

Remote Class Loading

CSE 794R/ECE 694R

1

Consider this Java Program

```
class PrintTime
{
    public static void main(String[] args)
    {
        System.out.println(new java.util.Date());
    }
}
```

CSE 794R/ECE 694R

2

Class Loading

- The following classes must be loaded:
 - PrintDate
 - java.lang.Object (why?)
 - java.util.Date

CSE 794R/ECE 694R

3

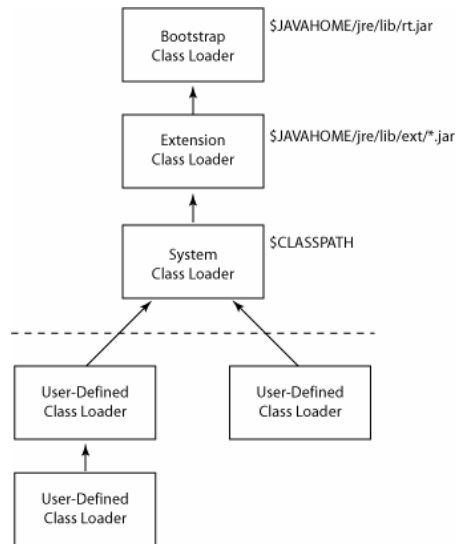
Class Loaders

- Class loaders are responsible for locating and loading necessary classes at runtime
 - bootstrap loader (standard classes)
 - extension loader (ext folder)
 - system loader (CLASSPATH)
 - user-defined loader(s)
- They form a hierarchy

CSE 794R/ECE 694R

4

Class Loader Hierarchy



CSE 794R/ECE 694R

5

Old Ideas

- classes
- objects
- interfaces
- instantiation
- inheritance

CSE 794R/ECE 694R

6

Classes Are Objects Too!

- `java.lang.Class`
 - Instances of the class `Class` are objects that represent classes and interfaces in a running Java application.
 - `ClassName.class` and `object.getClass()`, e.g., `Product.class` or `new Product().getClass()`
- Provides methods, e.g., `getClassLoader()` returns the `ClassLoader` that loaded this class (or null if it was the bootstrap loader)
 - `Product.class.getClassLoader()`
 - `new Product().getClass().getClassLoader()`

CSE 794R/ECE 694R

7

Classes vs Interfaces

- Good programming practice to program against an interface (instead of a class) whenever possible (why?)
- Programming against interfaces is also useful for dynamic loading

CSE 794R/ECE 694R

8

Remote Loading Basics

- Class loader can get bytecode over a network
 - interface (locally) available to client (rule-of-thumb: C's only public methods are those in CI)
 - loadClass uses the fully qualified name (e.g., "edu.ohio-state.cse.ase.rcleg.Product")

CSE 794R/ECE 694R

9

Understanding loadClass

- loadClass implementation:
 - 1) check if loaded by THIS loader (ClassLoader->findLoadedClass)
 - 2) if 1 fails, delegate to parent (super.loadClass)
 - 3) if 2 fails, findClass
- Net effect:
 - look for loaded class going UP
 - try to load new class going DOWN

CSE 794R/ECE 694R

10

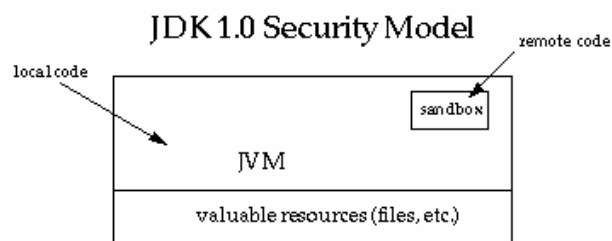
More About Class Loaders

- Class loaders create a "name space"
 - class A loaded by ClassLoader X is different from class A loaded by ClassLoader Y
- Cascading loading
 - after loading A, then load (classes of) members of A
 - start with the loader that succeeded in loading A, follow same rules
- Loading is *lazy* and *dynamic*

CSE 794R/ECE 694R

11

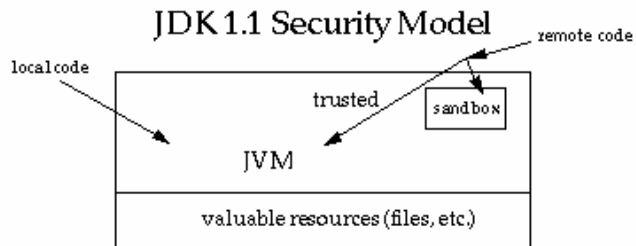
Evolution of Java Security



CSE 794R/ECE 694R

12

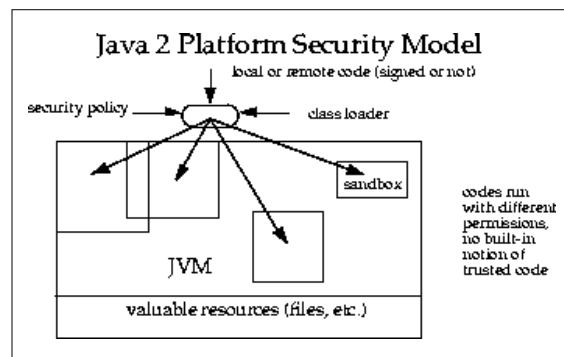
Evolution of Java Security cont.



CSE 794R/ECE 694R

13

Evolution of Java Security cont.



CSE 794R/ECE 694R

14

Security Policy

- "policy" : set of permissions
- two ways to control these permissions/constraints/properties:
 1. programmatically (i.e., within the code)
 2. descriptively (i.e., in a file read by the code)
 - can be edited by hand
 - or with "policytool"
- SEE: `$JAVA_HOME/jre/lib/java.policy`

CSE 794R/ECE 694R

15

Responsibilities

- class loader is responsible for:
 - consulting security policy
 - defining class with appropriate permissions
- securityManager can check if an op should be permitted
 - does the current thread have permission to do this?)
 - many methods: checkXXX (args...) throws SecurityException

CSE 794R/ECE 694R

16