Changing Java’s Semantics for Handling Null Pointer Exceptions

Masters Thesis Presentation
Kinga Dobolyi
Exceptions

- Linux on Air Algerie Airbus A330
Linux on Air Algerie Airbus A330

```
+ echo /seatapps/app_dir
/s atapp / pp_dir
+ echo Setu _Lib ary_Path
Setup_Library_Path
+ extort LD_LIBRARY_PATH=/engine.cram/lib:/lib:/lib:/usr/lib:/usr/X11R6/lib
s73/lib:/rhs73/usr/lib:/rhs73/usr/loca /lib
+ echo Test ldd
Test ldd _ RARY_PATH=/engine.cram/lib:/lib:/lib:/usr/lib:/usr/X11R6/lib
+ ldd /engine.cram/airsurfs73/usr/local/lib
    o libuga.so.1 => /engine.cram/lib/libuga.so.1 (0x40013000)
Test ldd    gl.so.1 => /engine.cram/lib/libugagl.so.1 ( x4 40000)
+ ldd /engine.cram/airsurfso6r => /engine.cram/lib/libfreetype.so.6 (x4:040d000)
    libd1.so.2 => /lib/libd1.so.2 (0x400b2000)
    libpthread.so.0 => /lib/libpthread.so.0 (0x400b5000) (0x40000000)
    libstdc++-libc6.2-2.so.3 => /usr/lib/libstdc++-libc6.2-2.so.3 (0x400
c
    libn.so.6d=>/lib/libn.so.6 (0x4010d000) (0x400b5000)
    libc.c=>/lib/libc-2.so.3 (0x400b3000)
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
+ echo Launching /engine/cram/airsurf
Launching /engine.cram/airsurf
+ /engine.cram/airsurf
0x00c6
sugarlib: Signal 11: Segmentation fault received.
Segmentation fault
```

Please press Enter to activate this console.
Exceptions

• How bad is this exception?
• What would you do, as a designer?
Exceptions

• What about this one?

“The data contained a zero where it shouldn’t have…”

“crashing the entire network and causing the ship to lose control of its propulsion system”
Exceptions

• What if we allowed the program to continue?
Exceptions

Your Account > Where's My Stuff? > Order Summary > Shipment Tracking

Information about shipment

Ship Carrier: USPS
Tracking Number: 9102009591871394870186
Status: In transit

Order #: 103-2140728-2736643
Shipment Date: May 12, 2007
Destination: Champaign, IL, USA
Estimated Arrival: May 30, 2007

Track your package

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Location</th>
<th>Event Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 15, 2007</td>
<td>---</td>
<td>HAZELWOOD MO US</td>
<td>Departure Scan</td>
</tr>
<tr>
<td>May 13, 2007</td>
<td>12:41:00 PM</td>
<td>PHILADELPHIA PA US</td>
<td>In transit</td>
</tr>
<tr>
<td>December 31, 1969</td>
<td>03:59:59 PM</td>
<td>US</td>
<td>Carrier notified to pick up package</td>
</tr>
<tr>
<td>May 12, 2007</td>
<td>---</td>
<td>US</td>
<td></td>
</tr>
</tbody>
</table>

Where's My Stuff?
- Track your recent orders.
- View or change your orders in Your Account.

Shipping & Returns
- See our shipping rates & policies.
- Return an item (here's our Returns Policy).

Need Help?
- Forgot your password? Click here.
- Redeem or buy a gift certificate.
- Visit our Help department.
Exceptions

- Increase availability
- Perhaps the program can continue
- Want exceptions to map into a total function (all input space is covered)
Exceptions

- NullPointerExceptions (NPEs) in Java
Overview

• Introduction
• NPE Background and motivation
• Proposed Technique
• Error Handling and Recovery Policies
• Experimental Results
• Related Work
• Conclusion
Introduction

• We want to prevent NPEs
• Create a total function
  – For valid dereferences
  – For invalid dereferences
Introduction

• **APPEND:**
  – Analysis of potential NPE sites
  – Insertion of error-handling code

• Compile time

• Recovery policies

• In object code
  – Reduces complexity
Background

• NPEs:
  – **Most common error** in Java programs [Cielecki 2006]
  – Frequent and catastrophic
  – Make programs unsafe
  – Top 10 web application security risks [Security Advisor Portal 2003]
Background

• Why are we not preventing them?
  – Conceptual errors
  – Unchecked vs checked exceptions
  – Manually impractical
Null Checking Analysis

- Not systematic
- Clutters code

```java
Personprs = database.getPerson(personID);
if (prs != null)
    println("Name: " + prs.getName());
if (prs != null && prs.getAddr() != null)
    println("Zipcode: " + prs.getAddr().getZip());
```
Do Programmers Put In Null Checks?

- 90% of null checking not taking place

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Call Chains</th>
<th>Required checks (total)</th>
<th>Programmer checks as % of total required</th>
<th>LOC</th>
<th>Required checks as % of LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>36</td>
<td>36</td>
<td>0%</td>
<td>241</td>
<td>15%</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>100%</td>
<td>284</td>
<td>0%</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>14</td>
<td>0%</td>
<td>86</td>
<td>12%</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>21</td>
<td>33%</td>
<td>315</td>
<td>4%</td>
</tr>
<tr>
<td>5</td>
<td>97</td>
<td>98</td>
<td>0%</td>
<td>207</td>
<td>47%</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>12</td>
<td>33%</td>
<td>128</td>
<td>6%</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>13</td>
<td>31%</td>
<td>101</td>
<td>9%</td>
</tr>
<tr>
<td>8</td>
<td>43</td>
<td>60</td>
<td>23%</td>
<td>319</td>
<td>14%</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>0%</td>
<td>273</td>
<td>0%</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>21</td>
<td>14%</td>
<td>137</td>
<td>13%</td>
</tr>
<tr>
<td>11</td>
<td>34</td>
<td>42</td>
<td>12%</td>
<td>230</td>
<td>16%</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>1</td>
<td>100%</td>
<td>229</td>
<td>0%</td>
</tr>
<tr>
<td>13</td>
<td>4</td>
<td>11</td>
<td>55%</td>
<td>223</td>
<td>2%</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>33</td>
<td>55%</td>
<td>217</td>
<td>7%</td>
</tr>
<tr>
<td>15</td>
<td>39</td>
<td>61</td>
<td>0%</td>
<td>185</td>
<td>33%</td>
</tr>
<tr>
<td>16</td>
<td>13</td>
<td>19</td>
<td>32%</td>
<td>206</td>
<td>9%</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>0</td>
<td>100%</td>
<td>171</td>
<td>0%</td>
</tr>
<tr>
<td>18</td>
<td>18</td>
<td>19</td>
<td>0%</td>
<td>94</td>
<td>20%</td>
</tr>
<tr>
<td>TOTAL</td>
<td>352</td>
<td>463</td>
<td>15%</td>
<td>3646</td>
<td>13%</td>
</tr>
</tbody>
</table>
Problems with NPEs

• Many sources implies multiple catch blocks
• Breakdown of encapsulation and information hiding
• Some programming idioms make static analysis unattractive
Goals

• Prevent all NPEs
  – Continued execution
  – Total function

• Automatic

• Transparent
  – Low overhead
  – Space, speed
Proposed Technique

• Analysis
  – Locate potential NPEs

• Transformation
  – Insert null check as a guard
  – Use user-specified recovery policy
Proposed Technique

• Input
  – Source code or byte code (unannotated)
  – Global recovery policy (default)
  – Context specific recovery policies (optional)

• Output
  – Transformed source code or bytecode
  – Guaranteed* free from NPEs
Example

• Before

```java
if (!name.equals("xsd:schema")) {
    // decls for instance only
    attrImpl.addAttribute(
        "",
        "",
        "xmlns:" + NSConstants.SCHEMA_INSTANCE_NS_PREFIX,
        "CDATA",
        NSConstants.SCHEMA_INSTANCE_NS_NAME);
}
```

• After

```java
if (r1 == null)
{
    r1 = new String();
}
if (! (r1.equals("xsd:schema")))
{
    r3.addAttribute("","", "xmlns:xsi", "CDATA", "
    http://www.w3.org/2001/XMLSchema-instance");
}
```
Example

• Before

```java
if (attrs == null || attrs.getLength() == 0) {
    attrImpl = new AttrImpl();
} else {
    attrImpl = new AttrImpl(attrs);
}
```

• After

```java
AttrImpl r3 = null;
label_0:
{
    if (r2 != null && r2.getLength() != 0)
    {
        r3 = new AttrImpl(r2);
        break label_0;
    }
}

    r3 = new AttrImpl();
} // end label_0:
```
Finding Potential NPEs

• Tradeoff

• Conservative flow-sensitive intraprocedural dataflow analysis
  – Constructor calls
  – Global field accesses (i.e., System.out)
  – Static function calls
  – Array accesses (i.e., p[i])
Soundness

• Does not change correct execution
  – *Assumes*: Programs do not rely on NPEs

• Correctness of exceptional execution
  – *Assumes*: correct user-defined recovery policies
Error Handling Transformations

• Call default constructor

```java
1 if (r4 == null)
2     r4 = new Vector();
3 r6 = virtualinvoke r4.<java.lang.Vector:
4     java.lang.String toString()>();
```

• Skip statements
• User defined recovery actions
User-defined recovery policy

• First class object
  – Manipulated and executed during compilation
• applicable
• apply

Input: The program context $C$ and an error location $L$.

\[
\text{if } \text{logging.applicable}(C,L) \text{ then } \\
C,L \leftarrow \text{logging.apply}(C,L) \\
\text{return } (C,L)
\]
User defined recovery policies

• Composable
  – Global policy
  – Context-specific policy
    • Target object
    • Class context
    • Method

• Data Structure

Consistency
User defined recovery policy

Input: The program context \( C \) and an error location \( L \).

1. if the \textit{dereferenced object} at \( L \) has a policy \( P_1 \)
   \( \land P_1.\text{applicable}(C,L) \) then
   return \( P_1.\text{apply}(C,L) \)
2. if the \textit{context class} at \( L \) in \( C \) has a policy \( P_2 \)
   \( \land P_2.\text{applicable}(C,L) \) then
   return \( P_2.\text{apply}(C,L) \)
3. else if the \textit{context method} at \( L \) in \( C \) has a policy \( P_3 \)
   \( \land P_3.\text{applicable}(C,L) \) then
   return \( P_3.\text{apply}(C,L) \)
4. else
5. if \textit{logging} \( \text{applicable}(C,L) \) then
   \( C,L \leftarrow \text{logging}.\text{apply}(C,L) \)
6. end if
7. if \textit{constructor} \( \text{applicable}(C,L) \) then
   \( C,L \leftarrow \text{constructor}.\text{apply}(C,L) \)
8. end if
9. return \( (C,L) \)
10. end if
User defined Data Structure Consistency

**Input:** The program context $C$ and an error location $L$.

```plaintext
if other_policy.applicable($C,L$) then
  $C,L \leftarrow$ other_policy.apply($C,L$)
end if

for all database writes $W(x)$ reached by $L$ do
  $C,L \leftarrow$ replace $W(x)$ by “if invariant($x$) then $W(x)$ else throw new DatabaseException()”
end for

return $C,L$
```
Experimental Results

• Effectiveness
  – Preventing NPEs in sample code
  – Preventing NPEs in Java Standard Library
  – Runtime overhead
  – Class file size
Experimental Results

• Used default policy, which composes:
  – Skip
  – Default constructor (available 65% of the time)

```plaintext
if constructor.applicable(C,L) then
    C,L ← constructor.apply(C,L)
else if skip.applicable(C,L) then
    C,L ← skip.apply(C,L)
return C,L
```
Experimental Results: Example

• Error in URL library class
• System.out.println(v1.indexOf(
    new URL("file", null,
    "C:\\jdk1.1.6\\src\\test" +
    i + ".txt")));


Experimental Results

- Average slowdown 1.3%
Experimental Results

- Increase in null checking:

<table>
<thead>
<tr>
<th>Benchmark Program</th>
<th>Null Checks Normal</th>
<th>Null Checks With APPEND</th>
<th>Increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScriptZip</td>
<td>9</td>
<td>9932</td>
<td>1100x</td>
</tr>
<tr>
<td>HTMLParser</td>
<td>170499</td>
<td>623361</td>
<td>3.66x</td>
</tr>
<tr>
<td>Skaringa</td>
<td>371</td>
<td>1732</td>
<td>4.66x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Benchmark Program</th>
<th>Executed Null Checks Normal</th>
<th>Executed Null Checks With APPEND</th>
<th>Normal as % of Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScriptZip</td>
<td>0</td>
<td>19848</td>
<td>0%</td>
</tr>
<tr>
<td>HTMLParser</td>
<td>190384</td>
<td>1146002</td>
<td>14%</td>
</tr>
<tr>
<td>Skaringa</td>
<td>296</td>
<td>1360</td>
<td>18%</td>
</tr>
</tbody>
</table>
Experimental Results

• Growth in byte code size = 22%

![Relative Size Chart for JavaScriptZip v1.0.3, HTML Parser v1.1, Skarina r3p7, Java Standard Library]
Related Work

- FindBugs (Pugh)
- Acceptability oriented and failure oblivious computing (Rinard)
- Soft computations
- AOP
Summary

• We want to prevent NPEs
• Create a total function
  – For valid dereferences
  – For invalid dereferences
Conclusion

• **APPEND:**
  – Analysis of potential NPE sites
  – Insertion of error-handling code
• Compile time
• Recovery policies
• In object code
  – Reduces complexity
• Low overhead