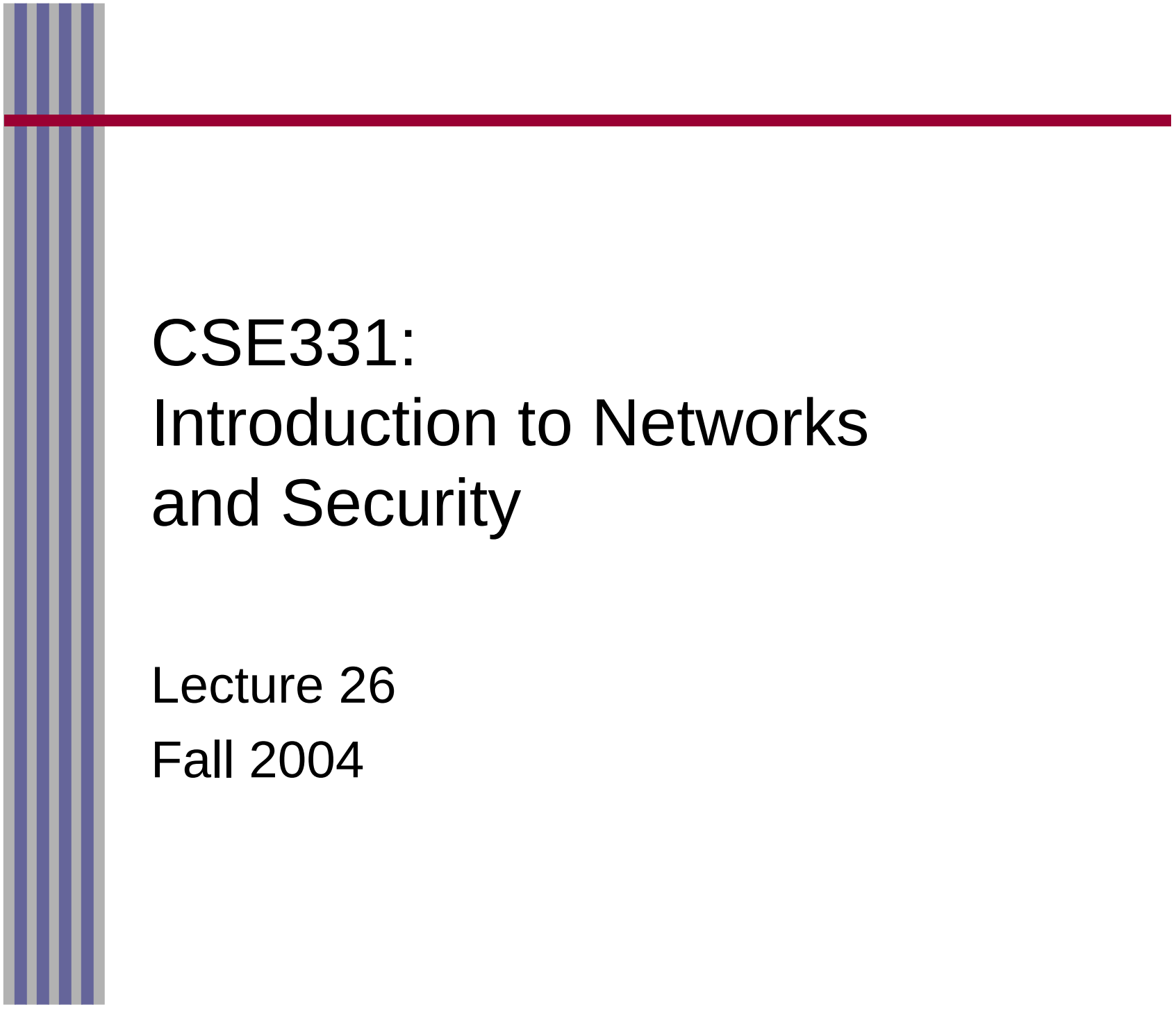


CSE331: Introduction to Networks and Security

Lecture 26

Fall 2004

Announcements

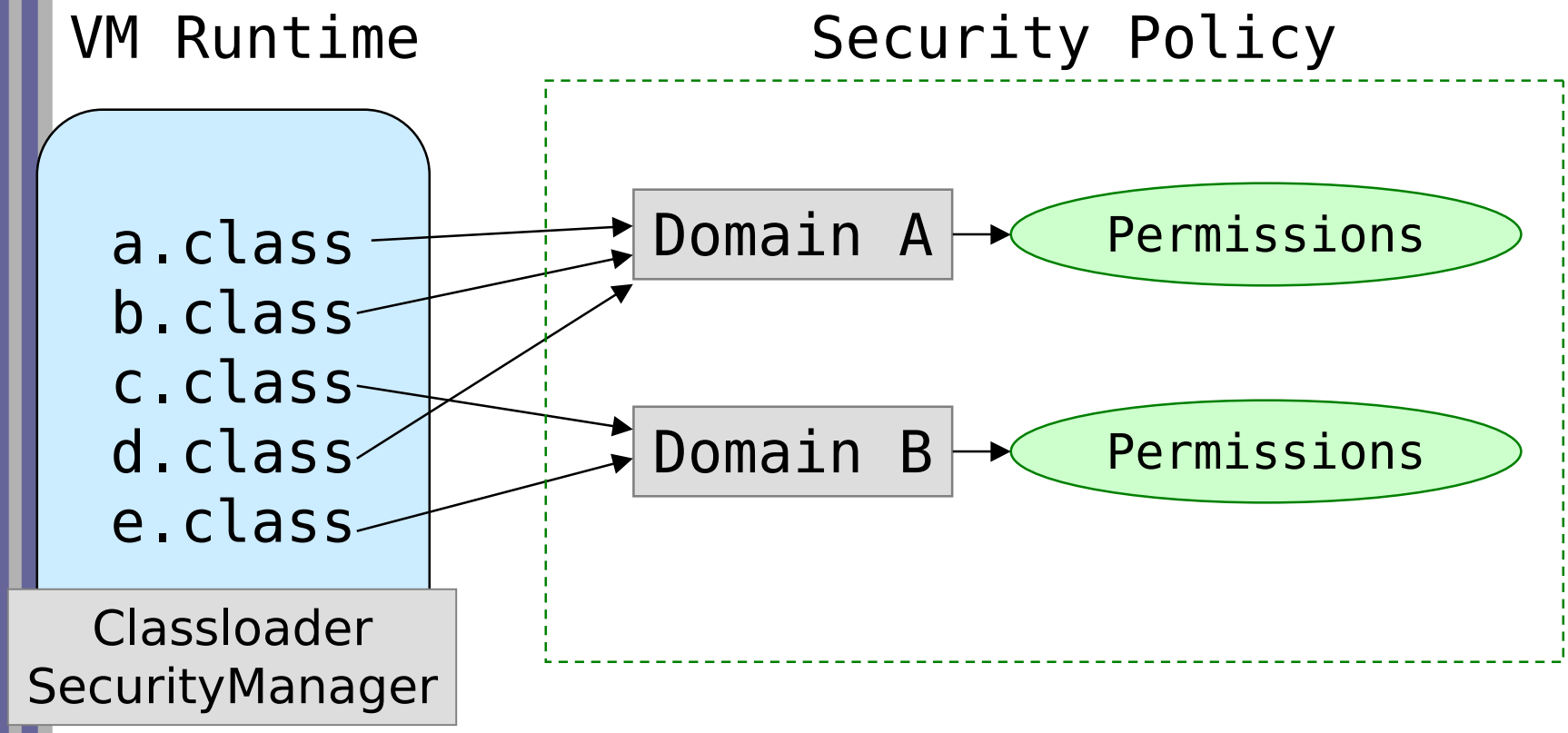
- Midterm 2 will be Monday, Nov. 15th.
  - Covers material since midterm 1

- Today:
  - Java/C# access control model

Access Control for Applets

- What level of granularity?
  - Applets can touch some parts of the file system but not others
  - Applets can make network connections to some locations but not others
- Different code has different levels of trustworthiness
  - www.l33t-hax0rs.com vs. www.java.sun.com
- Trusted code can call untrusted code
  - e.g. to ask an applet to repaint its window
- Untrusted code can call trusted code
  - e.g. the paint routine may load a font
- How is the access control policy specified?

Java Security Model



http://java.sun.com/j2se/1.4.2/docs/guide/security/spec/security-specTOC.fm.html

Kinds of Permissions

- java.security.Permission Class

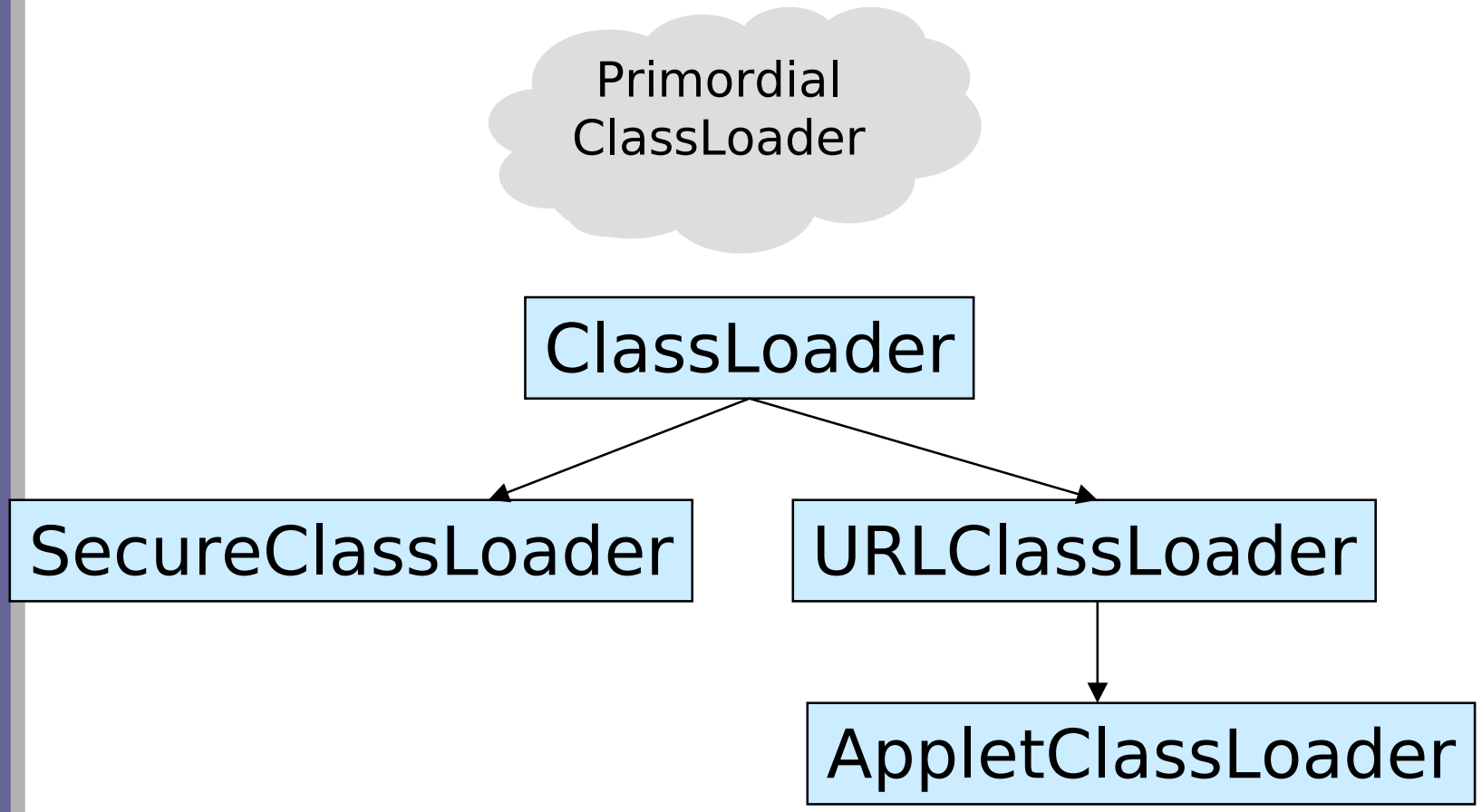
```
perm = new java.io.FilePermission("/tmp/abc","read");

java.security.AllPermission
java.security.SecurityPermission
java.security.UnresolvedPermission
java.awt.AWTPermission
java.io.FilePermission
java.io.SerializablePermission
java.lang.reflect.ReflectPermission
java.lang.RuntimePermission
java.net.NetPermission
java.net.SocketPermission
…
```

Code Trustworthiness

- How does one decide what protection domain the code is in?
  - Source (e.g. local or applet)
  - Digital signatures
  - C# calls this “evidence based”
- How does one decide what permissions a protection domain has?
  - Configurable – administrator file or command line
- Enforced by the classloader

Classloader Hierarchy



Classloader Resolution

- When loading the first class of an application, a new instance of the URLClassLoader is used.
- When loading the first class of an applet, a new instance of the AppletClassLoader is used.
- When java.lang.Class.ForName is directly called, the primordial class loader is used.
- If the request to load a class is triggered by a reference to it from an existing class, the class loader for the existing class is asked to load the class.

- Exceptions and special cases… (e.g. web browser may reuse applet loader)

Example Java Policy

```
grant codeBase “http://www.l33t-hax0rz.com/*” {
  permission java.io.FilePermission(“/tmp/*”, “read,write”);
}

grant codeBase “file://$JAVA_HOME/lib/ext/*” {
  permission java.security.AllPermission;
}

grant signedBy “trusted-company.com” {
  permission java.net.SocketPermission(…);
  permission java.io.FilePermission(“/tmp/*”, “read,write”);
  …
}
```

Policy information stored in:

$JAVA_HOME/lib/security/java.policy
$USER_HOME/.java.policy
(or passed on command line)

Example Trusted Code

Code in the System protection domain

```
void fileWrite(String filename, String s) {
  SecurityManager sm = System.getSecurityManager();
  if (sm != null) {
    FilePermission fp = new FilePermission(filename,“write”);
    sm.checkPermission(fp);
    /* … write s to file filename (native code) … */
  } else {
    throw new SecurityException();
  }
}
```

```
public static void main(…) {
  SecurityManager sm = System.getSecurityManager();
  FilePermission fp = new FilePermission(“/tmp/*”,“write,…”);
  sm.enablePrivilege(fp);
  UntrustedApplet.run();
}
```

Example Client

Applet code obtained from
http://www.l33t-hax0rz.com/

```
class UntrustedApplet {
  void run() {
    ...
    s.FileWrite(“/tmp/foo.txt”, “Hello!”);
    ...
    s.FileWrite(“/home/stevez/important.tex”, “kwijibo”);
    ...
  }
}
```

Stack Inspection

- Stack frames are annotated with their protection domains and any enabled privileges.

- During inspection, stack frames are searched from most to least recent:
  - fail if a frame belonging to a domain not authorized for privilege is encountered
  - succeed if activated privilege is found to ben enabled in the frame

Stack Inspection Example

```
main(…){
 fp = new FilePermission(“/tmp/*”,“write,…”);
 sm.enablePrivilege(fp);
 UntrustedApplet.run();
}
```

Policy Database

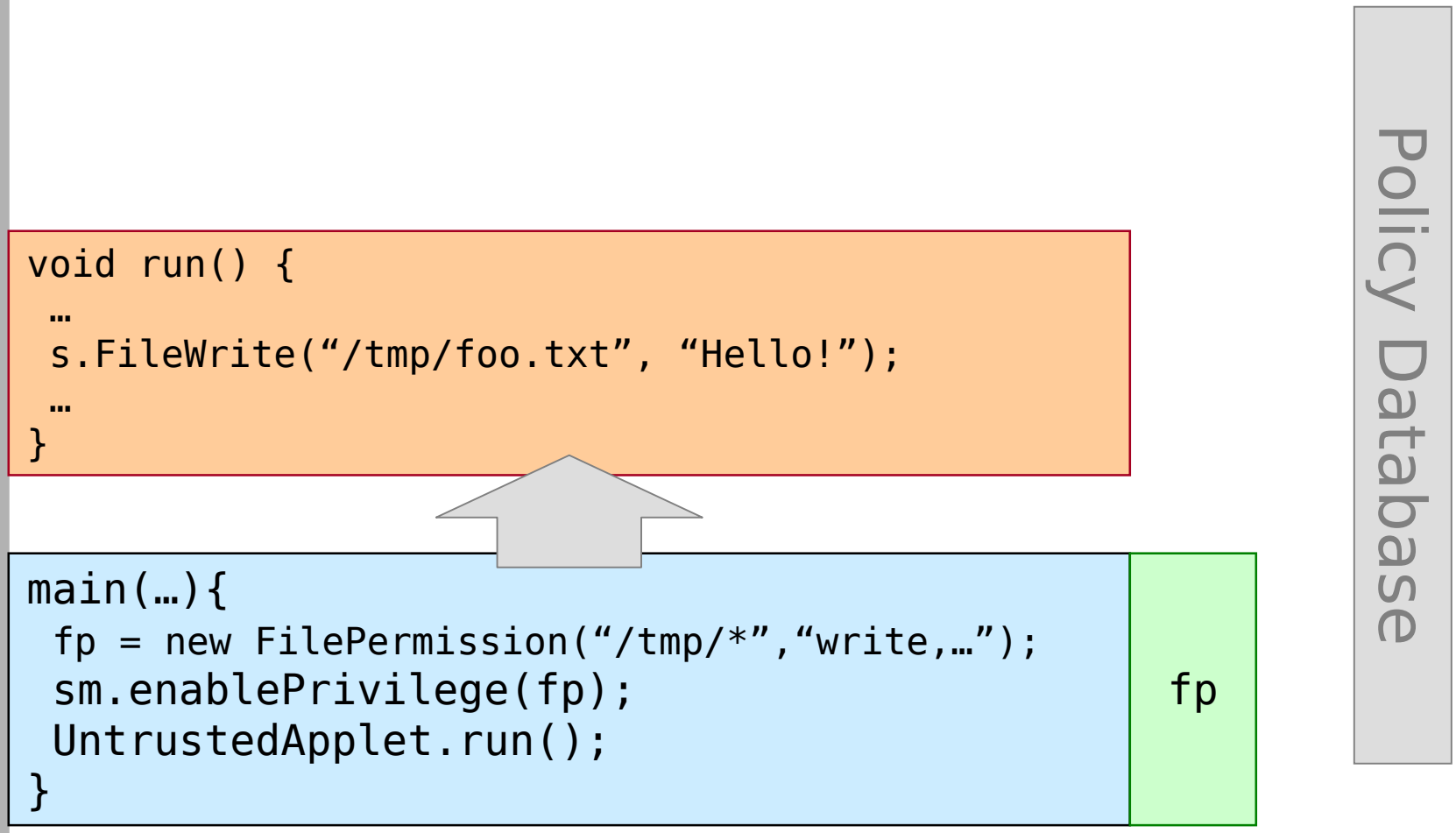
Stack Inspection Example

```
main(…){
 fp = new FilePermission(“/tmp/*”,“write,…”);
 sm.enablePrivilege(fp);
 UntrustedApplet.run();
}
```

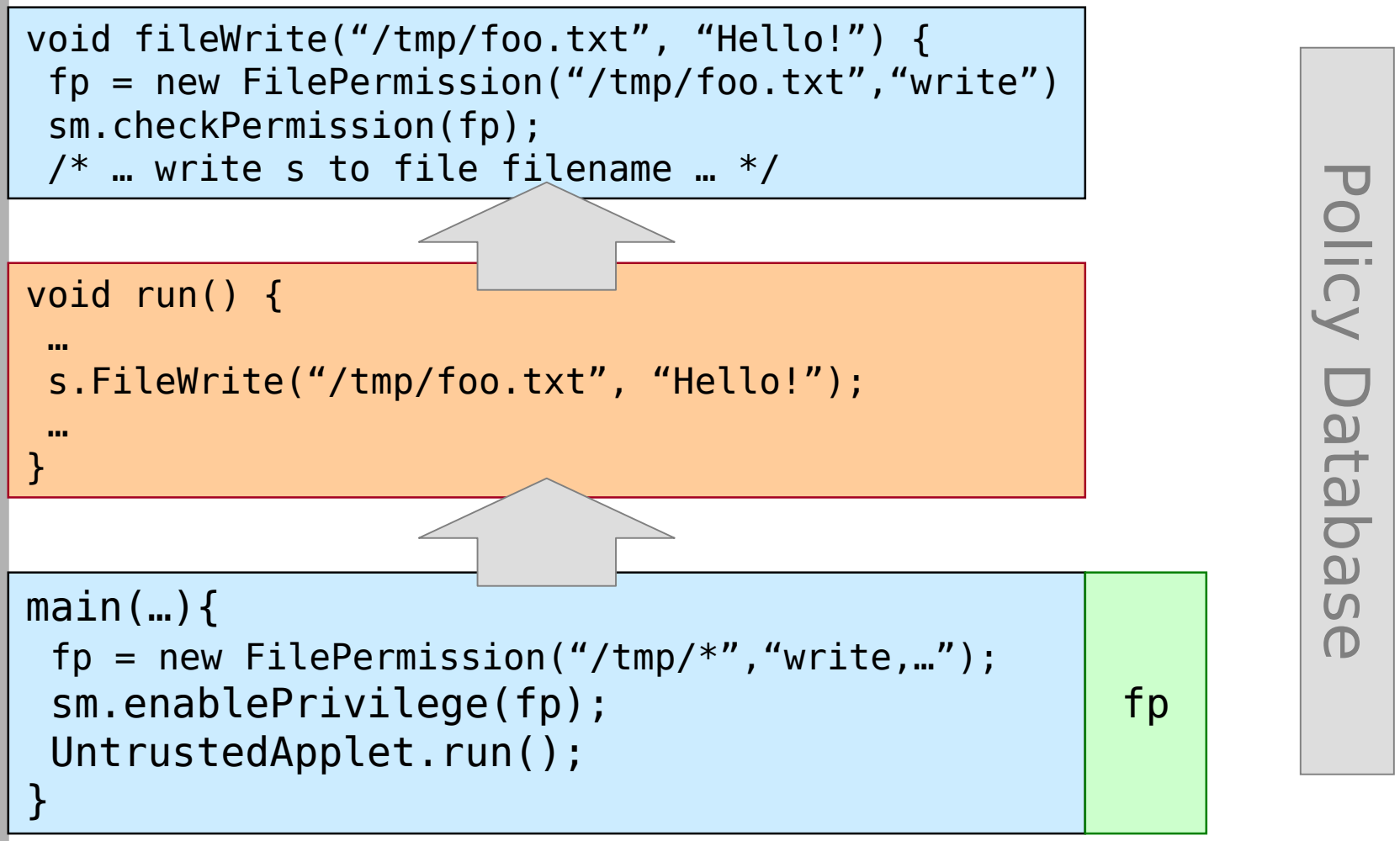
fp

Policy Database

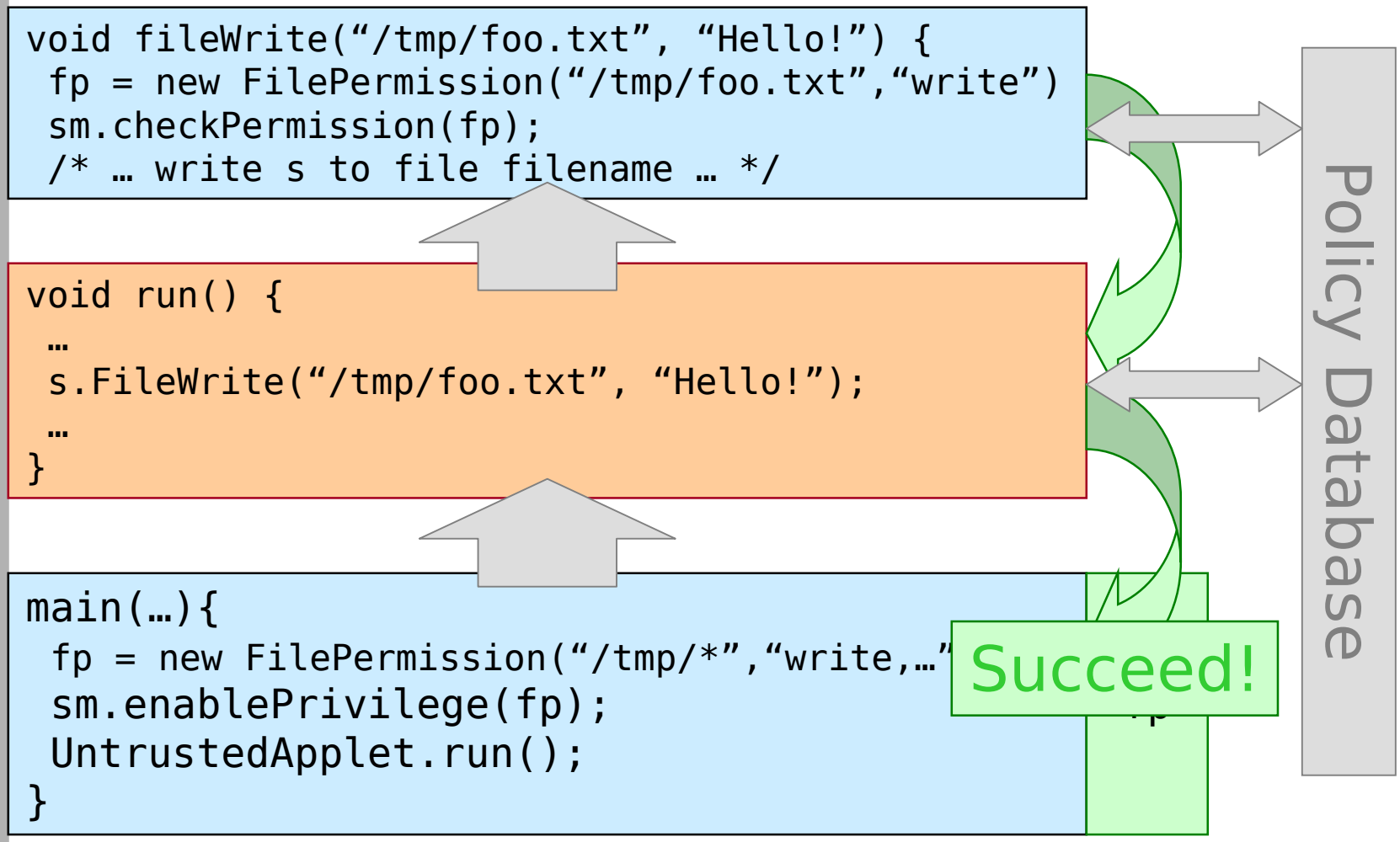
Stack Inspection Example



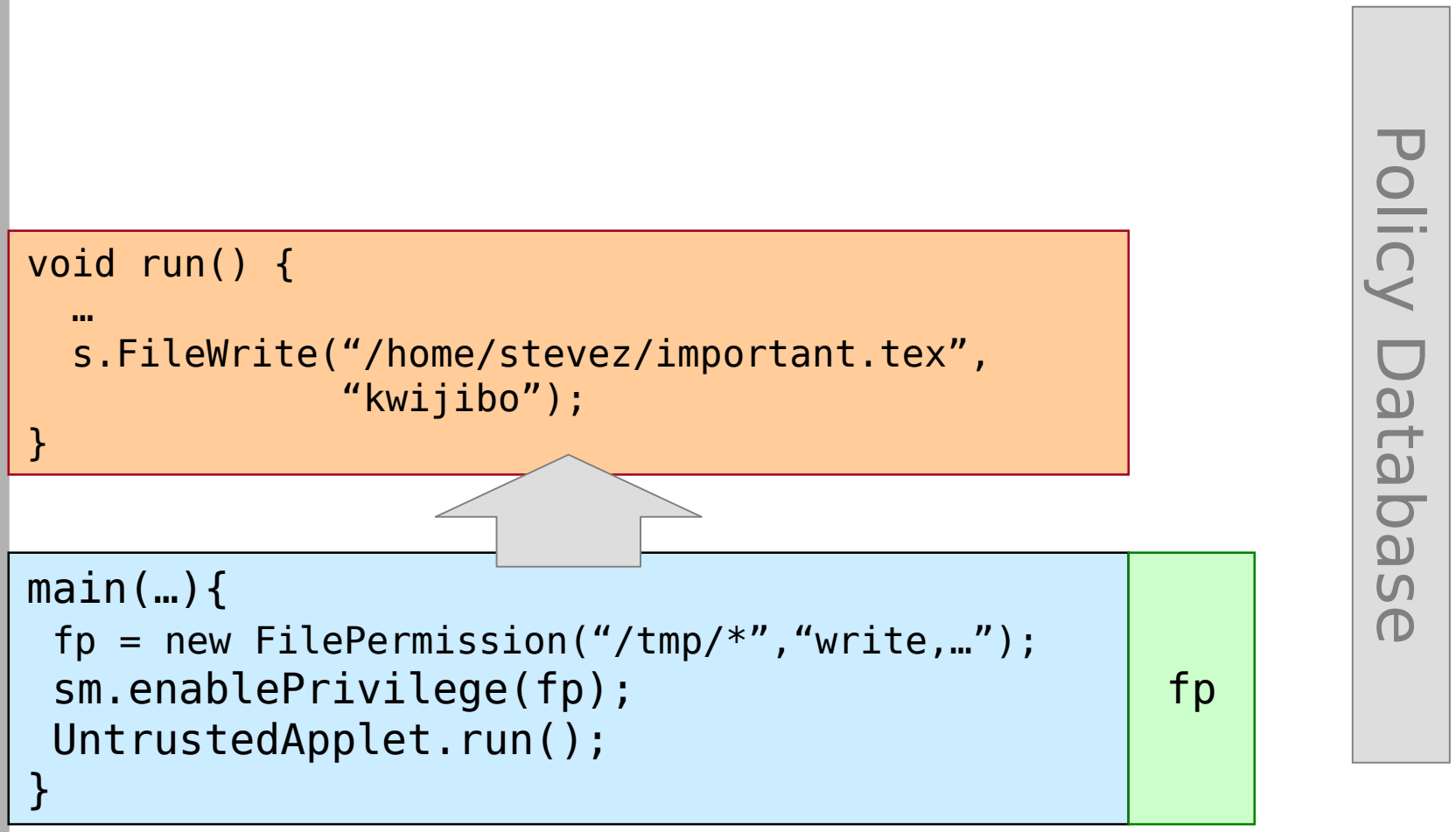
Stack Inspection Example



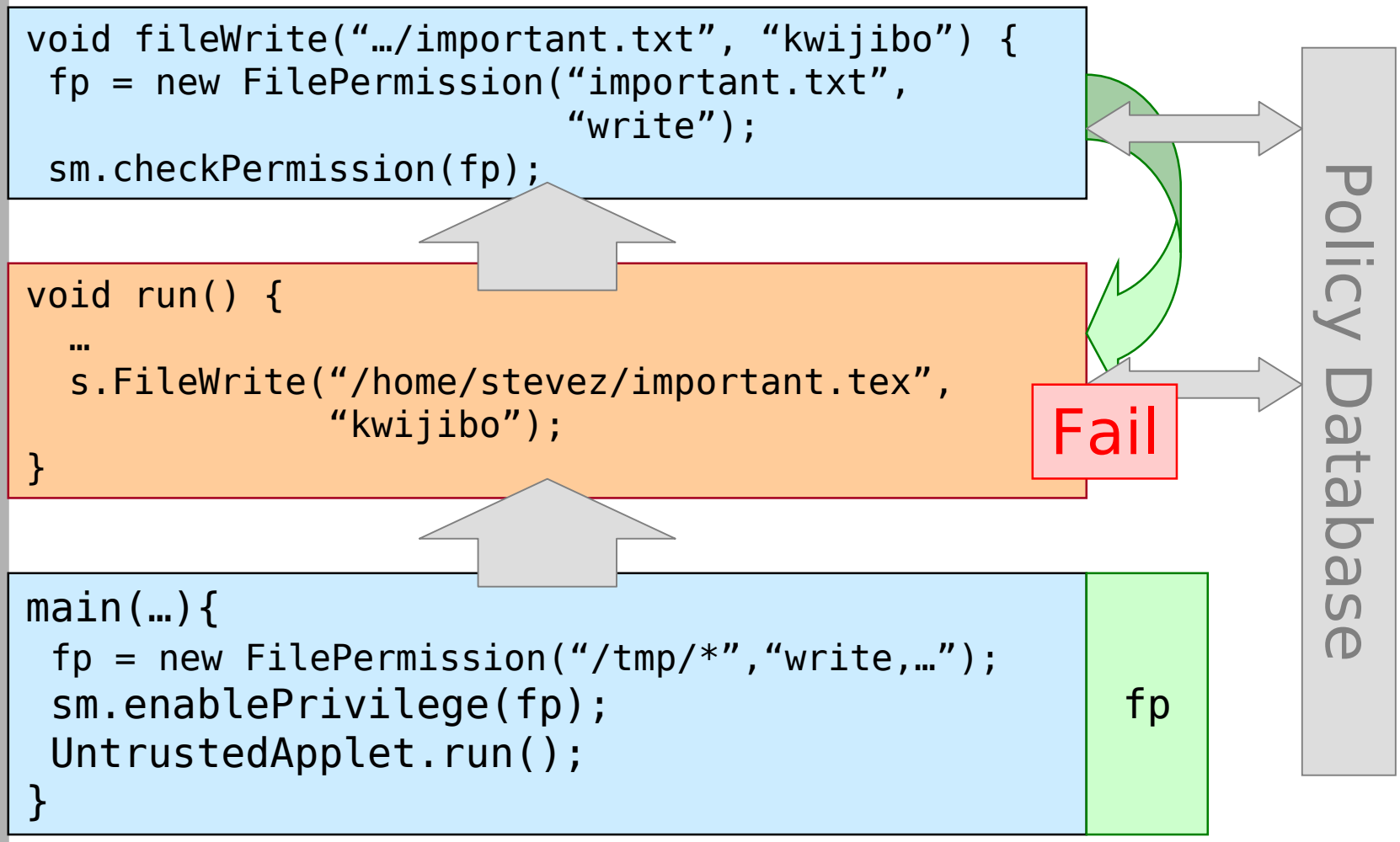
Stack Inspection Example



Stack Inspection Example



Stack Inspection Example



Other Possibilities

- The `fileWrite` method could enable the write permission itself
  - Potentially dangerous, should not base the file to write on data from the applet
  - … but no enforcement in Java (information flow would help here)
- A trusted piece of code could *disable* a previously granted permission
  - Terminate the stack inspection early

Stack Inspection Algorithm

```
checkPermission(T) {
  // loop newest to oldest stack frame
  foreach stackFrame {
    if (local policy forbids access to T by class executing in
        stack frame) throw ForbiddenException;

    if (stackFrame has enabled privilege for T)
      return;  // allow access

    if (stackFrame has disabled privilege for T)
      throw ForbiddenException;
  }

  // end of stack
  if (Netscape || ...) throw ForbiddenException;
  if (MS IE4.0 || JDK 1.2 || ...) return;
}
```