

Build Your Own Type System for **Fun** and **Profit**

Werner M. Dietl

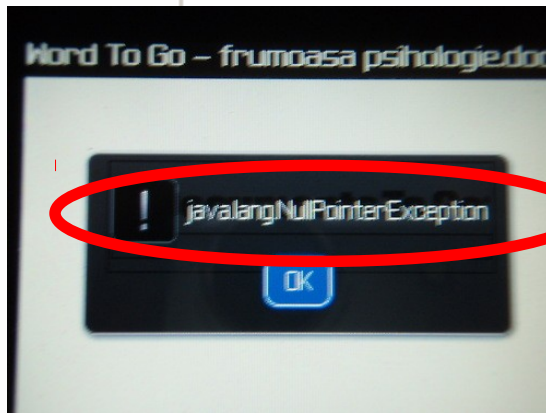
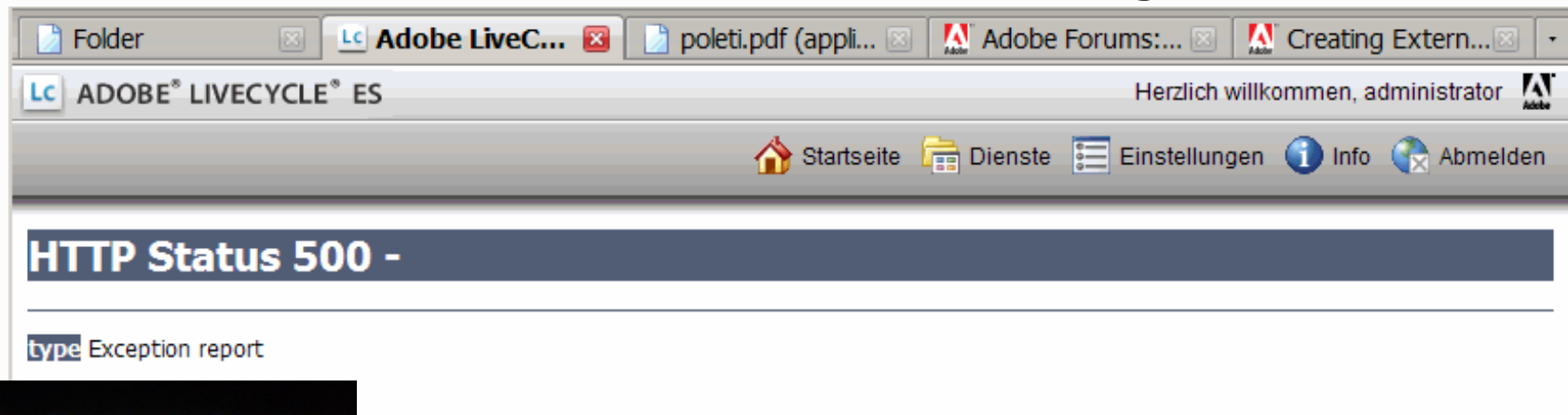
Michael D. Ernst

<http://types.cs.washington.edu/checker-framework/>



University of Washington
Computer Science & Engineering

Software has too many errors

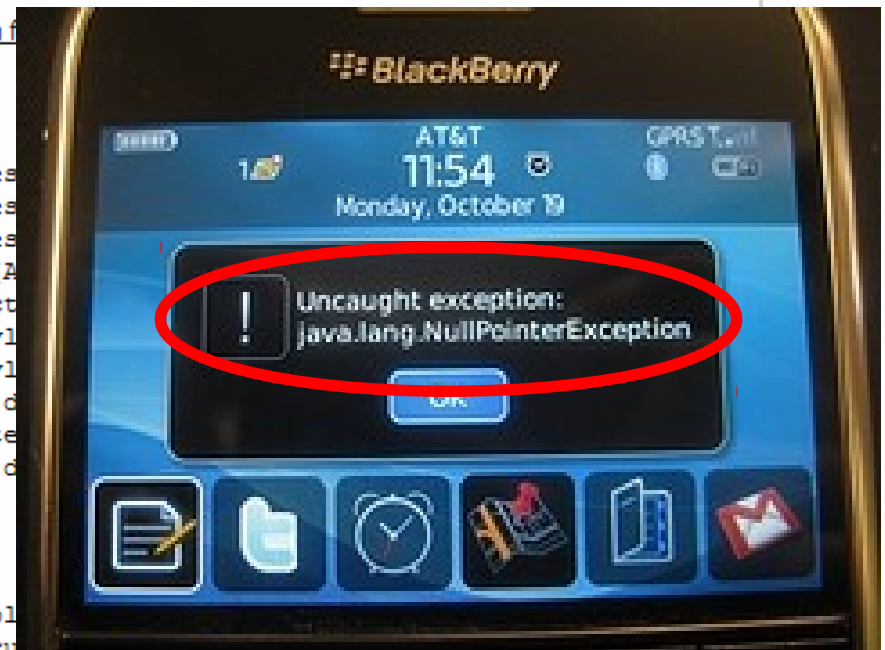


entered an internal error () that prevented it from f

Exception
com.adobe.edc.ui.policy.PolicyEditAction.initPol
com.adobe.edc.ui.policy.PolicyEditAction.doExecu
com.cc.framework.adapter.struts.ActionUtil.execute (Unknown Source)
com.cc.framework.adapter.struts.FWAction.execute (Unknown Source)
org.apache.struts.action.RequestProcessor.processActionPerform (RequestProcessor.java:431)
org.apache.struts.action.RequestProcessor.process (RequestProcessor.java:236)
org.apache.struts.action.ActionServlet.process (ActionServlet.java:1196)
org.apache.struts.action.ActionServlet.doGet (ActionServlet.java:414)

root cause

java.lang.NullPointerException
com.adobe.edc.ui.policy.PolicyEditAction.initPol
com.adobe.edc.ui.policy.PolicyEditAction.doExecu
com.cc.framework.adapter.struts.ActionUtil.execute (Unknown Source)
com.cc.framework.adapter.struts.FWAction.execute (Unknown Source)
com.cc.framework.adapter.struts.FWAction.execute (Unknown Source)
org.apache.struts.action.RequestProcessor.processActionPerform (RequestProcessor.java:431)
org.apache.struts.action.RequestProcessor.process (RequestProcessor.java:236)
org.apache.struts.action.ActionServlet.process (ActionServlet.java:1196)
org.apache.struts.action.ActionServlet.doGet (ActionServlet.java:414)



Java's type system is too weak

Type checking prevents many errors

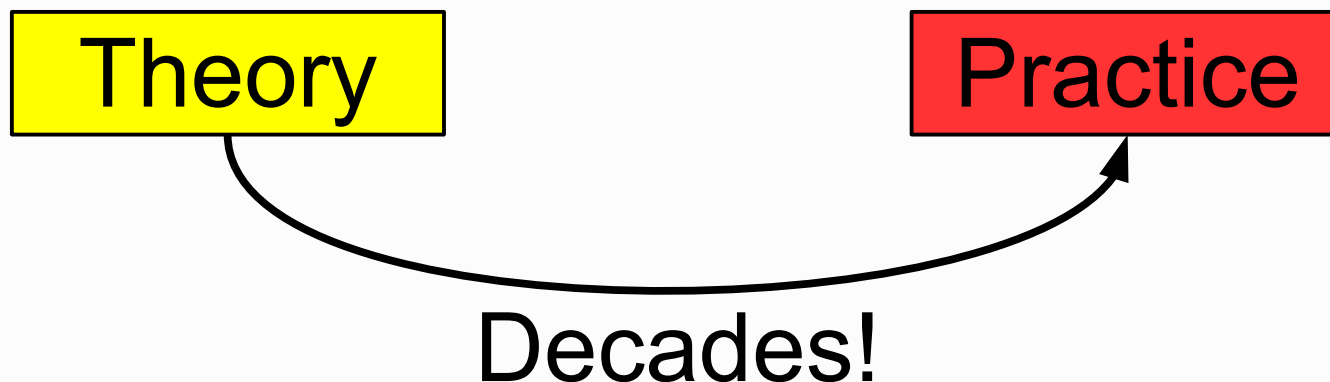
```
int i = "hello";
```

Type checking doesn't prevent **enough** errors

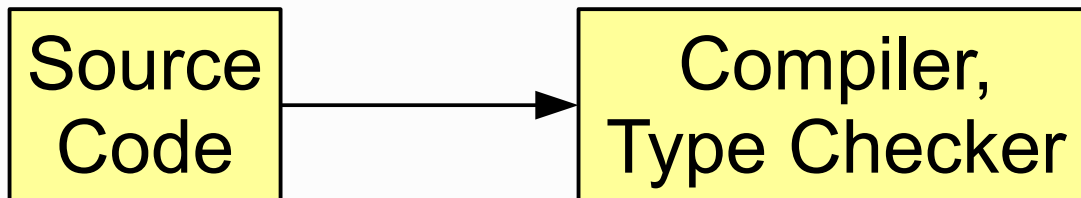
```
System.console().readLine();  
Collections.emptyList().add("one");  
dbStatement.executeQuery(userInput);
```

Better type systems can help!

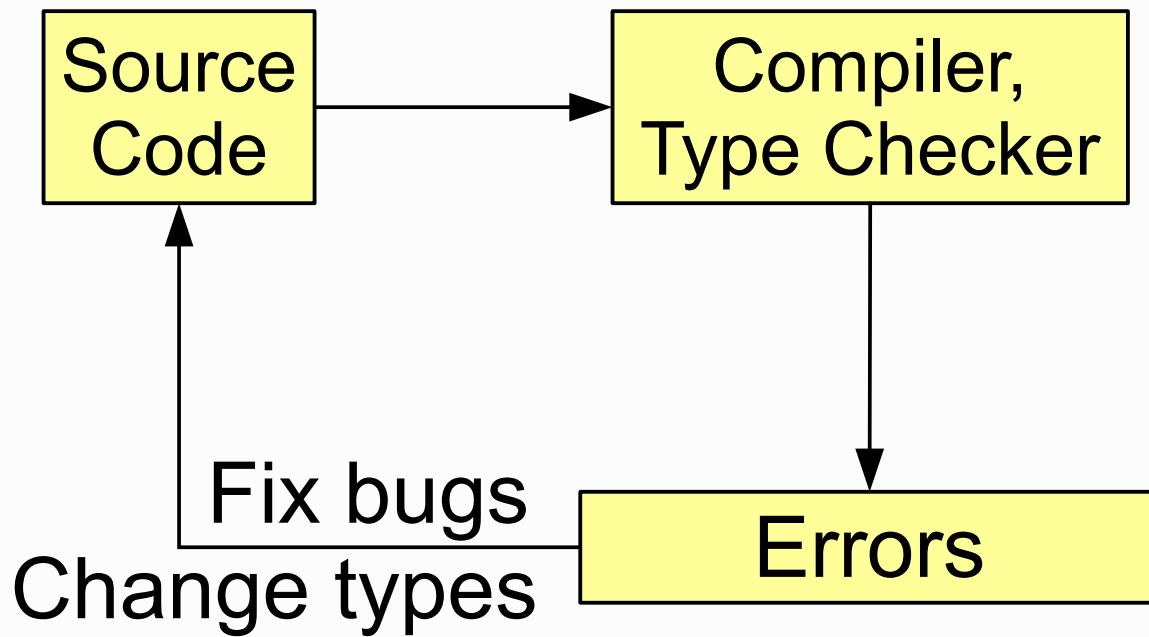
- Null-pointer exceptions [Fähndrich & Leino '03]
- Unwanted mutations [Tschantz & Ernst '05]
- Concurrency errors [Boyapati et al. '02, Cunningham et al. '07]
- ... many more!



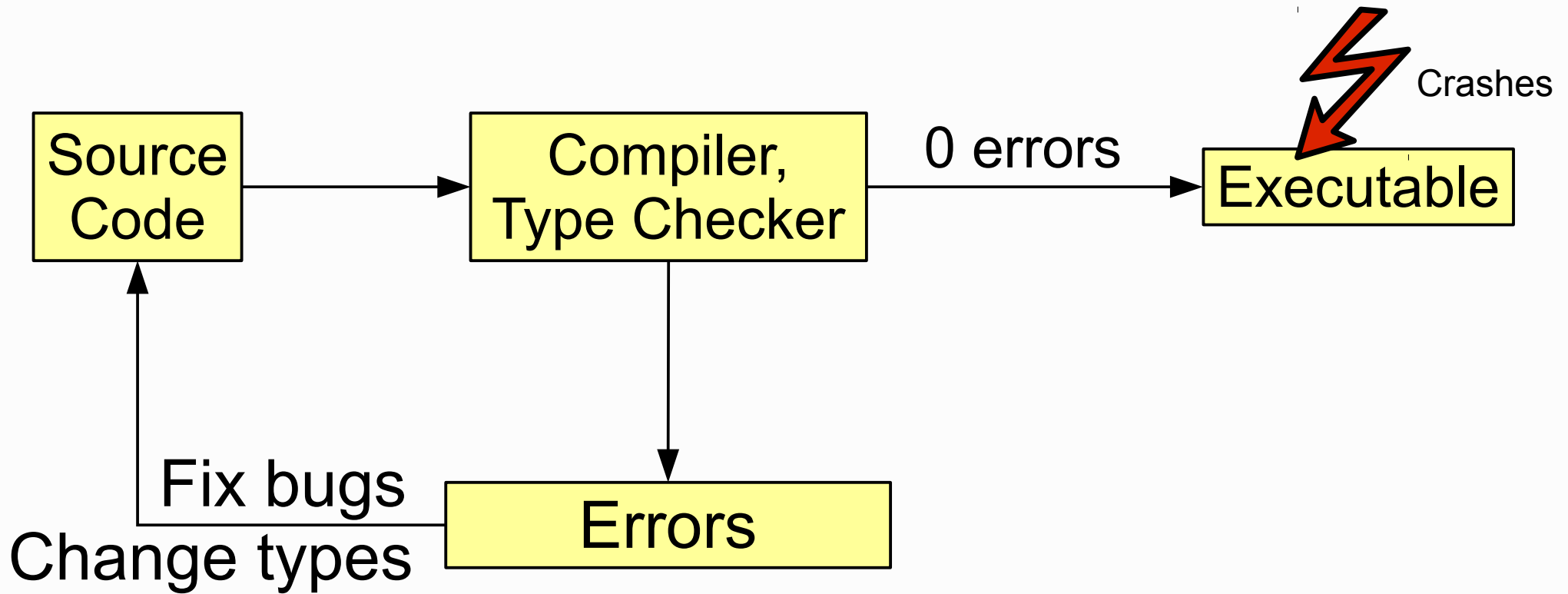
Type system



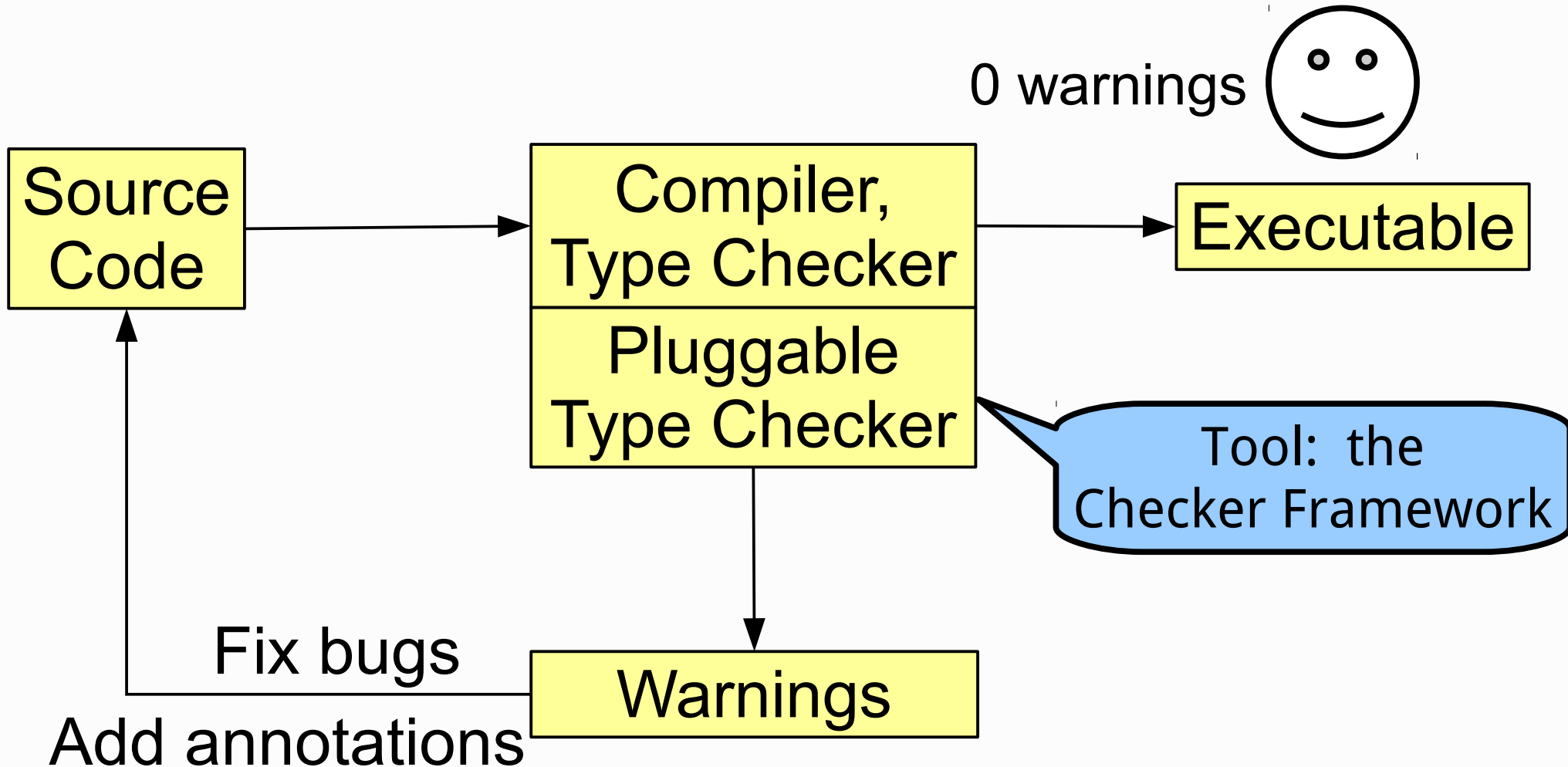
Type system



Type system



Pluggable type system

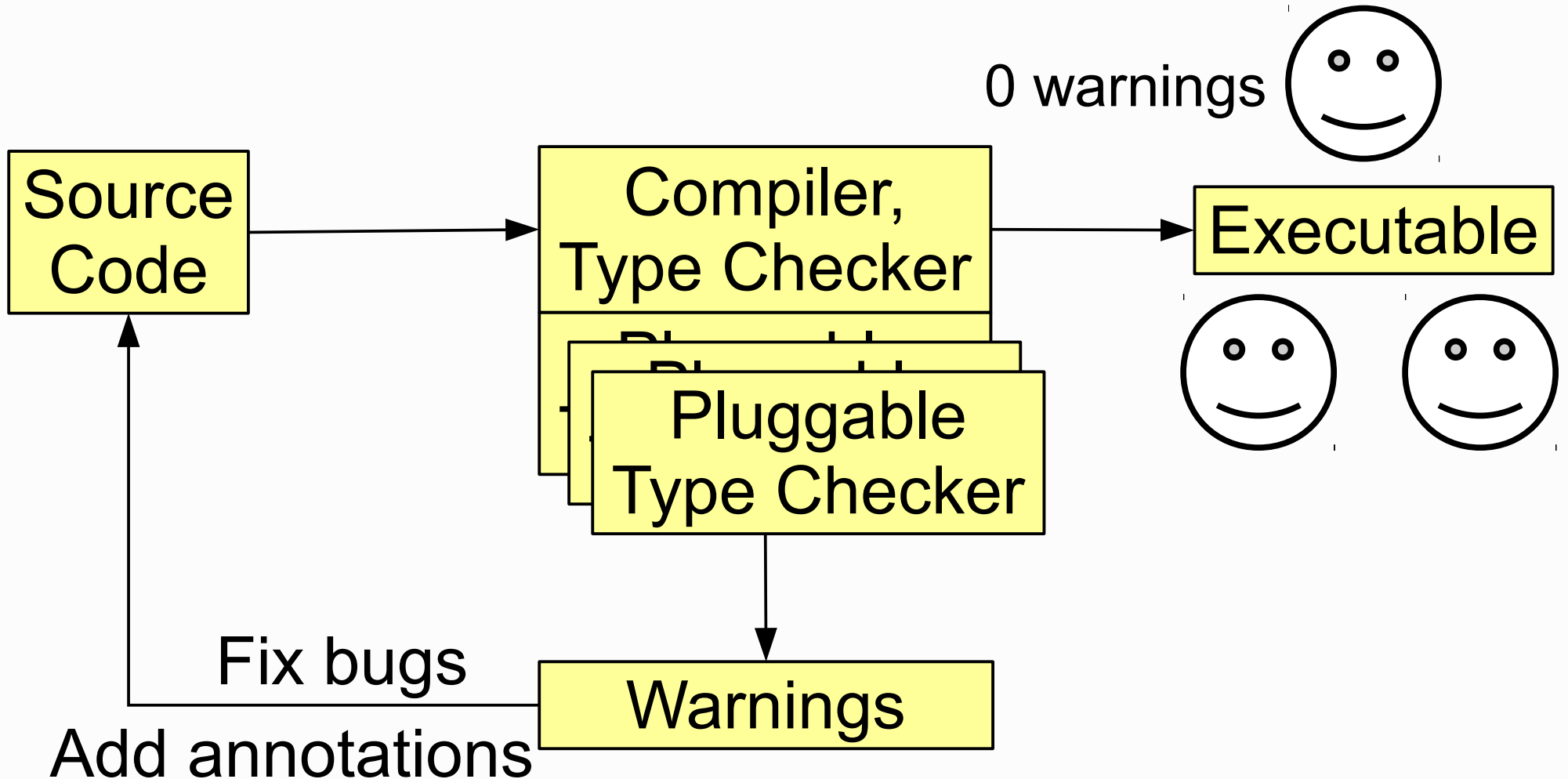


```
@Nullable Object myref;
```

```
...
```

```
myref.toString() // Warning!
```


Pluggable type systems



The Checker Framework

A framework for pluggable type checkers

Plugs a type checker into the Java compiler

Easy to use:

```
javac -processor EncryptionChecker ...
```

Ant, Maven, Eclipse

Free download:

<http://types.cs.washington.edu/checker-framework/>

Type annotation syntax

Annotations on all occurrences of types

```
@Untainted String query;  
List<@NonNull String> strings;  
myGraph = (@Immutable Graph) tmpGraph;  
class UnmodifiableList<T>  
    implements @ReadOnly List<@ReadOnly T> {}
```

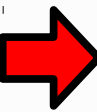
Stored in classfile

Handled by javac, javadoc, javap, ...

Part of Java 8, but you can use it with Java 5/6/7

- Backward compatible: write as a `/*@comment*/`

Outline

- Pluggable type systems
-  • Case studies & demo
- Type system examples
- How to build your own type system

Checker Framework experience

Type checkers reveal important latent bugs

- Ran on >3 million LOC of real-world code
- Found hundreds of user-visible bugs

Annotation overhead is low

- Mean 2.6 annotations per kLOC

Case study subject programs

Swing:

610 kLOC

Lucene:

Xerces:

OpenJDK (17

Daikon:

JabRef:

YaCy:

Google Colle

GanttProject

ASM:

Checker Fram

Annotation File Utilities:

17 kLOC

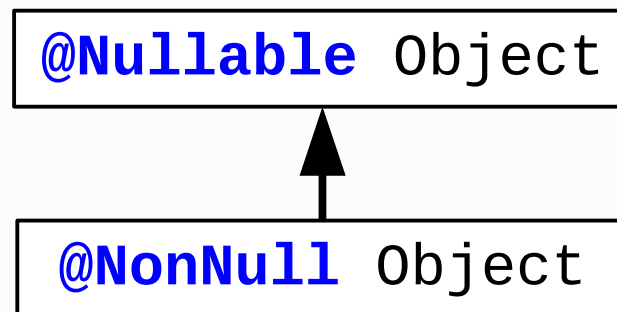
Approach:

- Write annotations
- Run compiler
- Each warning = program bug or incorrect annotation
- Fix the problem
- Rerun the compiler
- No more warnings →

Property guaranteed!

Null Pointer Exception type system

- Runtime behavior to prevent:
NullPointerException
- Legal operations:
Only dereference non-null references.
- Types of data:
@NonNull reference is never null.
@Nullable reference may be null.



Null-pointer crash in Google Collections

```
class ForMapWithDefault {  
    @Nullable Object defaultValue;  
    public int hashCode() {  
        return map.hashCode() +  
            defaultValue.hashCode();  
    }  
}
```

`java.lang.NullPointerException`

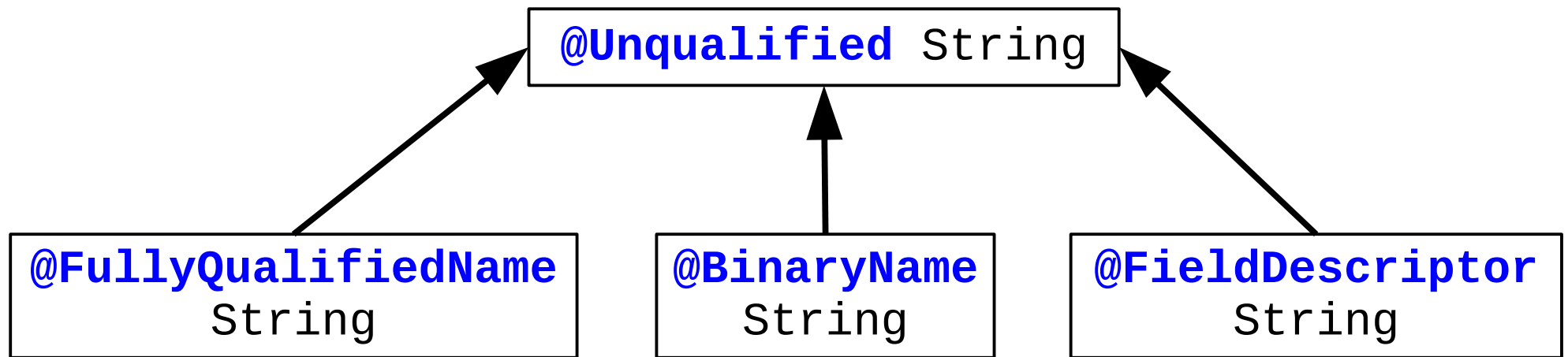
Found 9 such crashes, despite:

- 45000 tests (2/3 of the LOC)
- Uses FindBugs @Nullable annotations, no FindBugs warnings

Java signatures type system

- Runtime behavior to prevent:
`ClassNotFoundException`
- Legal operations:
`Class.forName` takes a binary name
- Types of data:
 - Unqualified strings: `"Hello, world!"`
 - Fully qualified names: `"package.Outer.Inner"`
 - Binary names: `"package.Outer$Inner"`
 - Field descriptors: `"Lpackage/Outer$Inner;"`

Java signatures type system



- **Types of data**

Unqualified strings:

"Hello, world!"

Fully qualified names:

"package.Outer.Inner"

Binary names:

"package.Outer\$Inner"

Field descriptors:

"Lpackage/Outer\$Inner;"

Java signature bugs found

Example from `java.lang.Class`:

```
static Class<?> forName(String className)
```

“Returns the Class object associated with the class or interface with the given string name. ...

Parameters:

className - the **fully qualified name** of the desired class”

`java.lang.ClassNotFoundException`

```
Class.forName("package.Outer.Inner")
```

```
Class.forName("package.Outer$Inner")
```

OK!

Signature String Checker

Found 11 crashing bugs in OpenJDK.

Found 13 crashing bugs in libraries.

Example annotations:

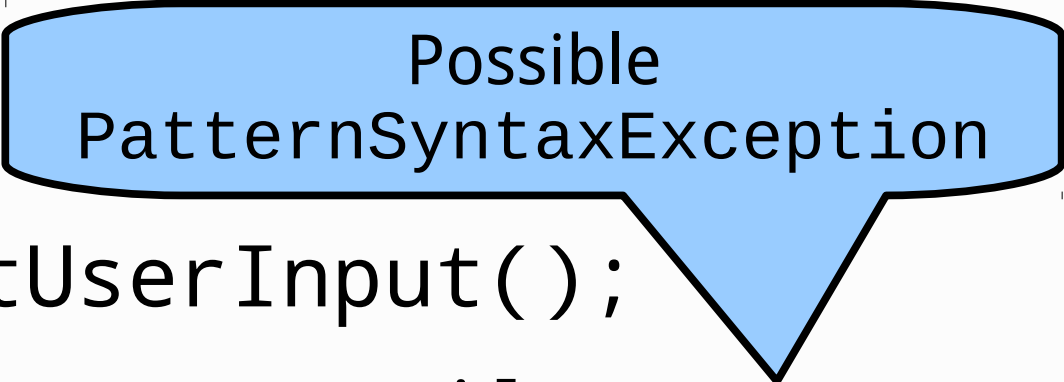
```
class Class<T> {  
    Class<?> forName(@BinaryName String className);  
    @BinaryName String getName();  
    @FullyQualifiedName String getCanonicalName();  
}
```

```
String name = myclass.getCanonicalName();  
Class.forName(name); // Warning
```

Regular expression type system

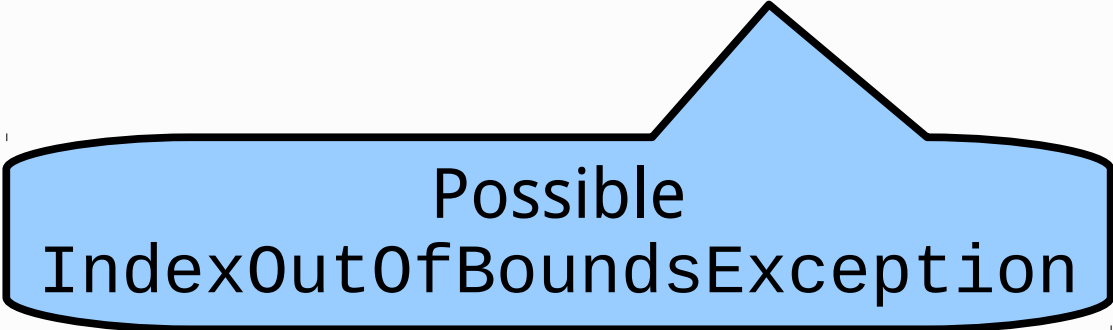
- **Runtime behavior to prevent:**
PatternSyntaxException and
IndexOutOfBoundsException.
- **Legal operations:**
Pattern.compile arg is a regex.
Matcher.group requires a regex with
minimum group count.
- **Types of data:**
@Regex String: is a regex, has a group count
@Unqualified String: might not be a regex

Example: Regular expressions



Possible
PatternSyntaxException


```
String regex = getUserInput();  
Pattern pat = Pattern.compile(regex);  
Matcher mat = pat.matcher(content);  
if (mat.matches()) {  
    println("Group: " + mat.group(4));  
}
```



Possible
IndexOutOfBoundsException

DEMO!

Outline

- Pluggable type systems
- Case studies & demo
-  • Type system examples
- How to build your own type system

What we saw so far

- What is pluggable type checking?
- What is the Checker Framework?
- Three examples:
 - Nullness Checker
 - Signature Checker
 - Regex Checker

Brainstorming new type checkers

1. What runtime behavior do you wish to prevent?
 - Properties of data, not of code structure, timing, ...
2. What operations are legal and illegal?
3. What types of data are there?

A sampling of type checkers

Property you care about:

- Tainting
- Java type signatures
- Null dereferences
- Concurrency
- Mutability & side effects
- Fake enumerations
- Internationalization
- Regular expressions
- Object encapsulation
- Energy efficiency
- Equality tests

Annotation to use:

@Tainted

@BinaryName

@Nullable

@Lock, @GuardedBy

@Immutable

@SwingCompassDirection

@Localized

@Regex

@Rep, @Peer, @Any

@Approx, @Precise

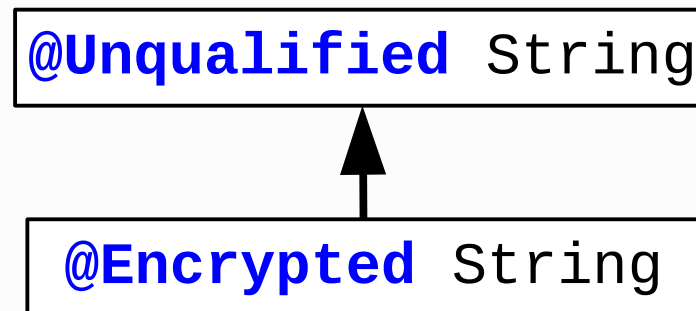
@Interned

Outline

- Pluggable type systems
- Case studies & demo
- Type system examples
- ➔ • How to build your own type system

Encryption type system

- **Runtime behavior to prevent:**
Plaintext data sent over the network
- **Legal operations:**
send() takes encrypted data
- **Types of data:**
@Encrypted String: definitely encrypted
@Unqualified String: might be unencrypted



Encrypted communication

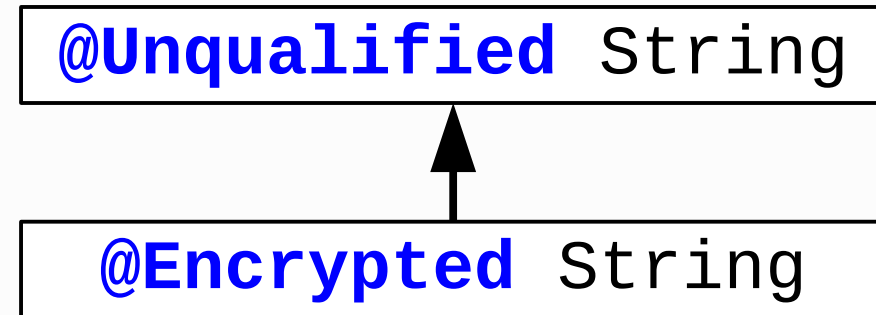
```
void send(@Encrypted String msg) {...}  
  
@Encrypted String msg1 = ...;  
send(msg1);    // OK  
  
String msg2 = ....;  
send(msg2);    // Warning!
```

Encrypted communication

```
void send(@Encrypted String msg) {...}  
  
@Encrypted String msg1 = ...;  
send(msg1);    // OK  
  
String msg2 = ....;  
send(msg2);    // Warning!
```

The complete checker:

```
@TypeQualifier  
@SubtypeOf(Unqualified.class)  
@Target(ElementType.TYPE_USE)  
public @interface Encrypted {}
```



Parts of a Type Checker

1. Type qualifiers

- What qualifiers exist and how do they relate?

@Unqualified String



@Encrypted String

2. Qualifier introduction rules

- Implicit qualifiers

"Hello, world!"

3. Type Rules

- Meaning of each operation including whether it is legal

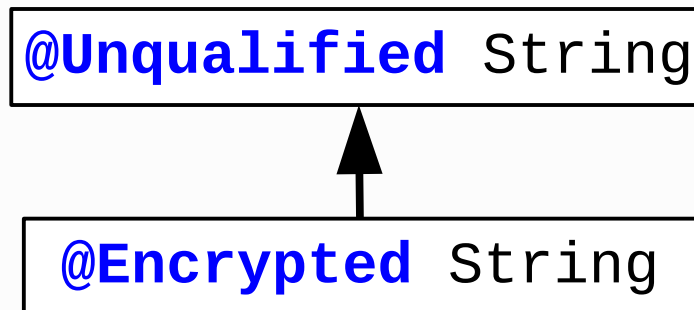
send(myCreditCardNum);

Part 1: Qualifiers and their relations

@TypeQualifier

@SubtypeOf(Unqualified.class)

public @interface **Encrypted** {}



Part 2: Implicit qualifiers

```
@TypeQualifier
@SubtypeOf({})
@ImplicitFor(
    trees = {Tree.Kind.NULL_LITERAL},
    typeNames = {Void.class})
public @interface Nullable {}
```

null



type is @Nullable

Part 3: Type rules

- Default AST visitor enforces standard object-oriented typing rules
 - Assignments, method calls, returns
 - Method overriding
 - Type well-formedness
- Customize where needed

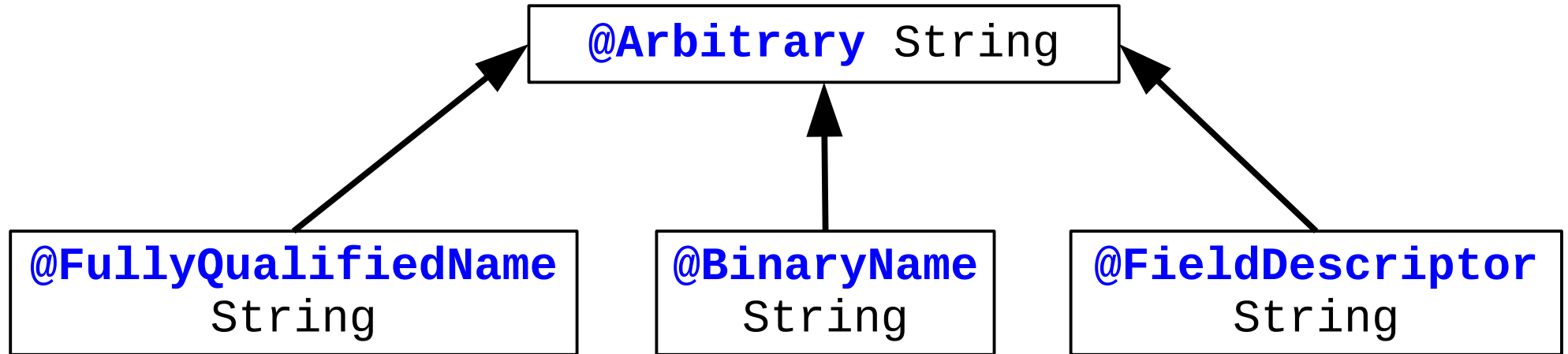


x must be @NonNull

x.f

```
@Override public Void  
visitMemberSelect(MemberSelectTree node, void p) {  
    checkForNullability(node.getExpression(),  
                        "dereference.of.nullable");  
    return super.visitMemberSelect(node, p);  
}
```

Java signatures type system



- **Types of data:**

Arbitrary strings:

"Hello, world!"

Fully qualified names:

"package.Outer.Inner"

Binary names:

"package.Outer\$Inner"

Field descriptors:

"Lpackage/Outer\$Inner;"

Signature String Checker

Type Qualifiers

```
@TypeQualifier
@SubtypeOf({Unqualified.class})
@ImplicitFor(stringPatterns="^[A-Za-z_][A-Za-z_0-9]*(\\.[A-Za-z_][A-Za-z_0-9]*)*(\\[\\])*$" )
public @interface FullyQualifiedName {}

@TypeQualifier
@SubtypeOf({Unqualified.class})
@ImplicitFor(stringPatterns="^[A-Za-z_][A-Za-z_0-9]*(\\.[A-Za-z_][A-Za-z_0-9]*)*(\\$[A-Za-z_][A-Za-z_0-9]*)?(\\[\\])*$" )
public @interface BinaryName {}

@TypeQualifier
@SubtypeOf({Unqualified.class})
@ImplicitFor(stringPatterns="^[\\[BCDFIJSZ]|L[A-Za-z_][A-Za-z_0-9]*(/[A-Za-z_][A-Za-z_0-9]*)*(\\$[A-Za-z_][A-Za-z_0-9]*)?(;|;)$" )
public @interface FieldDescriptor {}
```

```
@TypeQualifier
@SubtypeOf({BinaryName.class, FullyQualifiedName.class})
public @interface SourceName {}

@TypeQualifier
@SubtypeOf({Unqualified.class})
public @interface MethodDescriptor {}

@TypeQualifier
@SubtypeOf({BinaryName.class, FieldDescriptor.class, SourceName.class, FullyQualifiedName.class, MethodDescriptor.class})
@ImplicitFor(trees={Tree.Kind.NULL_LITERAL})
public @interface SignatureBottom {}
```

```
@TypeQualifiers({BinaryName.class, FullyQualifiedName.class, SourceName.class, FieldDescriptor.class, Unqualified.class, MethodDescriptor.class, SignatureBottom.class})
public final class SignatureChecker
extends BaseTypeChecker {}
```

Type Checker

Complex checkers are possible

Nullness Checker is actually 3 checkers:

- Nullness itself
- Object initialization
- Keys in maps
 - Infers when `Map.get` returns non-null

Powerful analyses are built in

Example: flow-sensitive type inference

```
@Nullable Object x = ...;  
x.toString();           // Warning!  
if (x != null) {  
    x.toString();       // OK  
}
```

Contributes to low annotation overhead

More tools

- Inference tools
- Annotation tools to insert annotations
- Specification files for libraries

The Checker Framework

- Powerful framework for type checkers
- Built-in checkers are ready to use
 - Identify many latent bugs
- Easy to create your own type system
 - 3 rules for designing a type system

Improve your code today!

<http://types.cs.washington.edu/checker-framework/>