1 Example

1.1 Summary

```
Ground truth doc:  We precompute the AttributeDefinition of the target attribute in order to being able to use much
Synth origin doc:  faster implementations are provided instead to implement additional and more sophisticated or intuitive or useful but
Synth refact doc:  more details here and above and for creating or updating elements are two or fewer than
```

1.2 Original

```
public void setAttributeDefinitions(final AttributeDefinitions attributeDefinitions) {
  Validate.notNull(attributeDefinitions, "Attribute Definitions cannot be null");
  // We precompute the AttributeDefinition of the target attribute in order to being able to use much
  // faster methods for setting/replacing attributes on the ElementAttributes implementation
  this.targetAttributeDefinition = attributeDefinitions.forName(TEMPLATE_MODE, TARGET_ATTR_NAME);
}
```

1.3 Synthetic

```
public void setAttributeDefinitions(final AttributeDefinitions attributeDefinitions) {
  Validate.notNull(attributeDefinitions, "Attribute Definitions cannot be null");
  // faster implementations are provided instead to implement additional and more sophisticated or intuitive or useful
  // faster methods for setting/replacing attributes on the ElementAttributes implementation
  this.targetAttributeDefinition = attributeDefinitions.forName(TEMPLATE_MODE, TARGET_ATTR_NAME);
}
```

1.4 Variant

```
public void setAttributeDefinitions(final AttributeDefinitions sensualizeDistinctness) {
  Validate.notNull(sensualizeDistinctness, "Attribute Definitions cannot be null");
  //  more details here and above and for creating or updating elements are two or fewer than
  // faster methods for setting/replacing attributes on the ElementAttributes implementation
  this.targetAttributeDefinition = sensualizeDistinctness.forName(TEMPLATE_MODE, TARGET_ATTR_NAME);
}
```

1.5 Comment

After renaming `attributeDefinitions` to `sensualizeDistinctness`. The comment does not make sense.

1.6 Discrepancy

```
ROUGE score before refactoring: 0.3422913719943423
ROUGE score after refactoring: 0.3352192362093352
Relative difference: 0.021097046413502178
Put 0.021097046413502178 in (9, renameTokens)
```

2 Example

2.1 Summary

```
Ground truth doc:  We precompute the AttributeDefinition of the target attribute in order to being able to use much
Synth origin doc:  faster implementations are provided instead to implement additional and more sophisticated or intuitive or useful but
Synth refact doc:  Helped in subclass methods for performance testing or by writing the below comments for adding
```

2.2 Original

```java
public void setAttributeDefinitions(final AttributeDefinitions attributeDefinitions) {
    Validate.notNull(attributeDefinitions, "Attribute Definitions cannot be null");
    // We precompute the AttributeDefinition of the target attribute in order to being able to use much
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = attributeDefinitions.forName(TEMPLATE_MODE, TARGET_ATTR_NAME);
}
```

2.3 Synthetic

```java
public void setAttributeDefinitions(final AttributeDefinitions attributeDefinitions) {
    Validate.notNull(attributeDefinitions, "Attribute Definitions cannot be null");
    // faster implementations are provided instead to implement additional and more sophisticated or intuitive or useful but
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = attributeDefinitions.forName(TEMPLATE_MODE, TARGET_ATTR_NAME);
}
```

2.4 Variant

```java
public void setAttributeDefinitions(final AttributeDefinitions externalizeRedefinition) {
    Validate.notNull(externalizeRedefinition, "Attribute Definitions cannot be null");
    // Helped in subclass methods for performance testing or by writing the below comments for adding
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = externalizeRedefinition.forName(TEMPLATE_MODE, TARGET_ATTR_NAME);
}
```

2.5 Comment

After renaming `attributeDefinitions` to `externalizeRedefinition`, the comment changes. The comment does not make sense.

2.6 Discrepancy

```
ROUGE score before refactoring: 0.3422913719943423
ROUGE score after refactoring: 0.15841584158415842
Relative difference: 1.1607142857142856
Put 1.1607142857142856 in (9, renameTokens)
```

3 Example

3.1 Summary

```
Ground truth doc:  We precompute the AttributeDefinition of the target attribute in order to being able to use much
Synth origin doc:  faster implementations are required for the generation but cannot override or replace attributes instead of just
Synth refact doc:  faster initialization methods and then call several different and distinct values for this definition and the
```

3.2 Original

```
public void setAttributeDefinitions(final AttributeDefinitions attributeDefinitions) {
    Validate.notNull(attributeDefinitions, "Attribute Definitions cannot be null");
    // We precompute the AttributeDefinition of the target attribute in order to being able to use much
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = attributeDefinitions.forName(getTemplateMode(), this.targetAttrCompleteName);
}
```

3.3 Synthetic

```
public void setAttributeDefinitions(final AttributeDefinitions attributeDefinitions) {
    Validate.notNull(attributeDefinitions, "Attribute Definitions cannot be null");
    // faster implementations are required for the generation but cannot override or replace attributes instead of just
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = attributeDefinitions.forName(getTemplateMode(), this.targetAttrCompleteName);
}
```

3.4 Variant

```
public void setAttributeDefinitions(final AttributeDefinitions assignDistinctness) {
    Validate.notNull(assignDistinctness, "Attribute Definitions cannot be null");
    // faster initialization methods and then call several different and distinct values for this definition and the
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = assignDistinctness.forName(getTemplateMode(), this.targetAttrCompleteName);
}
```

3.5 Comment

After renaming `attributeDefinitions` to `assignDistinctness`, the comment changes. The comment does not make sense.

3.6 Discrepancy

```
ROUGE score before refactoring: 0.4200848656294201
ROUGE score after refactoring: 0.056577086280056574
Relative difference: 6.425000000000001
Put 6.425000000000001 in (11, renameTokens)
```

4 Example

4.1 Summary

```
Ground truth doc:  We precompute the AttributeDefinition of the target attribute in order to being able to use much
Synth origin doc:  faster implementations are required for the generation but cannot override or replace attributes instead of just
Synth refact doc:  convenience calls are replaced to support non standard methods but with only support on standard objects
```

4.2 Original

```
public void setAttributeDefinitions(final AttributeDefinitions attributeDefinitions) {
    Validate.notNull(attributeDefinitions, "Attribute Definitions cannot be null");
    // We precompute the AttributeDefinition of the target attribute in order to being able to use much
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = attributeDefinitions.forName(getTemplateMode(), this.targetAttrCompleteName);
}
```

4.3 Synthetic

```
public void setAttributeDefinitions(final AttributeDefinitions attributeDefinitions) {
    Validate.notNull(attributeDefinitions, "Attribute Definitions cannot be null");
    // faster implementations are required for the generation but cannot override or replace attributes instead of just
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = attributeDefinitions.forName(getTemplateMode(), this.targetAttrCompleteName);
}
```

4.4 Variant

```
public void setAttributeDefinitions(final AttributeDefinitions evaluateSharpness) {
    Validate.notNull(evaluateSharpness, "Attribute Definitions cannot be null");
    // convenience calls are replaced to support non standard methods but with only support on standard objects
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = evaluateSharpness.forName(getTemplateMode(), this.targetAttrCompleteName);
}
```

4.5 Comment

After renaming `attributeDefinitions` to `evaluateSharpness`, the comment changes. The comment does not make sense.

4.6 Discrepancy

```
ROUGE score before refactoring: 0.4200848656294201
ROUGE score after refactoring: 0.37057991513437055
Relative difference: 0.13358778625954212
Put 0.13358778625954212 in (11, renameTokens)
```

5 Example

5.1 Summary

```
Ground truth doc:  We precompute the AttributeDefinition of the target attribute in order to being able to use much
Synth origin doc:  faster implementations are required for the generation but cannot override or replace attributes instead of just
Synth refact doc:  faster implementations or also use these are preferred for all other or even using other than
```

5.2 Original

```
public void setAttributeDefinitions(final AttributeDefinitions attributeDefinitions) {
    Validate.notNull(attributeDefinitions, "Attribute Definitions cannot be null");
    // We precompute the AttributeDefinition of the target attribute in order to being able to use much
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = attributeDefinitions.forName(getTemplateMode(), this.targetAttrCompleteName);
}
```

5.3 Synthetic

```
public void setAttributeDefinitions(final AttributeDefinitions attributeDefinitions) {
    Validate.notNull(attributeDefinitions, "Attribute Definitions cannot be null");
    // faster implementations are required for the generation but cannot override or replace attributes instead of just
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = attributeDefinitions.forName(getTemplateMode(), this.targetAttrCompleteName);
}
```

5.4 Variant

```
public void setAttributeDefinitions(final AttributeDefinitions attributeDefinitions) {
    Validate.notNull( "Attribute Definitions cannot be null",attributeDefinitions);
    // faster implementations or also use these are preferred for all other or even using other than
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = attributeDefinitions.forName( this.targetAttrCompleteName,getTemplateMode());
}
```

5.5 Comment

After swapping the argument order, the comment changes. The comment does not make sense.

5.6 Discrepancy

```
ROUGE score before refactoring: 0.4200848656294201
ROUGE score after refactoring: 0.43847241867043846
Relative difference: -0.041935483870967676
Put -0.041935483870967676 in (11, permuteArgumentOrder)
```

6 Example

6.1 Summary

```
Ground truth doc:  char[] are mutable but this is not an issue as this class is package-protected and the code from
Synth origin doc:  Note the thread in process the callbacks are for which event listeners and thus that it identifies a
Synth refact doc:  Note the context change occurs wherever a call or action from this instance which originated within a transaction instance
```

6.2 Original

```
public void startEvent(final char[] id, final char[] event) {
    // char[] are mutable but this is not an issue as this class is package-protected and the code from
    // which this method is called is under control
    this.newEvent = true;
    this.id = id;
    this.event = event;
}
```

6.3 Synthetic

```
public void startEvent(final char[] id, final char[] event) {
    // Note the thread in process the callbacks are for which event listeners and thus that it identifies a
    // which this method is called is under control
    this.newEvent = true;
    this.id = id;
    this.event = event;
}
```

6.4 Variant

```
public void startEvent(final char[] id, final char[] response) {
    // Note the context change occurs wherever a call or action from this instance which originated within a transaction instance
    // which this method is called is under control
    this.newEvent = true;
    this.id = id;
    this.response = response;
}
```

6.5 Comment

After renaming `event` to `response`, the comment changes. The comment does not make sense.

6.6 Discrepancy

```
ROUGE score before refactoring: 0.36507936507936506
ROUGE score after refactoring: 0.11507936507936507
Relative difference: 2.1724137931034484
Put 2.1724137931034484 in (10, renameTokens)
```

7 Example

7.1 Summary

```
Ground truth doc:  char[] are mutable but this is not an issue as this class is package-protected and the code from
Synth origin doc:  Note the thread in process the callbacks are for which event listeners and thus that it identifies a
Synth refact doc:  Note the thread in process the call stack from to here and which frame is being tracked are which
```

7.2 Original

```
public void startEvent(final char[] id, final char[] event) {
    // char[] are mutable but this is not an issue as this class is package-protected and the code from
    // which this method is called is under control
    this.newEvent = true;
    this.id = id;
    this.event = event;
}
```

7.3 Synthetic

```
public void startEvent(final char[] id, final char[] event) {
    // Note the thread in process the callbacks are for which event listeners and thus that it identifies a
    // which this method is called is under control
    this.newEvent = true;
    this.id = id;
    this.event = event;
}
```

7.4 Variant

```
public void startEvent(final char[] event,final char[] id) {
    // Note the thread in process the call stack from to here and which frame is being tracked are which
    // which this method is called is under control
    this.newEvent = true;
    this.id = id;
    this.event = event;
}
```

7.5 Comment

After swapping the argument order, the comment changes. The comment does not make sense.

7.6 Discrepancy

```
ROUGE score before refactoring: 0.36507936507936506
ROUGE score after refactoring: 0.3373015873015873
Relative difference: 0.08235294117647045
Put 0.08235294117647045 in (10, permuteArgumentOrder)
```

8 Example

8.1 Summary

```
Ground truth doc:  char[] are mutable but this is not an issue as this class is package-protected and the code from
Synth origin doc:  Note the thread in process the callbacks are for which event listeners and thus that it identifies a
Synth refact doc:  this object and it parent objects and any children are hidden if a parent control the element whose object
```

8.2 Original

```
public void startEvent(final char[] id, final char[] event) {
  // char[] are mutable but this is not an issue as this class is package-protected and the code from
  // which this method is called is under control
  this.newEvent = true;
  this.id = id;
  this.event = event;
}
```

8.3 Synthetic

```
public void startEvent(final char[] id, final char[] event) {
  // Note the thread in process the callbacks are for which event listeners and thus that it identifies a
  // which this method is called is under control
  this.newEvent = true;
  this.id = id;
  this.event = event;
}
```

8.4 Variant

```
public void startEvent(final char[] id, final char[] event) {
  // this object and it parent objects and any children are hidden if a parent control the element whose object
  // which this method is called is under control
  this.newEvent = true;

  this.id = id;
  this.event = event;
}
```

8.5 Comment

After adding extra whitespace, the comment changes. The comment does not make sense.

8.6 Discrepancy

```
ROUGE score before refactoring: 0.36507936507936506
ROUGE score after refactoring: 0.3412698412698413
Relative difference: 0.06976744186046503
Put 0.06976744186046503 in (10, addWhitespace)
```

9 Example

9.1 Summary

```
Ground truth doc:  No need to control overflow here. The fact that this has overflow will be used as a flag to determine
Synth origin doc:  Note here is an exception that may result when the implementation checks whether and if the operation succeeds to
Synth refact doc:  Make assumption that we already determined here is the caller of it and also know now we already determine
```

9.2 Original

```
@Override
public void flush() throws IOException {
  // No need to control overflow here. The fact that this has overflow will be used as a flag to determine
  // that further write operations are actually needed by means of the isOverflown() method.
  this.os.flush();
}
```

9.3 Synthetic

```
@Override
public void flush() throws IOException {
  // Note here is an exception that may result when the implementation checks whether and if the operation succeeds to
  // that further write operations are actually needed by means of the isOverflown() method.
  this.os.flush();
}
```

9.4 Variant

```
@Override
public void level() throws IOException {
  // Make assumption that we already determined here is the caller of it and also know now we already determine
  // that further write operations are actually needed by means of the isOverflown() method.
  this.os.level();
}
```

9.5 Comment

After renaming `flush` to `level`, the comment changes. The comment does not make sense.

9.6 Discrepancy

```
ROUGE score before refactoring: 0.3645320197044335
ROUGE score after refactoring: 0.45689655172413796
Relative difference: -0.20215633423180598
Put -0.20215633423180598 in (9, renameTokens)
```

10 Example

10.1 Summary

```
Ground truth doc:  No need to control overflow here. The fact that this has overflow will be used as a flag to determine
Synth origin doc:  Note here is an exception that may result when the implementation checks whether and if the operation succeeds to
Synth refact doc:  Note here is an assertion that would tell that this writer knows now as is in case we detect
```

10.2 Original

```
@Override
public void flush() throws IOException {
  // No need to control overflow here. The fact that this has overflow will be used as a flag to determine
  // that further write operations are actually needed by means of the isOverflown() method.
  this.os.flush();
}
```

10.3 Synthetic

```
@Override
public void flush() throws IOException {
  // Note here is an exception that may result when the implementation checks whether and if the operation succeeds to
  // that further write operations are actually needed by means of the isOverflown() method.
  this.os.flush();
}
```

10.4 Variant

```
public void flush() throws IOException {
  @Override
  // Note here is an assertion that would tell that this writer knows now as is in case we detect
  // that further write operations are actually needed by means of the isOverflown() method.
  this.os.flush();
}
```

10.5 Comment

After reordering lines, the comment changes. The reordering is invalid, because `@Override` is a method annotation.

10.6 Discrepancy

```
ROUGE score before refactoring: 0.3645320197044335
ROUGE score after refactoring: 0.35467980295566504
Relative difference: 0.02777777777777774
Put 0.02777777777777774 in (9, swapMultilineNoDeps)
```

11 Example

11.1 Summary

```
Ground truth doc:  No need to control overflow here. The fact that this has overflow will be used as a flag to determine
Synth origin doc:  Note here is an exception that may result when the writer checks whether there exist problems or otherwise to
Synth refact doc:  Make assumption to ensure there s an even size file and if required then it seems the user will
```

11.2 Original

```
@Override
public void flush() throws IOException {
  // No need to control overflow here. The fact that this has overflow will be used as a flag to determine
  // that further write operations are actually needed by means of the isOverflown() method.
  this.writer.flush();
}
```

11.3 Synthetic

```
@Override
public void flush() throws IOException {
  // Note here is an exception that may result when the writer checks whether there exist problems or otherwise to
  // that further write operations are actually needed by means of the isOverflown() method.
  this.writer.flush();
}
```

11.4 Variant

```
@Override
public void kick() throws IOException {
  // Make assumption to ensure there s an even size file and if required then it seems the user will
  // that further write operations are actually needed by means of the isOverflown() method.
  this.writer.kick();
}
```

11.5 Comment

After renaming `flush` to `kick`, the comment changes. The comment does not make sense.

11.6 Discrepancy

```
ROUGE score before refactoring: 0.35714285714285715
ROUGE score after refactoring: 0.18719211822660098
Relative difference: 0.9078947368421054
Put 0.9078947368421054 in (9, renameTokens)
```

12 Example

12.1 Summary

```
Ground truth doc:  release memory
Synth origin doc:  callers
Synth refact doc:  call listeners
```

12.2 Original

```
@Override
protected void onDestroy() {
  super.onDestroy();
  // release memory
  if (mRecyclerView != null) {
    mRecyclerView.destroy();
    mRecyclerView = null;
  }
}
```

12.3 Synthetic

```
@Override
protected void onDestroy() {
  super.onDestroy();
  // callers
  if (mRecyclerView != null) {
    mRecyclerView.destroy();
    mRecyclerView = null;
  }
}
```

12.4 Variant

```
@Override
protected void onDestroy() {
  super.onDestroy();
  // call listeners
  if (86RecyclerReconsider != null) {
    86RecyclerReconsider.destroy();
    86RecyclerReconsider = null;
  }
}
```

12.5 Comment

After renaming `mRecyclerView` to `86RecyclerReconsider`. This renaming is invalid because Java identifiers cannot start with a number.

12.6 Discrepancy

```
ROUGE score before refactoring: 0.0
ROUGE score after refactoring: 0.017964071856287425
Relative difference: -1.0
Put -1.0 in (13, renameTokens)
```