

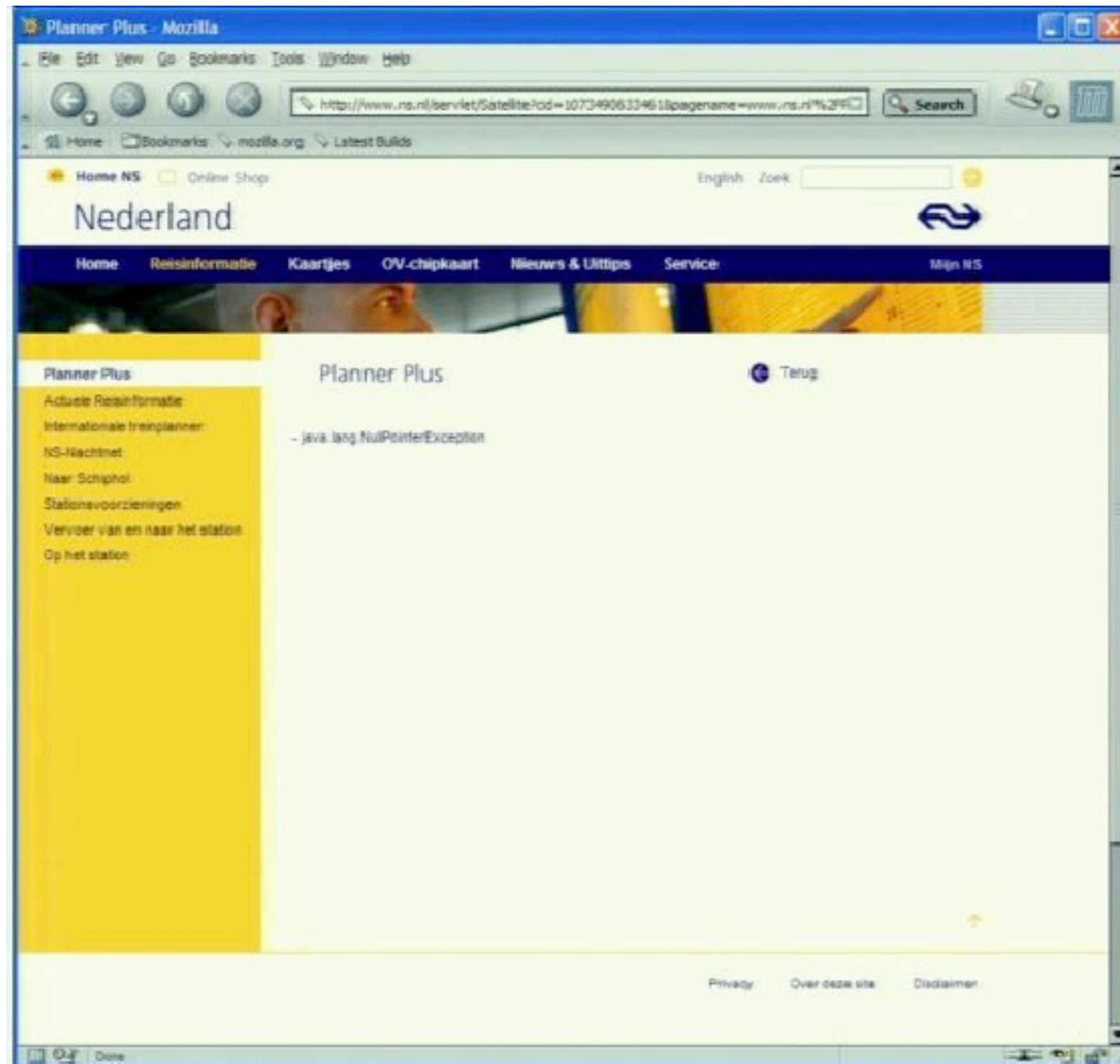
# Preventing bugs with pluggable type-checking



Mahmood Ali  
Checkers Developer  
MIT

```
print(@Readonly Object x) {  
    List<@Nonnull String> lst;  
    ...  
}
```

# Motivation



# Motivation

- `System.console().readLine();`
  - `NullPointerException`
- `Collections.emptyList().add("One");`
  - `UnsupportedOperationException`

# Motivation: Silent Errors

- ```
Date date = new Date(0);  
myMap.put(date, "Java Epoch");  
date.setYear(70);  
myMap.put(date, "Linux Epoch");
```
- Corrupted Map
- ```
dbStatement.executeQuery(userQuery)
```
- SQL Injection Attack

# Java Type System

- **Problem:** It cannot express important properties
  - Nullness, interning, mutation, encryption, tainting, etc

# Java Type System

- **Problem:** It cannot express important properties
  - Nullness, interning, mutation, encryption, tainting, etc
- **Idea!**

```
@Immutable Date date = new Date(0);  
date.setTime(70);    // compile-time error
```

# Pluggable Type Systems

# Pluggable Type Systems

- Design a type system to solve a specific problem
- Annotate your code with type qualifiers
  - or use a type inference tool!
- Use Type Checkers to detect violations (errors)



# Agenda

## **Type Qualifiers**

### **Checkers Available Today**

Checkers demo

User experience

### **Writing a new checker**

Taintness checker

# Type Qualifiers

- Java 7 annotations written on types

```
@Untainted String query;
```

```
List<@NonNull String> names;
```

- Backward compatible: within comments

- **Use it today!**

```
class UnmodifiableList<T>  
    implements /*@ReadOnly*/ List<T> { ... }
```

# Type Qualifiers: Benefits

- **Improve documentation**
- **Find bugs** in programs
- Guarantee the **absence of errors**
- Aid compilers and analysis tools
- Reduce the need for assertions and run-time checks

# Agenda

## **Type Qualifiers**

### **Checkers Available Today**

Checkers demo

User experience

### **Writing a new checker**

Taintness checker

# Checkers Available Today

- @NotNull: null dereference
- @Interned: incorrect equality tests
- @ReadOnly: incorrect mutation and side-effects
- Many other simple checkers
  - Security: encryption, tainting, access control
  - Encoding: SQL, URL, ASCII/Unicode
- ...

# Using Checkers

- Designed as compiler plug-ins (i.e. annotation processors)
- Use familiar error messages

```
% javac -processor NullnessChecker MyFile.java
```

```
MyFile.java:9: incompatible types.  
found    : @Nullable String  
required: @NonNull String  
    nonNull = nullable;  
              ^
```

# Checkers Demo

## Nullness and Mutation Demo

# Checkers are Featureful

- Full type systems: assignments, overriding, etc
- Polymorphic (Java Generics)
- Flow-sensitive type qualifier inference
- Qualifier defaults
- Warning suppression



# Checkers are Effective

- Scales to > 200,000 LOC
- Each checker found errors in each code base it ran on
- **Nullness Tool Comparison: 4 KLOC code base**

	Bugs		False Positives	Annotations
	Found	Missed		
<b>Null Checker</b>	8	0	4	35
<b>FindBugs</b>	0	8	1	0
<b>Jlint</b>	0	8	8	0
<b>PMD</b>	0	8	0	0

False Positives suppressed by an annotation or assertion

# Checkers are Usable

- Tool support: javac, Ant, Eclipse, Netbeans
- Not too verbose
  - @NonNull: 1 annotation per 75 lines
  - @Interned: 124 annotations in 220 KLOC revealed 11 errors
- Fewer annotations in new code
- Inference tools: nullness, mutability

# Agenda

## **Type Qualifiers**

### **Checkers Available Today**

Checkers demo

User experience

### **Writing a new checker**

Taintness checker

# SQL Injection Attack

- SQL query constructed using unfiltered user input

- Server code:

```
query = "SELECT * FROM users "  
        + "WHERE name '" + userInput + "'";
```

- User inputs: `a' or 't'='t`

- Result

```
query => SELECT * FROM users  
         WHERE name='a' or 't'='t';
```

# Taintness Checker: Code

- Full Checker:

```
@TypeQualifier
@SubtypeOf(Unqualified.class)
@ImplicitFor(trees = { STRING_LITERAL })
public @interface Untainted { }
```

- How to use it

```
javac -processor BasicChecker \
      -Aquals=Untainted javaFiles ...
```

# Checker Demos

## Taintness Checker

# Summary

- Pluggable Type-Checkers
  - Featureful, Effective, and Usable
- Programmers can
  - find and prevent bugs at compile time
  - obtain guarantees that program is free of certain errors
  - create custom qualifiers and type-checkers

Download:

<http://pag.csail.mit.edu/jsr308>