Examples of code snippets and tool warnings

This document gives seven examples of code snippets used in our dataset: 1-2 snippets from each of the six prior datasets we leveraged. For those snippets, we also show warnings of different kinds produced by every tool we used in our study. Along with the examples, we present the results of our manual validation of warnings (for the full validation and results see the document “Tool warning analysis”).

dataset 1, snippet 8

2 warnings, both from **openjml**

lines 159-168 in Tasks.java.

simple-datasets > src > main > java > cog_complexity_validation_datasets > One > Tasks.java > Tasks > main9(String[])

```
159     public static void main8(String[] args) {
160         int number = 323;
161         int result = 0;
162
163         while (number!= 0) {
164             result = result + number % 10;
165             number = number / 10;
166         }
167         System.out.println(result);
168     }
```

cog_complexity_validation_datasets/One/Tasks.java:164: verify: The prover cannot establish an assertion (**ArithmeticOperationRange**) in method main8: overflow in int sum

```
        result = result + number % 10;
                    ^
```

cog_complexity_validation_datasets/One/Tasks.java:164: verify: The prover cannot establish an assertion (**ArithmeticOperationRange**) in method main8: underflow in int sum

```
        result = result + number % 10;
                    ^
```

Both are false positives, since the loop is bounded and over/underflow never occurs.

VERIFIED.

dataset 2, snippet 8

3 warnings, all from the typestate checker (JaTyC).

This snippet is lines 244-252 in Tasks.java.

simple-datasets > src > main > java > cog_complexity_validation_datasets > One > Tasks.java > Tasks > main9(String[])

```
244     public static void main14(String[] args) {
245         String word = "Hello";
246         String result = new String();
247
248         for ( int j = word.length() - 1; j >= 0; j-- )
249             result += word.charAt(j);
250
251         System.out.println(word);
252     }
```

simple-datasets/src/main/java/cog_complexity_validation_datasets/One/Tasks.java:249: warning: **Cannot assign: cannot cast** from Unknown to Shared{java.lang.String} | Null
        result += word.charAt(j);

Clearly a false positive, since chars cannot be null.

simple-datasets/src/main/java/cog_complexity_validation_datasets/One/Tasks.java:244: warning: [result[85] StringConcat ToString charAt(word, j)] **did not complete its protocol** (found: Unknown)
  public static void main14(String[] args) {
                ^

simple-datasets/src/main/java/cog_complexity_validation_datasets/One/Tasks.java:244: warning: [result[85] = result[85] StringConcat ToString charAt(word, j)] **did not complete its protocol** (found: Unknown)
  public static void main14(String[] args) {
                ^

These two errors are a bit nonsensical, since I'm not sure to what they're referring. The two are slightly different (note the "result[85]" vs "result[85] = result[85]"). In general, the TS checker issues this error when a protocol isn't finished (e.g., when you forget to close a stream). I'm not sure what protocol it's trying to enforce here, but it's definitely a false positive.

dataset 3, snippet 30

5 warnings:

- 1 from openjml
- 4 from the typestate checker

lines 902-908 in Tasks_1 (Note: the comments are the original ones or were added by Munoz et al.)

simple-datasets > src > main > java > cog_complexity_validation_datasets > Three > Tasks_1.java > Tasks_1 > s31()

```
896     /**
897      * Sets the currently chosen <code>MapTransform</code>.
898      * @param mt The transform that should be applied to a
899      *      <code>Tile</code> that is clicked on the map.
900      */
901     //SNIPPET_STARTS
902     public void setMapTransform(MapTransform mt) {
903         currentMapTransform = mt;
904         MapControlsAction mca = (MapControlsAction) freeColClient.getActionManager().getFreeColAction
            (MapControlsAction.ID);
905         if (mca.getMapControls() != null) {
906             mca.getMapControls().update(mt);
907         } // Added to allow compilation
908     } // Added to allow compilation
```

openjml:

cog_complexity_validation_datasets/Three/Tasks_1.java:904: verify: The prover cannot establish an assertion (**PossiblyBadCast**) in method setMapTransform: a java.lang.Object cannot be proved to be a cog_complexity_validation_datasets.Three.MapControlsAction

```
    MapControlsAction mca = (MapControlsAction) freeColClient.getActionManager().getFreeColAction(MapControlsAction.ID);
                            ^
```

Redundant with the javac cast, so this is clearly intentional and therefore an FP.

typestate:

simple-datasets/src/main/java/cog_complexity_validation_datasets/Three/Tasks_1.java:903: warning: Cannot assign because [this.currentMapTransform] **is not accessible here**

```
    currentMapTransform = mt;
                ^
```

simple-datasets/src/main/java/cog_complexity_validation_datasets/Three/Tasks_1.java:904: warning: **Cannot access** [cog_complexity_validation_datasets.Three.MapControlsAction.ID]

```
    MapControlsAction mca = (MapControlsAction) freeColClient.getActionManager().getFreeColAction(MapControlsAction.ID);
                                                                                                  ^
```

simple-datasets/src/main/java/cog_complexity_validation_datasets/Three/Tasks_1.java:904: warning**: Unsafe cast**

```
    MapControlsAction mca = (MapControlsAction) freeColClient.getActionManager().getFreeColAction(MapControlsAction.ID);
                            ^
```

simple-datasets/src/main/java/cog_complexity_validation_datasets/Three/Tasks_1.java:904: warning: **Cannot call getActionManager on null**

```
        MapControlsAction mca = (MapControlsAction)
freeColClient.getActionManager().getFreeColAction(MapControlsAction.ID);
                                                  ^
```

First and second indicate need for access annotations; fourth indicates need for nullability annotation. 3rd is redundant with javac. All FPs.

VERIFIED.

dataset 3, snippet 79

16 warnings:

- 9 from the CF
- 7 from typestate

lines 594-605 of Tasks_3:

simple-datasets > src > main > java > cog_complexity_validation_datasets > Three > Tasks_3.java > Tasks_3 > s79()

```
594 public int s79() {
595     for (int j = 0; j < fieldcount; j++) {
596         int i = Column.compare(session.database.collation, a[cols[j]],
597                 b[cols[j]], coltypes[cols[j]]);
598
599         if (i != 0) {
600             return i;
601         }
602     }
603
604     return 0;
605 } // Added to allow compilation
```

CF:

/home/authors/Code-Complexity-Research/complexity-verification-project/simple-datasets/src/main/java/cog_complexity_validation_datasets/Three/Tasks_3.java:596: warning: [array.access.unsafe.low] Potentially **unsafe array access**: the index could be negative.

```
        int i = Column.compare(session.database.collation, a[cols[j]],
                                                           ^
  found   : @LowerBoundUnknown int
  required: an integer >= 0 (@NonNegative or @Positive)
```

/home/authors/Code-Complexity-Research/complexity-verification-project/simple-datasets/src/main/java/cog_complexity_validation_datasets/Three/Tasks_3.java:596: warning: [array.access.unsafe.high] Potentially **unsafe array access**: the index could be larger than the array's bound

```
        int i = Column.compare(session.database.collation, a[cols[j]],
                                                           ^
  found   : @UpperBoundUnknown int
  required: @IndexFor("this.a") or @LTLengthOf("this.a") -- an integer less than this.a's length
```

/home/authors/Code-Complexity-Research/complexity-verification-project/simple-datasets/src/main/java/cog_complexity_validation_datasets/Three/Tasks_3.java:596: warning: [array.access.unsafe.high.range] Potentially **unsafe array access**: the index could be larger than the array's bound

```
        int i = Column.compare(session.database.collation, a[cols[j]],
                                                           ^
  index type found: @IntRange(from=-2147483648) int
  array type found: @UnknownVal int @UnknownVal []
  required        : index of type @IndexFor("this.cols") or @LTLengthOf("this.cols"), or array of type
@MinLen(-9223372036854775808)
```

/home/authors/Code-Complexity-Research/complexity-verification-project/simple-datasets/src/main/java/cog_complexity_validation_datasets/Three/Tasks_3.java:597: warning: [array.access.unsafe.low] Potentially **unsafe array access:** the index could be negative.

```
                b[cols[j]], coltypes[cols[j]]);
                    ^
  found   : @LowerBoundUnknown int
```

required: an integer >= 0 (@NonNegative or @Positive)
/home/authors/Code-Complexity-Research/complexity-verification-project/simple-datasets/src/main/java/cog_complexity_validation_datasets/Three/Tasks_3.java:597: warning: [array.access.unsafe.high] Potentially **unsafe array access:** the index could be larger than the array's bound
b[cols[j]], coltypes[cols[j]]);
^
found : @UpperBoundUnknown int
required: @IndexFor("this.b") or @LTLengthOf("this.b") -- an integer less than this.b's length
/home/authors/Code-Complexity-Research/complexity-verification-project/simple-datasets/src/main/java/cog_complexity_validation_datasets/Three/Tasks_3.java:597: warning: [array.access.unsafe.high.range] Potentially **unsafe array access:** the index could be larger than the array's bound
b[cols[j]], coltypes[cols[j]]);
^
index type found: @IntRange(from=-2147483648) int
array type found: @UnknownVal int @UnknownVal []
required : index of type @IndexFor("this.cols") or @LTLengthOf("this.cols"), or array of type @MinLen(-9223372036854775808)
/home/authors/Code-Complexity-Research/complexity-verification-project/simple-datasets/src/main/java/cog_complexity_validation_datasets/Three/Tasks_3.java:597: warning: [array.access.unsafe.low] Potentially **unsafe array access**: the index could be negative.
b[cols[j]], coltypes[cols[j]]);
^
found : @LowerBoundUnknown int
required: an integer >= 0 (@NonNegative or @Positive)
/home/authors/Code-Complexity-Research/complexity-verification-project/simple-datasets/src/main/java/cog_complexity_validation_datasets/Three/Tasks_3.java:597: warning: [array.access.unsafe.high] Potentially **unsafe array access**: the index could be larger than the array's bound
b[cols[j]], coltypes[cols[j]]);
^
found : @UpperBoundUnknown int
required: @IndexFor("this.coltypes") or @LTLengthOf("this.coltypes") -- an integer less than this.coltypes's length
/home/authors/Code-Complexity-Research/complexity-verification-project/simple-datasets/src/main/java/cog_complexity_validation_datasets/Three/Tasks_3.java:597: warning: [array.access.unsafe.high.range] Potentially **unsafe array access**: the index could be larger than the array's bound
b[cols[j]], coltypes[cols[j]]);
^
index type found: @IntRange(from=-2147483648) int
array type found: @UnknownVal int @UnknownVal []
required : index of type @IndexFor("this.cols") or @LTLengthOf("this.cols"), or array of type @MinLen(-9223372036854775808)

These warnings all come from four missing facts:

- fieldcount is equal to the length of the cols[] array
- elements of the cols array are indices for the a, b, and coltypes arrays

Both of those are easy to express with the Index Checker, which would remove the warnings, so we can consider them FPs. Tbh, I'm a bit surprised that OpenJML didn't warn on this. I bet it timed out. (I checked, it did. Nice.)

typestate:

simple-datasets/src/main/java/cog_complexity_validation_datasets/Three/Tasks_3.java:597: **warning: Cannot call** [#helpers.arrayAccess] on Shared{int[]} | Null

```
                b[cols[j]], coltypes[cols[j]]);
                 ^
simple-datasets/src/main/java/cog_complexity_validation_datasets/Three/Tasks_3.java:596: warning: Cannot call
[#helpers.arrayAccess] on Shared{int[]} | Null
        int i = Column.compare(session.database.collation, a[cols[j]],
                                                 ^
simple-datasets/src/main/java/cog_complexity_validation_datasets/Three/Tasks_3.java:597: warning: Cannot call
[#helpers.arrayAccess] on Shared{int[]} | Null
                b[cols[j]], coltypes[cols[j]]);
                     ^
simple-datasets/src/main/java/cog_complexity_validation_datasets/Three/Tasks_3.java:597: warning: Cannot call
[#helpers.arrayAccess] on Shared{int[]} | Null
                b[cols[j]], coltypes[cols[j]]);
                                 ^
simple-datasets/src/main/java/cog_complexity_validation_datasets/Three/Tasks_3.java:596: warning: Cannot access
field [database] of null
        int i = Column.compare(session.database.collation, a[cols[j]],
                                 ^
simple-datasets/src/main/java/cog_complexity_validation_datasets/Three/Tasks_3.java:596: warning: Cannot call
[#helpers.arrayAccess] on Shared{int[]} | Null
        int i = Column.compare(session.database.collation, a[cols[j]],
                                                   ^
simple-datasets/src/main/java/cog_complexity_validation_datasets/Three/Tasks_3.java:597: warning: Cannot call
[#helpers.arrayAccess] on Shared{int[]} | Null
                b[cols[j]], coltypes[cols[j]]);
                                 ^
```

All are false positives: all result from missing specifications about nullability.

VERIFIED

dataset 6, snippet 12

We got 94 warnings:

- 91 from the typestate checker
- 1 from infer
- 2 from openjml
- 0 from the checker framework (CF)

This snippet is lines 93-144 of CarReport.java.

dataset6 > src > main > java > CarReport.java > CarReport > SetupWebDavSyncDialogActivity

```
93      public class SetupWebDavSyncDialogActivity extends Activity {
94          // s12: me.kuehle.carreport.gui.dialog.SetupWebDavSyncDialogActivity.onCreate(android.os.Bundle)
95      //    @Override // Removed to allow compilation
96          public void onCreate(Bundle savedInstanceState) {
97              super.onCreate(savedInstanceState);
98              setContentView(R.layout.activity_setup_webdav_sync);
99              getWindow().setLayout(ViewGroup.LayoutParams.MATCH_PARENT, ViewGroup.LayoutParams.WRAP_CONTENT);
100
101             mEdtUrl = (EditText) findViewById(R.id.edt_url);
102             mEdtUrl.addTextChangedListener(new TextWatcher() {
103                 //    @Override // Removed to allow compilation
104                 public void beforeTextChanged(CharSequence s, int start, int count, int after) {
105                 }
106
107                 //    @Override // Removed to allow compilation
108                 public void onTextChanged(CharSequence s, int start, int before, int count) {
109                 }
110
111                 //    @Override // Removed to allow compilation
112                 public void afterTextChanged(Editable s) {
113                     mTxtTrustCertificateDescription.setVisibility(View.GONE);
114                     mTxtTrustCertificate.setVisibility(View.GONE);
115                     mChkTrustCertificate.setChecked(b:false);
116                     mChkTrustCertificate.setVisibility(View.GONE);
117                 }
118             });
119             mEdtUserName = (EditText) findViewById(R.id.edt_user_name);
120             mEdtPassword = (EditText) findViewById(R.id.edt_password);
121             mTxtTrustCertificateDescription = (TextView) findViewById(R.id.txt_trust_certificate_description);
122             mTxtTrustCertificate = (TextView) findViewById(R.id.txt_trust_certificate);
123             mChkTrustCertificate = (CheckBox) findViewById(R.id.chk_trust_certificate);
124
125             mTxtTrustCertificateDescription.setVisibility(View.GONE);
126             mTxtTrustCertificate.setVisibility(View.GONE);
127             mChkTrustCertificate.setVisibility(View.GONE);
128
129             mBtnOk = (Button) findViewById(R.id.btn_ok);
130             mBtnOk.setOnClickListener(new View.OnClickListener() {
131                 //    @Override // Removed to allow compilation
132                 public void onClick(View v) {
133                     onOkClick();
134                 }
135             });
136             findViewById(R.id.btn_cancel).setOnClickListener(new View.OnClickListener() {
137                 //    @Override // Removed to allow compilation
138                 public void onClick(View v) {
139                     setResult(Activity.RESULT_CANCELED);
140                     finish();
141                 }
142             });
143         }
144     }
```

OpenJML:
./CarReport.java:114: verify: The prover cannot establish an assertion (**PossiblyNullDeReference**) in method afterTextChanged

```
            mTxtTrustCertificate.setVisibility(View.GONE);
                                ^
./CarReport.java:115: verify: The prover cannot establish an assertion (PossiblyNullDeReference) in method afterTextChanged
            mChkTrustCertificate.setChecked(false);
```

These are definitely false positives, because these fields are set just below those lines (and this is a callback).

Infer:

```
dataset6/src/main/java/CarReport.java:99: error: Null Dereference
  object returned by `this$0.getWindow()` could be null and is dereferenced at line 99.
   97.            super.onCreate(savedInstanceState);
   98.            setContentView(R.layout.activity_setup_webdav_sync);
   99. >           getWindow().setLayout(ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.WRAP_CONTENT);
  100.
  101.            mEdtUrl = (EditText) findViewById(R.id.edt_url);
```

TRUE POSITIVE: This looks like an artifact of our stubbing-out process: getWindow() is defined to always return null, and Infer has deduced that and is reporting for that reason.

Typestate Checker:

- many, many **"cannot access"** warnings (about ⅔ of the total warnings) like these:

```
dataset6/src/main/java/CarReport.java:139: warning: Cannot access [CarReport.Activity.RESULT_CANCELED]
            setResult(Activity.RESULT_CANCELED);
                      ^
dataset6/src/main/java/CarReport.java:114: warning: Cannot access [this.mTxtTrustCertificate]
            mTxtTrustCertificate.setVisibility(View.GONE);
            ^
dataset6/src/main/java/CarReport.java:113: warning: Cannot access [CarReport.View.GONE]
            mTxtTrustCertificateDescription.setVisibility(View.GONE);
                                            ^
dataset6/src/main/java/CarReport.java:113: warning: Cannot access [this.mTxtTrustCertificateDescription]
            mTxtTrustCertificateDescription.setVisibility(View.GONE);
            ^
dataset6/src/main/java/CarReport.java:116: warning: Cannot access [CarReport.View.GONE]
            mChkTrustCertificate.setVisibility(View.GONE);
                                 ^
dataset6/src/main/java/CarReport.java:114: warning: Cannot access [CarReport.View.GONE]
            mTxtTrustCertificate.setVisibility(View.GONE);
                                 ^
dataset6/src/main/java/CarReport.java:115: warning: Cannot access [this.mChkTrustCertificate]
            mChkTrustCertificate.setChecked(false);
            ^
```

These are caused by the typestate checker's insistence that fields cannot be public without a human writing an annotation, which is overly conservative. Therefore, they are false positives.

- some **"unsafe cast"** warnings (about ⅙ of the total), like these:

```
dataset6/src/main/java/CarReport.java:120: warning: Unsafe cast
```

```
        mEdtPassword = (EditText) findViewById(R.id.edt_password);
                    ^
dataset6/src/main/java/CarReport.java:121: warning: Unsafe cast
        mTxtTrustCertificateDescription = (TextView) findViewById(R.id.txt_trust_certificate_description);
                                    ^
dataset6/src/main/java/CarReport.java:123: warning: Unsafe cast
        mChkTrustCertificate = (CheckBox) findViewById(R.id.chk_trust_certificate);
                        ^
dataset6/src/main/java/CarReport.java:129: warning: Unsafe cast
        mBtnOk = (Button) findViewById(R.id.btn_ok);
              ^
dataset6/src/main/java/CarReport.java:101: warning: Unsafe cast
        mEdtUrl = (EditText) findViewById(R.id.edt_url);
               ^
dataset6/src/main/java/CarReport.java:122: warning: Unsafe cast
        mTxtTrustCertificate = (TextView) findViewById(R.id.txt_trust_certificate);
                        ^
dataset6/src/main/java/CarReport.java:119: warning: Unsafe cast
        mEdtUserName = (EditText) findViewById(R.id.edt_user_name);
                    ^
```

These casts are perfectly safe, so these are false positives, too.

- some **“cannot assign**” warnings, like these (about ⅙ of the total):

```
dataset6/src/main/java/CarReport.java:119: warning: Cannot assign because [this.mEdtUserName] is not accessible
here
        mEdtUserName = (EditText) findViewById(R.id.edt_user_name);
                   ^
dataset6/src/main/java/CarReport.java:120: warning: Cannot assign because [this.mEdtPassword] is not accessible
here
        mEdtPassword = (EditText) findViewById(R.id.edt_password);
```

These are false positives: the checker is enforcing an ownership model that is stronger than the standard Java one, but which this code does not (and should not) obey. It is safe to assign a field, and these fields are accessible under standard Java rules.

There are no other warnings.

VERIFIED.

dataset 9, snippet 2

10 warnings, 5 each from the typestate checker and openJML.

The snippet is lines 458-471 of CodeSnippets.java. (Note: the comments are the original ones or were added by Munoz et al.)

dataset9 > src > main > java > CodeSnippets.java > CodeSnippets > logAndEmailSeriousProblemS113(Throwable, HttpServletRequest)

```
453     /**
454     * Informs the webmaster of an unexpected problem (Exception “ex”)
455     * with the deployed application (indicated by “aRequest”).
456     */
457     //SNIPPET_STARTS_2
458     public void logAndEmailSeriousProblemS112(Throwable ex, HttpServletRequest aRequest)
459     {
460     /* Define local variable. */
461     TroubleTicket troubleTicket = new TroubleTicket(ex, aRequest);
462     /* Log message. */
463     fLogger.severe(msg:"TOP LEVEL CATCHING Throwable.");
464     fLogger.severe(troubleTicket.toString());
465     /* Log message again. */
466     System.out.println(x:"SERIOUS PROBLEM OCCURRED.");// changed to allow compilation
467     System.out.println(troubleTicket.toString());// changed to allow compilation
468     /* Update context and mail trouble ticket. */
469     aRequest.getSession().getServletContext().
470     setAttribute(MOST_RECENT_TROUBLE_TICKET, troubleTicket);
471     }
```

OpenJML:

./CodeSnippets.java:466: verify: The prover cannot establish an assertion (**InvariantLeaveCaller**: /home/authors/openjml/specs/java/io/PrintStream.jml:35:) in method logAndEmailSeriousProblemS112: (Caller: CodeSnippets.logAndEmailSeriousProblemS112(java.lang.Throwable,javax.servlet.http.HttpServletRequest), Callee: java.io.PrintStream.println(java.lang.String))

```
    System.out.println("SERIOUS PROBLEM OCCURRED.");// changed to allow compilation
                  ^
```

./CodeSnippets.java:466: verify: The prover cannot establish an assertion (**InvariantLeaveCaller**: /home/authors/openjml/specs/java/io/PrintStream.jml:42:) in method logAndEmailSeriousProblemS112: (Caller: CodeSnippets.logAndEmailSeriousProblemS112(java.lang.Throwable,javax.servlet.http.HttpServletRequest), Callee: java.io.PrintStream.println(java.lang.String))

```
    System.out.println("SERIOUS PROBLEM OCCURRED.");// changed to allow compilation
                  ^
```

./CodeSnippets.java:467: verify: The prover cannot establish an assertion (**InvariantLeaveCaller**: /home/authors/openjml/specs/java/io/PrintStream.jml:42:) in method logAndEmailSeriousProblemS112: (Caller: CodeSnippets.logAndEmailSeriousProblemS112(java.lang.Throwable,javax.servlet.http.HttpServletRequest), Callee: java.io.PrintStream.println(java.lang.String))

```
    System.out.println(troubleTicket.toString());// changed to allow compilation
                  ^
```

./CodeSnippets.java:467: verify: The prover cannot establish an assertion (**InvariantLeaveCaller**: /home/authors/openjml/specs/java/io/PrintStream.jml:35:) in method logAndEmailSeriousProblemS112: (Caller: CodeSnippets.logAndEmailSeriousProblemS112(java.lang.Throwable,javax.servlet.http.HttpServletRequest), Callee: java.io.PrintStream.println(java.lang.String))

```
    System.out.println(troubleTicket.toString());// changed to allow compilation
                  ^
```

./CodeSnippets.java:470: verify: The prover cannot establish an assertion (**InvariantEntrance**: /home/authors/openjml/specs/java/lang/CharSequence.jml:30:) in method logAndEmailSeriousProblemS112: (Caller: CodeSnippets.logAndEmailSeriousProblemS112(java.lang.Throwable,javax.servlet.http.HttpServletRequest), Callee: javax.servlet.ServletContext.setAttribute(java.lang.String,java.lang.Object))

```
    setAttribute(MOST_RECENT_TROUBLE_TICKET, troubleTicket);
```

The first four errors are all saying that System.out's print stream might not be closed. Obviously false positives!

The last error is referring to the internal specification of java.lang.CharSequence, suggesting that the internal character array hasn't been allocated. That is obviously false, since the constant string here is actually a string constant. So this is also an FP.

typestate:

```
dataset9/src/main/java/CodeSnippets.java:464: warning: Cannot access [CodeSnippets.fLogger]
    fLogger.severe(troubleTicket.toString());
    ^

dataset9/src/main/java/CodeSnippets.java:463: warning: Cannot access [CodeSnippets.fLogger]
    fLogger.severe("TOP LEVEL CATCHING Throwable.");
    ^

dataset9/src/main/java/CodeSnippets.java:469: warning: Cannot call setAttribute on null
    aRequest.getSession().getServletContext().
                                  ^

dataset9/src/main/java/CodeSnippets.java:470: warning: Cannot access
[CodeSnippets.MOST_RECENT_TROUBLE_TICKET]
    setAttribute(MOST_RECENT_TROUBLE_TICKET, troubleTicket);
            ^

dataset9/src/main/java/CodeSnippets.java:469: warning: Cannot call getServletContext on null
    aRequest.getSession().getServletContext().
                  ^
```

The 3 "cannot access" warnings are obviously FPs. The "cannot call" warnings are a bit trickier: effectively, they are claiming that aRequest and aRequest.getSession() might both be null. getSession() cannot return null according to its docs. aRequest is a method parameter, so if it were nullable a human would need to indicate that, but the comments don't do that. We can definitely regard both as false positives.

dataset f, snippet 13

simple-datasets > src > main > java > fMRI_Study_Classes > RecursiveFibonacciVariant.java > RecursiveFibonacciVariant

```java
1  package fMRI_Study_Classes;
2
3  public class RecursiveFibonacciVariant {
4      public static void run() {
5          int number = 4;
6          System.out.print(compute(number));
7      }
8
9      //SNIPPET_STARTS
10     public static int compute(int number) {
11         if (number <= 1) {
12             return 1;
13         }
14
15         return compute(number - 2) + compute(number - 4);
16     }
17 }
```

2 warnings from OpenJML:

fMRI_Study_Classes/RecursiveFibonacciVariant.java:14: verify: The prover cannot establish an assertion (**ArithmeticOperationRange**) in method compute: underflow in int sum

```
    return compute(number - 2) + compute(number - 4);
                   ^
```

fMRI_Study_Classes/RecursiveFibonacciVariant.java:14: verify: The prover cannot establish an assertion (**ArithmeticOperationRange**) in method compute: overflow in int sum

```
    return compute(number - 2) + compute(number - 4);
```

The underflow is obviously a false positive, since compute() of any number less than 1 is 1 (with no recursive call).

The overflow warning is trickier. I think technically this code can overflow, if a sufficiently-large number is given as input. However, one could easily write a specification that limits the inputs to numbers that would not result in an overflow, and that spec would be verifiable. So, I think we can regard this as a case of a missing annotation.

VERIFIED.