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

```
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

```
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

```
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: 0.027777777777777774
Put 0.027777777777777774 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)
```

13 Example

13.1 Summary

```
Ground truth doc:  This method is meant to be overriden. By default, no local variables
Synth origin doc:   this parameter before any attribute local values before tag parsing is executed is not
Synth refact doc:   The value to the given expression as map key if found then value as
```

13.2 Original

```
protected Map<String, Object> computeAdditionalLocalVariables(final ITemplateContext context, final IProcessableElementTag tag, final Attribut
    // This method is meant to be overriden. By default, no local variables
    // will be set.
    return null;
}
```

13.3 Synthetic

```
protected Map<String, Object> computeAdditionalLocalVariables(final ITemplateContext context, final IProcessableElementTag tag, final Attribut
    //  this parameter before any attribute local values before tag parsing is executed is not
    // will be set.
    return null;
}
```

13.4 Variant

```
protected Map<String, Object> resolveAccumulativeTopicalVector(final ITemplateContext context, final IProcessableElementTag tag, final AttributeNa
    //  The value to the given expression as map key if found then value as
    // will be set.
    return null;
}
```

13.5 Comment

TODO.

13.6 Discrepancy

```
Rouge score before refactoring: 0.7447447447447447
Rouge score after refactoring: 0.10810810810810811
Relative difference: 5.888888888888888
```

Put 5.8888 in (complexity=6, SCT=renameTokens)

14 Example

14.1 Summary

```
Ground truth doc:  This method is meant to be overriden. By default, no local variables
Synth origin doc:   this parameter before any attribute local values before tag parsing is executed is not
Synth refact doc:   this parameter should always exist since not more than this variable is expected to
```

14.2 Original

```
protected Map<String, Object> computeAdditionalLocalVariables(final ITemplateContext context, final IProcessableElementTag tag, final Attribut
    // This method is meant to be overriden. By default, no local variables
    // will be set.
    return null;
}
```

14.3 Synthetic

```
protected Map<String, Object> computeAdditionalLocalVariables(final ITemplateContext context, final IProcessableElementTag tag, final Attribut
    //  this parameter before any attribute local values before tag parsing is executed is not
    // will be set.
    return null;
}
```

14.4 Variant

```
protected Map<String, Object> computeAdditionalLocalVariables( final AttributeName attributeName, final IStandardExpression ex
pression,final ITemplateContext context, final IProcessableElementTag tag, final String attributeValue) {
    //  this parameter should always exist since not more than this variable is expected to
    // will be set.
    return null;
}
```

14.5 Comment

TODO.

14.6 Discrepancy

```
Rouge score before refactoring: 0.7447447447447447
Rouge score after refactoring: 0.7417417417417418
Relative difference: 0.004048582995951325
```

Put 0.0040 in (complexity=6, SCT=permuteArgumentOrder)

15 Example

15.1 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:   also use this as alternative or add to a group with other groups for performance in
```

15.2 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);
}
```

15.3 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 but
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = attributeDefinitions.forName(TEMPLATE_MODE, TARGET_ATTR_NAME);
}
```

15.4 Variant

```
public void setAttributeDefinitions(final AttributeDefinitions judgeExplanation) {
    Validate.notNull(judgeExplanation, "Attribute Definitions cannot be null");
    //  also use this as alternative or add to a group with other groups for performance in
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = judgeExplanation.forName(TEMPLATE_MODE, TARGET_ATTR_NAME);
}
```

15.5 Comment

TODO.

15.6 Discrepancy

```
Rouge score before refactoring: 0.3422913719943423
Rouge score after refactoring: 0.24328147100424327
Relative difference: 0.40697674418604657
```

Put 0.4069 in (complexity=9, SCT=renameTokens)

16 Example

16.1 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:   slower versions use these two versions as opposedto using this class which also defines more
```

16.2 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);
}
```

16.3 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 but
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = attributeDefinitions.forName(TEMPLATE_MODE, TARGET_ATTR_NAME);
}
```

16.4 Variant

```
public void setAttributeDefinitions(final AttributeDefinitions ascribeExplanation) {
    Validate.notNull(ascribeExplanation, "Attribute Definitions cannot be null");
    //  slower versions use these two versions as opposedto using this class which also defines more
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = ascribeExplanation.forName(TEMPLATE_MODE, TARGET_ATTR_NAME);
}
```

16.5 Comment

TODO.

16.6 Discrepancy

```
Rouge score before refactoring: 0.3422913719943423
Rouge score after refactoring: 0.14002828854314003
Relative difference: 1.4444444444444444
```

Put 1.4444 in (complexity=9, SCT=renameTokens)

17 Example

17.1 Summary

```
Ground truth doc:  We precompute the AttributeDefinition of the target attribute in order to being able to use much
Synth origin doc:   slower versions are implemented with these additional methods which call additional methods for avoiding duplicate values
Synth refact doc:   add some utility code or use these two and three to use generic types to allow
```

17.2 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, this.targetAttributeCompleteName);
}
```

17.3 Synthetic

```
public void setAttributeDefinitions(final AttributeDefinitions attributeDefinitions) {
    Validate.notNull(attributeDefinitions, "Attribute Definitions cannot be null");
    //  slower versions are implemented with these additional methods which call additional methods for avoiding duplicate values
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = attributeDefinitions.forName(TEMPLATE_MODE, this.targetAttributeCompleteName);
}
```

17.4 Variant

```
public void setAttributeDefinitions(final AttributeDefinitions anthropomorphiseExplanation) {
    Validate.notNull(anthropomorphiseExplanation, "Attribute Definitions cannot be null");
    //  add some utility code or use these two and three to use generic types to allow
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = anthropomorphiseExplanation.forName(TEMPLATE_MODE, this.targetAttributeCompleteName);
}
```

17.5 Comment

TODO.

17.6 Discrepancy

```
Rouge score before refactoring: 0.36633663366336633
Rouge score after refactoring: 0.16124469589816123
Relative difference: 1.2719298245614037
```

Put 1.2719 in (complexity=9, SCT=renameTokens)

18 Example

18.1 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:   helper functions for implementing a subclass and providing a lot to provide for simplicity and for
```

18.2 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);
}
```

18.3 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);
}
```

18.4 Variant

```
public void setAttributeDefinitions(final AttributeDefinitions assignSharpness) {
    Validate.notNull(assignSharpness, "Attribute Definitions cannot be null");
    //  helper functions for implementing a subclass and providing a lot to provide for simplicity and for
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = assignSharpness.forName(getTemplateMode(), this.targetAttrCompleteName);
}
```

18.5 Comment

TODO.

18.6 Discrepancy

```
Rouge score before refactoring: 0.4200848656294201
Rouge score after refactoring: 0.05233380480905234
Relative difference: 7.027027027027027
```

Put 7.0270 in (complexity=11, SCT=renameTokens)

19 Example

19.1 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:   slower implementations or also other slower or equivalent methods are encouraged and can instead invoke only
```

19.2 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);
}
```

19.3 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);
}
```

19.4 Variant

```
public void setAttributeDefinitions(final AttributeDefinitions attributeDefinitions) {
    Validate.notNull(attributeDefinitions, "Attribute Definitions cannot be null");
    //  slower implementations or also other slower or equivalent methods are encouraged and can instead invoke only
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = attributeDefinitions.forName( this.targetAttrCompleteName,getTemplateMode());
}
```

19.5 Comment

TODO.

19.6 Discrepancy

```
Rouge score before refactoring: 0.4200848656294201
Rouge score after refactoring: 0.3338048090523338
Relative difference: 0.2584745762711866
```

Put 0.2584 in (complexity=11, SCT=permuteArgumentOrder)

20 Example

20.1 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:   also call addOptions or apply options on it and addTo instead to implement any
```

20.2 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);
}
```

20.3 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);
}
```

20.4 Variant

```
public void setAttributeDefinitions(final AttributeDefinitions accreditAccount) {
    Validate.notNull(accreditAccount, "Attribute Definitions cannot be null");
    //  also call addOptions or apply options on it and addTo instead to implement any
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = accreditAccount.forName(getTemplateMode(), this.targetAttrCompleteName);
}
```

20.5 Comment

TODO.

20.6 Discrepancy

```
Rouge score before refactoring: 0.4200848656294201
Rouge score after refactoring: 0.11598302687411598
Relative difference: 2.6219512195121952
```

Put 2.6219 in (complexity=11, SCT=renameTokens)

21 Example

21.1 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:   helper functions to implement more specific behavior and other general purpose behavior to call in this
```

21.2 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);
}
```

21.3 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);
}
```

21.4 Variant

```
public void setAttributeDefinitions(final AttributeDefinitions attributeDefinitions) {
    Validate.notNull( "Attribute Definitions cannot be null",attributeDefinitions);
    //  helper functions to implement more specific behavior and other general purpose behavior to call in this
    // faster methods for setting/replacing attributes on the ElementAttributes implementation
    this.targetAttributeDefinition = attributeDefinitions.forName(getTemplateMode(), this.targetAttrCompleteName);
}
```

21.5 Comment

TODO.

21.6 Discrepancy

```
Rouge score before refactoring: 0.4200848656294201
Rouge score after refactoring: 0.1669024045261669
Relative difference: 1.516949152542373
```

Put 1.5169 in (complexity=11, SCT=permuteArgumentOrder)

22 Example

22.1 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 in that this method or context in other code is ignored since only that method called
```

22.2 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;
}
```

22.3 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;
}
```

22.4 Variant

```
public void startEvent(final char[] id, final char[] byproduct) {
    //  Note the context in that this method or context in other code is ignored since only that method called
    // which this method is called is under control
    this.newEvent = true;
    this.id = id;
    this.byproduct = byproduct;
}
```

22.5 Comment

TODO.

22.6 Discrepancy

```
Rouge score before refactoring: 0.36507936507936506
Rouge score after refactoring: 0.3531746031746032
Relative difference: 0.03370786516853921
```

Put 0.0337 in (complexity=10, SCT=renameTokens)

23 Example

23.1 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:   notifying os that this object will flush out data without notice and informing that we can detect when
```

23.2 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();
}
```

23.3 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();
}
```

23.4 Variant

```
@Override
public void flat-bottom() throws IOException {
    //  notifying os that this object will flush out data without notice and informing that we can detect when
    // that further write operations are actually needed by means of the isOverflown() method.
    this.os.flat-bottom();
}
```

23.5 Comment

TODO.

23.6 Discrepancy

```
Rouge score before refactoring: 0.3645320197044335
Rouge score after refactoring: 0.07758620689655173
Relative difference: 3.6984126984126986
```

Put 3.6984 in (complexity=9, SCT=renameTokens)

24 Example

24.1 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
```

24.2 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();
}
```

24.3 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();
}
```

24.4 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();
}
```

24.5 Comment

TODO.

24.6 Discrepancy

```
Rouge score before refactoring: 0.3645320197044335
Rouge score after refactoring: 0.35467980295566504
Relative difference: 0.027777777777777774
```

Put 0.0277 in (complexity=9, SCT=swapMultilineNoDeps)

25 Example

25.1 Summary

```
Ground truth doc:  Model inserted BEFORE can never be processable
Synth origin doc:   return nullableInstanceObjectsList
Synth refact doc:   return nullableInstanceBuilderInstanceObject
```

25.2 Original

```
public void insertBefore(final IModel model) {
    resetAllButVariablesOrAttributes();
    Validate.notNull(model, "Model cannot be null");
    this.insertBeforeModel = true;
    this.insertBeforeModelValue = model;
    // Model inserted BEFORE can never be processable
}
```

25.3 Synthetic

```
public void insertBefore(final IModel model) {
    resetAllButVariablesOrAttributes();
    Validate.notNull(model, "Model cannot be null");
    this.insertBeforeModel = true;
    this.insertBeforeModelValue = model;
    //  return nullableInstanceObjectsList
}
```

25.4 Variant

```
public void insertBefore(final IModel display) {
    resetAllButVariablesOrAttributes();
    Validate.notNull(display, "Model cannot be null");
    this.insertBeforeModel = true;
    this.insertBeforeModelValue = display;
    //  return nullableInstanceBuilderInstanceObject
}
```

25.5 Comment

TODO.

25.6 Discrepancy

```
Rouge score before refactoring: 0.049773755656108594
Rouge score after refactoring: 0.042986425339366516
Relative difference: 0.1578947368421052
```

Put 0.1578 in (complexity=8, SCT=renameTokens)

26 Example

26.1 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:   Note here is an exception because there isn't enough logic in it and thus no assumption made there
```

26.2 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();
}
```

26.3 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();
}
```

26.4 Variant

```
@Override
public void feed() throws IOException {
    //  Note here is an exception because there isn't enough logic in it and thus no assumption made there
    // that further write operations are actually needed by means of the isOverflown() method.
    this.writer.feed();
}
```

26.5 Comment

TODO.

26.6 Discrepancy

```
Rouge score before refactoring: 0.35714285714285715
Rouge score after refactoring: 0.3817733990147783
Relative difference: -0.064516129032258
```

Put -0.064 in (complexity=9, SCT=renameTokens)

27 Example

27.1 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:   Not a check of this implementation to check for any errors but also it seems useful here as to
```

27.2 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();
}
```

27.3 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();
}
```

27.4 Variant

```
public void flush() throws IOException {
Override
 // Not a check of this implementation to check for any errors but also it seems useful here as to
    // that further write operations are actually needed by means of the isOverflown() method.
    this.writer.flush();
}
```

27.5 Comment

TODO.

27.6 Discrepancy

```
Rouge score before refactoring: 0.35714285714285715
Rouge score after refactoring: 0.13669950738916256
Relative difference: 1.6126126126126126
```

Put 1.6126 in (complexity=9, SCT=swapMultilineNoDeps)