
d	day-of-month	number	10
Q/q	quarter-of-year	number/text	3; 03; Q3; 3rd quarter
Y	week-based-year	year	1996; 96
w	week-of-week-based-year	number	27
W	week-of-month	number	4
E	day-of-week	text	Tue; Tuesday; T
e/c	localized day-of-week	number/text	2; 02; Tue; Tuesday; T
F	week-of-month	number	3
a	am-pm-of-day	text	PM
h	clock-hour-of-am-pm (1-12)	number	12
K	hour-of-am-pm (0-11)	number	0
k	clock-hour-of-am-pm (1-24)	number	0
H	hour-of-day (0-23)	number	0
m	minute-of-hour	number	30
s	second-of-minute	number	55
S	fraction-of-second	fraction	978
A	milli-of-day	number	1234
n	nano-of-second	number	987654321
N	nano-of-day	number	1234000000
V	time-zone ID	zone-id	America/Los_Angeles; Z; -08:30
z	time-zone name	zone-name	Pacific Standard Time; PST
O	localized zone-offset	offset-O	GMT+8; GMT+08:00; UTC-08:00;
X	zone-offset 'Z' for zero	offset-X	Z; -08; -0830; -08:30; -083015; -08:30:15;
x	zone-offset	offset-x	+0000; -08; -0830; -08:30; -083015; -08:30:15;
Z	zone-offset	offset-Z	+0000; -0800; -08:00;
p	pad next	pad modifier	1
'	escape for text	delimiter	
''	single quote	literal	'
[	optional section start		
]	optional section end		
#	reserved for future use		
{	reserved for future use		
}	reserved for future use		

The count of pattern letters determine the format. See DateTimeFormatter for a user-focused description of the patterns. The following tables define how the pattern letters map to the builder.

Date fields: Pattern letters to output a date.

Pattern	Count	Equivalent builder methods
G	1	appendText(ChronoField.ERA, TextStyle.SHORT)
GG	2	appendText(ChronoField.ERA, TextStyle.SHORT)
GGG	3	appendText(ChronoField.ERA, TextStyle.SHORT)
GGGG	4	appendText(ChronoField.ERA, TextStyle.FULL)
GGGGG	5	appendText(ChronoField.ERA, TextStyle.NARROW)
u	1	appendValue(ChronoField.YEAR, 1, 19, SignStyle.NORMAL);
uu	2	appendValueReduced(ChronoField.YEAR, 2, 2000);
uuu	3	appendValue(ChronoField.YEAR, 3, 19, SignStyle.NORMAL);
u..u	4..n	appendValue(ChronoField.YEAR, n, 19, SignStyle.EXCEEDS_PAD);
y	1	appendValue(ChronoField.YEAR_OF_ERA, 1, 19, SignStyle.NORMAL);
yy	2	appendValueReduced(ChronoField.YEAR_OF_ERA, 2, 2000);
yyy	3	appendValue(ChronoField.YEAR_OF_ERA, 3, 19, SignStyle.NORMAL);
y..y	4..n	appendValue(ChronoField.YEAR_OF_ERA, n, 19, SignStyle.EXCEEDS_PAD);
Y	1	append special localized WeekFields element for numeric week-based-year
YY	2	append special localized WeekFields element for reduced numeric week-based-year 2 digits;
YYY	3	append special localized WeekFields element for numeric week-based-year (3, 19, SignStyle.NORMAL);
Y..Y	4..n	append special localized WeekFields element for numeric week-based-year (n, 19, SignStyle.EXCEEDS_PAD);
Q	1	appendValue(IsoFields.QUARTER_OF_YEAR);
QQ	2	appendValue(IsoFields.QUARTER_OF_YEAR, 2);
QQQ	3	appendText(IsoFields.QUARTER_OF_YEAR, TextStyle.SHORT)
QQQQ	4	appendText(IsoFields.QUARTER_OF_YEAR, TextStyle.FULL)
QQQQQ	5	appendText(IsoFields.QUARTER_OF_YEAR, TextStyle.NARROW)
q	1	appendValue(IsoFields.QUARTER_OF_YEAR);
qq	2	appendValue(IsoFields.QUARTER_OF_YEAR, 2);
qqq	3	appendText(IsoFields.QUARTER_OF_YEAR, TextStyle.SHORT_STANDALONE)
qqqq	4	appendText(IsoFields.QUARTER_OF_YEAR, TextStyle.FULL_STANDALONE)
qqqqq	5	appendText(IsoFields.QUARTER_OF_YEAR, TextStyle.NARROW_STANDALONE)
M	1	appendValue(ChronoField.MONTH_OF_YEAR);
MM	2	appendValue(ChronoField.MONTH_OF_YEAR, 2);
MMM	3	appendText(ChronoField.MONTH_OF_YEAR, TextStyle.SHORT)
MMMM	4	appendText(ChronoField.MONTH_OF_YEAR, TextStyle.FULL)
MMMMM	5	appendText(ChronoField.MONTH_OF_YEAR, TextStyle.NARROW)
L	1	appendValue(ChronoField.MONTH_OF_YEAR);
LL	2	appendValue(ChronoField.MONTH_OF_YEAR, 2);
LLL	3	appendText(ChronoField.MONTH_OF_YEAR, TextStyle.SHORT_STANDALONE)
LLLL	4	appendText(ChronoField.MONTH_OF_YEAR, TextStyle.FULL_STANDALONE)
LLLLL	5	appendText(ChronoField.MONTH_OF_YEAR, TextStyle.NARROW_STANDALONE)
w	1	append special localized WeekFields element for numeric week-of-year
ww	2	append special localized WeekFields element for numeric week-of-year, zero-padded
W	1	append special localized WeekFields element for numeric week-of-month
d	1	appendValue(ChronoField.DAY_OF_MONTH)
dd	2	appendValue(ChronoField.DAY_OF_MONTH, 2)
D	1	appendValue(ChronoField.DAY_OF_YEAR)
DD	2	appendValue(ChronoField.DAY_OF_YEAR, 2)
DDD	3	appendValue(ChronoField.DAY_OF_YEAR, 3)
F	1	appendValue(ChronoField.ALIGNED_DAY_OF_WEEK_IN_MONTH)
E	1	appendText(ChronoField.DAY_OF_WEEK, TextStyle.SHORT)
EE	2	appendText(ChronoField.DAY_OF_WEEK, TextStyle.SHORT)
EEE	3	appendText(ChronoField.DAY_OF_WEEK, TextStyle.SHORT)
EEEE	4	appendText(ChronoField.DAY_OF_WEEK, TextStyle.FULL)
EEEEE	5	appendText(ChronoField.DAY_OF_WEEK, TextStyle.NARROW)

```
  e      1     append special localized WeekFields element for numeric day-of-week
  ee     2     append special localized WeekFields element for numeric day-of-week, zero-padded
  eee    3     appendText(ChronoField.DAY_OF_WEEK, TextStyle.SHORT)
  eeee   4     appendText(ChronoField.DAY_OF_WEEK, TextStyle.FULL)
  eeeee  5     appendText(ChronoField.DAY_OF_WEEK, TextStyle.NARROW)
  c      1     append special localized WeekFields element for numeric day-of-week
  ccc    3     appendText(ChronoField.DAY_OF_WEEK, TextStyle.SHORT_STANDALONE)
  cccc   4     appendText(ChronoField.DAY_OF_WEEK, TextStyle.FULL_STANDALONE)
  ccccc  5     appendText(ChronoField.DAY_OF_WEEK, TextStyle.NARROW_STANDALONE)
```

Time fields: Pattern letters to output a time.

```
 Pattern  Count  Equivalent builder methods
 -------  -----  --------------------------
  a       1      appendText(ChronoField.AMPM_OF_DAY, TextStyle.SHORT)
  h       1      appendValue(ChronoField.CLOCK_HOUR_OF_AMPM)
  hh      2      appendValue(ChronoField.CLOCK_HOUR_OF_AMPM, 2)
  H       1      appendValue(ChronoField.HOUR_OF_DAY)
  HH      2      appendValue(ChronoField.HOUR_OF_DAY, 2)
  k       1      appendValue(ChronoField.CLOCK_HOUR_OF_DAY)
  kk      2      appendValue(ChronoField.CLOCK_HOUR_OF_DAY, 2)
  K       1      appendValue(ChronoField.HOUR_OF_AMPM)
  KK      2      appendValue(ChronoField.HOUR_OF_AMPM, 2)
  m       1      appendValue(ChronoField.MINUTE_OF_HOUR)
  mm      2      appendValue(ChronoField.MINUTE_OF_HOUR, 2)
  s       1      appendValue(ChronoField.SECOND_OF_MINUTE)
  ss      2      appendValue(ChronoField.SECOND_OF_MINUTE, 2)

  S..S    1..n   appendFraction(ChronoField.NANO_OF_SECOND, n, n, false)
  A       1      appendValue(ChronoField.MILLI_OF_DAY)
  A..A    2..n   appendValue(ChronoField.MILLI_OF_DAY, n)
  n       1      appendValue(ChronoField.NANO_OF_SECOND)
  n..n    2..n   appendValue(ChronoField.NANO_OF_SECOND, n)
  N       1      appendValue(ChronoField.NANO_OF_DAY)
  N..N    2..n   appendValue(ChronoField.NANO_OF_DAY, n)
```

Zone ID: Pattern letters to output ZoneId.

```
 Pattern  Count  Equivalent builder methods
 -------  -----  --------------------------
  VV      2      appendZoneId()
  z       1      appendZoneText(TextStyle.SHORT)
  zz      2      appendZoneText(TextStyle.SHORT)
  zzz     3      appendZoneText(TextStyle.SHORT)
  zzzz    4      appendZoneText(TextStyle.FULL)
```

Zone offset: Pattern letters to output ZoneOffset.

```
 Pattern  Count  Equivalent builder methods
 -------  -----  --------------------------
  O       1      appendLocalizedOffsetPrefixed(TextStyle.SHORT);
  OOOO    4      appendLocalizedOffsetPrefixed(TextStyle.FULL);
  X       1      appendOffset("+HHmm","Z")
  XX      2      appendOffset("+HHMM","Z")
  XXX     3      appendOffset("+HH:MM","Z")
  XXXX    4      appendOffset("+HHMMss","Z")
  XXXXX   5      appendOffset("+HH:MM:ss","Z")
  x       1      appendOffset("+HHmm","+00")
  xx      2      appendOffset("+HHMM","+0000")
  xxx     3      appendOffset("+HH:MM","+00:00")
  xxxx    4      appendOffset("+HHMMss","+0000")
```

```
xxxxx   5     appendOffset("+HH:MM:ss","+00:00")
Z       1     appendOffset("+HHMM","+0000")
ZZ      2     appendOffset("+HHMM","+0000")
ZZZ     3     appendOffset("+HHMM","+0000")
ZZZZ    4     appendLocalizedOffset(TextStyle.FULL);
ZZZZZ   5     appendOffset("+HH:MM:ss","Z")
```

Modifiers: Pattern letters that modify the rest of the pattern:

```
Pattern  Count  Equivalent builder methods
-------  -----  --------------------------
[       1      optionalStart()
]       1      optionalEnd()
p..p    1..n   padNext(n)
```

Any sequence of letters not specified above, unrecognized letter or reserved character will throw an exception. Future versions may add to the set of patterns. It is recommended to use single quotes around all characters that you want to output directly to ensure that future changes do not break your application.

Note that the pattern string is similar, but not identical, to SimpleDateFormat. The pattern string is also similar, but not identical, to that defined by the Unicode Common Locale Data Repository (CLDR/LDML). Pattern letters 'X' and 'u' are aligned with Unicode CLDR/LDML. By contrast, SimpleDateFormat uses 'u' for the numeric day of week. Pattern letters 'y' and 'Y' parse years of two digits and more than 4 digits differently. Pattern letters 'n', 'A', 'N', and 'p' are added. Number types will reject large numbers.

Parameters:
pattern - the pattern to add, not null
Returns:
this, for chaining, not null
Throws:
IllegalArgumentException - if the pattern is invalid

8. Do you think the documentation mentioned above (Example-1) is bloated? *

Mark only one oval.

- ◯ Strongly agree
- ◯ Agree
- ◯ Neutral
- ◯ Disagree
- ◯ Strongly disagree

2. Bloated Documentation

Definition: Documentations that are excessively lengthy and verbose. Hence, boring to read and understand.

Example-2

Method Prototype:
public float[] toCIEXYZ(float[] colorvalue)

Documentation:
Transforms a color value assumed to be in this ColorSpace into the CS_CIEXYZ conversion color space.
This method transforms color values using relative colorimetry, as defined by the ICC Specification. This means that the XYZ values returned by this method are represented relative to the D50 white point of the CS_CIEXYZ color space. This representation is useful in a two-step color conversion process in which colors are transformed from an input color space to CS_CIEXYZ and then to an output color space. This representation is not the same as the XYZ values that would be measured from the given color value by a colorimeter. A further transformation is necessary to compute the XYZ values that would be measured using current CIE recommended practices. The paragraphs below explain this in more detail.

The ICC standard uses a device independent color space (DICS) as the mechanism for converting color from one device to another device. In this architecture, colors are converted from the source device's color space to the ICC DICS and then from the ICC DICS to the destination device's color space. The ICC standard defines device profiles which contain transforms which will convert between a device's color space and the ICC DICS. The overall conversion of colors from a source device to colors of a destination device is done by connecting the device-to-DICS transform of the profile for the source device to the DICS-to-device transform of the profile for the destination device. For this reason, the ICC DICS is commonly referred to as the profile connection space (PCS). The color space used in the methods toCIEXYZ and fromCIEXYZ is the CIEXYZ PCS defined by the ICC Specification. This is also the color space represented by ColorSpace.CS_CIEXYZ.

The XYZ values of a color are often represented as relative to some white point, so the actual meaning of the XYZ values cannot be known without knowing the white point of those values. This is known as relative colorimetry. The PCS uses a white point of D50, so the XYZ values of the PCS are relative to D50. For example, white in the PCS will have the XYZ values of D50, which is defined to be X=.9642, Y=1.000, and Z=0.8249. This white point is commonly used for graphic arts applications, but others are often used in other applications.

To quantify the color characteristics of a device such as a printer or monitor, measurements of XYZ values for particular device colors are typically made. For purposes of this discussion, the term device XYZ values is used to mean the XYZ values that would be measured from device colors using current CIE recommended practices.

Converting between device XYZ values and the PCS XYZ values returned by this method corresponds to converting between the device's color space, as represented by CIE colorimetric values, and the PCS. There are many factors involved in this process, some of which are quite subtle. The most important, however, is the adjustment made to account for differences between the device's white point and the white point of the PCS. There are many techniques for doing this and it is the subject of much current research and controversy. Some commonly used methods are XYZ scaling, the von Kries transform, and the Bradford transform. The proper method to use depends upon each particular application.

The simplest method is XYZ scaling. In this method each device XYZ value is converted to a PCS XYZ value by multiplying it by the ratio of the PCS white point (D50) to the device white point.

Xd, Yd, Zd are the device XYZ values
Xdw, Ydw, Zdw are the device XYZ white point values
Xp, Yp, Zp are the PCS XYZ values
Xd50, Yd50, Zd50 are the PCS XYZ white point values

Xp = Xd * (Xd50 / Xdw)
Yp = Yd * (Yd50 / Ydw)
Zp = Zd * (Zd50 / Zdw)

Conversion from the PCS to the device would be done by inverting these equations:

Xd = Xp * (Xdw / Xd50)

Yd = Yp * (Ydw / Yd50)
Zd = Zp * (Zdw / Zd50)

Note that the media white point tag in an ICC profile is not the same as the device white point. The media white point tag is expressed in PCS values and is used to represent the difference between the XYZ of device illuminant and the XYZ of the device media when measured under that illuminant. The device white point is expressed as the device XYZ values corresponding to white displayed on the device. For example, displaying the RGB color (1.0, 1.0, 1.0) on an sRGB device will result in a measured device XYZ value of D65. This will not be the same as the media white point tag XYZ value in the ICC profile for an sRGB device.

Specified by:
toCIEXYZ in class ColorSpace
Parameters:
colorvalue - a float array with length of at least the number of components in this ColorSpace.
Returns:
a float array of length 3.
Throws:
ArrayIndexOutOfBoundsException - if array length is not at least the number of components in this ColorSpace.

9. Do you think the documentation mentioned above (Example-2) is bloated? *

Mark only one oval.

- Strongly agree
- Agree
- Neutral
- Disagree
- Strongly disagree

2. Bloated Documentation

Definition: Documentations that are excessively lengthy and verbose. Hence, boring to read and understand.

10. Based on your experience of the last three months, how frequently did you observe the bloated documentation? *

Mark only one oval.

- ◯ Never
- ◯ Once or twice
- ◯ Occasionally
- ◯ Frequently
- ◯ No opinion

11. How severe was the bloated documentation problem to complete your development task, when you last observed it? *

Mark only one oval.

- ◯ Not a Problem
- ◯ Moderate (kind of irritating)
- ◯ Severe (I wasted a lot of time on this but figured it out)
- ◯ Blocker (I could not get past it and picked another api)
- ◯ No opinion

12. Any comment about your experience with the bloated documentation problems?

_____

_____

_____

_____

_____

3. Tangled Documentation

Definition: Documentations with a complex description that is difficult to understand.

Example-1

Method Prototype:
public NamingEnumeration<SearchResult> search(Name name, Attributes matchingAttributes, String[] attributesToReturn)

Documentation:
Searches in a single context for objects that contain a specified set of attributes, and retrieves selected attributes. The search is performed using the default `SearchControls` settings. For an object to be selected, each attribute in `matchingAttributes` must match some attribute of the object. If `matchingAttributes` is empty or null, all objects in the target context are returned.

An attribute _A_ 1 in `matchingAttributes` is considered to match an attribute _A_ 2 of an object if _A_ 1 and _A_ 2 have the same identifier, and each value of _A_ 1 is equal to some value of _A_ 2. This implies that the order of values is not significant, and that _A_ 2 may contain "extra" values not found in _A_ 1 without affecting the comparison. It also implies that if _A_ 1 has no values, then testing for a match is equivalent to testing for the presence of an attribute _A_ 2 with the same identifier.

The precise definition of "equality" used in comparing attribute values is defined by the underlying directory service. It might use the `Object.equals` method, for example, or might use a schema to specify a different equality operation. For matching based on operations other than equality (such as substring comparison) use the version of the `search` method that takes a filter argument. When changes are made to this `DirContext`, the effect on enumerations returned by prior calls to this method is undefined. If the object does not have the attribute specified, the directory will ignore the nonexistent attribute and return the requested attributes that the object does have. A directory might return more attributes than was requested (see **Attribute Type Names** in the class description), but is not allowed to return arbitrary, unrelated attributes.

Specified by: `search` in interface `DirContext`
Parameters: `name` \- the name of the context to search `matchingAttributes` \- the attributes to search for. If empty or null, all objects in the target context are returned. `attributesToReturn` \- the attributes to return. null indicates that all attributes are to be returned; an empty array indicates that none are to be returned.
Returns: a non-null enumeration of `SearchResult` objects. Each `SearchResult` contains the attributes identified by `attributesToReturn` and the name of the corresponding object, named relative to the context named by `name`.
Throws: `NamingException` \- if a naming exception is encountered

13. Do you think the documentation mentioned above (Example-1) is Tangled? *

Mark only one oval.

- ◯ Strongly agree
- ◯ Agree
- ◯ Neutral
- ◯ Disagree
- ◯ Strongly disagree

3. Tangled Documentation

Definition: Documentations with a complex description that is difficult to understand.

Example-2

Method Prototype:
public static ResourceBundle getBundle(String baseName, Locale targetLocale, ClassLoader loader, ResourceBundle.Control control)

Documentation:
Returns a resource bundle using the specified base name, target locale, class loader and control. Unlike the `getBundle` factory methods with no `control` argument, the given `control` specifies how to locate and instantiate resource bundles. Conceptually, the bundle loading process with the given `control` is performed in the following steps.

1. This factory method looks up the resource bundle in the cache for the specified `baseName`, `targetLocale` and `loader`. If the requested resource bundle instance is found in the cache and the time-to-live periods of the instance and all of its parent instances have not expired, the instance is returned to the caller. Otherwise, this factory method proceeds with the loading process below.

2. The `control.getFormats` method is called to get resource bundle formats to produce bundle or resource names. The strings `""java.class""` and `""java.properties""` designate class-based and property-based resource bundles, respectively. Other strings starting with `""java.""` are reserved for future extensions and must not be used for application-defined formats. Other strings designate application-defined formats.

3. The `control.getCandidateLocales` method is called with the target locale to get a list of _candidate`Locale`s_ for which resource bundles are searched.

4. The `control.newBundle` method is called to instantiate a `ResourceBundle` for the base bundle name, a candidate locale, and a format. (Refer to the note on the cache lookup below.) This step is iterated over all combinations of the candidate locales and formats until the `newBundle` method returns a `ResourceBundle` instance or the iteration has used up all the combinations. For example, if the candidate locales are `Locale(""de"", ""DE"")`, `Locale(""de"")` and `Locale("""")` and the formats are `""java.class""` and `""java.properties""`, then the following is the sequence of locale-format combinations to be used to call `control.newBundle`.

`Locale`	`format`
`Locale(""de"", ""DE"")`	`java.class`
`Locale(""de"", ""DE"")`	`java.properties`
`Locale(""de"")`	`java.class`
`Locale(""de"")`	`java.properties`
`Locale("""")`	`java.class`
`Locale("""")`	`java.properties`

5. If the previous step has found no resource bundle, proceed to Step 6. If a bundle has been found that is a base bundle (a bundle for `Locale("""")`), and the candidate locale list only contained `Locale("""")`, return the bundle to the caller. If a bundle has been found that is a base bundle, but the candidate locale list contained locales other than Locale(""""), put the bundle on hold and proceed to Step 6. If a bundle has been found that is not a base bundle, proceed to Step 7.

6. The `control.getFallbackLocale` method is called to get a fallback locale (alternative to the current target locale) to try further finding a resource bundle. If the method returns a non-null locale, it becomes the next target locale and the loading process starts over from Step 3. Otherwise, if a base bundle was found and put on hold in a previous Step 5, it is returned to the caller now. Otherwise, a MissingResourceException is thrown.

7. At this point, we have found a resource bundle that's not the base bundle. If this bundle set its parent during its instantiation, it is returned to the caller. Otherwise, its parent chain is instantiated based on the list of candidate locales from which it was found. Finally, the bundle is returned to the caller.

During the resource bundle loading process above, this factory method looks up the cache before calling the `control.newBundle` method. If the time-to-live period of the resource bundle found in the cache has expired, the factory method calls the `control.needsReload` method to determine whether the resource bundle needs to be reloaded. If reloading is required, the factory method calls `control.newBundle` to reload the resource bundle. If

`control.newBundle` returns `null`, the factory method puts a dummy resource bundle in the cache as a mark of nonexistent resource bundles in order to avoid lookup overhead for subsequent requests. Such dummy resource bundles are under the same expiration control as specified by `control`.

All resource bundles loaded are cached by default. Refer to `control.getTimeToLive` for details.

The following is an example of the bundle loading process with the default `ResourceBundle.Control` implementation.

Conditions:

* Base bundle name: `foo.bar.Messages`

* Requested `Locale`: `Locale.ITALY`

* Default `Locale`: `Locale.FRENCH`

* Available resource bundles: `foo/bar/Messages_fr.properties` and `foo/bar/Messages.properties`

First, `getBundle` tries loading a resource bundle in the following sequence.

* class `foo.bar.Messages_it_IT`

* file `foo/bar/Messages_it_IT.properties`

* class `foo.bar.Messages_it`

* file `foo/bar/Messages_it.properties`

* class `foo.bar.Messages`

* file `foo/bar/Messages.properties`

At this point, `getBundle` finds `foo/bar/Messages.properties`, which is put on hold because it's the base bundle. `getBundle` calls `control.getFallbackLocale(""foo.bar.Messages"", Locale.ITALY)` which returns `Locale.FRENCH`. Next, `getBundle` tries loading a bundle in the following sequence.

* class `foo.bar.Messages_fr`

* file `foo/bar/Messages_fr.properties`

* class `foo.bar.Messages`

* file `foo/bar/Messages.properties`

`getBundle` finds `foo/bar/Messages_fr.properties` and creates a `ResourceBundle` instance. Then, `getBundle` sets up its parent chain from the list of the candidate locales. Only `foo/bar/Messages.properties` is found in the list and `getBundle` creates a `ResourceBundle` instance that becomes the parent of the instance for `foo/bar/Messages_fr.properties`.

Parameters:
`baseName` \- the base name of the resource bundle, a fully qualified class name
`targetLocale` \- the locale for which a resource bundle is desired
`loader` \- the class loader from which to load the resource bundle
`control` \- the control which gives information for the resource bundle loading process

Returns:
a resource bundle for the given base name and locale

Throws:
`NullPointerException` \- if `baseName`, `targetLocale`, `loader`, or `control` is `null`

`MissingResourceException` \- if no resource bundle for the specified base name can be found
`IllegalArgumentException` \- if the given `control` doesn't perform properly (e.g., `control.getCandidateLocales` returns null.) Note that validation of `control` is performed as needed.

14. Do you think the documentation mentioned above (Example-2) is Tangled? *

Mark only one oval.

- Strongly agree
- Agree
- Neutral
- Disagree
- Strongly disagree

3. Tangled Documentation

Definition: Documentations with a complex description that is difficult to understand.

15. Based on your experience of the last three months, how frequently did you observe the tangled documentation? *

Mark only one oval.

- Never
- Once or twice
- Occasionally
- Frequently
- No opinion

16. How severe was the tangled documentation problem to complete your development task, when you last observed it? *

Mark only one oval.

- ◯ Not a Problem
- ◯ Moderate (kind of irritating)
- ◯ Severe (I wasted a lot of time on this but figured it out)
- ◯ Blocker (I could not get past it and picked another api)
- ◯ No opinion

17. Any comment about your experience with the tangled documentation problems?

_____

_____

_____

_____

_____

4. Excessive Structural Info

Definition: Documentations with description that contains too many structural syntax or structure. For example, the Javadoc of the java.lang.Object class Javadoc lists all the hundreds of subclasses of the class.

Example-1

Method Prototype:
public void printStackTrace()

Documentation:
Prints this throwable and its backtrace to the standard error stream. This method prints a stack trace for this `Throwable` object on the error output stream that is the value of the field `System.err`. The first line of output contains the result of the `toString()` method for this object. Remaining lines represent data previously recorded by the method `fillInStackTrace()`.

The format of this information depends on the implementation, but the following example may be regarded as typical:

```
> >     java.lang.NullPointerException
>           at MyClass.mash(MyClass.java:9)
>           at MyClass.crunch(MyClass.java:6)
>           at MyClass.main(MyClass.java:3)
>
```

This example was produced by running the program:

```
    class MyClass {

    public static void main(String[] args) {

       crunch(null);

    }

    static void crunch(int[] a) {

       mash(a);

    }

    static void mash(int[] b) {

       System.out.println(b[0]);

    }

 }
```

The backtrace for a throwable with an initialized, non-null cause should generally include the backtrace for the cause. The format of this information depends on the implementation, but the following example may be regarded as typical:

```
    HighLevelException: MidLevelException: LowLevelException
       at Junk.a(Junk.java:13)
       at Junk.main(Junk.java:4)

 Caused by: MidLevelException: LowLevelException
       at Junk.c(Junk.java:23)
       at Junk.b(Junk.java:17)
       at Junk.a(Junk.java:11)

 Caused by: LowLevelException
       at Junk.e(Junk.java:30)
```

```
    at Junk.d(Junk.java:27)
    at Junk.c(Junk.java:21)
```

Note the presence of lines containing the characters `""..."".`. These lines indicate that the remainder of the stack trace for this exception matches the indicated number of frames from the bottom of the stack trace of the exception that was caused by this exception (the ""enclosing"" exception). This shorthand can greatly reduce the length of the output in the common case where a wrapped exception is thrown from same method as the ""causative exception"" is caught.

The above example was produced by running the program:

```
public class Junk {

public static void main(String args[]) {

    try {

        a();

    } catch(HighLevelException e) {

        e.printStackTrace();

    }

}

static void a() throws HighLevelException {

    try {

        b();

    } catch(MidLevelException e) {

        throw new HighLevelException(e);

    }

}

static void b() throws MidLevelException {

    c();

}

static void c() throws MidLevelException {

    try {

        d();

    } catch(LowLevelException e) {

        throw new MidLevelException(e);

    }

}
```

```
    ´
    static void d() throws LowLevelException {
       e();
    }
    static void e() throws LowLevelException {
        throw new LowLevelException();
    }
}

class HighLevelException extends Exception {
    HighLevelException(Throwable cause) { super(cause); }
}

class MidLevelException extends Exception {
    MidLevelException(Throwable cause)  { super(cause); }
}

class LowLevelException extends Exception {
}
```

As of release 7, the platform supports the notion of _suppressed exceptions_(in conjunction with the `try`-with-resources statement). Any exceptions that were suppressed in order to deliver an exception are printed out beneath the stack trace. The format of this information depends on the implementation, but the following example may be regarded as typical:

```
Exception in thread ""main"" java.lang.Exception: Something happened
at Foo.bar(Foo.java:10)
at Foo.main(Foo.java:5)
Suppressed: Resource$CloseFailException: Resource ID = 0
        at Resource.close(Resource.java:26)
        at Foo.bar(Foo.java:9)
```

Note that the ""... n more"" notation is used on suppressed exceptions just at it is used on causes. Unlike causes, suppressed exceptions are indented beyond their ""containing exceptions.""

An exception can have both a cause and one or more suppressed exceptions:

```
Exception in thread ""main"" java.lang.Exception: Main block at Foo3.main(Foo3.java:7)
```

```
Suppressed: Resource$CloseFailException: Resource ID = 2
        at Resource.close(Resource.java:26)
        at Foo3.main(Foo3.java:5)

Suppressed: Resource$CloseFailException: Resource ID = 1
        at Resource.close(Resource.java:26)
        at Foo3.main(Foo3.java:5)

Caused by: java.lang.Exception: I did it
        at Foo3.main(Foo3.java:8)
```

Likewise, a suppressed exception can have a cause:

```
Exception in thread ""main"" java.lang.Exception: Main block

at Foo4.main(Foo4.java:6)

Suppressed: Resource2$CloseFailException: Resource ID = 1
        at Resource2.close(Resource2.java:20)
        at Foo4.main(Foo4.java:5)

Caused by: java.lang.Exception: Rats, you caught me
        at Resource2$CloseFailException.<init>(Resource2.java:45)
```

18. Do you think the documentation mentioned above (Example-1) is Excessive Structured? *

Mark only one oval.

- ◯ Strongly agree
- ◯ Agree
- ◯ Neutral
- ◯ Disagree
- ◯ Strongly disagree

4. Excessive Structural Info

Definition: Documentations with description that contains too many structural syntax or structure. For example, the Javadoc of the java.lang.Object class Javadoc lists all the hundreds of subclasses of the class.

19. Based on your experience of the last three months, how frequently did you observe the excessive structured documentation? *

Mark only one oval.

- ◯ Never
- ◯ Once or twice
- ◯ Occasionally
- ◯ Frequently
- ◯ No opinion

20. How severe was the excessive structured documentation problem to complete your development task, when you last observed it? *

Mark only one oval.

- ◯ Not a Problem
- ◯ Moderate (kind of irritating)
- ◯ Severe (I wasted a lot of time on this but figured it out)
- ◯ Blocker (I could not get past it and picked another api)
- ◯ No opinion

21. Any comment about your experience with the excessive structured documentation problems?

_____

_____

_____

_____

_____

5. Fragmented Documentation

Definition: Documentations where the information of documentation (related to an API element) is scattered over too many pages or sections.

Example-1

Method Prototype:
public void setMinimum(int n)

Documentation:
Sets the progress bar's minimum value (stored in the progress bar's data model) to `n`.

The data model (a `BoundedRangeModel` instance) handles any mathematical issues arising from assigning faulty values. See the `BoundedRangeModel` documentation for details.

If the minimum value is different from the previous minimum, all change listeners are notified.

Parameters: `n` \- the new minimum

22. Do you think the documentation mentioned above (Example-1) is Fragmented? *

Mark only one oval.

- Strongly agree
- Agree
- Neutral
- Disagree
- Strongly disagree

5. Fragmented Documentation

Definition: Documentations where the information of documentation (related to an API element) is scattered over too many pages or sections.

23. Based on your experience of the last three months, how frequently did you observe the fragmented documentation? *

Mark only one oval.

- Never
- Once or twice
- Occasionally
- Frequently
- No opinion

24. How severe was the fragmented documentation problem to complete your development task, when you last observed it? *

Mark only one oval.

- ◯ Not a Problem
- ◯ Moderate (kind of irritating)
- ◯ Severe (I wasted a lot of time on this but figured it out)
- ◯ Blocker (I could not get past it and picked another api)
- ◯ No opinion

25. Any comment about your experience with the fragmented documentation problems?

_____

_____

_____

_____

_____

Suggestion

26. Do you think the documentation smells hinder the productivity of software developers?

Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Maybe

27. Do you think these documentation smells should be avoided to improve the software development process?

Mark only one oval.

- Yes
- No
- Maybe

28. Which documentation smell you face the most?

Mark only one oval.

- Lazy
- Bloated
- Tangled
- Excessive Structured
- Fragmented

29. Do you face any documentation smell other than these documentation smells? If yes, would you please describe it briefly?

Email

30. Please provide your email if you want to get a summary of the survey responses.