Leyden

Ashutosh Mehra

Fosdem 2025

Red Hat

Project Leyden is ...

Project Leyden is ...

- About improving startup, warmup and footprint
- Focuses on shifting computation
  - Forward or backward
    - GC - forward shift
    - CDS archive - backward shift

Slow startup

Slow to start because...

- Computations are done just-in-time/on-demand/lazily
  - Makes the Java platform dynamic
  - Gives flexibility and extensibility to Java developers
  - JVM benefits in multiple ways
    - Adaptability
    - Exploiting the cpu and hardware capabilities
    - Avoid wasted effort
  - At the expense of ... guess what? ... startup time

Opportunity ...

- Classes and links are stabilized
- Computations are repeated on every run
- Opportunity to shift computation ahead of time

```
public class HelloWorld {
  public static void main(String args[]) {
    System.out.println("Hello Fosdem");
  }
}
```

javac

```
0000000 feca beba 0000 4300 1d00 000a 0002 0703
0000010 0400 000c 0005 0106 1000 616a 6176 6c2f
0000020 6e61 2f67 624f 656a 7463 0001 3c06 6e69
0000030 7469 013e 0300 2928 0956 0800 0900 0007
0000040 0c0a 0b00 0c00 0001 6a10 7661 2f61 616c
…
```

javap

```
Constant pool:
   …
   #7 = Fieldref           #8.#9          //
java/lang/System.out:Ljava/io/PrintStream;
   #8 = Class              #10            // java/lang/System
   #9 = NameAndType        #11:#12        // out:Ljava/io/PrintStream;
   …
  #13 = String             #14            // Hello World
  #14 = Utf8               Hello World
  #15 = Methodref          #16.#17        //
java/io/PrintStream.println:(Ljava/lang/String;)V
  #16 = Class              #18            // java/io/PrintStream
  …


  public static void main(java.lang.String[]);
    descriptor: ([Ljava/lang/String;)V
    flags: (0x0009) ACC_PUBLIC, ACC_STATIC
    Code:
      stack=2, locals=1, args_size=1
         0: getstatic     #7   // Field
java/lang/System.out:Ljava/io/PrintStream;
         3: ldc           #13                  // String Hello World
         5: invokevirtual #15  // Method
java/io/PrintStream.println:(Ljava/lang/String;)V
         8: return
```

```
public class HelloWorld {
  public static void main(String args[]) {
    System.out.println("Hello Fosdem");
  }
}
```

javac

```
0000000 feca beba 0000 4300 1d00 000a 0002 0703
0000010 0400 000c 0005 0106 1000 616a 6176 6c2f
0000020 6e61 2f67 624f 656a 7463 0001 3c06 6e69
0000030 7469 013e 0300 2928 0956 0800 0900 0007
0000040 0c0a 0b00 0c00 0001 6a10 7661 2f61 616c
…
```

javap

```
public class System {
  …
  public static PrintStream out;
  …
}
```

```
public class PrintStream {
  …
}
```

```
public class String {
  …
}
```

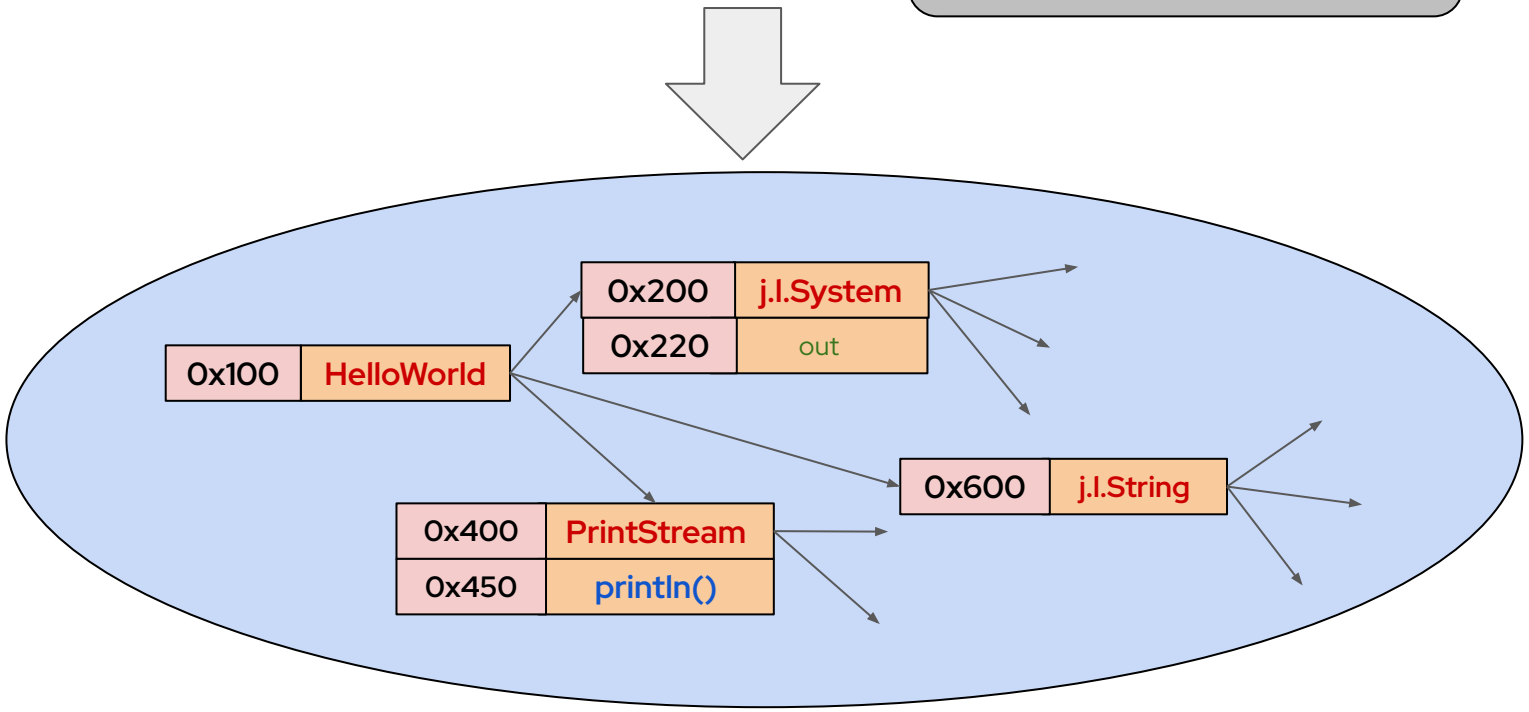
```
Constant pool:
   …
   #7 = Fieldref           #8.#9          //
java/lang/System.out:Ljava/io/PrintStream;
   #8 = Class              #10            // java/lang/System
   #9 = NameAndType        #11:#12        // out:Ljava/io/PrintStream;
   …
  #13 = String             #14            // Hello Fosdem
  #14 = Utf8               Hello Fosdem
  #15 = Methodref          #16.#17        //
java/io/PrintStream.println:(Ljava/lang/String;)V
  #16 = Class              #18            // java/io/PrintStream
  …


  public static void main(java.lang.String[]);
    descriptor: ([Ljava/lang/String;)V
    flags: (0x0009) ACC_PUBLIC, ACC_STATIC
    Code:
      stack=2, locals=1, args_size=1
         0: getstatic     #7   // Field java/lang/System.out:Ljava/io/PrintStream;
         3: ldc           #13                  // String Hello World
         5: invokevirtual #15  // Method
java/io/PrintStream.println:(Ljava/lang/String;)V
         8: return
```

```
public class HelloWorld {
  public static void main(String args[]) {
    System.out.println("Hello Fosdem");
  }
}
```

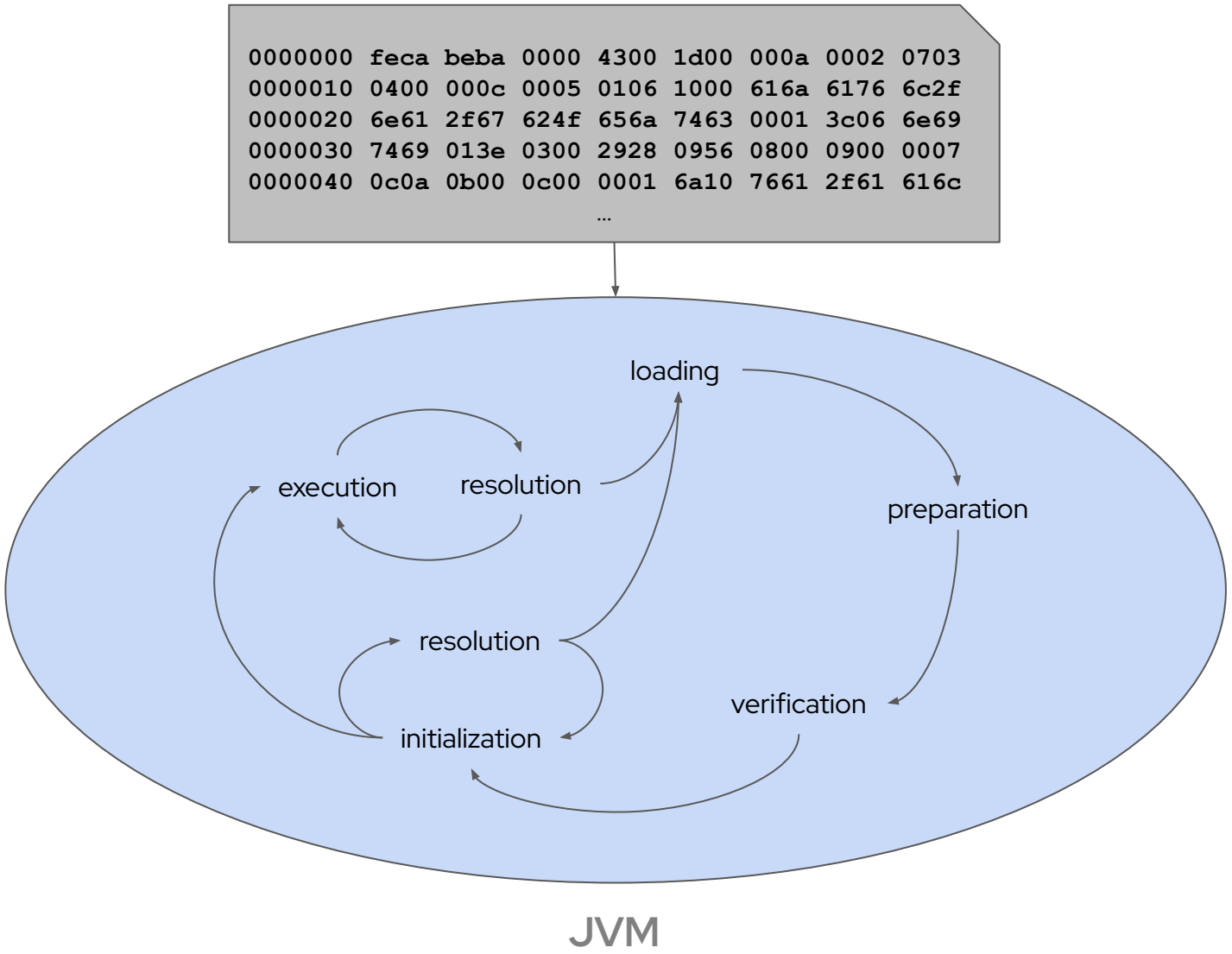
```
public class System {
  …
  public static PrintStream out;
  …
}
```

```
public class String {
  …
}
```

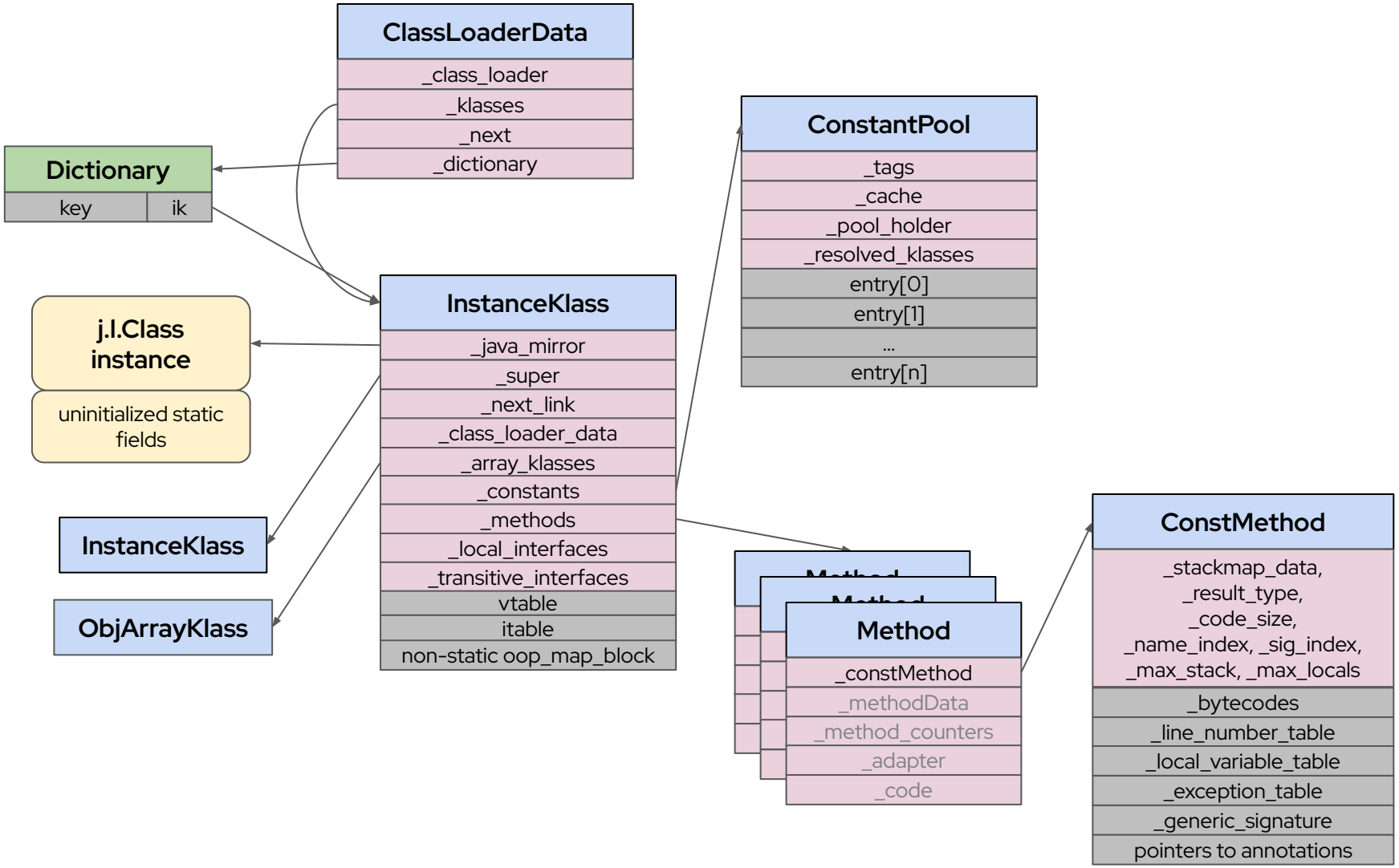
```
public class PrintStream {
  …
}
```



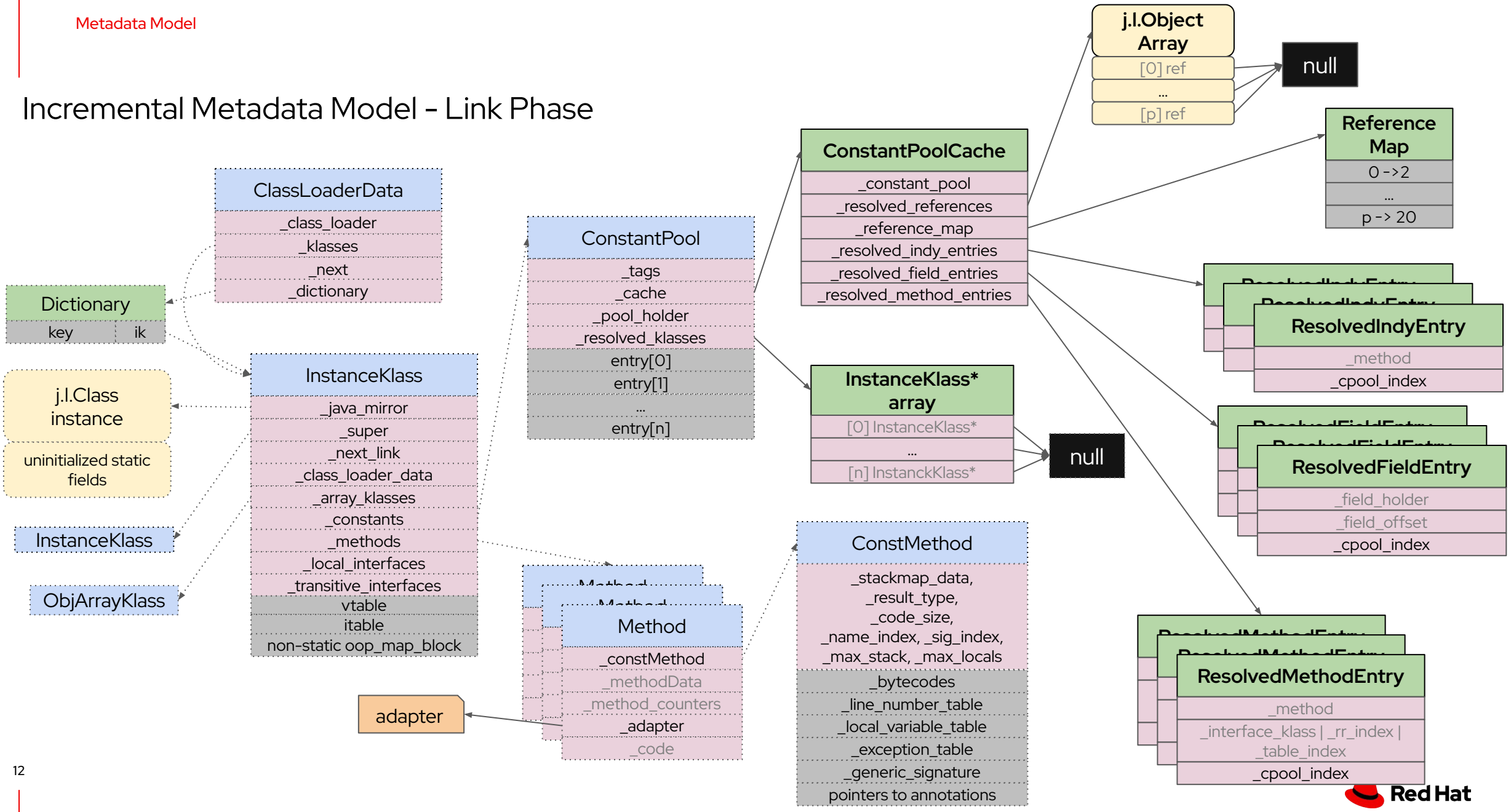
JVM


0000000 feca beba 0000 4300 1d00 000a 0002 0703
0000010 0400 000c 0005 0106 1000 616a 6176 6c2f
0000020 6e61 2f67 624f 656a 7463 0001 3c06 6e69
0000030 7469 013e 0300 2928 0956 0800 0900 0007
0000040 0c0a 0b00 0c00 0001 6a10 7661 2f61 616c
...
loading
execution
resolution
preparation
resolution
verification
initialization
JVM

Incremental Metadata Model - Load Phase

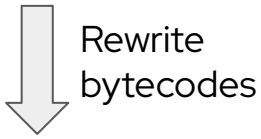


Incremental Metadata Model - Link Phase



Field resolution in Action

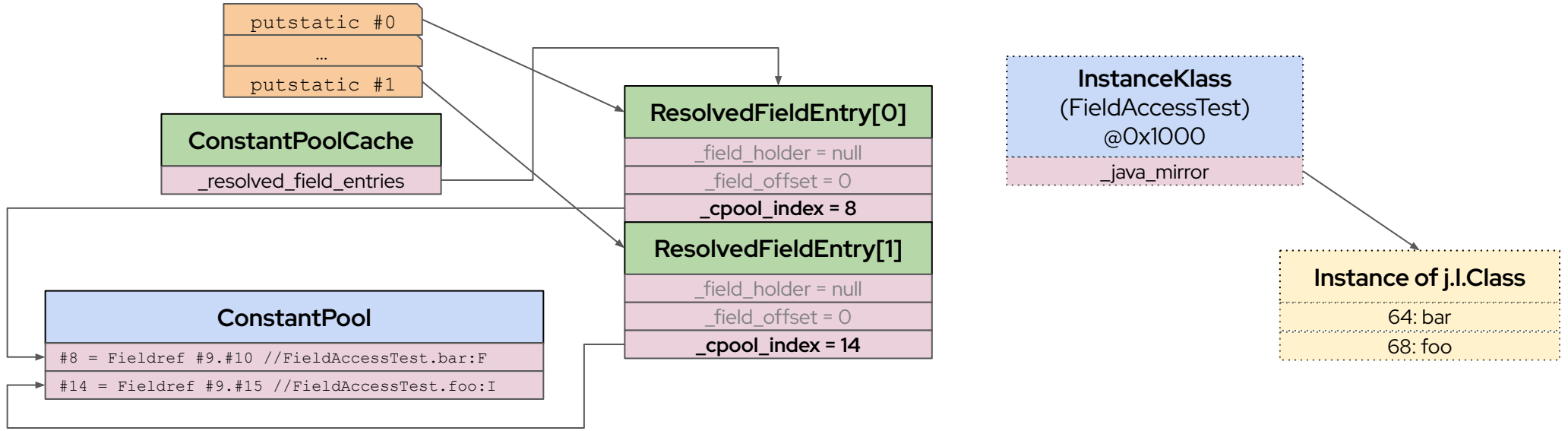
```
0: ldc           #7  // float 150.0f
2: putstatic     #8  // Field bar:F
5: bipush        20
7: putstatic     #14 // Field foo:I
10: return
```



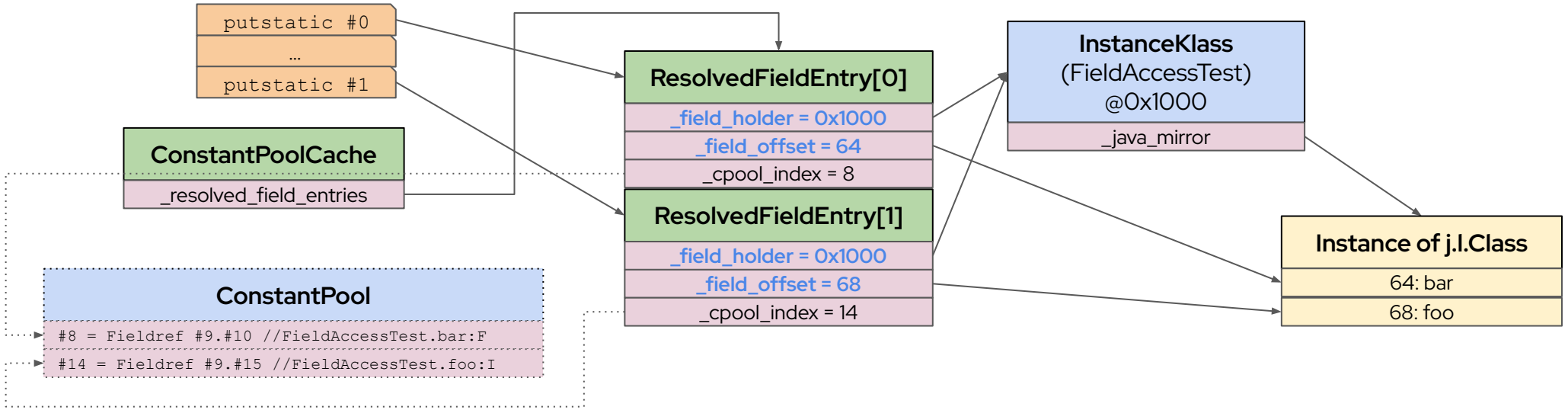

```
0: ldc        #7  // float 150.0f
2: putstatic  #0  // index in resolved_field_entries
5: bipush     20
7: putstatic  #1  // index in resolved_field_entries
10: return
```

```
Constant pool:
   #1 = Methodref          #2.#3          // java/lang/Object."<init>":()V
   #2 = Class              #4             // java/lang/Object
   #3 = NameAndType        #5:#6          // "<init>":()V
   #4 = Utf8               java/lang/Object
   #5 = Utf8               <init>
   #6 = Utf8               ()V
   #7 = Float              150.0f
   #8 = Fieldref           #9.#10         // FieldAccessTest.bar:F
   #9 = Class              #11            // FieldAccessTest
  #10 = NameAndType        #12:#13        // bar:F
  #11 = Utf8               FieldAccessTest
  #12 = Utf8               bar
  #13 = Utf8               F
  #14 = Fieldref           #9.#15         // FieldAccessTest.foo:I
  #15 = NameAndType        #16:#17        // foo:I
  #16 = Utf8               foo
  #17 = Utf8               I
  ...
```

Before Resolution

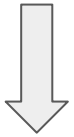


After Resolution



LDC resolution in Action

```
0: ldc  #7  // String Fosdem
2: invokestatic  #9  // Method count:(Ljava/lang/String;)I
5: pop
6: ldc  #15  // String OpenJDK
8: invokestatic  #9   // Method count:(Ljava/lang/String;)I
```

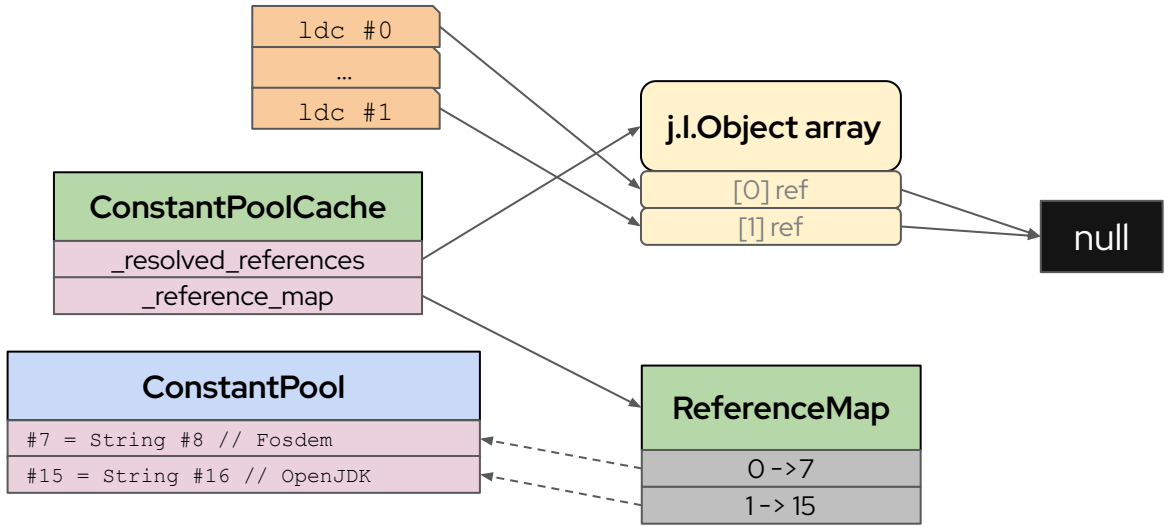


Rewrite bytecodes

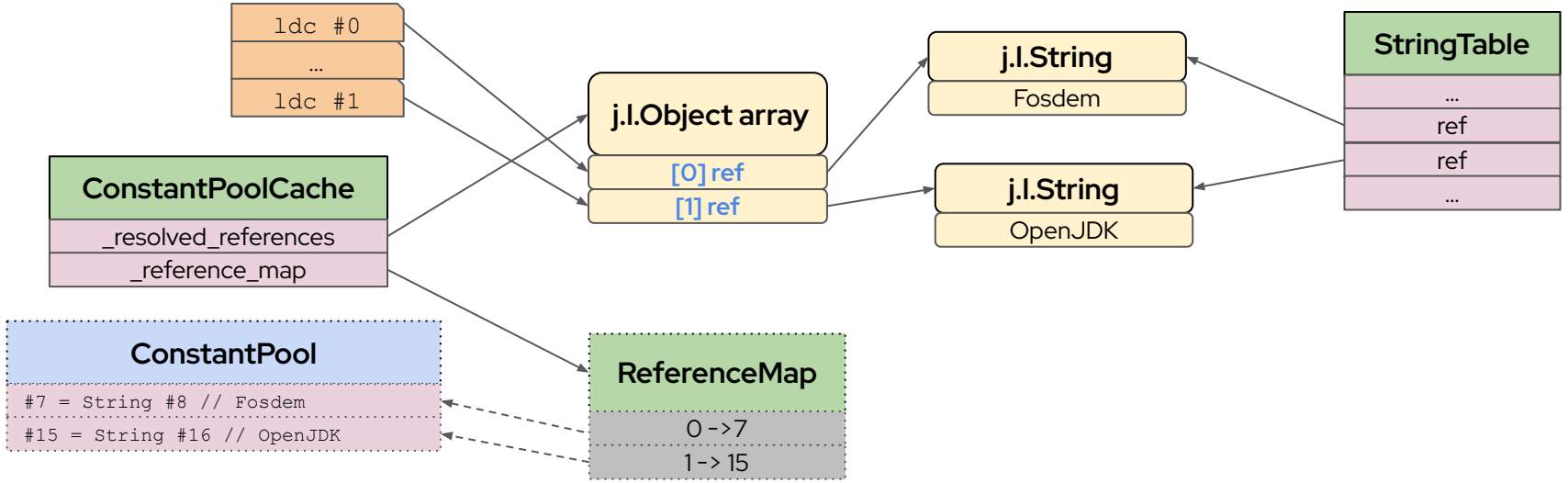
```
0: ldc  #0  // Index in resolved_references
2: invokestatic  #9  // Method count:(Ljava/lang/String;)I
5: pop
6: ldc  #1  // Index in resolved_references
8: invokestatic  #9   // Method count:(Ljava/lang/String;)I
```

```
Constant pool:
  ...
   #7 = String             #8             // Fosdem
   #8 = Utf8               Fosdem
   #9 = Methodref          #10.#11        // LDCTest.count:(Ljava/lang/String;)I
  #10 = Class              #12            // LDCTest
  #11 = NameAndType        #13:#14        // count:(Ljava/lang/String;)I
  #12 = Utf8               LDCTest
  #13 = Utf8               count
  #14 = Utf8               (Ljava/lang/String;)I
  #15 = String             #16            // OpenJDK
  #16 = Utf8               OpenJDK
  ...
```

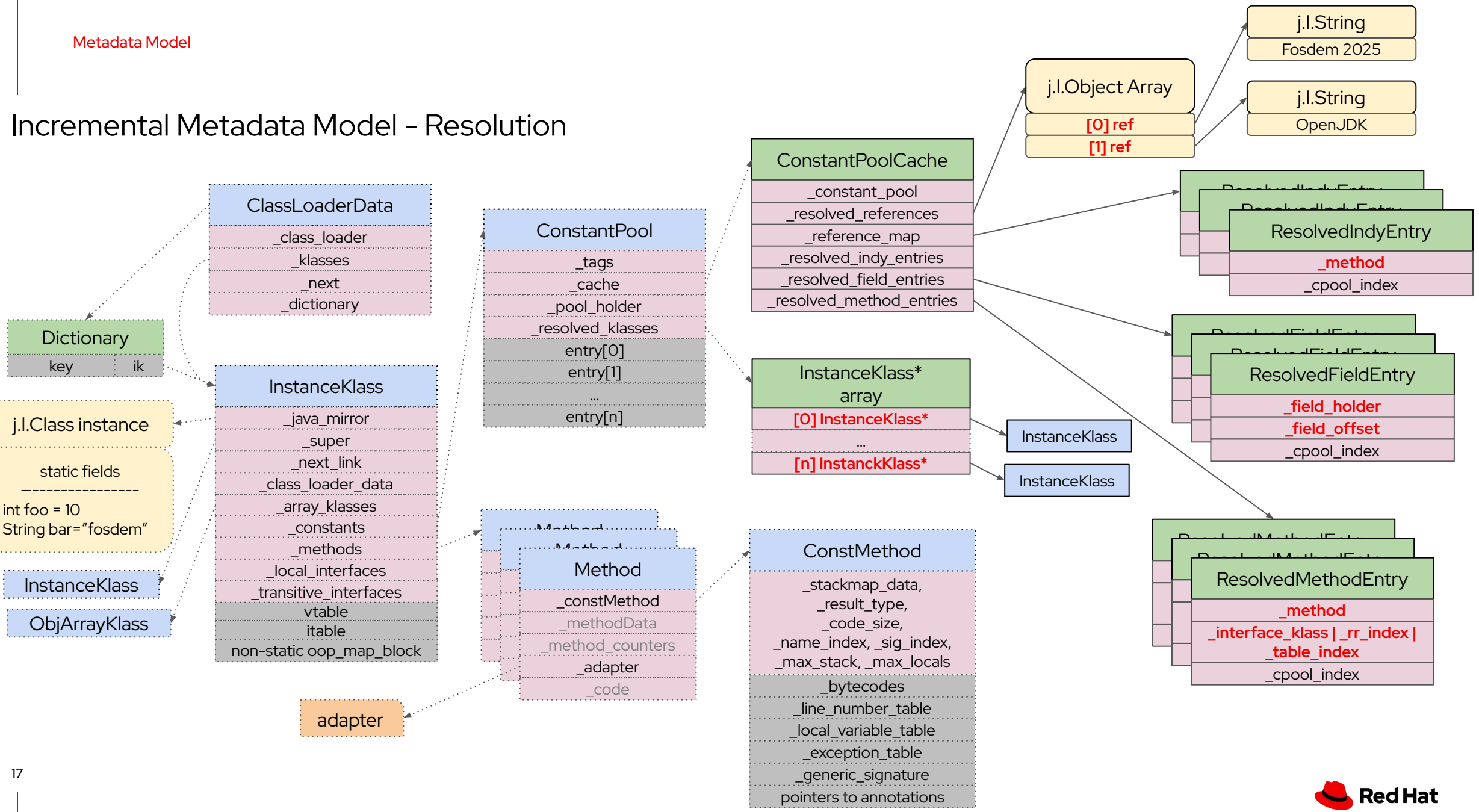
Before Resolution



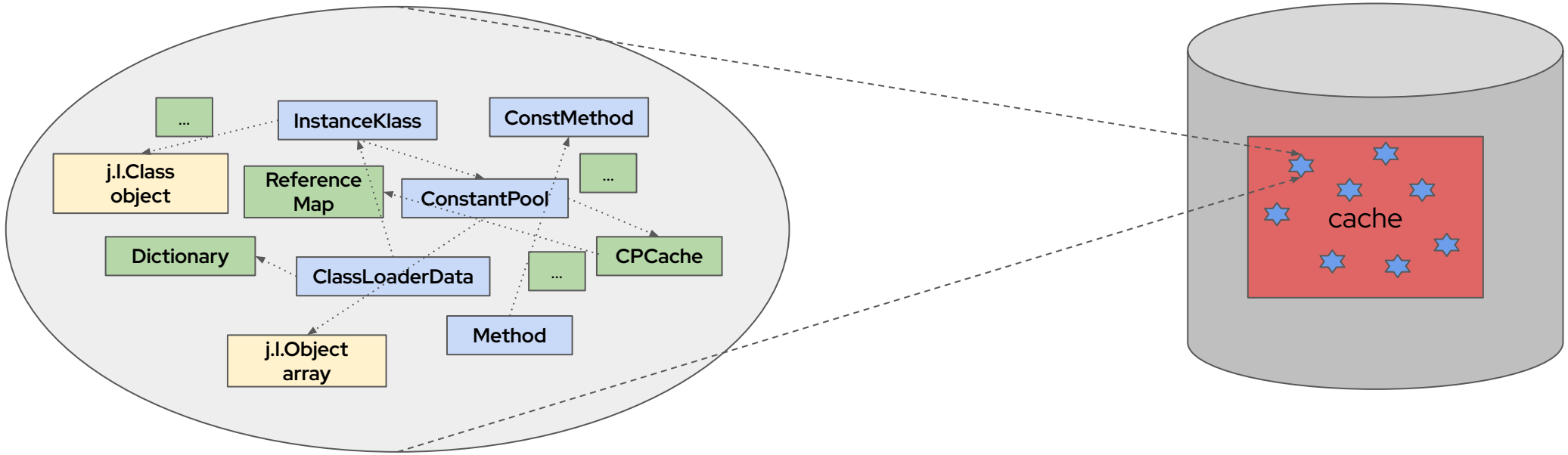
After Resolution



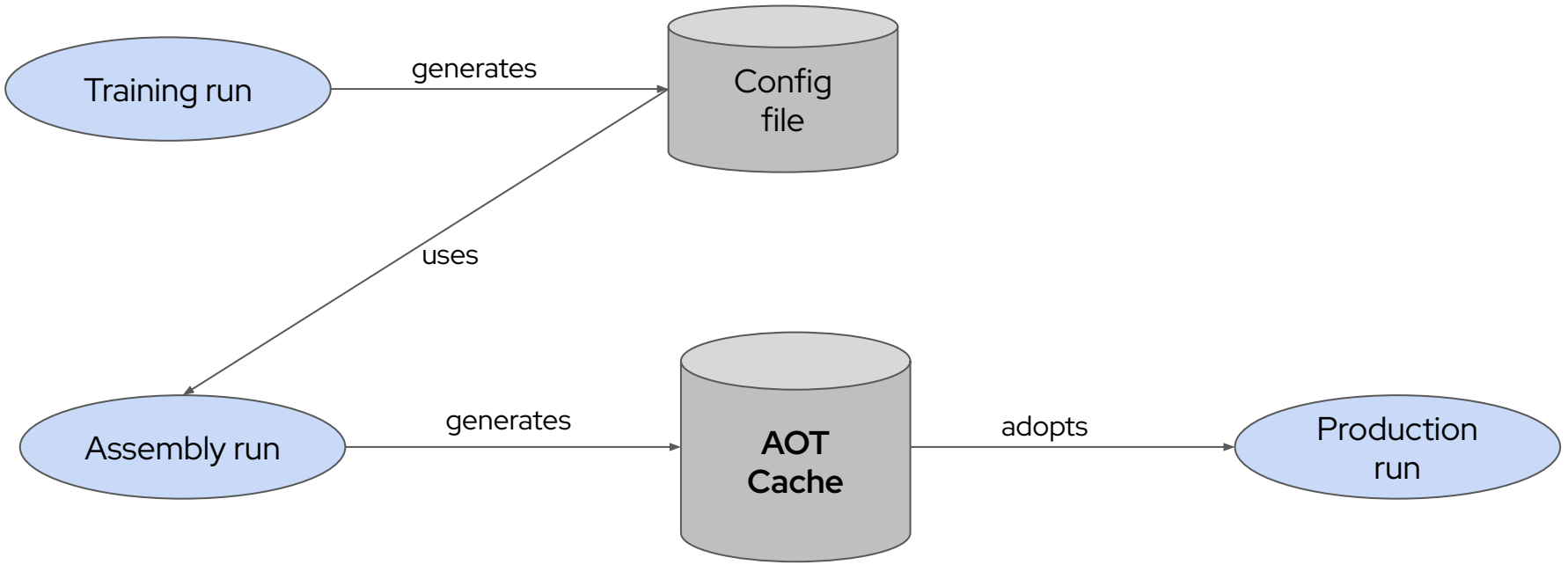
Incremental Metadata Model - Resolution

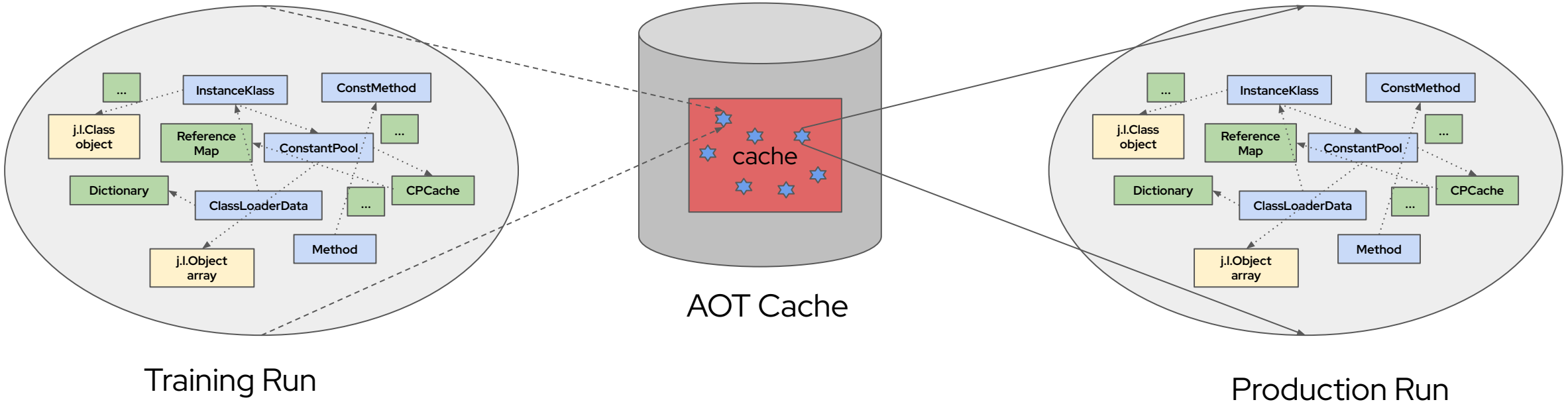


Red Hat


...
InstanceKlass
ConstMethod
j.l.Class object
Reference Map
ConstantPool
...
Dictionary
ClassLoaderData
...
CPCache
Method
j.l.Object array
cache

JEP 483 - AOT Class Loading and Linking




...
InstanceKlass
ConstMethod
j.l.Class object
Reference Map
ConstantPool
...
Dictionary
ClassLoaderData
...
CPCache
Method
j.l.Object array
cache
AOT Cache
Training Run
Production Run

Challenges

- User-defined classloaders
  - Lack of identity across runs
  - Do not have a well-defined behavior

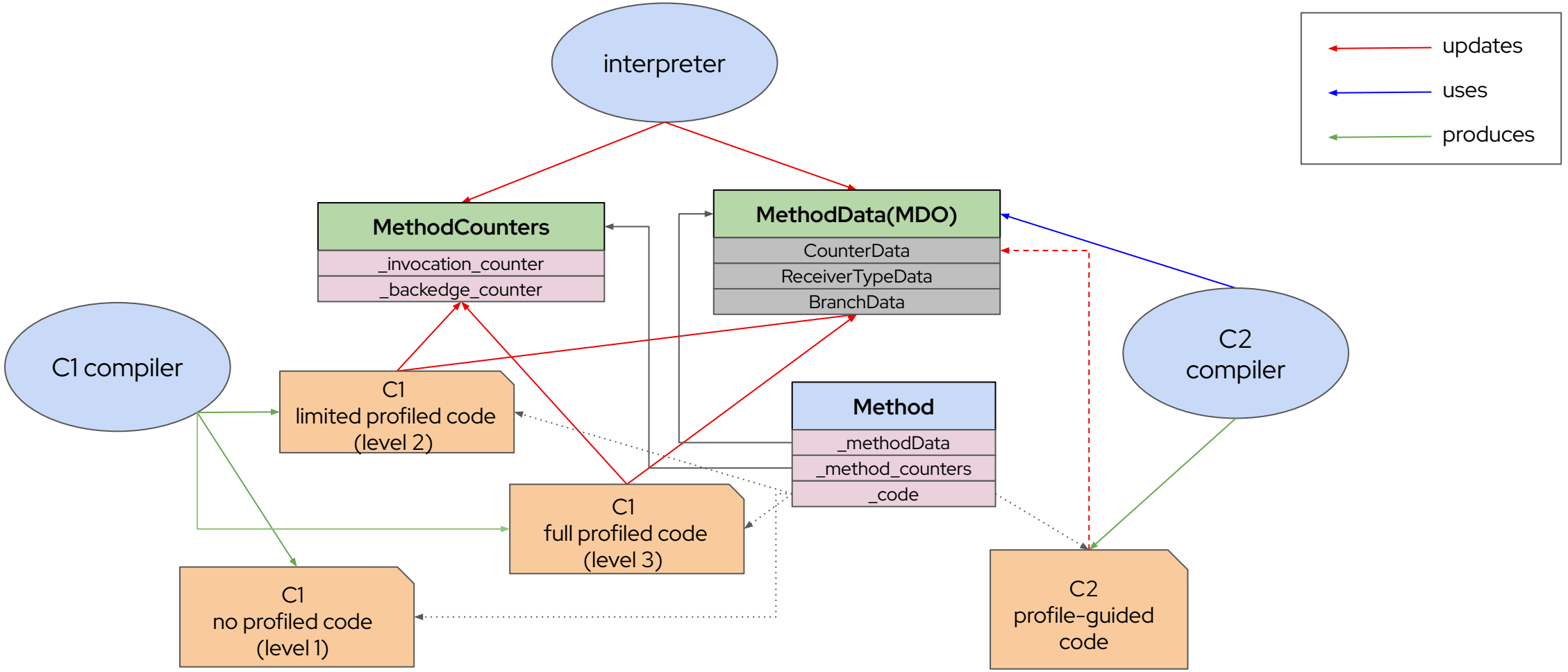
  Class initializers
  - Soupy in nature with side-effects
  - Can pull in dependency on the environment

Part 2 - What's next?

What’s next?

- AOT Profiling
- AOT Compilation

Profiling Model



AOT Profiling

AOT Profiling

- Archiving of Method Profiles planned in JDK-8325147
  - Schedule compilations earlier
  - Complements on-line profiles
  - May avoid some deopts & recompiles

AOT Compilation

AOT Compilation

- Archiving of Generated Code planned in JDK-8335368
  - Store compiled Java method code
    - Level 4 (C2) and Level 3 (C1)
  - Can still recompile (including after deopt)
  - Store generated stubs, Interpreter, adapters

When does it arrive?

When does it arrive?

- JEP-483 Ahead of Time Class Loading & Linking
  - Delivered to JDK24
- JEP-8325147 Ahead of Time Method Profiling
  - Likely for JDK25
- JEP-8335368 Ahead of Time Code Compilation
  - Possibly for JDK25 for Java methods and stubs
- Leyden EA build https://jdk.java.net/leyden/
- All features are available in Leyden repo prototype
  - https://github.com/openjdk/leyden/tree/premain

Q & A

Red Hat