ASPECTS AND CLASS-BASED SECURITY

A Survey of Interactions between Advice Weaving and the Java 2 Security Model

AOP and Advice Weaving

```
package trusted;

class IO {
  void perform(File f) {
    if (f.exists()) {
      // perform file I/O
    }
  }
}
```

AOP and Advice Weaving

```
package trusted;

class IO {
  void perform(File f) {
    if (f.exists()) {
      // perform file I/O
    }
  }
}
```

```
package untrusted;

aspect Manipulation {
  around(File f) :
      call(boolean File.exists())
      && target(f) {
    return f.delete();
  }
}
```

AOP and Advice Weaving

```
package trusted;

class IO {
  void perform(File f) {
    if (f.exists()) {
      // perform file I/O
    }
  }
}
```

```
package untrusted;

class Manipulation {
  static boolean around$1(File f) {


    return f.delete();
  }
}
```

AOP and Advice Weaving

```
package trusted;

class IO {
  void perform(File f) {
    if (Manipulation.around$1(f)) {
      // perform file I/O
    }
  }
}
```

```
package untrusted;

class Manipulation {
  static boolean around$1(File f) {


    return f.delete();
}
```

Advice Weaving

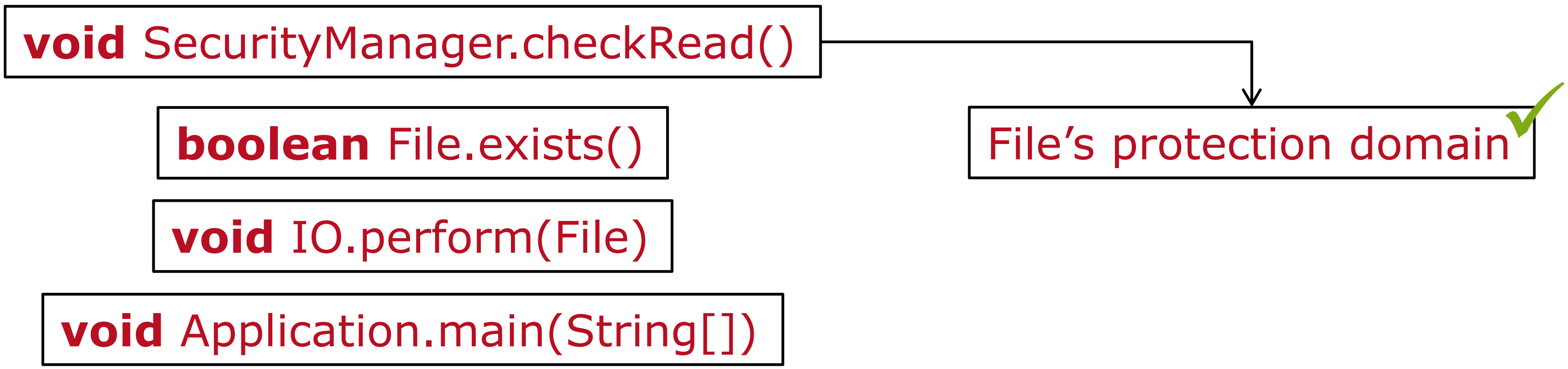
Questions Answered

I. How to control access, e.g., to delete()?

II. How to enforce namespace separation?

III. How to control advice weaving, e.g., at class IO?

Access Control



void SecurityManager.checkRead()

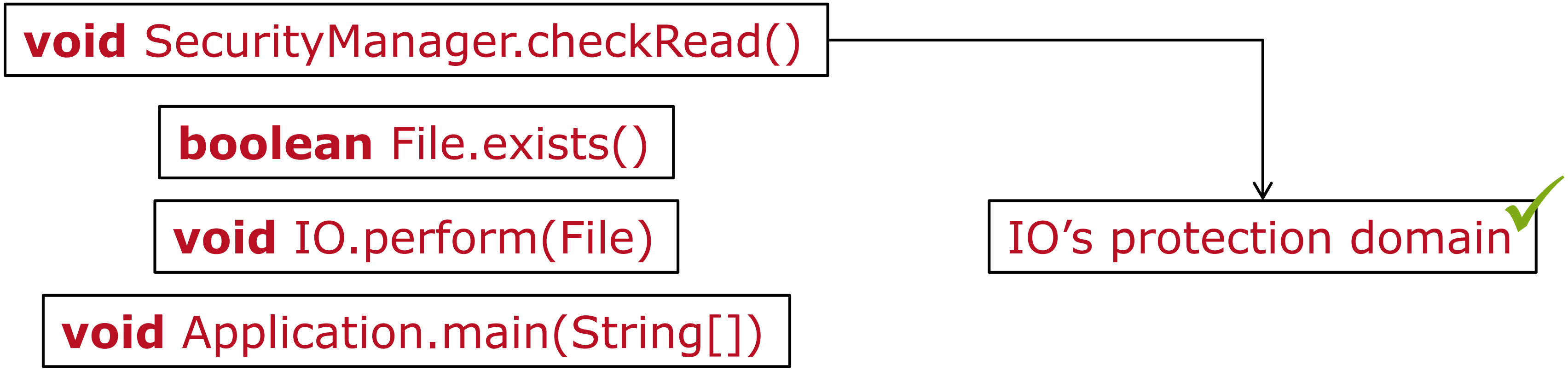
boolean File.exists()

void IO.perform(File)

void Application.main(String[])

File’s protection domain ✓

Access Control



void SecurityManager.checkRead()

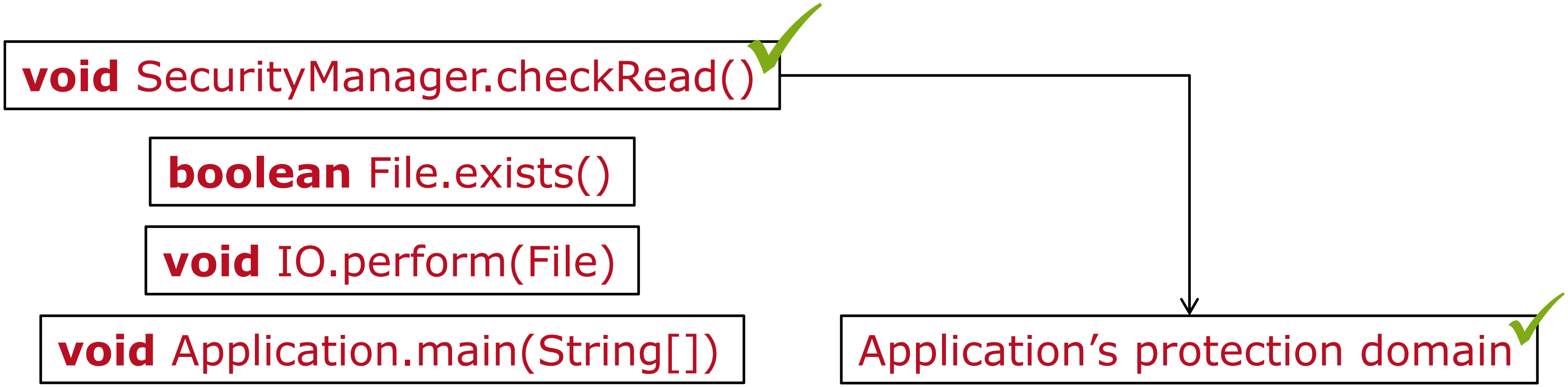
boolean File.exists()

void IO.perform(File)

void Application.main(String[])

IO’s protection domain ✓

Access Control



void SecurityManager.checkRead() ✓

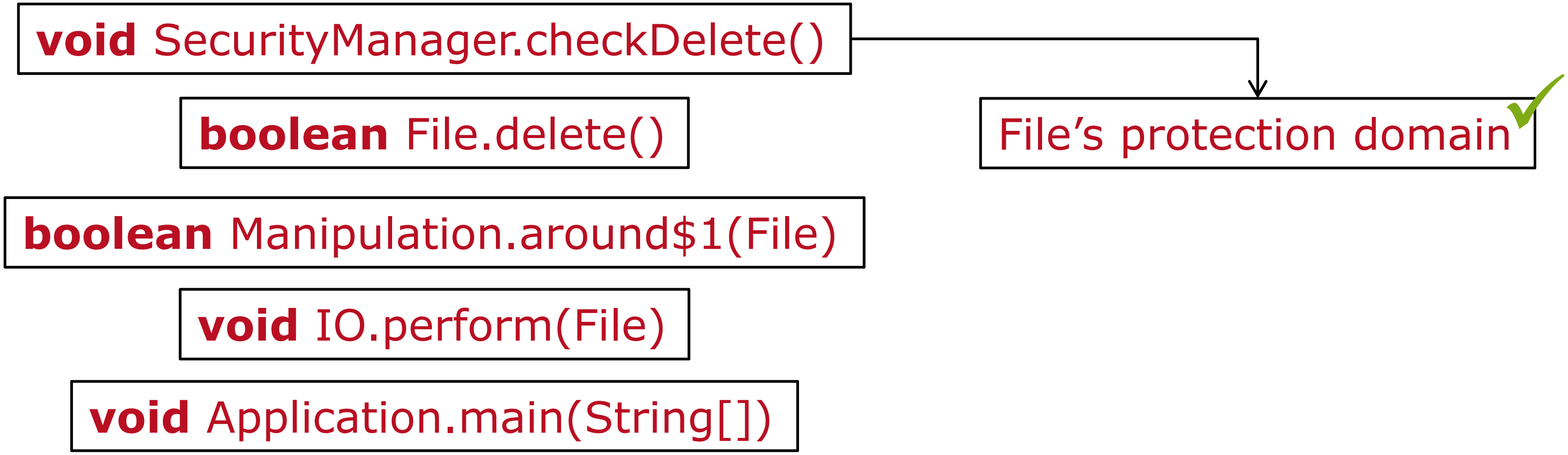
boolean File.exists()

void IO.perform(File)

void Application.main(String[])

Application’s protection domain ✓

Access Control (cont'd)



void SecurityManager.checkDelete()

boolean File.delete()

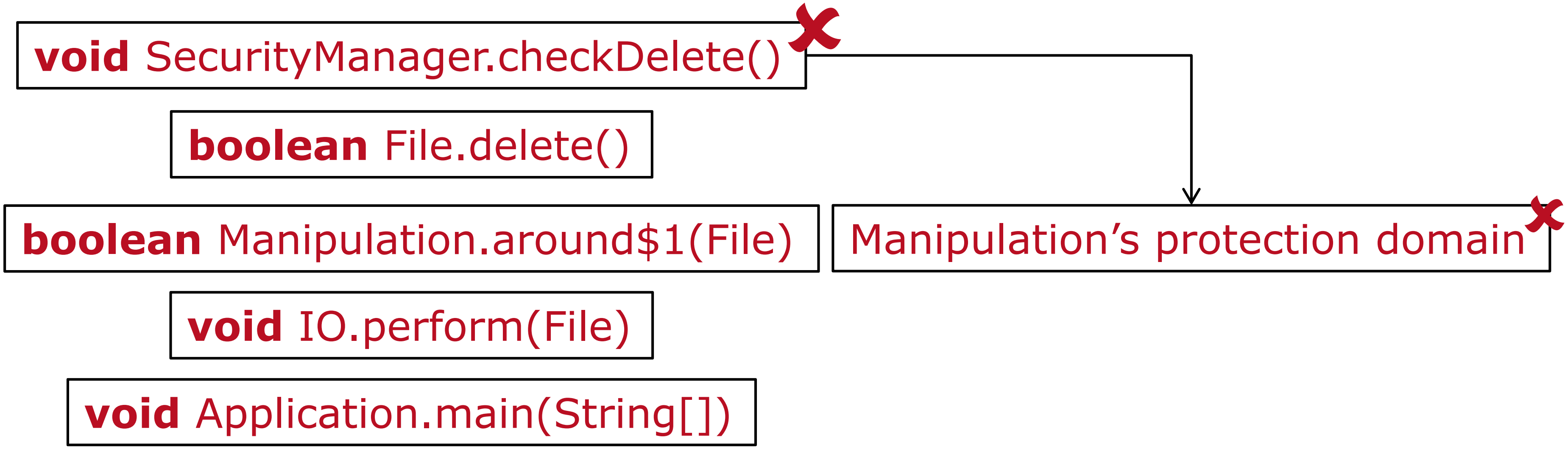
boolean Manipulation.around$1(File)

void IO.perform(File)

void Application.main(String[])

File’s protection domain ✓

Access Control (cont'd)



void SecurityManager.checkDelete()

boolean File.delete()

boolean Manipulation.around$1(File)

Manipulation’s protection domain

void IO.perform(File)

void Application.main(String[])

Advice Weaving and Inlining

```
package trusted;

class IO {
  void perform(File f) {
    if (Manipulation.around$1(f)) {
      // perform file I/O
    }
  }
}
```

```
package untrusted;

class Manipulation {
  static boolean around$1(File f) {


    return f.delete();
  }
}
```

Advice Weaving and Inlining

```
package trusted;

class IO {
  void perform(File f) {
    if (f.delete()) {
      // perform file I/O
    }
  }
}
```

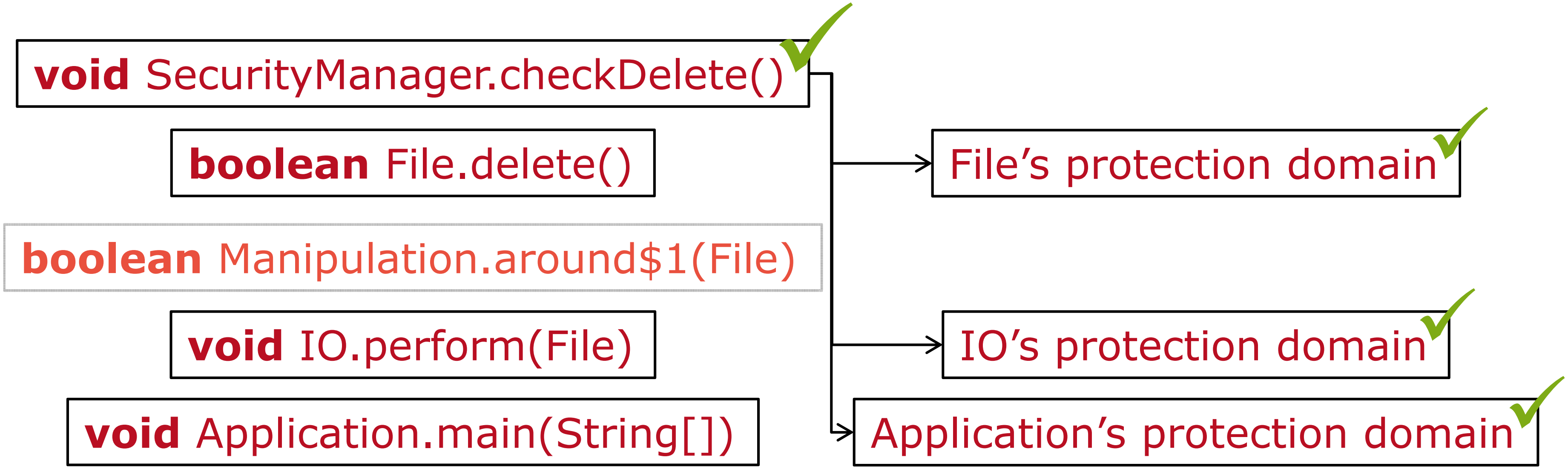
Advice Inlining

```
package untrusted;

class Manipulation {
  static boolean around$1(File f) {


    return f.delete();
  }
}
```

Access Control (cont’d)



void SecurityManager.checkDelete() ✓

boolean File.delete()

boolean Manipulation.around$1(File)

void IO.perform(File)

void Application.main(String[])

File’s protection domain ✓

IO’s protection domain ✓

Application’s protection domain ✓

Answer

I. Advice must be assigned its aspect’s protection domain.

Questions Answered

I. How to control access, e.g., to delete()?

II. How to enforce namespace separation?

III. How to control advice weaving, e.g., at class IO?

Namespace Separation

- Each Class has an associated class loader.
- It determines which classes are visible to it.
- Advice inlining can sever this association; the class loader of the woven class is used.

Answers (cont'd)

II. Advice must use its aspect's class loader to resolve references.

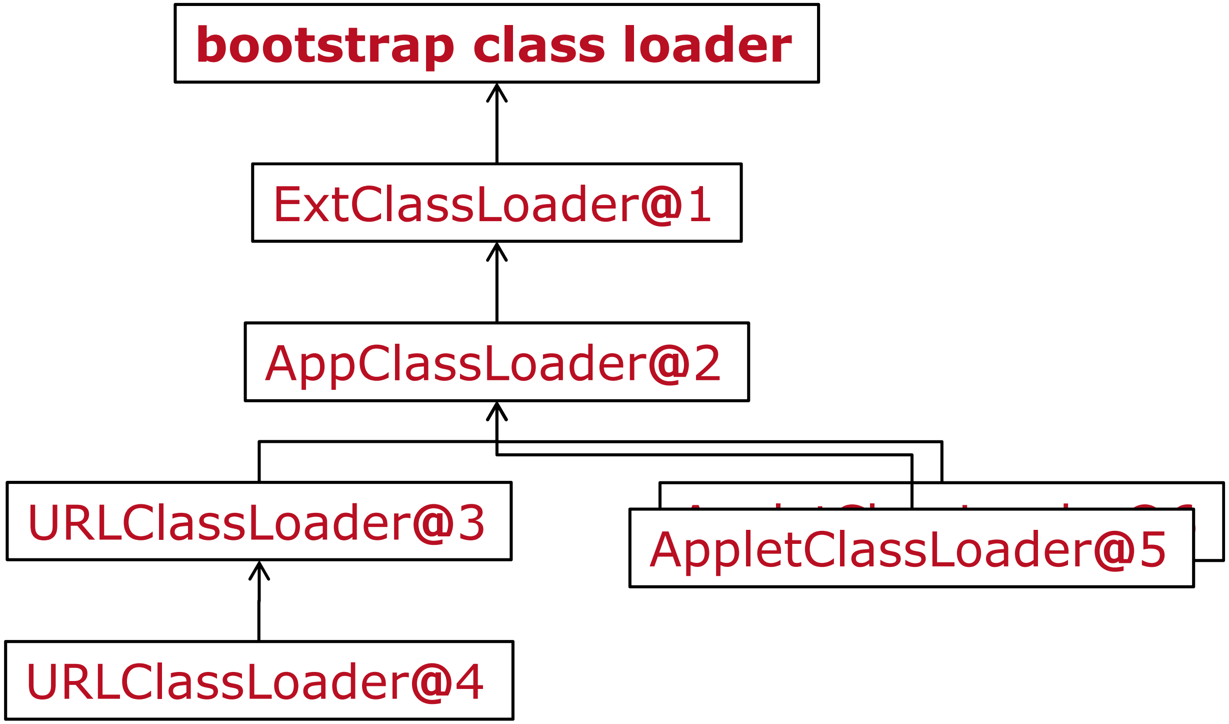
Questions Answered

I. How to control access, e.g., to delete()?

II. How to enforce namespace separation?

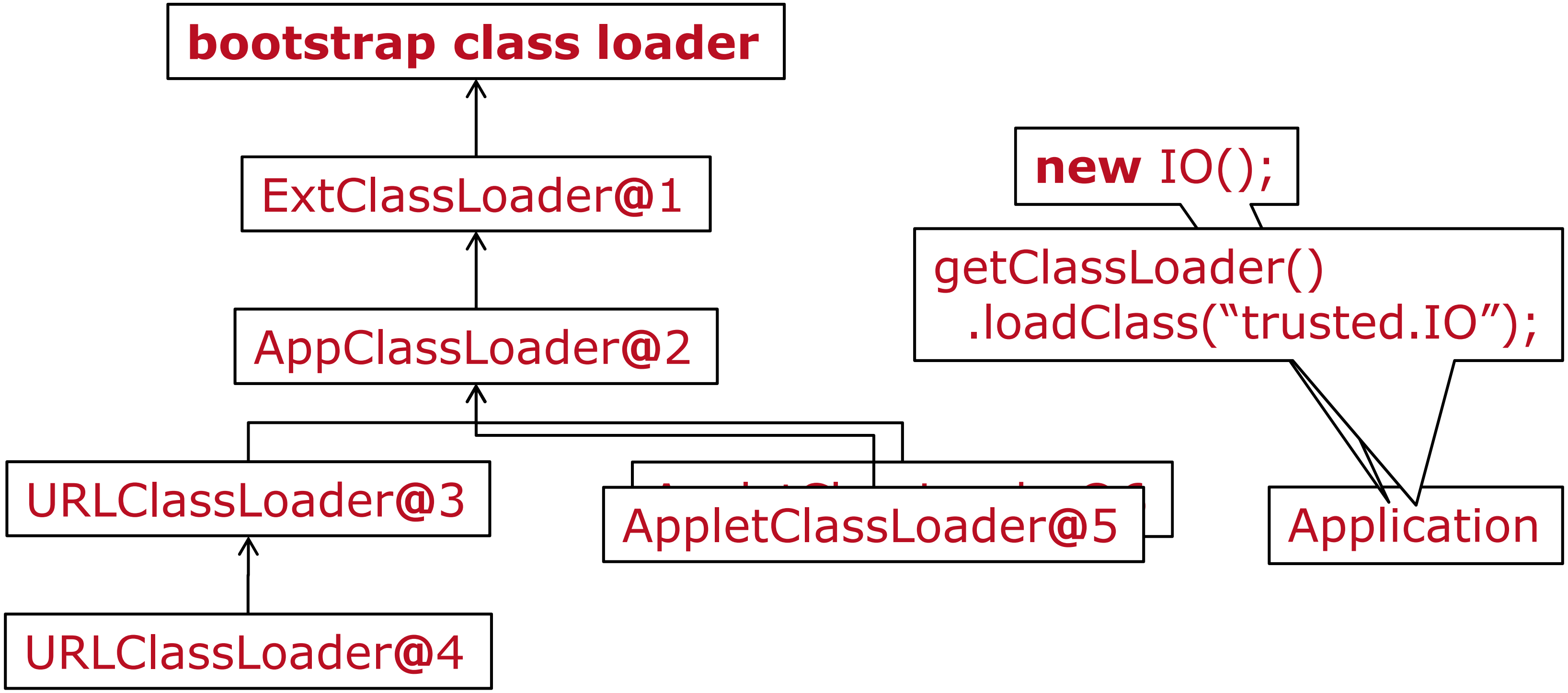
III. How to control advice weaving, e.g., at class IO?

Class Loader Hierarchy



Application

Class Loader Hierarchy



bootstrap class loader

ExtClassLoader@1

AppClassLoader@2

URLClassLoader@3

URLClassLoader@4

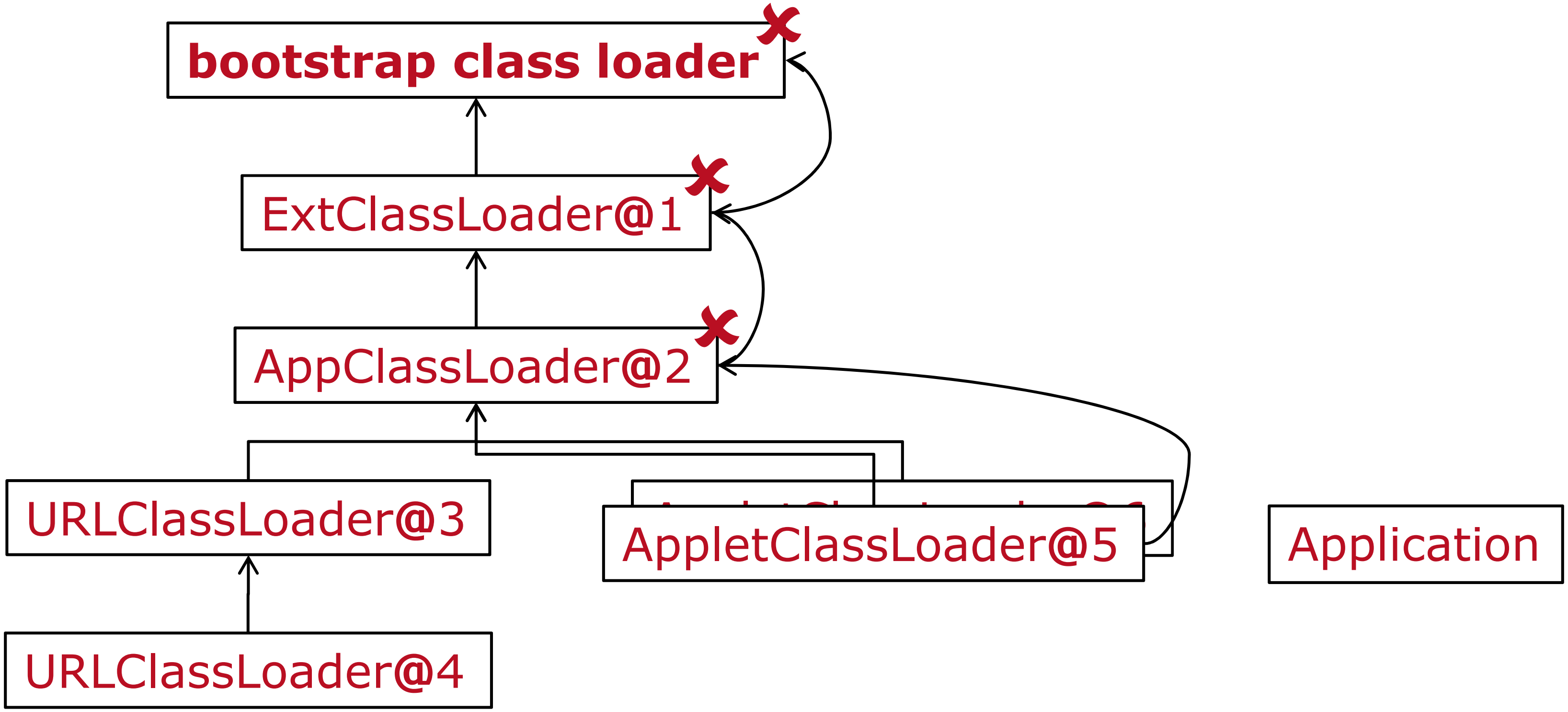
AppletClassLoader@5

Application

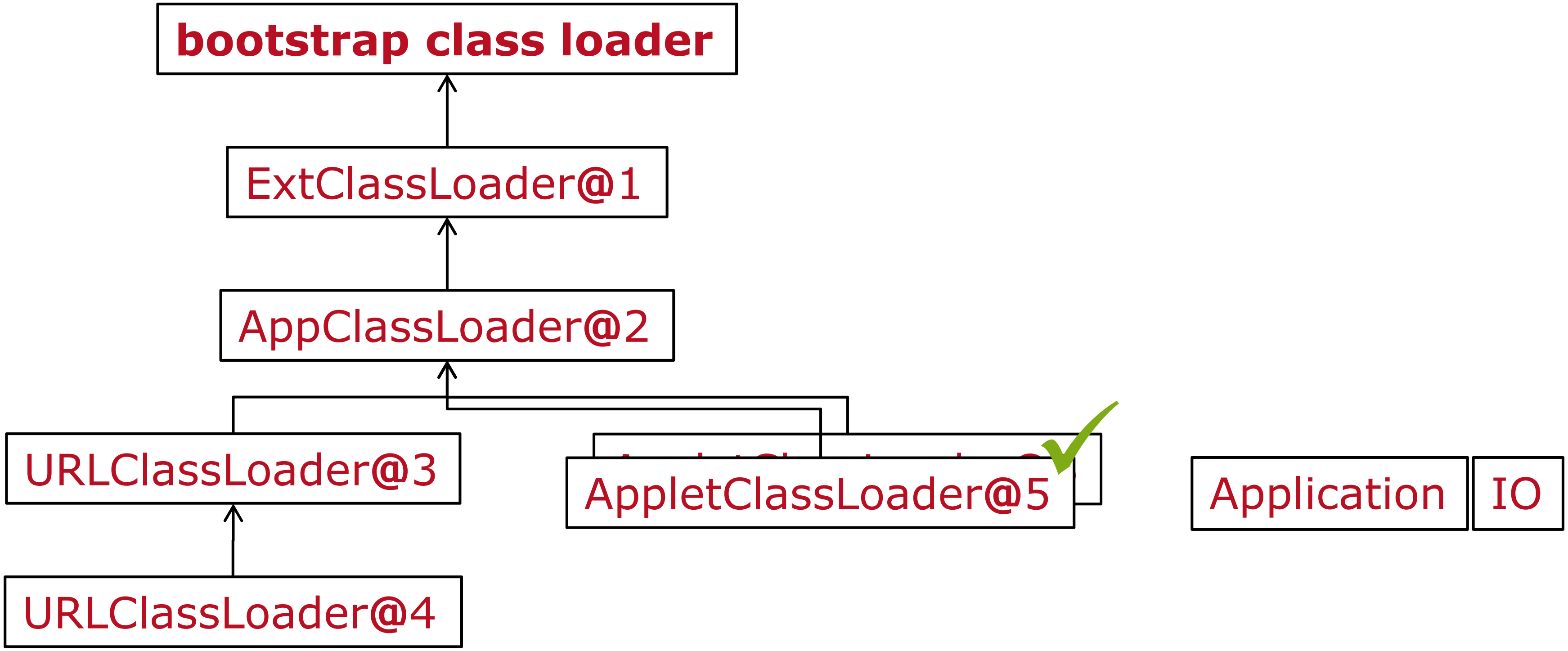
```
new IO();
```

```
getClassLoader()
  .loadClass("trusted.IO");
```

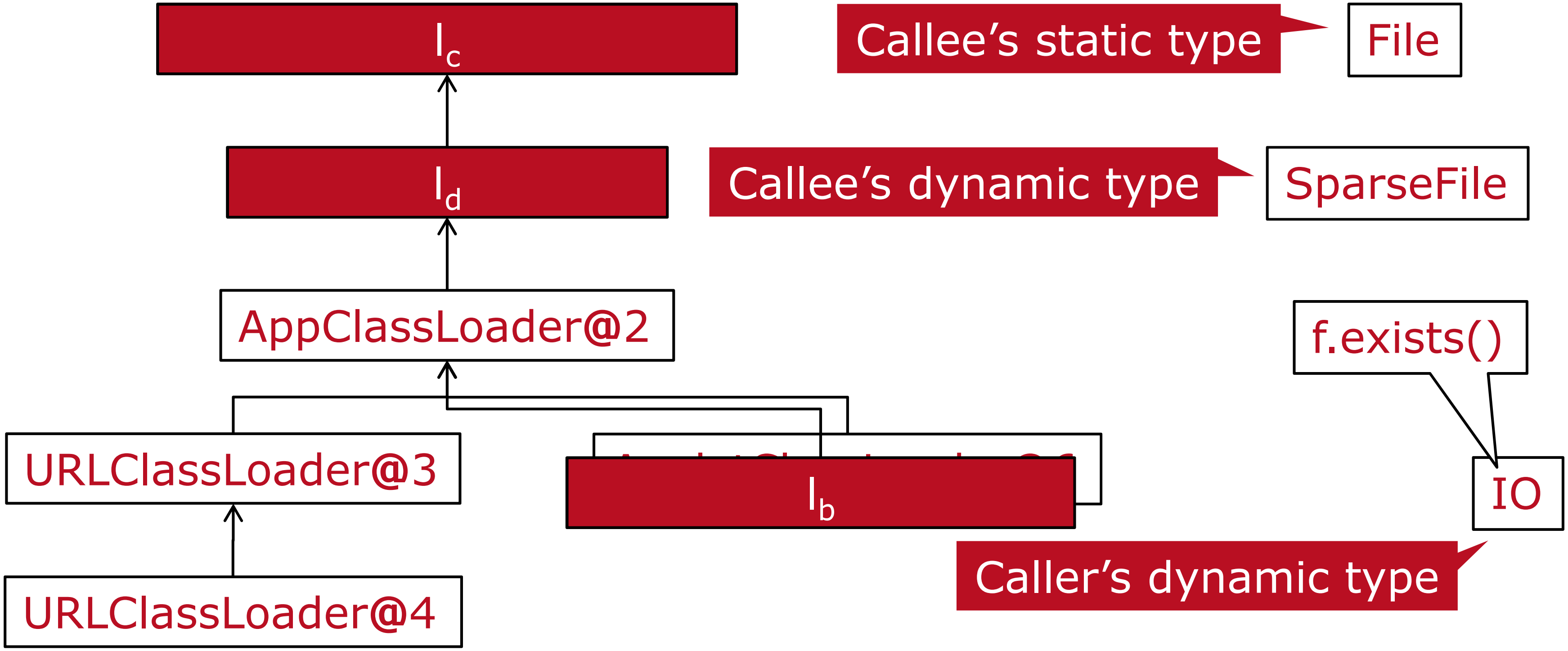
Class Loader Hierarchy



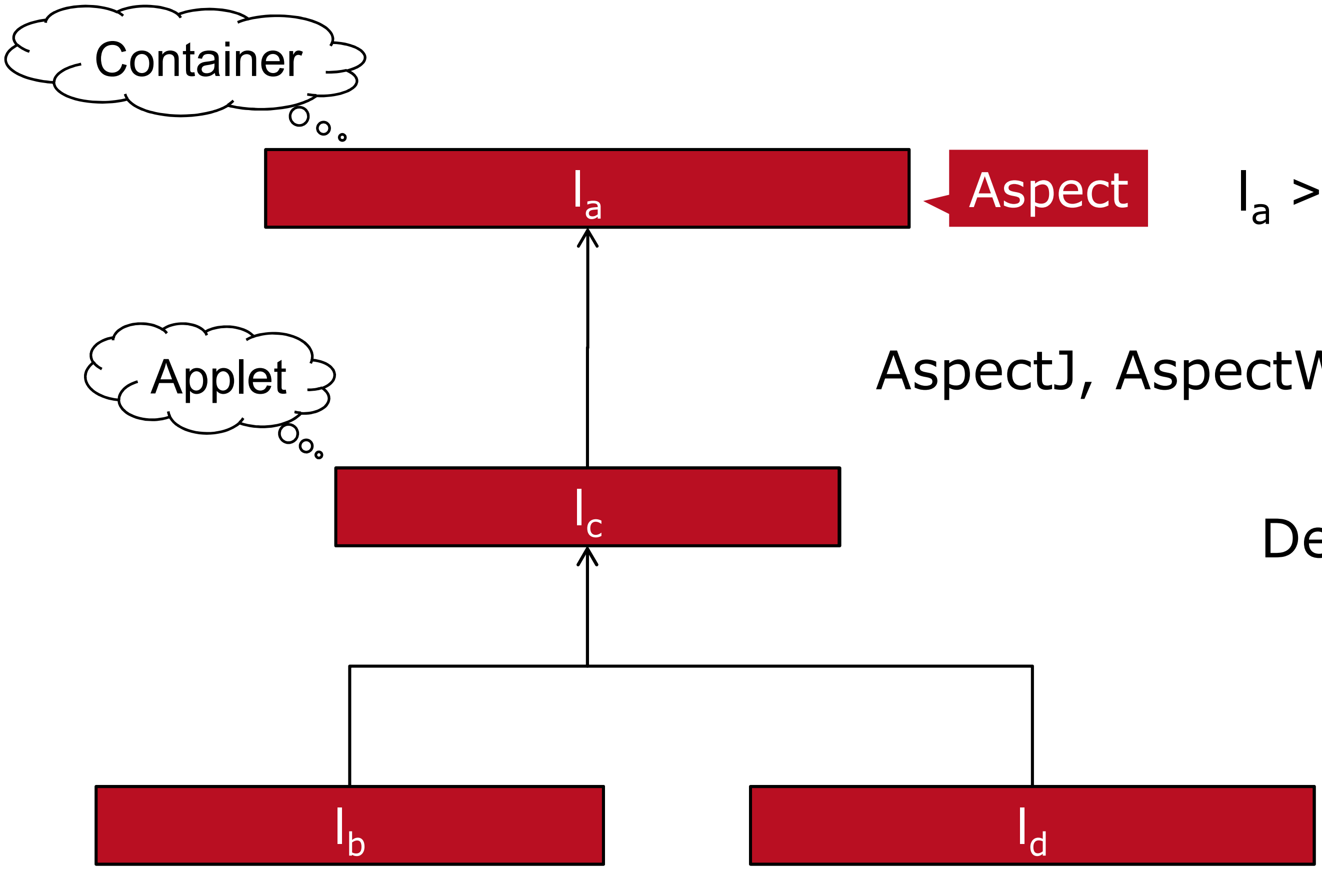
Class Loader Hierarchy



Class Loading Behavior



Survey

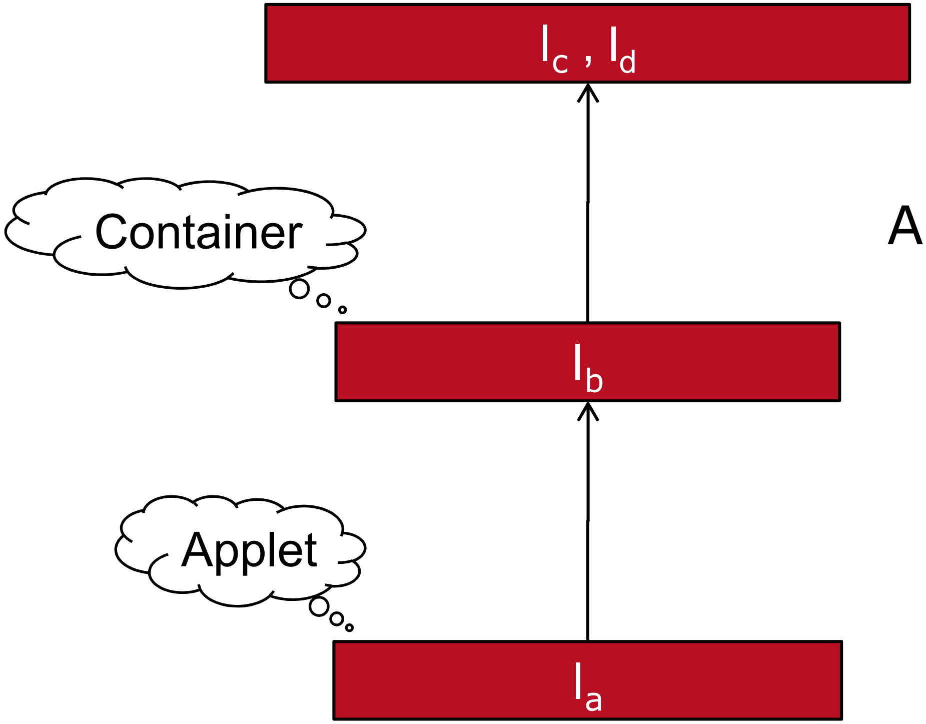


$l_a > l_b$, $l_a > l_c$, $l_a > l_d$

AspectJ, AspectWerkz, JAsCo ✓

Desired Design ✗

Survey (cont’d)

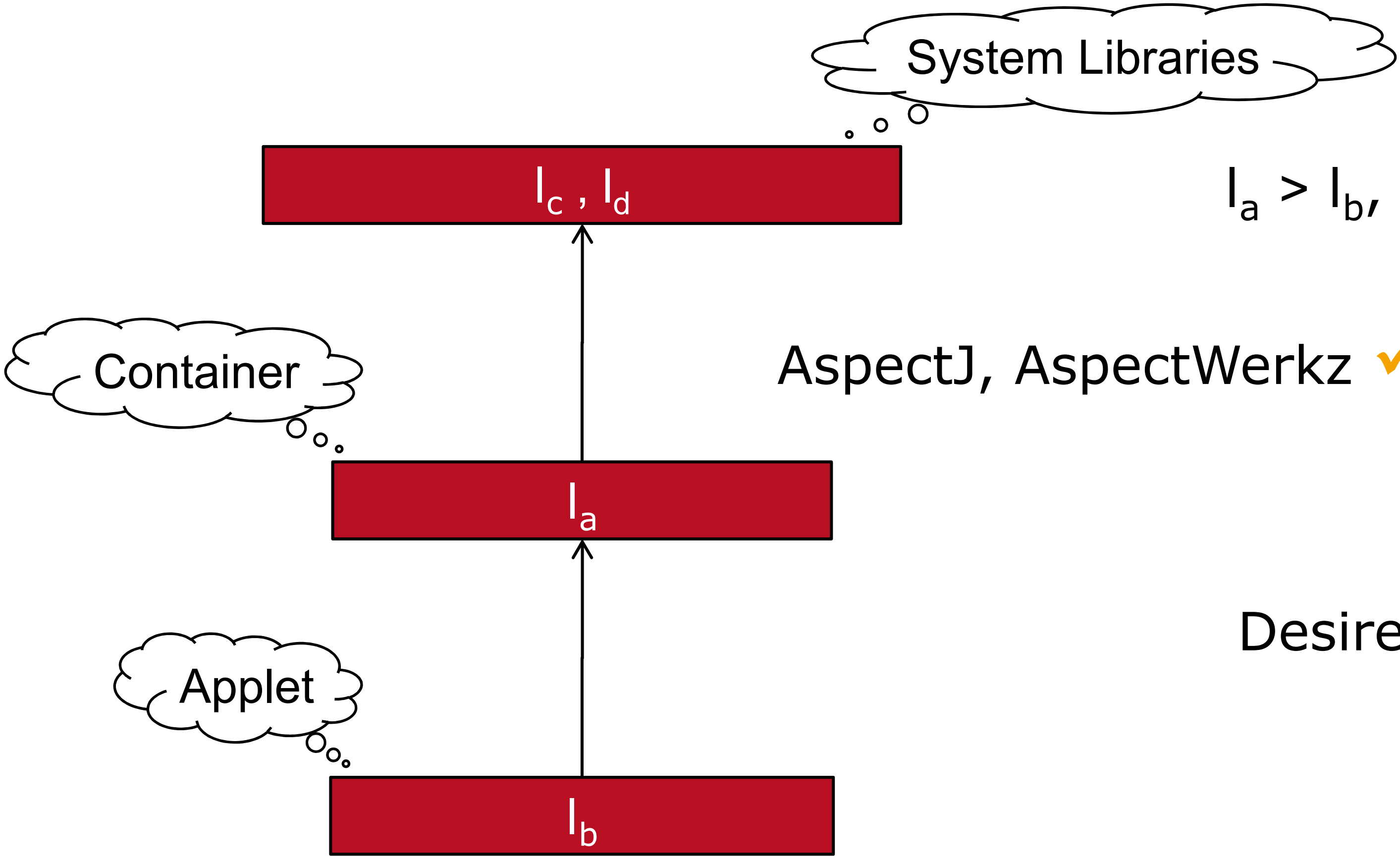


$l_a < l_b$, $l_a < l_c$, $l_a < l_d$

AspectJ, AspectWerkz, JAsCo ✗

Desired Design ✗

Survey (cont'd)



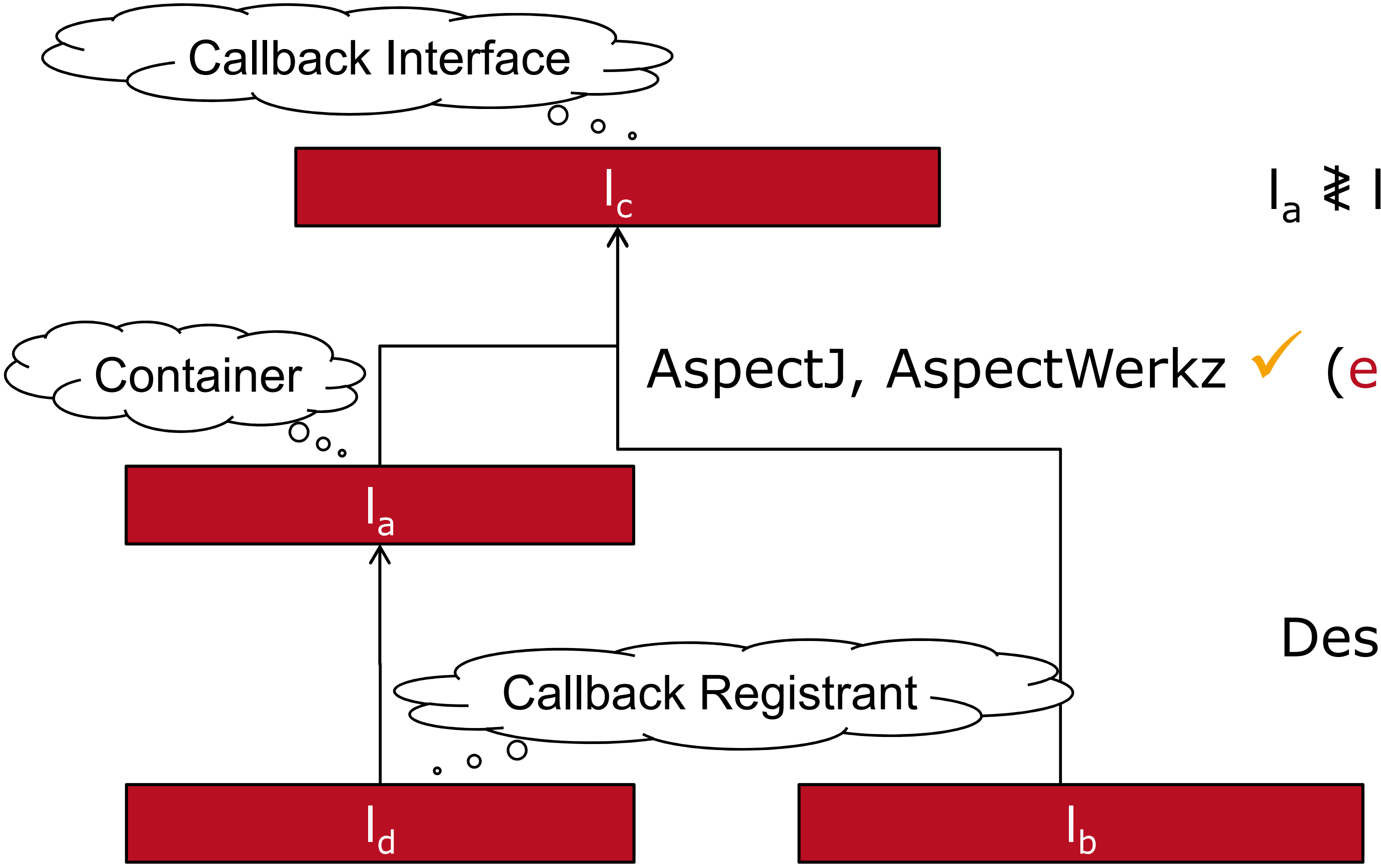
$l_a > l_b$, $l_a < l_c$, $l_a < l_d$

AspectJ, AspectWerkz ✓ (call only)

JAsCo ✗

Desired Design ✓

Survey (cont’d)



$I_a \ngtrless I_b, I_a < I_c, I_a > I_d$

AspectJ, AspectWerkz ✓ (execution only)

JAsCo ✓

Desired Design ✓

Survey Summarized

	I_b	I_c	I_d	AspectJ/Werkz	JAsCo	Desired Design
I_a	>	>	>	✓	✓	✗
	<	<	<	✗	✗	✗
	≹	<	<	✗	✗	✗
	<	<	≹	✗	✗	✗
	≹	<	≹	✗	✗	✗
	≹	≹	≹	✗	✗	✗
	≥	≤	≥	✓	✓	✓
	≥	<	<	✓call	✗	✓
	≥	<	≹	✓call	✗	✓
	<	<	≥	✓execution	✓	✓
	≹	<	≥	✓execution	✓	✓

Answers

I. Advice must be assigned its aspect’s protection domain.

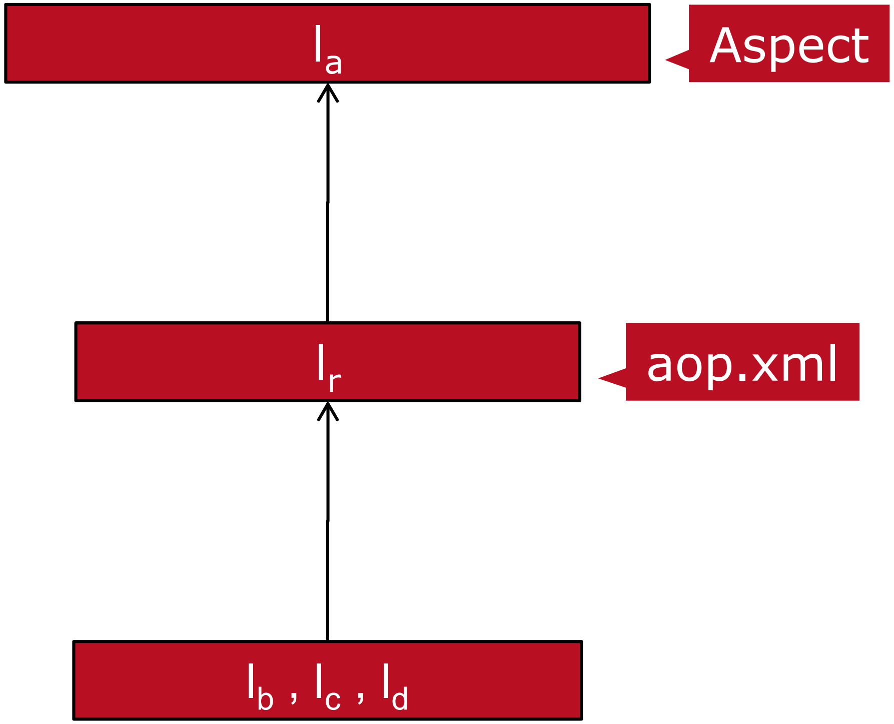
II. Advice must use its aspect’s class loader to resolve references.

III. Aspects should affect only calls when visible to the caller’s or callee’s dynamic type—unless the callee’s static type is visible.

Open Questions

- How to handle declaring resources, e.g., aop.xml?
- How to handle PrivilegedActions?

Declaring Resources



$l_a > l_b$, $l_a > l_c$, $l_a > l_d$

Open Questions

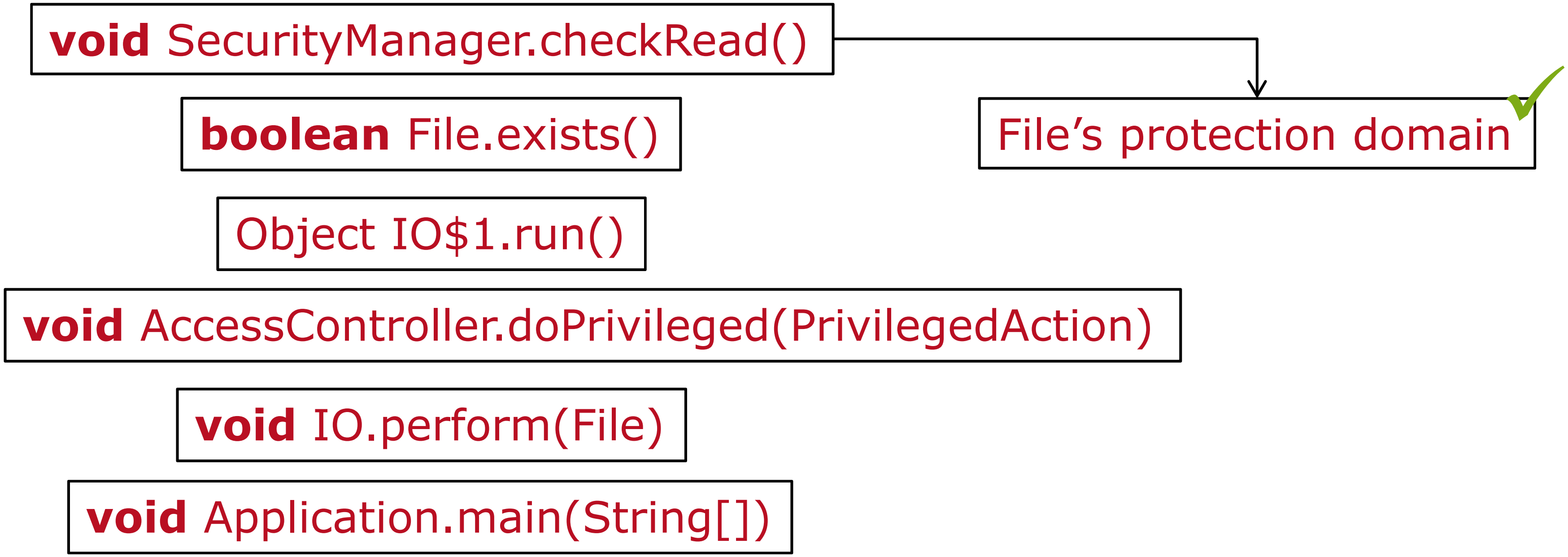
- How to handle declaring resources, e.g., aop.xml?
- How to handle PrivilegedActions?

Privileged Actions

```
package trusted;

class IO {
  void perform(File f) {
    AccessController.doPrivileged(new PrivilegedAction() {
      public Object run() {
        if (f.exists()) {
          // perform file I/O
        }
        return null;
      }
    });
  }
}
```

Privileged Actions (cont'd)



void SecurityManager.checkRead()

boolean File.exists()

Object IO$1.run()

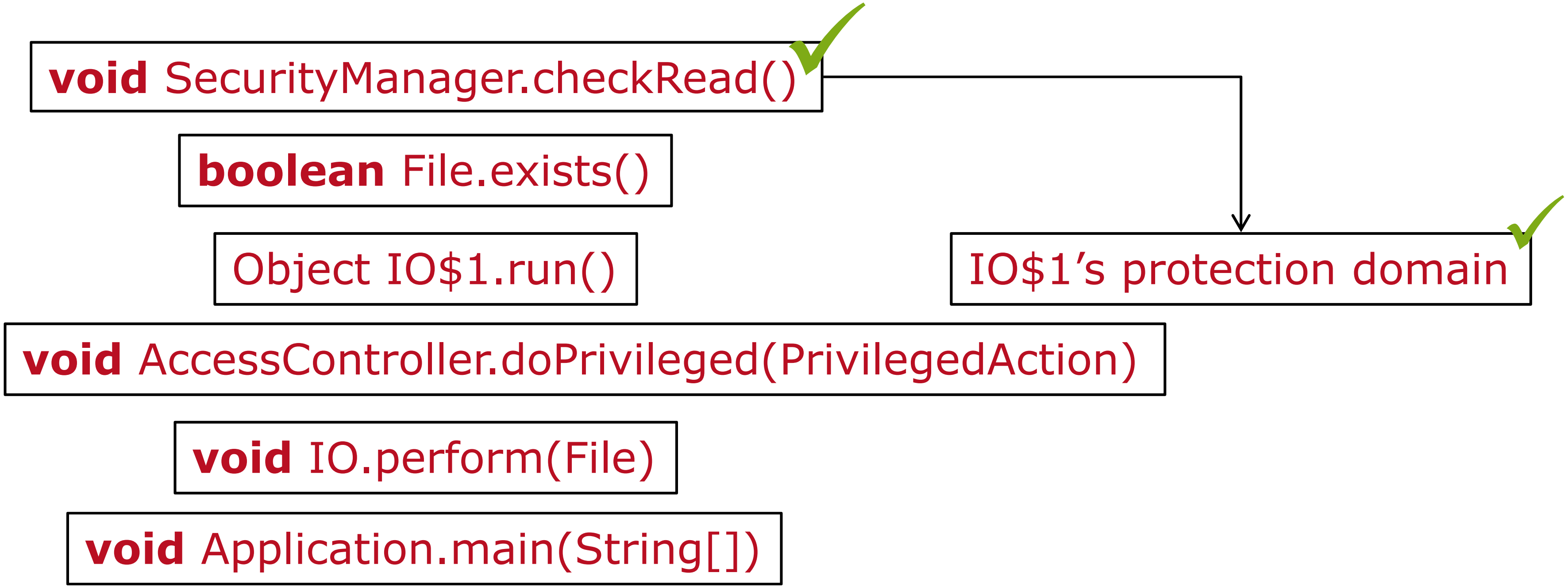
void AccessController.doPrivileged(PrivilegedAction)

void IO.perform(File)

void Application.main(String[])

File's protection domain ✓

Privileged Actions (cont’d)



void SecurityManager.checkRead() ✓

boolean File.exists()

Object IO$1.run()

IO$1’s protection domain ✓

void AccessController.doPrivileged(PrivilegedAction)

void IO.perform(File)

void Application.main(String[])