Java
Methods (MET)

Mario Alviano

University of Calabria, Italy

A.Y. 2016/2017

Validate method arguments

- ✗ Don't trust caller outside the trust boundary
- ✗ Don't trust *paladins* of efficiency
- ✓ You must perform callee validation on all public methods of your library
- ✓ You should perform callee validation on other methods whose arguments come from public methods

```
https://www.securecoding.cert.org/confluence/
display/java/MET00-J.+Validate+method+arguments
```

Never use assertions to validate method arguments

- ✗ Assentions are disabled in release bytecode
- ✗ Not validating at all is even worse
- ✓ Throw appropriate run-time exceptions

```
https://www.securecoding.cert.org/confluence/
display/java/MET01-J.+Never+use+assertions+to+
validate+method+arguments
```

Do not use deprecated or obsolete classes or methods

- ✗ If they have been deprecated, there must be a reason!
- ✗ Deprecated methods may leave your system inconsistent, and may be removed in the future
- ✓ Check the documentation for alternatives
- ✓ Obsolescence is also indicated in the documentation

```
https://www.securecoding.cert.org/confluence/
display/java/MET02-J.+Do+not+use+deprecated+or+
obsolete+classes+or+methods
```

Methods that perform a security check must be declared private or final

- ✗ Overriding may bypass security checks
- ✓ Private methods cannot be overridden
- ✓ Final methods cannot be overridden
- ✓ Methods of final classes cannot be overridden

```
https://www.securecoding.cert.org/confluence/
display/java/MET03-J.+Methods+that+perform+a+
security+check+must+be+declared+private+or+final
```

Do not increase the accessibility of overridden or hidden methods

- ✗ Protected and default accessibility may be increased
- ✓ Don’t use them for sensitive operations
- ✓ Declare them final to prevent overriding

```
https://www.securecoding.cert.org/confluence/
display/java/MET04-J.+Do+not+increase+the+
accessibility+of+overridden+or+hidden+methods
```

Ensure that constructors do not call overridable methods

- ✗ Overridden methods may access a partially initialized object
- ✓ Only call private or final methods in constructors

```
https://www.securecoding.cert.org/confluence/
display/java/MET05-J.+Ensure+that+constructors+do+
not+call+overridable+methods
```

Similar rule

Do not invoke overridable methods in clone()

Never declare a class method that hides a method declared in a superclass or superinterface

- ✗ Static methods cannot be overridden, in case they are hidden
- ✓ Hidden methods are determined during the compilation
- ✓ Overridden methods are determined at runtime

```
https://www.securecoding.cert.org/confluence/
display/java/MET07-J.+Never+declare+a+class+
method+that+hides+a+method+declared+in+a+
superclass+or+superinterface
```

Preserve the equality contract when overriding the equals() method

- ✓ `equals()` must be reflexive, symmetric and transitive
- ✓ It must be consistent: multiple invocations must return the same value
- ✓ It must return false if its argument is `null`

```
https://www.securecoding.cert.org/confluence/
display/java/MET08-J.+Preserve+the+equality+
contract+when+overriding+the+equals%28%29+method
```

Classes that define an equals() method must also define a hashCode() method

- ✗ The default `hashCode()` method usually returns an hash value computed on the reference address
- ✓ Invariant: `x.equals(y) == true` implies `x.hashCode() == y.hashCode()`
- ✓ If you override `equals()`, you must override `hashCode()`

```
https:
//www.securecoding.cert.org/confluence/display/
java/MET09-J.+Classes+that+define+an+equals%28%29+
method+must+also+define+a+hashCode%28%29+method
```

Follow the general contract when implementing the compareTo() method

- ✓ `compareTo()` must define a partial order
- ✓ If possible, define a total order

```
https://www.securecoding.cert.org/confluence/
display/java/MET10-J.+Follow+the+general+contract+
when+implementing+the+compareTo%28%29+method
```

Ensure that keys used in comparison operations are immutable

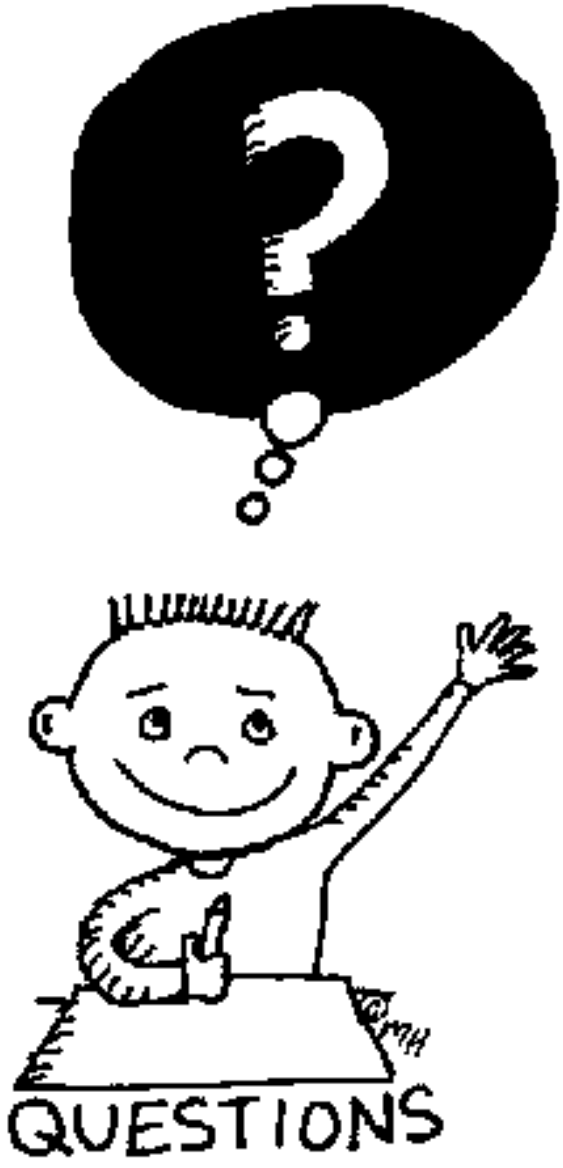
- ✗ Change a key used in map, lose the object
- ✓ Override `equals()` and `hashCode()`, and use only immutable fields
- ✓ Be aware of serialization and deserialization

```
https://www.securecoding.cert.org/confluence/
display/java/MET11-J.+Ensure+that+keys+used+in+
comparison+operations+are+immutable
```

Do not use finalizers

- ✗ May result into deadlocks, starvations, inconsistencies
- ✗ Do not extend classes with finalizers
- ✓ Use composition instead

```
https://www.securecoding.cert.org/confluence/
display/java/MET12-J.+Do+not+use+finalizers
```



END OF THE LECTURE