Changing Java's Semantics for Handling Null Pointer Exceptions

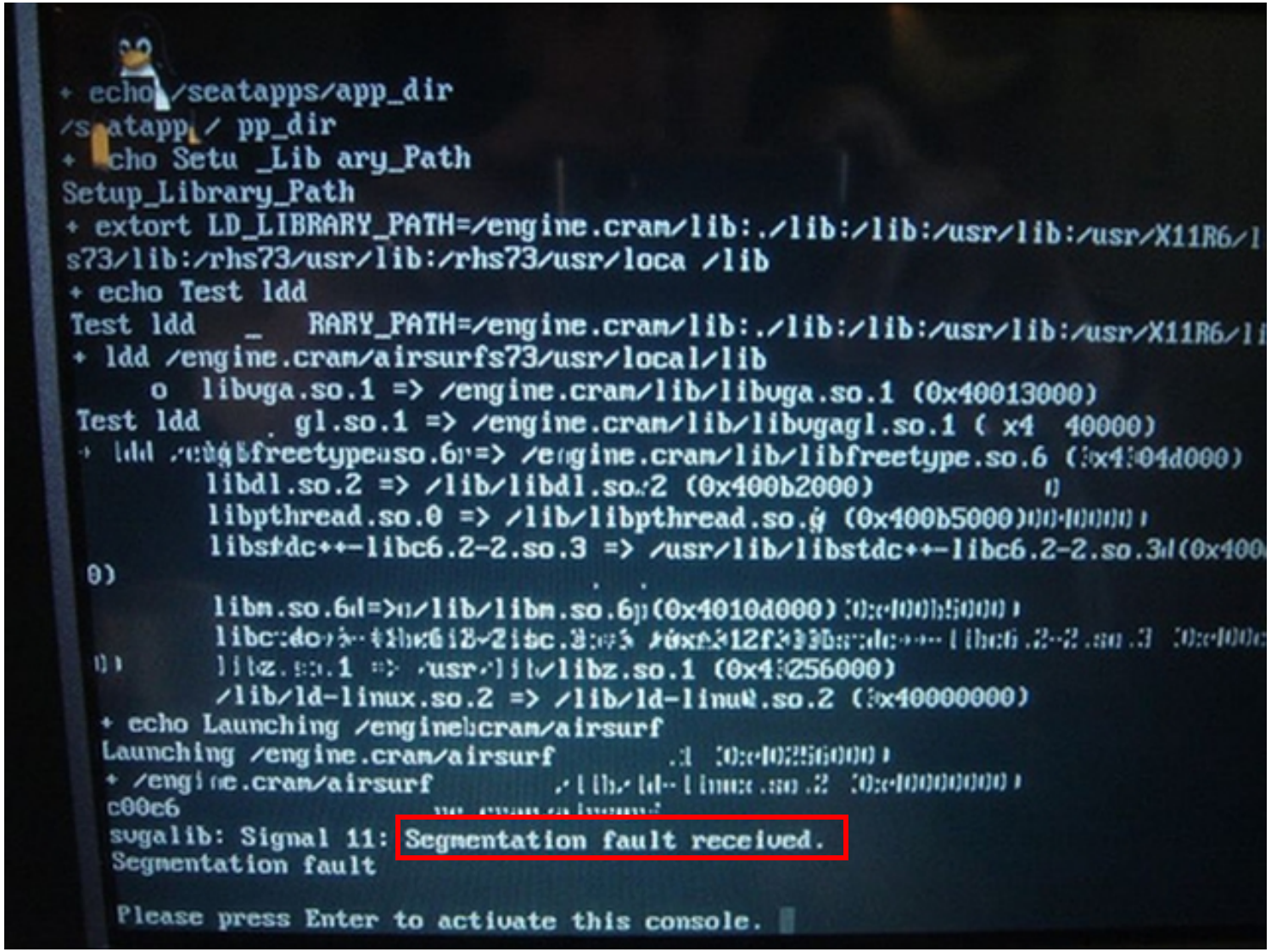
Masters Thesis Presentation

Kinga Dobolyi

Exceptions

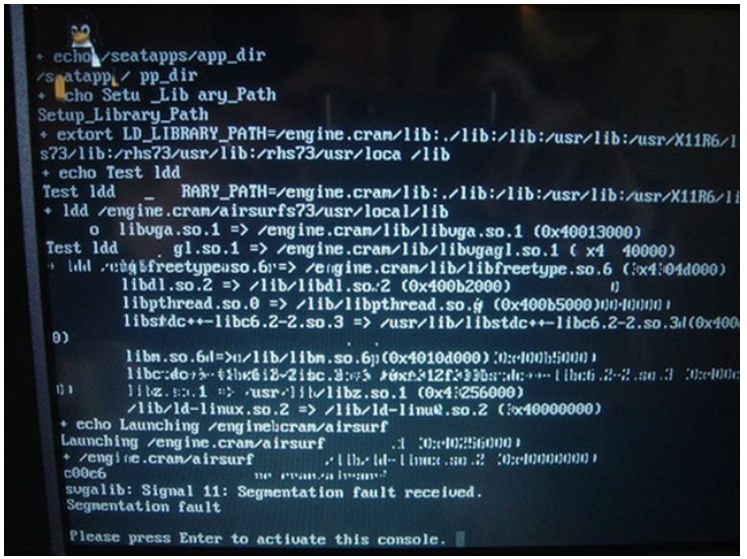
- Linux on Air Algerie Airbus A330

Linux on Air Algerie Airbus A330



Exceptions

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



Exceptions

- What about this one?

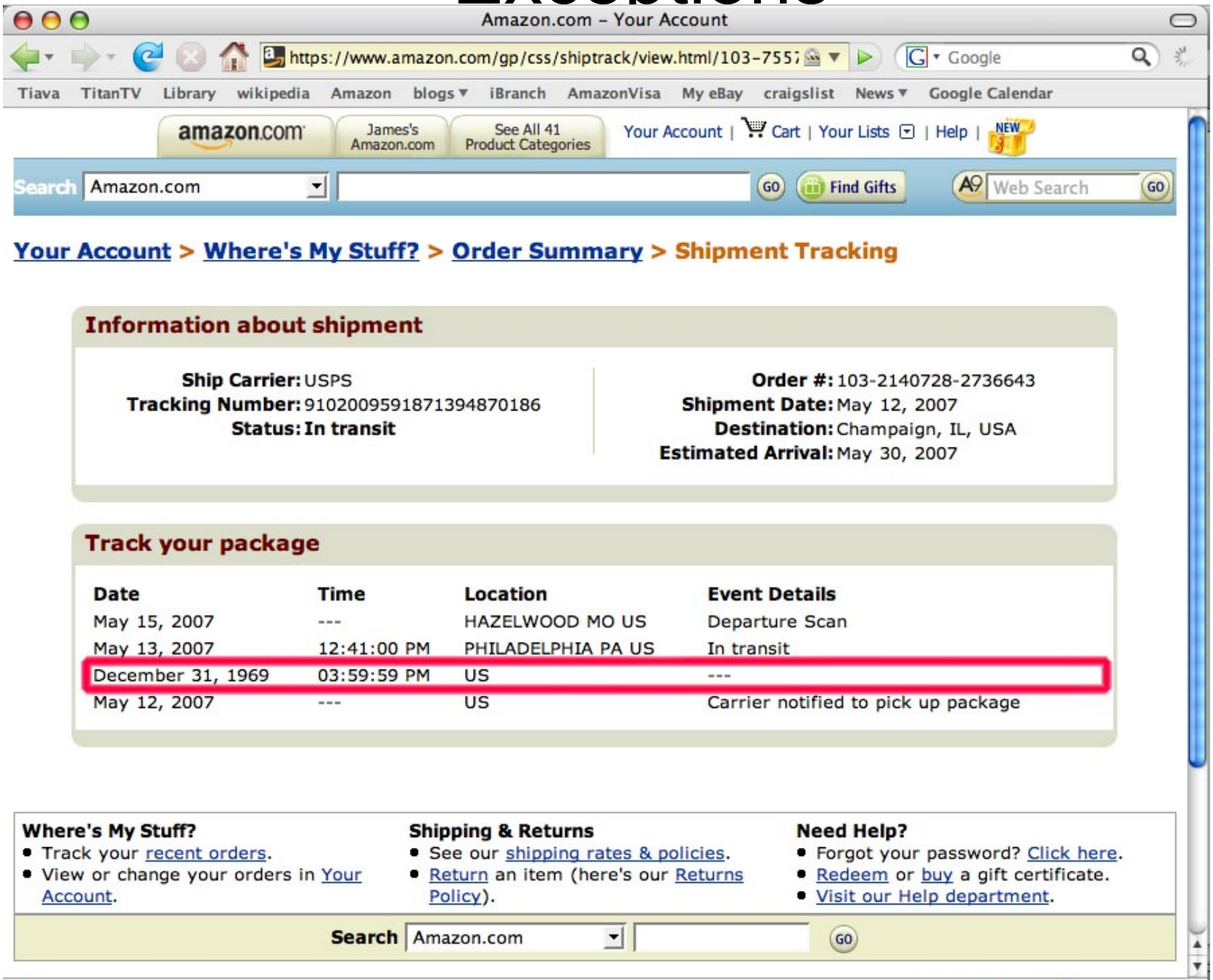
  “The data contained a zero where it shouldn’t have…”

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

Exceptions

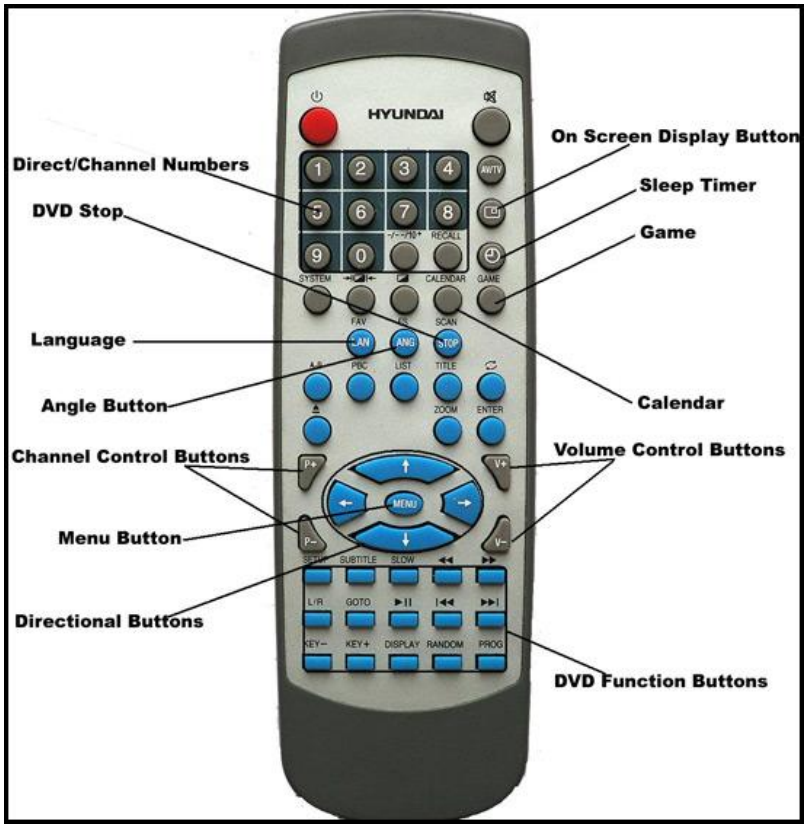
- What if we allowed the program to continue?

Exceptions


Amazon.com – Your Account
https://www.amazon.com/gp/css/shiptrack/view.html/103-755
Google
Tiava TitanTV Library wikipedia Amazon blogs iBranch AmazonVisa My eBay craigslist News Google Calendar
amazon.com
James's Amazon.com
See All 41 Product Categories
Your Account | Cart | Your Lists | Help |
Search Amazon.com GO Find Gifts Web Search GO
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
Date Time Location Event Details
May 15, 2007 --- HAZELWOOD MO US Departure Scan
May 13, 2007 12:41:00 PM PHILADELPHIA PA US In transit
December 31, 1969 03:59:59 PM US ---
May 12, 2007 --- US Carrier notified to pick up package
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.
Search Amazon.com GO

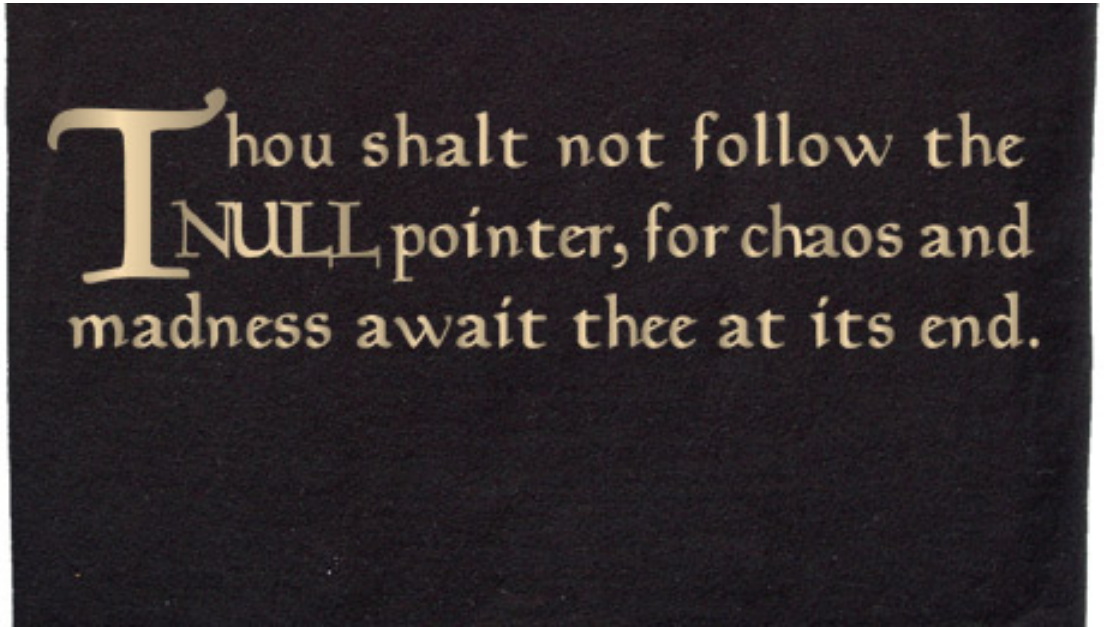
Exceptions

- 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

```
1  Person prs = database.getPerson(personID);
2  if (prs != null)
3    println("Name: " + prs.getName());
4  if (prs != null && prs.getAddr() != null)
5    println("Zipcode: " + prs.getAddr().getZip());
```

Do Programmers Put In Null Checks?

- 90% of null checking not taking place

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

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

```
44      if (!name.equals("xsd:schema")) {
45        // decls for instance only
46        attrImpl.addAttribute(
47          "",
48          "",
49          "xmlns:" + NSConstants.SCHEMA_INSTANCE_NS_PREFIX,
50          "CDATA",
51          NSConstants.SCHEMA_INSTANCE_NS_NAME);
52      }
```

- After

```
34          if (r1 == null)
35          {
36              r1 = new String();
37          }
38          if ( ! (r1.equals("xsd:schema")))
39          {
40              r3.addAttribute("", "", "xmlns:xsi", "CDATA", "
41                http://www.w3.org/2001/XMLSchema-instance");
42          }
```

Example

- Before

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

- After

```
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

```
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.

if `logging.applicable`(C,L) **then**

$C,L \leftarrow$ `logging.apply`(C,L)

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 dereferenced object at L has a policy P_1
   $\wedge$ P_1.applicable(C,L) **then**
2: **return** P_1.apply(C,L)
3: **else if** the context class at L in C has a policy P_2
   $\wedge$ P_2.applicable(C,L) **then**
4: **return** P_2.apply(C,L)
5: **else if** the context method at L in C has a policy P_3
   $\wedge$ P_3.applicable(C,L) **then**
6: **return** P_3.apply(C,L)
7: **else**
8: **if** logging.applicable(C,L) **then**
9: $C,L \leftarrow$ logging.apply(C,L)
10: **end if**
11: **if** constructor.applicable(C,L) **then**
12: $C,L \leftarrow$ constructor.apply(C,L)
13: **end if**
14: **return** (C,L)
15: **end if**

User defined Data Structure Consistency

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

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)

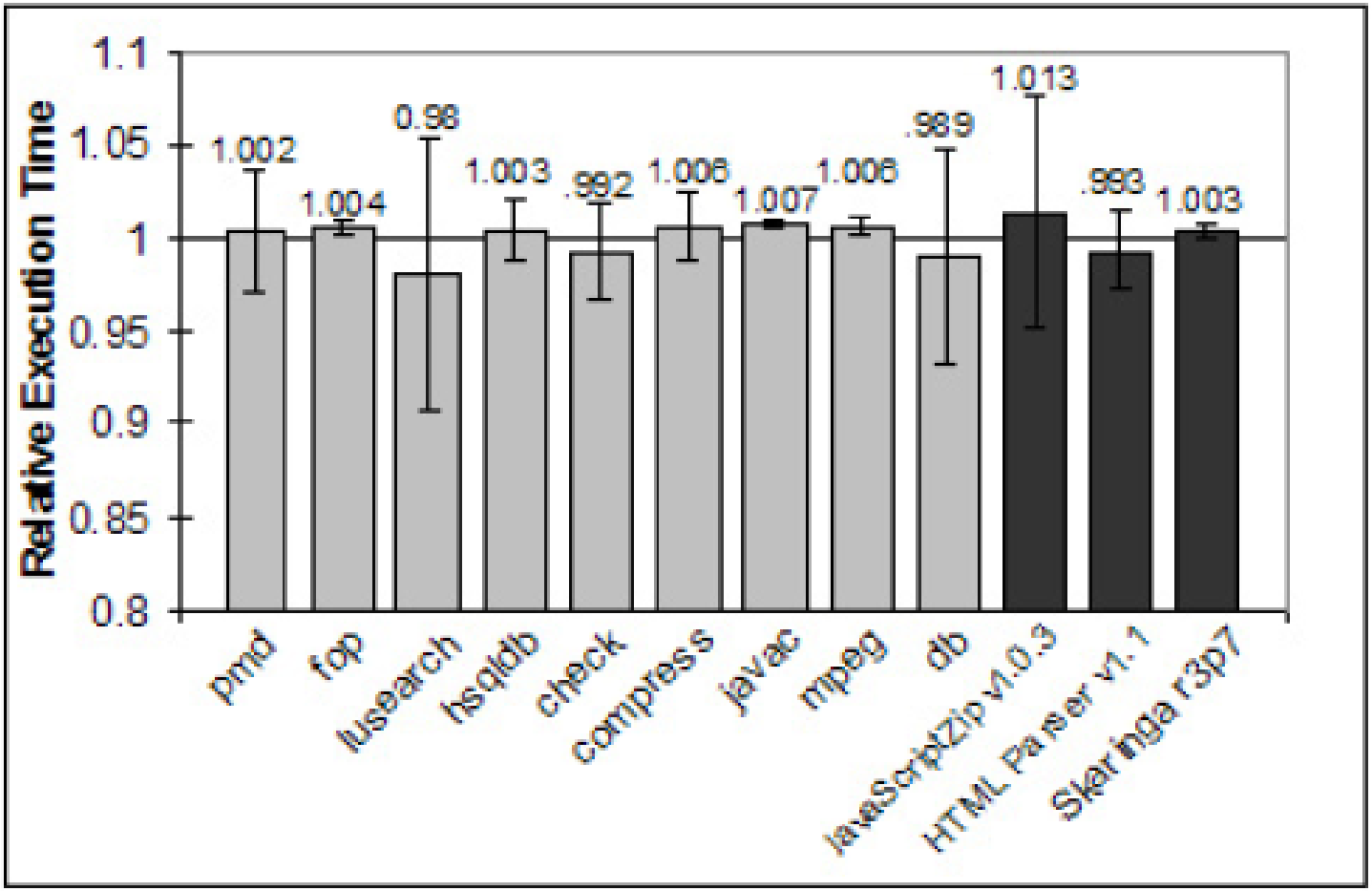
```
if constructor.applicable(C,L) then
        C,L ← constructor.apply(C,L)
else if skip.applicable(C,L) then
        C,L ← skip.apply(C,L)
return C,L
```

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

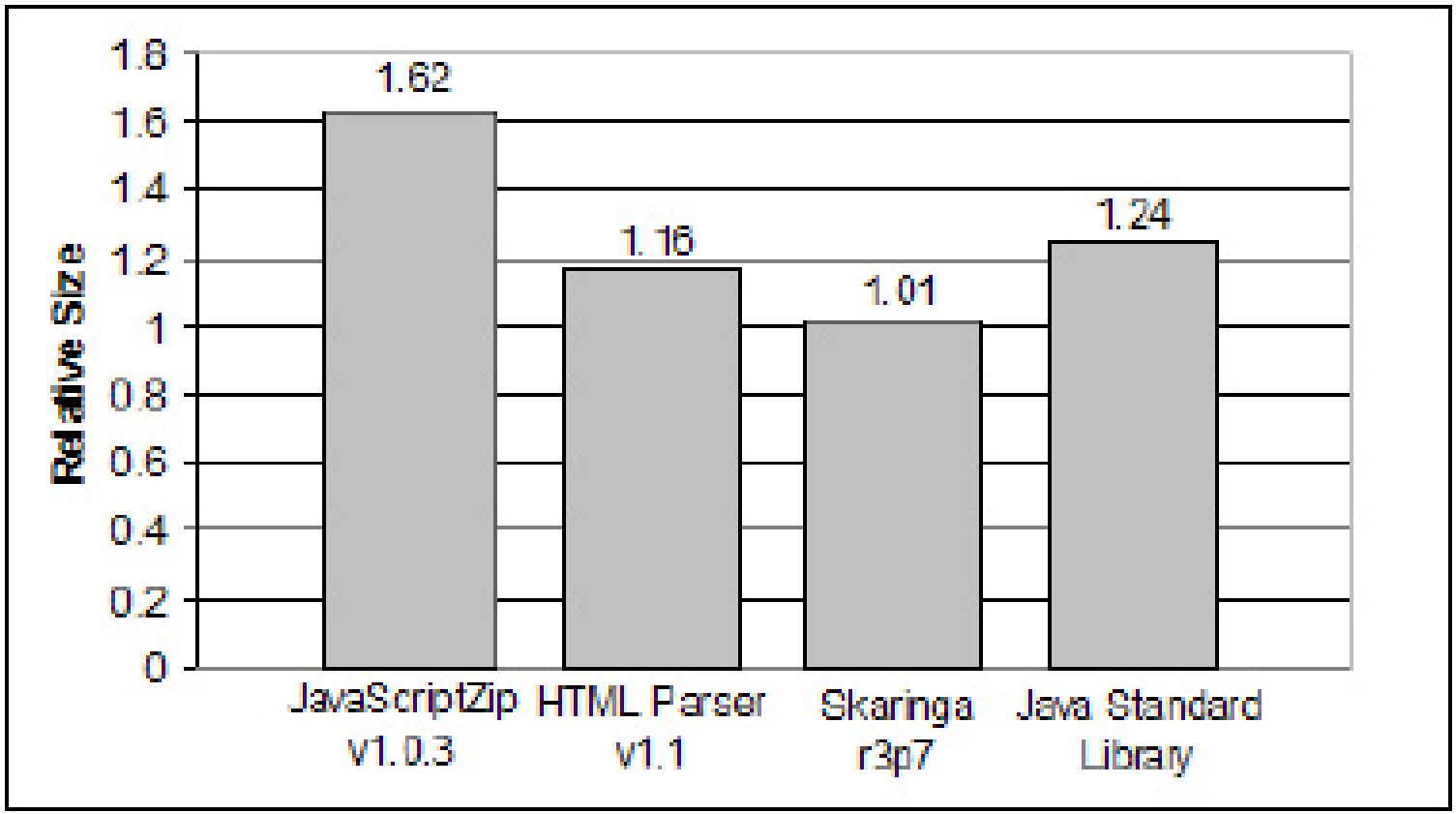
- Increase in null checking:

Benchmark Program	Null Checks		Increase
	Normal	With APPEND	
JAVASCRIPTZIP	9	9932	1100x
HTMLPARSER	170499	623361	3.66x
SKARINGA	371	1732	4.66x

Benchmark Program	Executed Null Checks		Normal as % of Total
	Normal	With APPEND	
JAVASCRIPTZIP	0	19848	0%
HTMLPARSER	190384	1146002	14%
SKARINGA	296	1360	18%

Experimental Results

- Growth in byte code size = 22%



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