InvocMap Overview

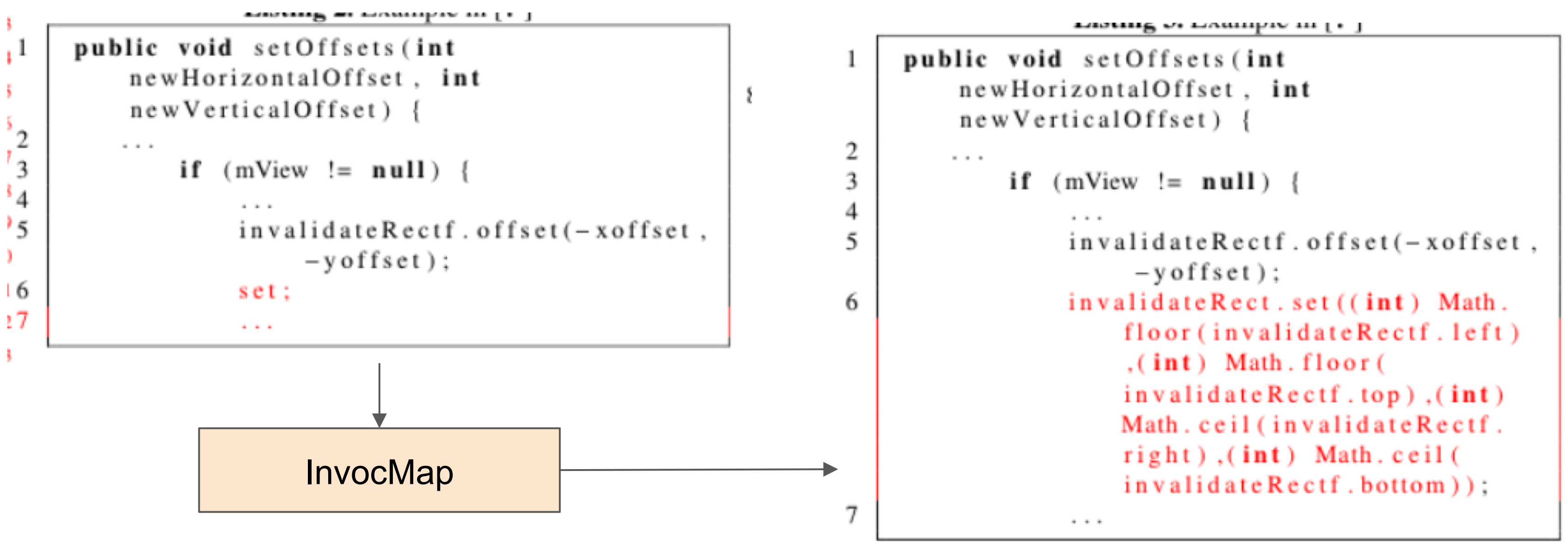
Overview

Implementing correct method invocation is an important task for software developers. However, this is challenging work, since the structure of method invocation can be complicated. In this paper, we propose InvocMap, a code completion tool allows developers to obtain an implementation of multiple method invocations from a list of method names inside code context. InvocMap is able to predict the nested method invocations which their names didn't appear in the list of input method names given by developers. To achieve this, we analyze the Method Invocations by four levels of abstraction. We build a Machine Translation engine to learn the mapping from the first level to the third level of abstraction of multiple method invocations, which only requires developers to manually add local variables from generated expression to get the final code. We evaluate our proposed approach on six popular libraries: JDK, Android, GWT, Joda-Time, Hibernate, and Xstream. With the training corpus of 2.86 million method invocations extracted from 1000 Java Github projects and the testing corpus extracted from 120 online forums code snippets, InvocMap achieves the accuracy rate up to 84 in F1- score depending on how much information of context provided along with method names, which outperforms the state-of-the-art Neural Machine Translation approach by 40% and shows its potential for auto code completion.

Input and Output

Input

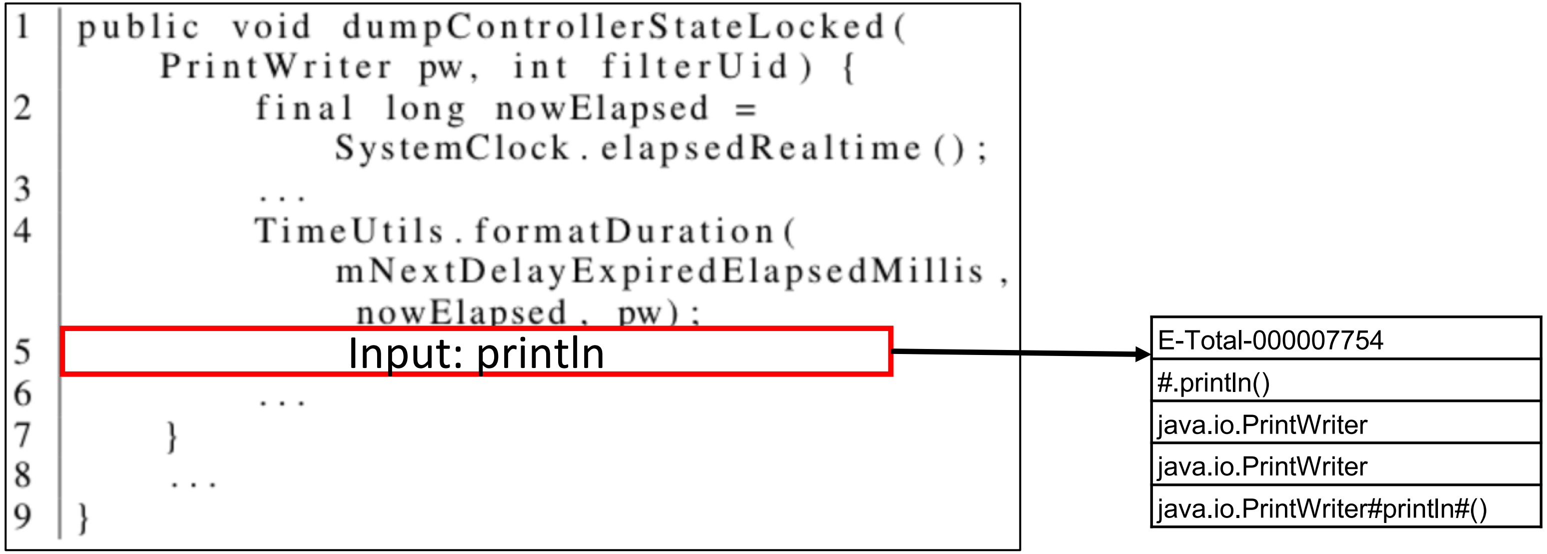
Output



```
public void setOffsets(int
    newHorizontalOffset, int
    newVerticalOffset) {
  ...
    if (mView != null) {
      ...
      invalidateRectf.offset(-xoffset,
          -yoffset);
      set;
      ...
```

```
public void setOffsets(int
    newHorizontalOffset, int
    newVerticalOffset) {
  ...
    if (mView != null) {
      ...
      invalidateRectf.offset(-xoffset,
          -yoffset);
      invalidateRect.set((int) Math.
          floor(invalidateRectf.left)
          ,(int) Math.floor(
          invalidateRectf.top),(int)
          Math.ceil(invalidateRectf.
          right),(int) Math.ceil(
          invalidateRectf.bottom));
      ...
```

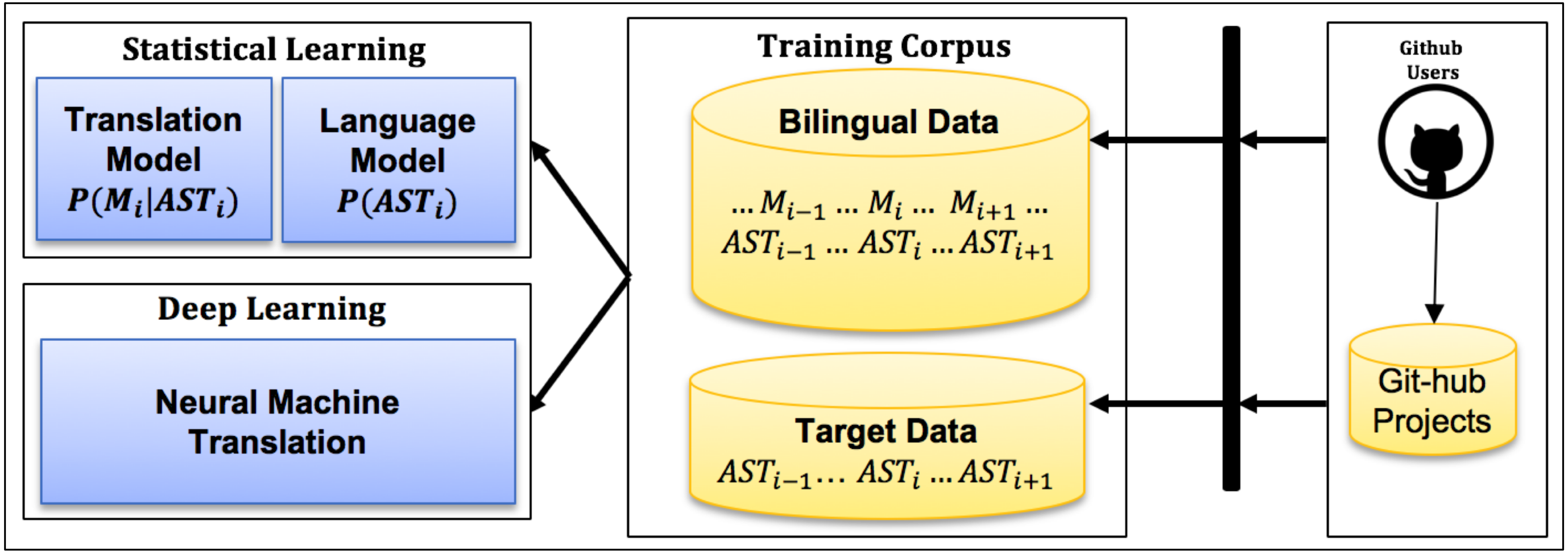
Input and Output (2)



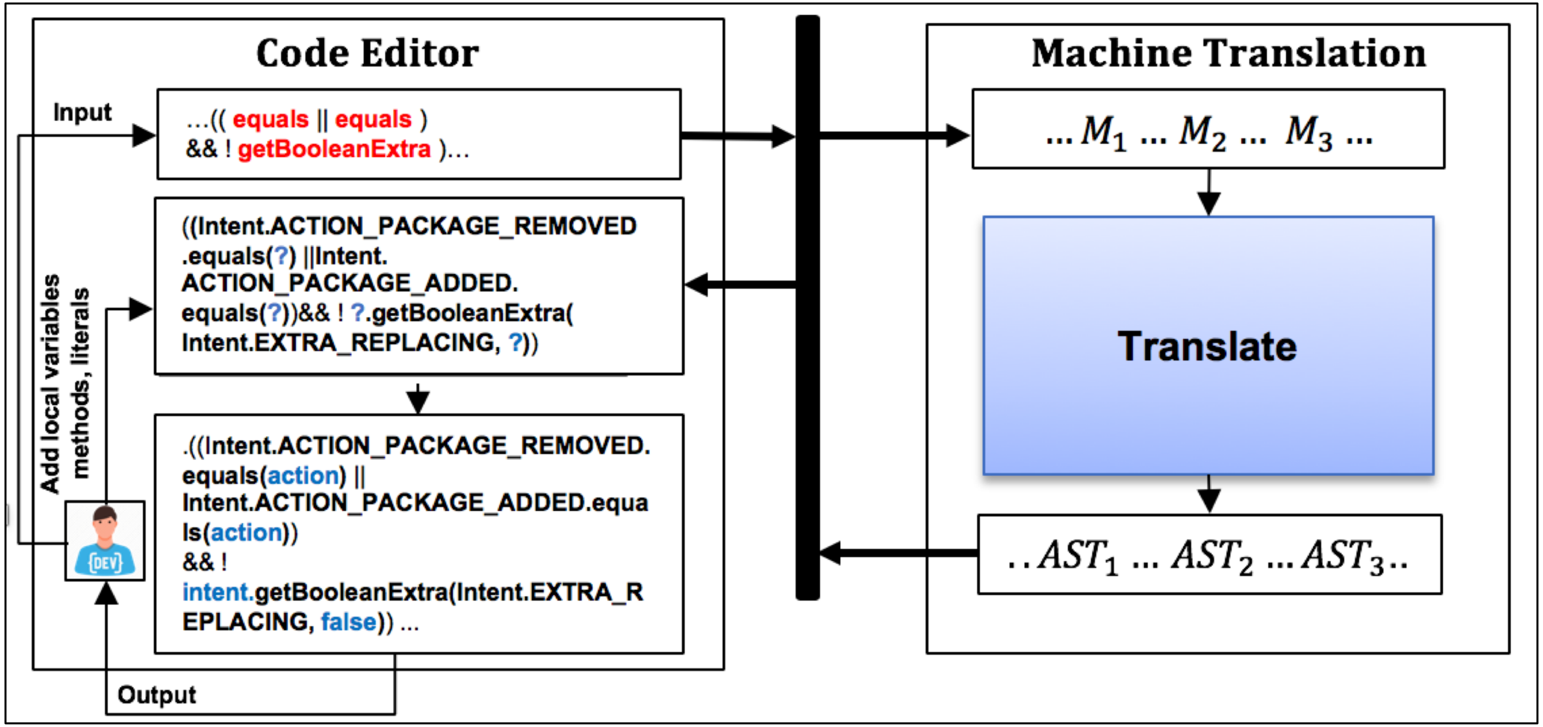
```
public void dumpControllerStateLocked(
    PrintWriter pw, int filterUid) {
        final long nowElapsed =
            SystemClock.elapsedRealtime();
        ...
        TimeUtils.formatDuration(
            mNextDelayExpiredElapsedMillis,
             nowElapsed, pw);
                    Input: println
        ...
    }
    ...
}
```

E-Total-000007754
#.println()
java.io.PrintWriter
java.io.PrintWriter
java.io.PrintWriter#println#()

InvocMap


Statistical Learning
Translation Model $P(M_i|AST_i)$
Language Model $P(AST_i)$
Deep Learning
Neural Machine Translation
Training Corpus
Bilingual Data
$\dots M_{i-1} \dots M_i \dots M_{i+1} \dots$
$AST_{i-1} \dots AST_i \dots AST_{i+1}$
Target Data
$AST_{i-1} \dots AST_i \dots AST_{i+1}$
Github Users
Git-hub Projects

InvocMap (2)



```
...(( equals || equals )
&& ! getBooleanExtra )...
```

```
((Intent.ACTION_PACKAGE_REMOVED
.equals(?) ||Intent.
ACTION_PACKAGE_ADDED.
equals(?))&& ! ?.getBooleanExtra(
Intent.EXTRA_REPLACING, ?))
```

```
.((Intent.ACTION_PACKAGE_REMOVED.
equals(action) ||
Intent.ACTION_PACKAGE_ADDED.equa
ls(action))
&& !
intent.getBooleanExtra(Intent.EXTRA_R
EPLACING, false)) ...
```

Evaluation: Accuracy of InvocMap by SMT

	Extrinsic Evaluation with Configuration 3								
Library	Correct	Incorrect	OOSource	OOTarget	OOVoc	Total	Precision	Recall	F1-Score
GWT	89	4	0	9	9	102	95.70%	90.82%	93.19%
Joda-Time	55	3	2	15	17	75	94.83%	76.39%	84.62%
JDK	174	32	0	44	44	250	84.47%	79.82%	82.08%
Android	82	11	0	13	13	106	88.17%	86.32%	87.23%
Hibernate	146	40	1	39	40	226	78.49%	78.49%	78.49%
Xstream	50	0	0	14	14	64	100.00%	78.13%	87.72%
Total	596	90	3	134	137	823	86.88%	81.31%	84.00%

Evaluation (2): Comparison between SMT and NMT

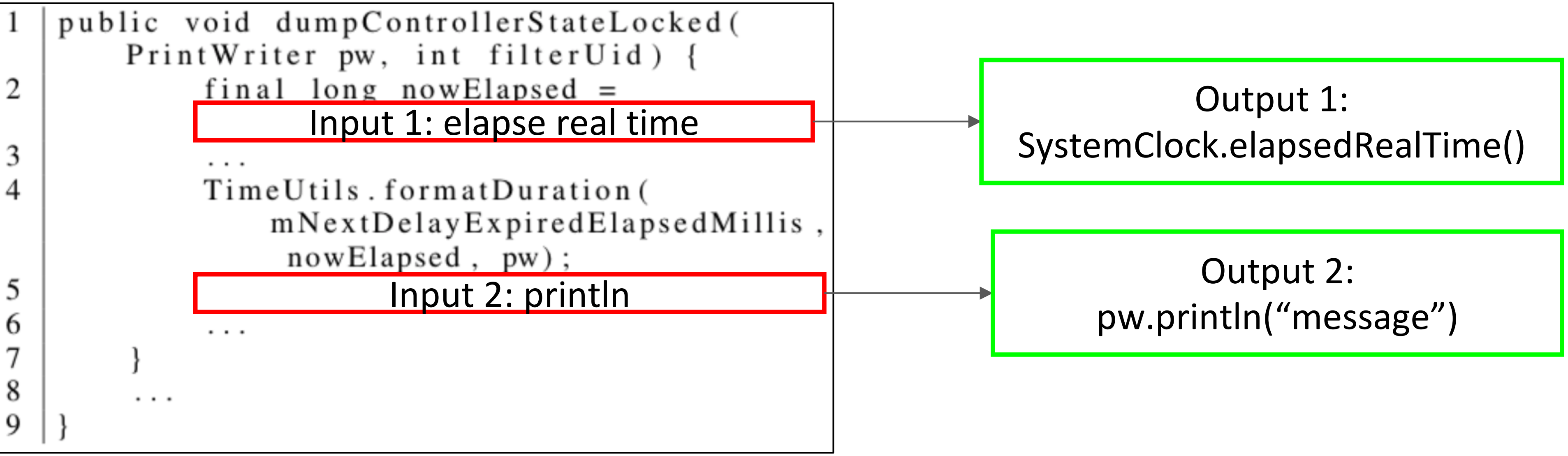
Configuration	Statistical Machine Translation			Neural Machine Translation		
Intrinsic	**Precision**	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
Configuration 1	70.43%	75.89%	73.06%	54.09%	46.18%	49.82%
Configuration 2	84.04%	78.98%	81.43%	46.77%	67.97%	55.41%
Configuration 3	89.33%	79.98%	84.40%	49.27%	69.09%	57.52%
Extrinsic	**Precision**	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
Configuration 1	62.54%	75.80%	68.53%	44.62%	49.71%	47.03%
Configuration 2	86.01%	81.16%	83.51%	33.38%	62.57%	43.54%
Configuration 3	86.88%	81.31%	84.00%	34.84%	63.56%	45.01%

Thank you

Appendix: Resolve multiple method names

Prior work: AnyCode [1]

Handle sequentially



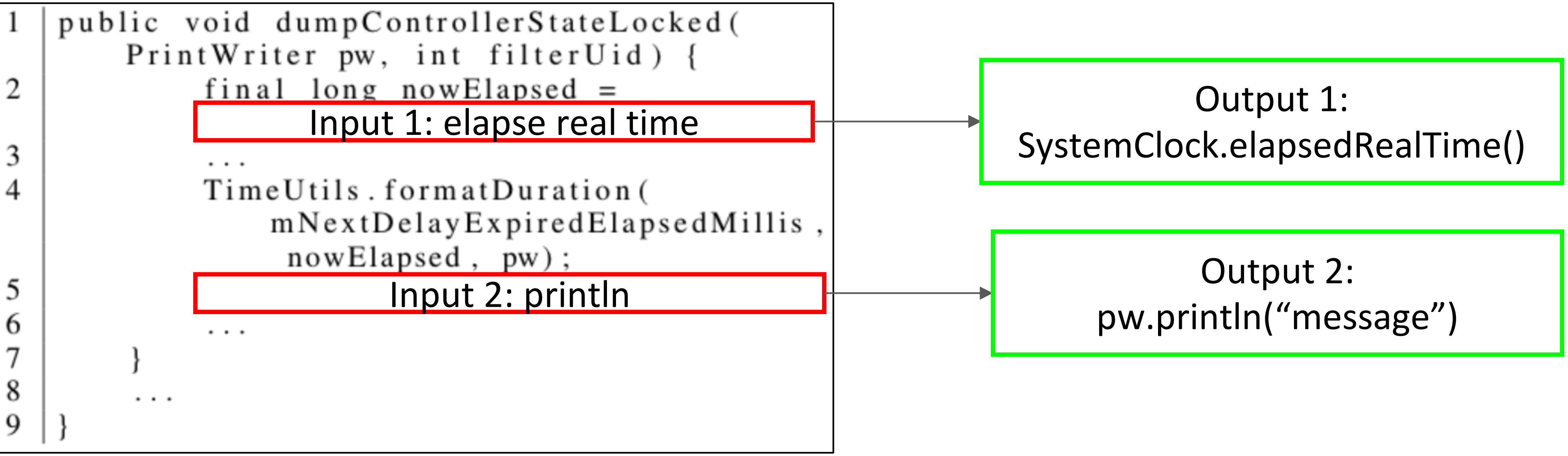
```
public void dumpControllerStateLocked(
    PrintWriter pw, int filterUid) {
        final long nowElapsed =
        ...
        TimeUtils.formatDuration(
            mNextDelayExpiredElapsedMillis,
             nowElapsed, pw);

        ...
    }
    ...
}
```

Input 1: elapse real time
Input 2: println
Output 1: SystemClock.elapsedRealTime()
Output 2: pw.println("message")

Appendix: Resolve multiple method names

InvocMap

Handle concurrently



```
public void dumpControllerStateLocked(
     PrintWriter pw, int filterUid) {
          final long nowElapsed =

          ...
          TimeUtils.formatDuration(
               mNextDelayExpiredElapsedMillis,
               nowElapsed, pw);

          ...
     }
     ...
}
```

Input 1: elapse real time
Output 1: SystemClock.elapsedRealTime()
Input 2: println
Output 2: pw.println("message")

References

1. AnyCode. https://lara.epfl.ch/~kuncak/papers/GveroKuncak15SynthesizingJavaExpressionsFreeFormQueries.pdf