Suspicious pieces of code found in DaCapo benchmark programs

Eric Bodden, Patrick Lam and Laurie Hendren

Sable Research Group
School of Computer Science
McGill University

March 7th, 2008

This document gives information about interesting pieces of code that we discovered in DaCapo.

```
String[]  selectNativeCode(org.osgi.framework.Bundle bundle) {
    ...
    if (bundleNativeCodes.size() == 0)
        return noMatches(optional);
    Iterator  iter  = bundleNativeCodes.iterator();
    BundleNativeCode highestRanking = (BundleNativeCode) iter.next();
    ...
}
```

Figure 1: Code in class org.eclipse.osgi.framework.internal.core.Framework, CVS revision 1.110
The code does not check `hasNext()` but rather tests `size()==0` on the collection.

```
private void setOutput(File newOutFile, Writer newWriter, boolean append) {
    ...
    Reader  fileIn  = null;
    try {
        openFile();
        fileIn  = new InputStreamReader(secureAction.getFileInputStream(oldOutFile),"UTF-8");
        copyReader(fileIn, this.writer);
    } catch (IOException e) {
    ...
}
```

Figure 2: Code in class org.eclipse.core.runtime.adaptor.EclipseLog, Eclipse version 3.1
The code creates a reader and then passes it to another method. After doing so, `setOutput` has to make sure not to close the reader, nor its input stream.

```
protected NameDeclaration findVariableHere(NameOccurrence occurrence) {
    if (occurrence.isThisOrSuper() || occurrence.getImage().equals(className)) {
        if (variableNames.isEmpty() && methodNames.isEmpty()) {
            return null;
        }
        if (!variableNames.isEmpty()) {
            return variableNames.keySet().iterator().next();
        }
        return methodNames.keySet().iterator().next();
    }
    ...
}
```

Figure 3: Code in class net.sourceforge.pmd.symboltable.ClassScope, SVN revision 5111
The code does not check `hasNext()` but rather tests `isEmpty()` on the collection.

```
public Iterator iterator () {
    return new Iterator() {
        Iterator i = list . iterator ();
        public void remove() {
            throw new UnsupportedOperationException();
        }
        public boolean hasNext() {
            return i.hasNext();
        }
        public Object next() {
            return i.next();
        }
    };
}
```

Figure 4: Delegating Iterator in class org.python.core.PyTuple, SVN revision 3673; although next() calls itself i.next() without calling i.hasNext(), this use is safe because i is used correctly whenever **this** (the wrapper) is used correctly

```
private List markUsages(IDataFlowNode inode) {
    ...
    for (Iterator k = ((List) entry.getValue()).iterator(); k.hasNext();) {
        addAccess(k, inode);
    }
    ...
}

...

private void addAccess(Iterator k, IDataFlowNode inode) {
    NameOccurrence occurrence = (NameOccurrence) k.next();
    ...
}
```

(a) SVN revision 4797

```
private List markUsages(IDataFlowNode inode) {
    ...
    for (NameOccurrence occurrence: entry.getValue()) {
        addAccess(occurrence, inode);
    }
    ...
}

...

private void addAccess(NameOccurrence occurrence, IDataFlowNode inode) {
    ...
}
```

(b) SVN revision 4993; code was fixed when switching to enhanced Java 5 for-loops

Figure 5: Suspicious code in class net.sourceforge.pmd.dfa.variableaccess.VariableAccessVisitor
The method `addAccess` extracts the iterator's first element without a check. This program is only sound because `addAccess` is only called by `markUsages`.

```
static String getLine(BufferedReader reader, int line) {
    if (reader == null)
        return "";
    try {
        String text=null;
        for(int i=0; i < line; i++) {
            text = reader.readLine();
        }
        return text;
    } catch (IOException ioe) {
        return null;
    }
}
```

(a) Suspicious code in class org.python.core.parser; reads from potentially closed reader

```
private final void FillBuff() throws java.io.IOException
{
    ...
    try {
        if ((i = inputStream.read(buffer, maxNextCharInd, available − maxNextCharInd)) == −1) {
            inputStream.close();
            throw new java.io.IOException();
        }
        else
            maxNextCharInd += i;
        return;
    }
    ...
}
```

(b) Suspicious code in class org.python.parser.ReaderCharStream; closes reader when it hits the end of file (the variable inputStream is actually of type Reader)

Figure 6: Suspicious code in jython, SVN revision 3673
Reads from a potentially closed reader.