Java
Object Orientation (OBJ)

Mario Alviano

University of Calabria, Italy

A.Y. 2016/2017

Limit accessibility of fields

- ✗ Invariants cannot be enforced for public nonfinal fields
- ✗ Similar for final fields that reference a mutable object
- ✓ Prefer private or package-private visibility
- ✗ Also protected may be insufficient
- ✓ Exception: classes like structs use public visibility

```
https:
//www.securecoding.cert.org/confluence/display/
java/OBJ01-J.+Limit+accessibility+of+fields
```

Preserve dependencies in subclasses when changing superclasses

- ✓ Always check subclasses of modified classes
- ✓ Preserve invariants

```
https://www.securecoding.cert.org/confluence/
display/java/OBJ02-J.+Preserve+dependencies+in+
subclasses+when+changing+superclasses
```

Prevent heap pollution

- ✗ Do not mix generically typed code with raw typed code
- ✓ Prefer generically typed code
- ✓ Use wrapper collections to interface legacy code
- ✗ Avoid variadic methods

```
https://www.securecoding.cert.org/confluence/
display/java/OBJ03-J.+Prevent+heap+pollution
```

Provide mutable classes with copy functionality to safely allow passing instances to untrusted code

- ✗ Do not trust mutable objects passed to untrusted methods
- ✓ Always pass copies to untrusted methods
- ✓ Provide copy constructors or implement `clone()`

```
https://www.securecoding.cert.org/confluence/
display/java/OBJ04-J.+Provide+mutable+classes+
with+copy+functionality+to+safely+allow+passing+
instances+to+untrusted+code
```

Do not return references to private mutable class members

- ✗ You compromise encapsulation
- ✓ Make copies of mutable objects
- ✓ Make copies of containers

```
https://www.securecoding.cert.org/confluence/
display/java/OBJ05-J.+Do+not+return+references+to+
private+mutable+class+members
```

Defensively copy mutable inputs and mutable internal components

- ✗ Do not introduce time-of-check, time-of-use (TOCTOU) vulnerabilities
- ✓ Make copies of untrusted mutable input objects
- ✓ Prefer copy constructors to `clone()`

```
https://www.securecoding.cert.org/confluence/
display/java/OBJ06-J.+Defensively+copy+mutable+
inputs+and+mutable+internal+components
```

Sensitive classes must not let themselves be copied

- ✗ Not having a copy constructor or a `clone()` method doesn't mean it can't be copied
- ✓ Prevent subclassing of classes and methods

```
https://www.securecoding.cert.org/confluence/
display/java/OBJ07-J.+Sensitive+classes+must+not+
let+themselves+be+copied
```

Do not expose private members of an outer class from within a nested class

- ✗ Private fields and methods of a class having a public inner class are weakened to package visibility
- ✓ Use public inner classes only if you understand this aspect
- ✓ Prefer private inner classes

```
https:
//www.securecoding.cert.org/confluence/display/
java/OBJ08-J.+Do+not+expose+private+members+of+an+
outer+class+from+within+a+nested+class
```

Compare classes and not class names

- ✗ Comparing class names may expose to mix-and-match attack
- ✓ Same name but different class loader means different classes
- ✓ Always compare the `class` object

```
https:
//www.securecoding.cert.org/confluence/display/
java/OBJ09-J.+Compare+classes+and+not+class+names
```

Do not use public static nonfinal fields

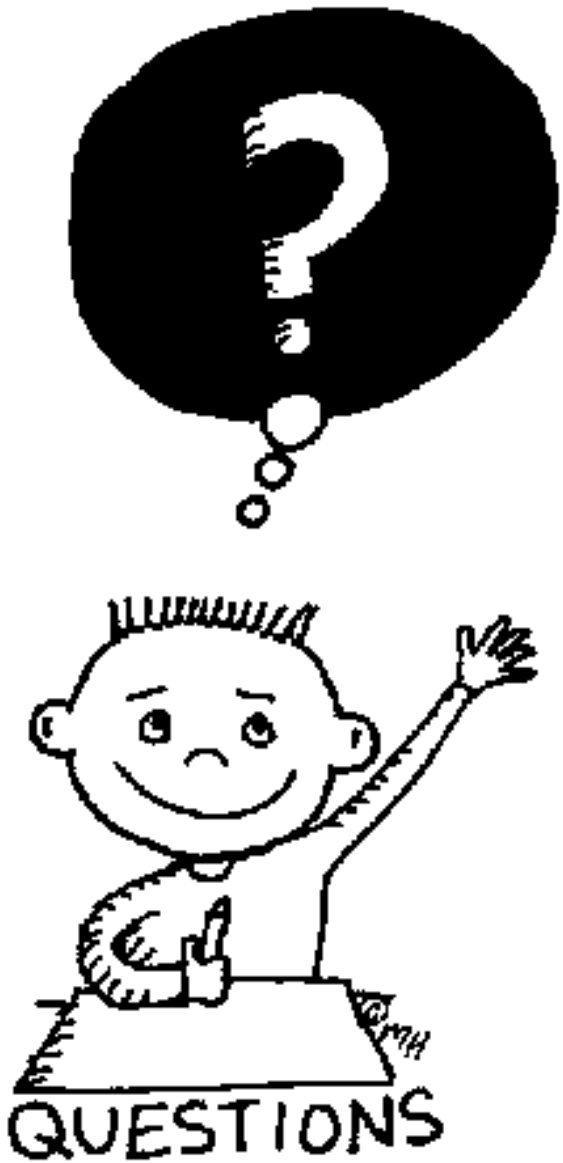
- ✗ Everyone can change them
- ✓ Add the `final` keyword

```
https://www.securecoding.cert.org/confluence/
display/java/OBJ10-J.+Do+not+use+public+static+
nonfinal+fields
```

Be wary of letting constructors throw exceptions

- ✗ Do not leave partially initialized objects
- ✓ Prevent Finalized Attacks
- ✓ Use public and private constructors
- ✓ Flag fully initialized objects

```
https://www.securecoding.cert.org/confluence/
display/java/OBJ11-J.+Be+wary+of+letting+
constructors+throw+exceptions
```



END OF THE LECTURE