Java Class Loader & Security

Bhanu Prakash Gopularam
Senior Engineer
Java Platform Group

Agenda

- Introduction
- Java Class Loader
- Java Class Loading Phases
- Custom Class loading
- Class Loader Exceptions
- Debugging Class loader Problems
- Questions

Java Language

Java Platform and Programming language introduced in 1995

- Java Language
  - General purpose object oriented programming language
  - Automatic storage management – GC
  - Platform independent code, security and network mobility

- Few Java Language Security features
  - Built in Security Architecture
  - Configurable policies and domains
  - Applet Sand box: Allows securely download and run untrusted Java programs over the network

Java Class Loader - Introduction

ClassLoader “Reads byte code into JVM”

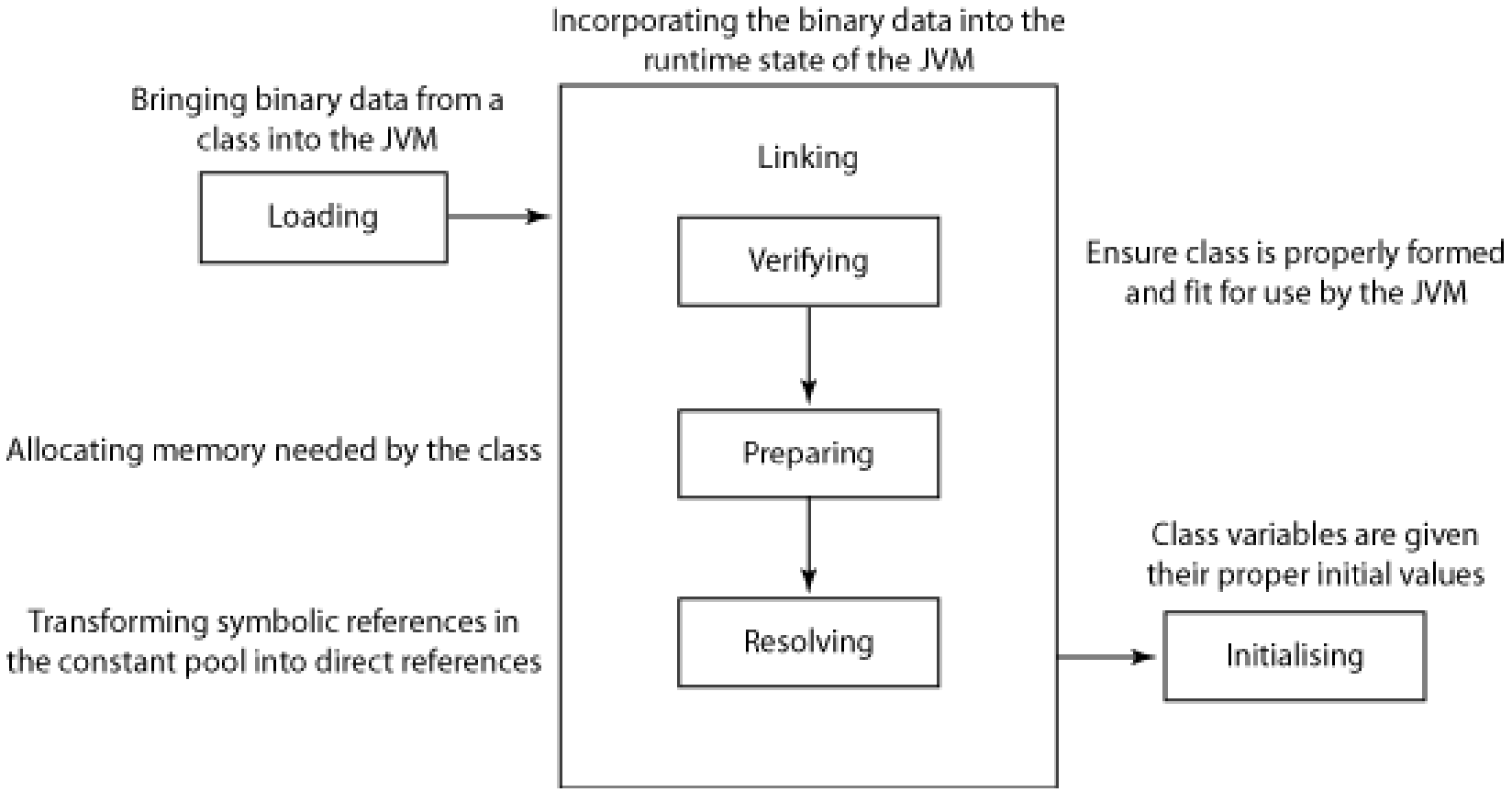
A class is defined by its

<class name, defining class loader>

Goals of Class Loader:

- Make first line of defense
- Guard system packages from fake classes and spoofing attacks
- Resolve symbolic references from one class to another

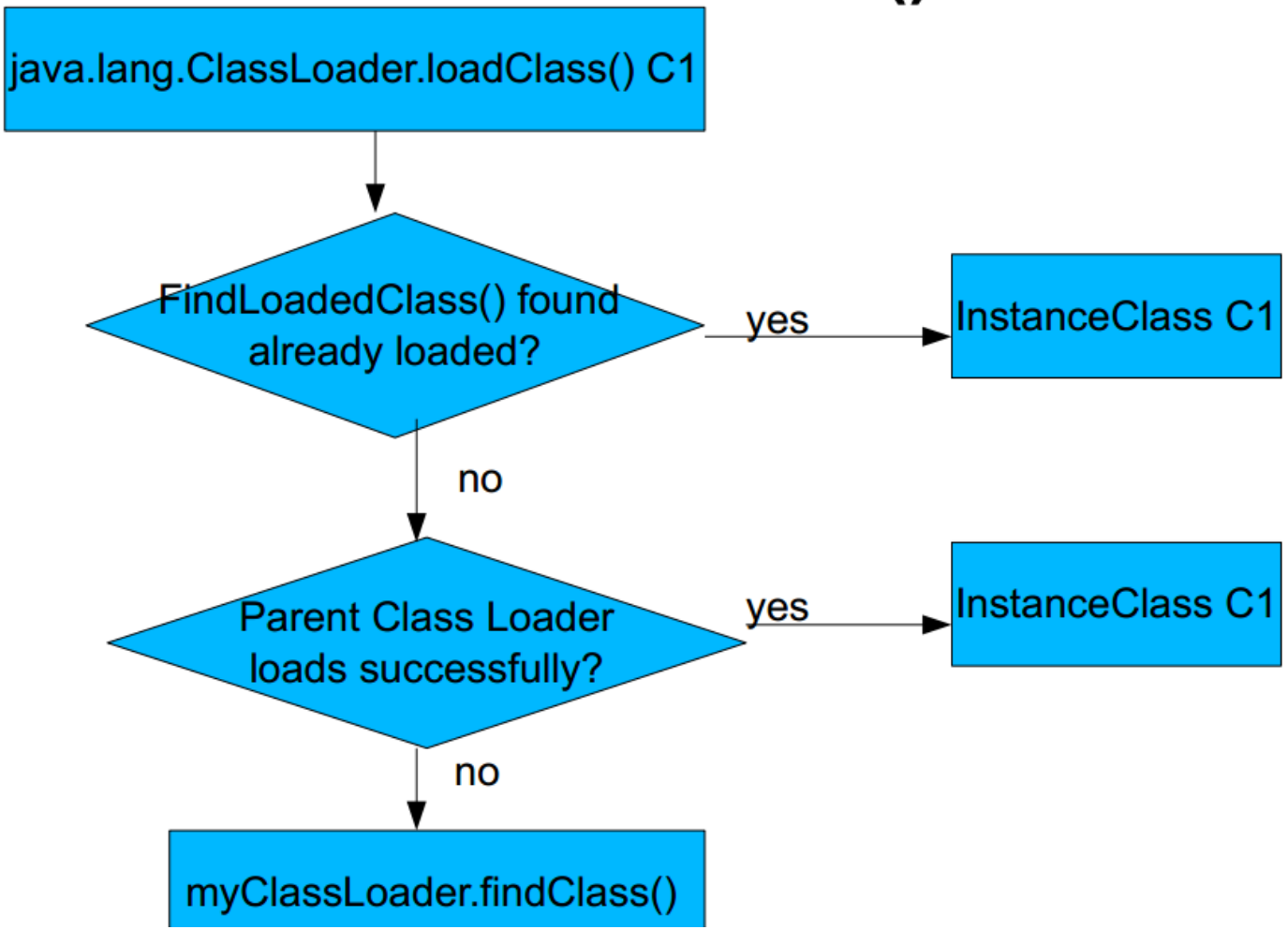
Java Class loading phases



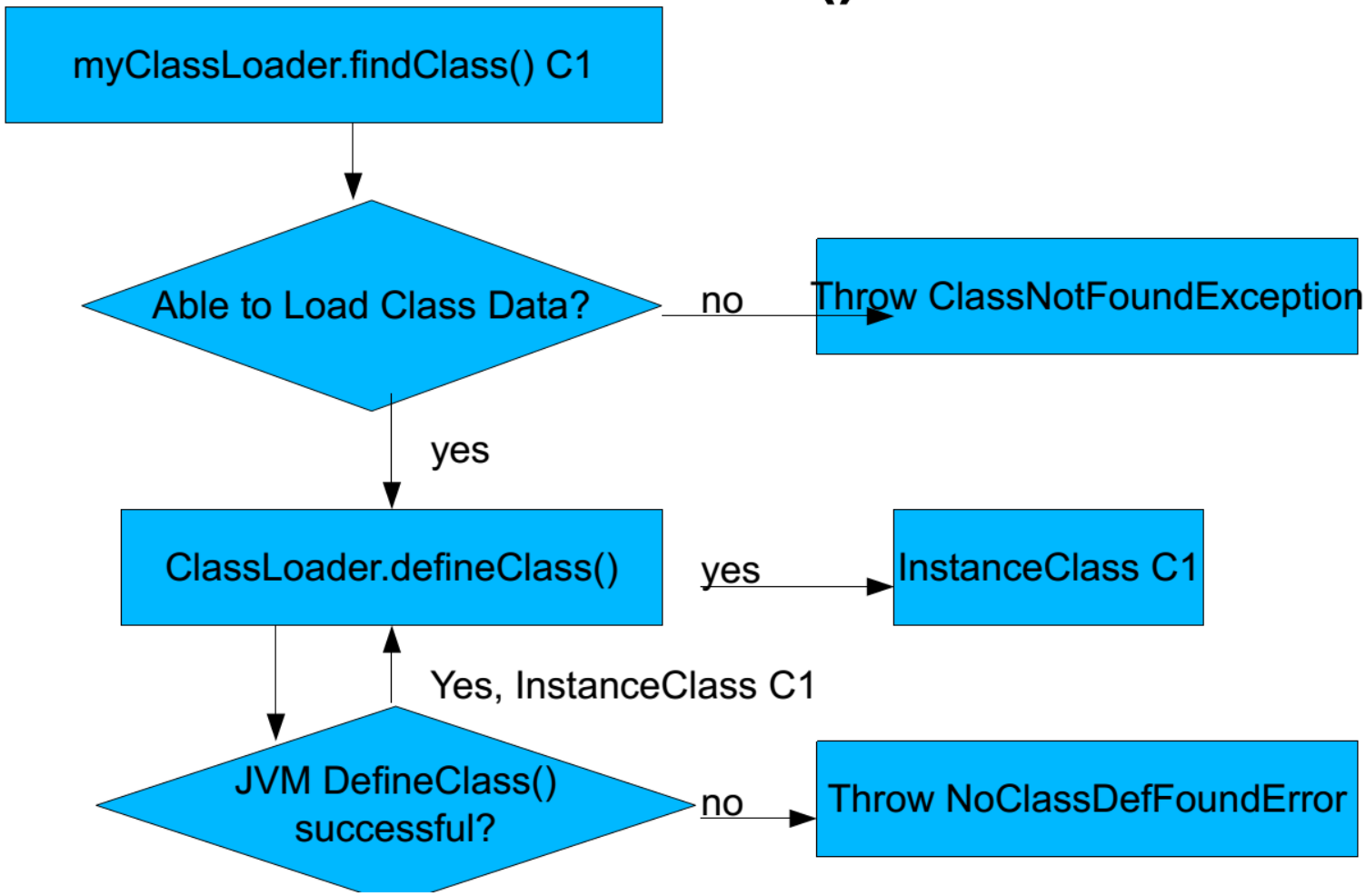
Verification Process:

1. Structural check
2. Semantic check
3. Byte code verification
4. Symbolic references check

Class Loader loadClass()



Class Loader findClass()



Java Class Loaders

1. Bootstrap or Primordial Class Loader
   - rt.jar
   - *-XbootClassPath* – use judiciously
   - System property *sun.boot.class.path*

2. Extension Class Loader
   - Installed optional packages, lib/ext (in JRE) or jre/lib/ext (in JDK)
   - $JRE_HOME/lib/ext
   - System property *java.ext.dirs*

3. Application Class Loader
   - Application classpath $CLASSPATH or -cp variable
   - System property *java.class.path*
   - Misleadingly it is also called as System Classloader
   - Can be changed using property *-Djava.system.class.loader*

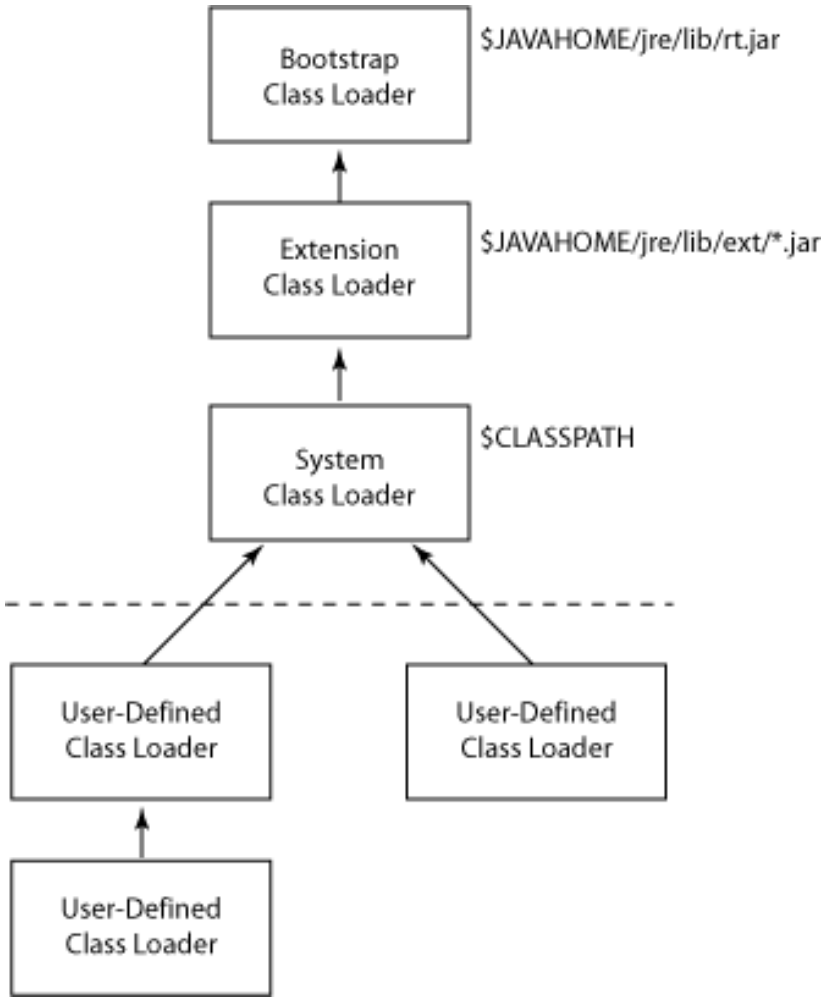
Java Class Loaders – contd..

4. SecureClassLoader
   - Adds support for code security model in JDK 1.2
     - Adds defineClass(String name, CodeSource)
     - Adds getPermissions(CodeSource)

5. URLClassLoader
   - Loads classes from specified url path (dir or jar file)
   - Extends from SecureClassLoader
     - Supports loading classes from URL code sources

6. Context Class Loader
   - Context class loader is provided by creator of thread
   - If Security Manager is present, checkPermission() is invoked with getClassLoader() call

Java Class loader Delegation

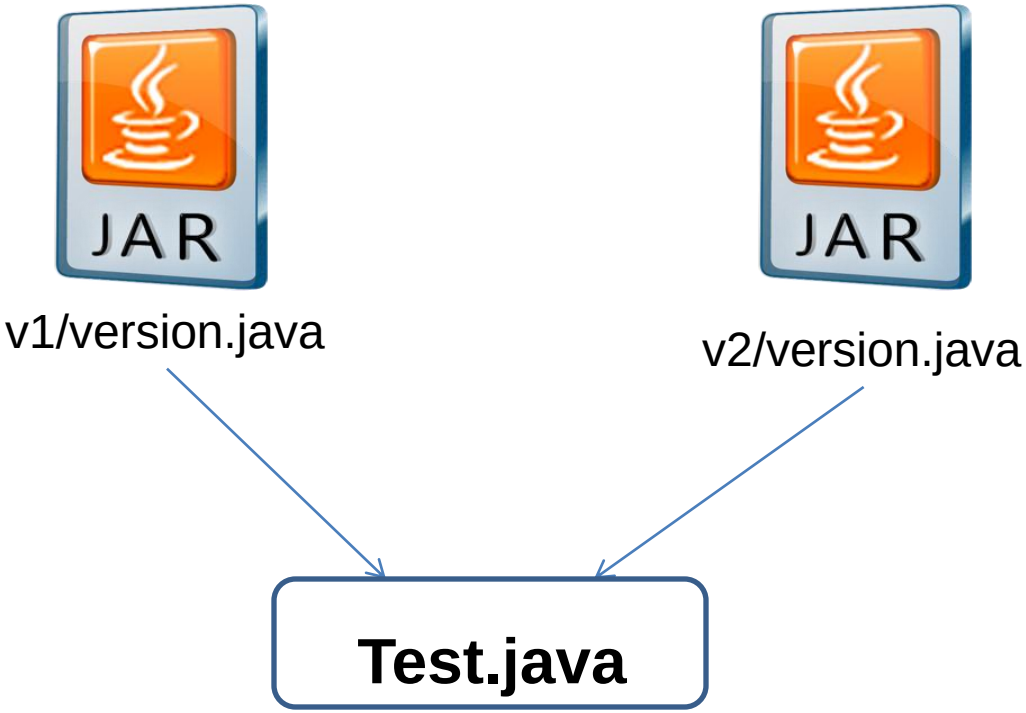


Why to write own class loader ?

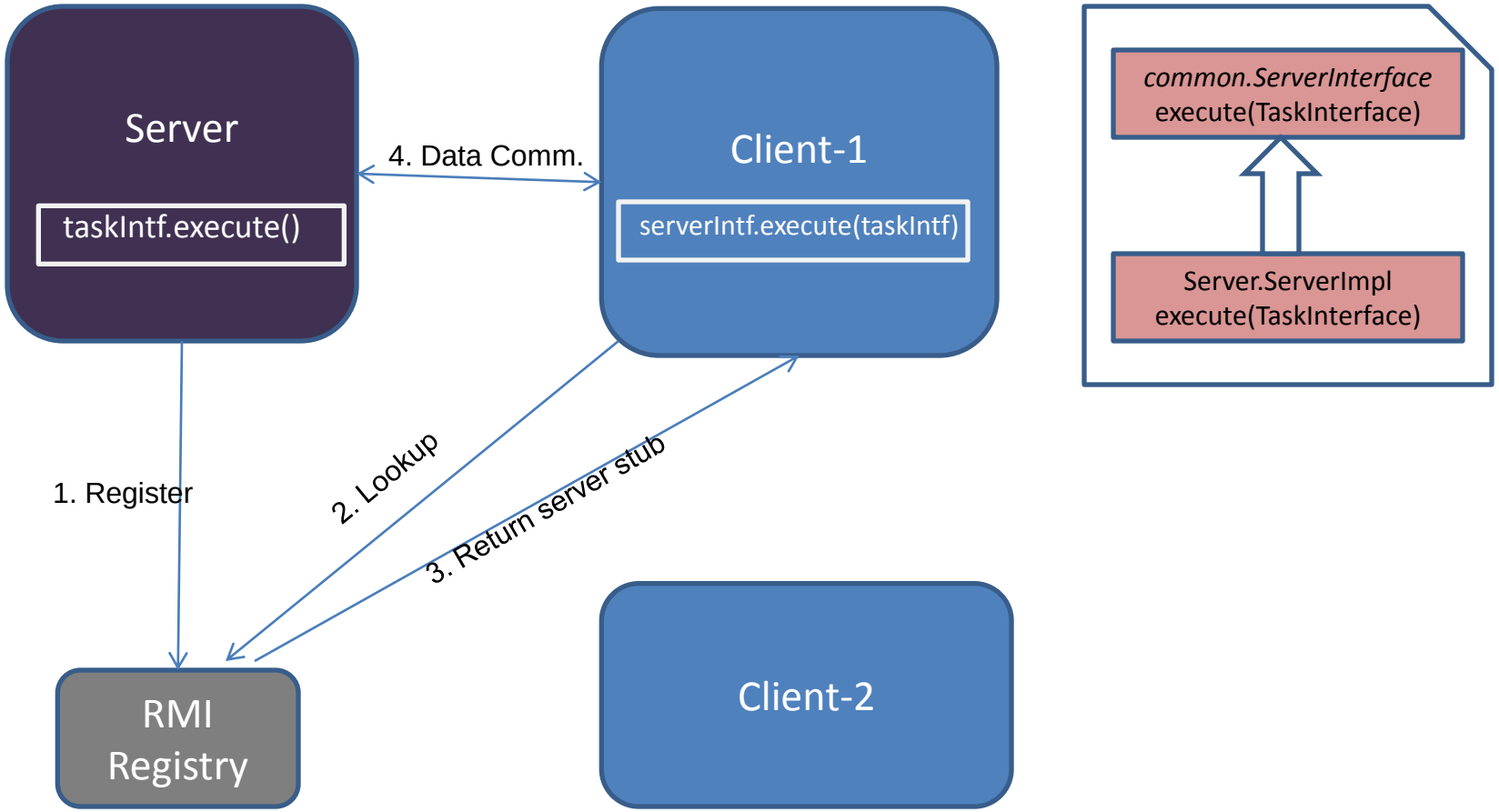
1. Alternative delegation model - Java EE web modules
   - ✓ Checks local repositories first, common folder in Tomcat. However loading of system classes remain unchanged
   - ✓ By instantiating class loader again, a class can be reloaded
2. Hot Deployment
   - ✓ Support upgrade
3. Class loader and Security
   - ✓ Add extra code after findClass() and before defineClass(), compression, encryption techniques
4. Modifying the class files
   - ✓ Add extra debugging logic

     Example: BCEL (Byte code engineering library) and ASM tools

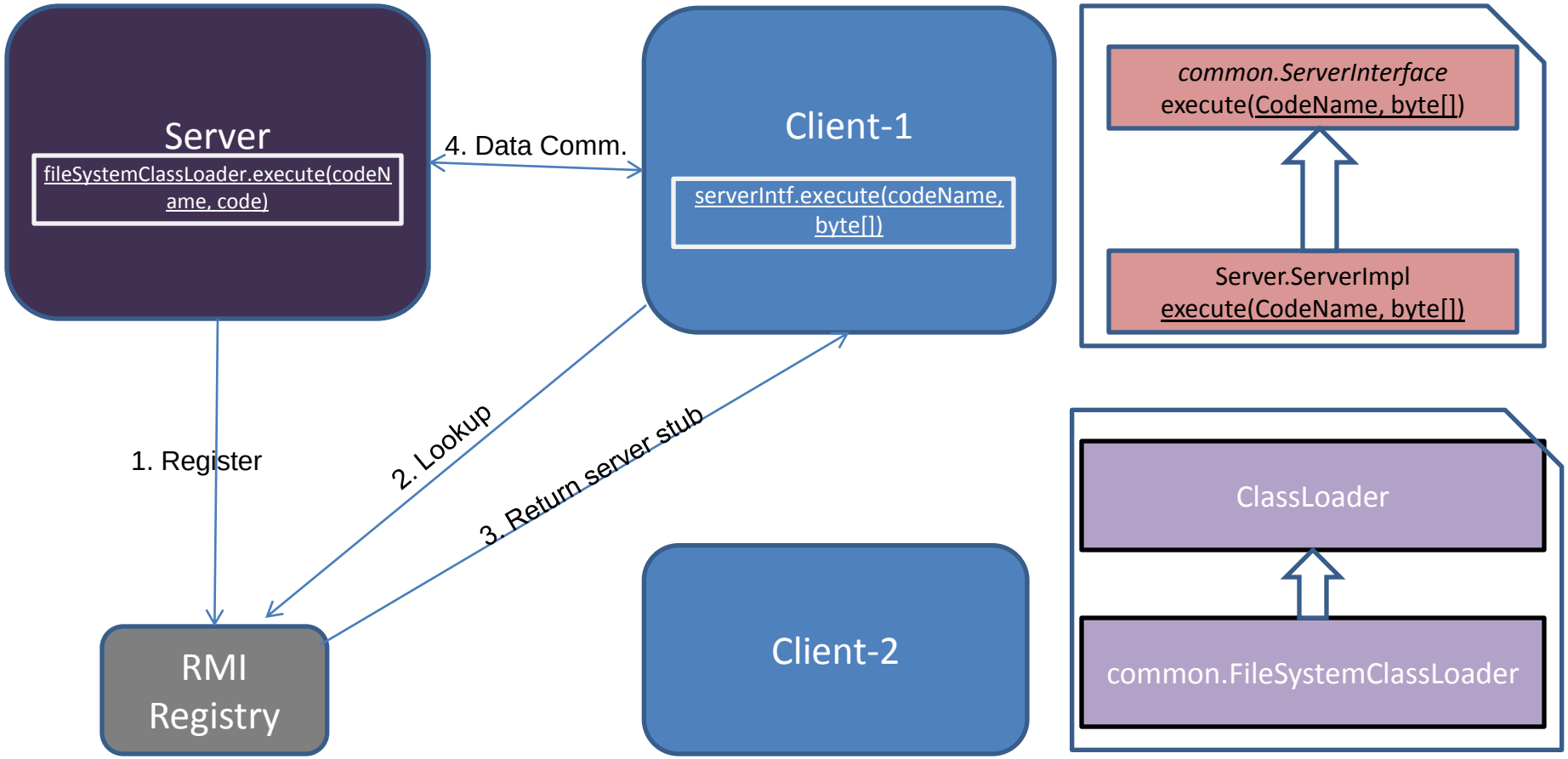
Example (1): Jars in same classpath



Example (2): RMI Execution Engine



Example (3): RMI Execution Engine



Class Loader Security

- Classes are separated using namespaces
- Built-in checks for identifying malicious classes
- Encloses class into ProtectionDomain
- Verification of code for valid signature
- Class File Verifier does various checks for integrity

ClassLoader Exceptions

1. ClassNotFoundException
   - ClassLoader.findSystemClass(), loadClass() fails
   - Wrong classloader is used or Dir is not added

   ✓ *Figure out what class loader and parent class loader and see why class cannot be loaded*

2. NoClassDefFoundError
   - Indicates linkage problem, Symbolic reference cannot be found.
   - Folder or source of class is not made available to parent class loader
   - Check the stacktrace to find the class name

   ✓ *Figure out class loader and missing symbolic link*
   ✓ *List parent class loaders recursively*

ClassLoader Exceptions – Contd.

3. ClassCastException
   - Casting an object to an unrelated class

   ✓ *Check for type and classloader used*

4. UnSatisfiedLinkError
   - System.loadLibrary(“solaris.image_converter”), loading JNI code

   ✓ *JVM is unable to find proper native library of class, check references*

5. ClassCircularityError
   - Thrown when some class is a indirect superclass of itself, an Interface extends itself or similar, mainly when diff versions of same library is loaded

   ✓ *Check for double class names in classpath*

Debugging Class Loading Problems

1. Use *java –verbose class HelloWord*
2. Use *javap –private HelloWord*
3. Linux check class file
   - *find *.jar –exec jar –tf '{}' \; | grep HelloWorld*
4. Use BCEL or ASM libraries, ByteCode visualizer for Eclipse

Questions - 1

- Difference between

  Class.forName() vs classLoader.loadClass()

Questions - 2

- In Java, what is the need for main method?

```
public static void main(String args[])
```

Questions - 3

- Guess first 4 bytes of a class file!

  Byte code generated by compiler need to have standard data at beginning of the file

Resources

1. The Java Language Specification, Java SE 8 Edition, https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf
2. The Java Virtual Machine Specification, Java SE 8 Edition, https://docs.oracle.com/javase/specs/jvms/se8/jvms8.pdf
3. Demystifying Java Platform Security Architecture, Ramesh Nagappan
4. Internals of Java Class Loading, Binildas Christudas, O'Reilly, OnJava.com
5. Core Security Patterns: Best Practices and Strategies for J2EE, Web Services and Identity Management, Sun MicroSystems, Prentice Hall
6. Java and JVM security vulnerabilities and their exploitation techniques
7. http://www.blackhat.com/presentations/bh-asia-02/LSD/bh-asia-02-lsd.pdf
8. GitHub URL - https://github.com/gopularam/developer/tree/master/Classloader
9. Slideshare URL - http://www.slideshare.net/bhanugopularam/java-class-loader-49366166

Thank You. Q/A