CS266 Software Reverse Engineering (SRE)
Reversing and Patching Java Bytecode

Teodoro (Ted) Cipresso
Senior Software Engineer, IBM
SRE Guest Lecture

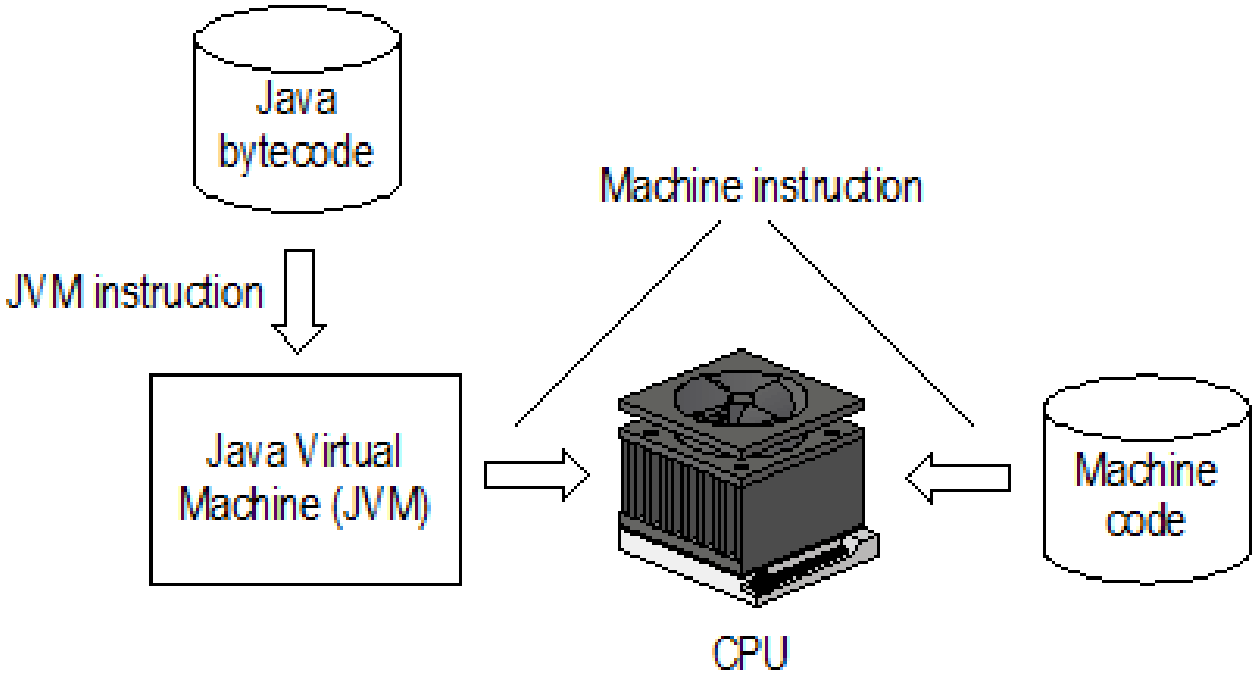
The information in this presentation is taken from the thesis “Software reverse engineering education” available at http://scholarworks.sjsu.edu/etd_theses/3734/ where all citations can be found.

Reversing and Patching Java Bytecode

- Good-quality Java source can often be generated from Java bytecode with little difficulty due to certain characteristics of bytecode:
  - Platform-independent (consistent) instruction set and layout/format.
  - Very rich, well-structured metadata about Classes, Methods, and Variables:
    - names and datatypes (e.g., String personName, Map personRecord).
    - Method signatures (includes Constructors).
- Generating HLL source (e.g., C/C++) from machine code is challenging due to high variation in the output of compilers on different platforms and unavoidable loss of information that occurs when compiling a HLL down to machine code.
  - *Note*: Symbolic information may be included in machine code when specifying a compiler's "debug" option (e.g., -g, -debug, TEST).

Reversing and Patching Java Bytecode (cont'd)

- The following diagram illustrates the difference in how Java bytecode and machine code are processed during execution:



Reversing and Patching Java Bytecode (cont'd)

- The following formal definitions of machine code and Java bytecode apply:
  - **Machine code**: "Machine code or machine language is a system of instructions and data executed directly by a computer's central processing unit" [14]. Machine code contains the platform-specific machine instructions to execute on the target processor.
  - **Java bytecode**: "Bytecode is the intermediate representation of Java programs just as assembler is the intermediate representation of C or C++ programs" [15]. Java bytecode contains platform-independent instructions that are translated to platform-specific instructions by a Java Virtual Machine.

Reversing and Patching Java Bytecode (cont'd)

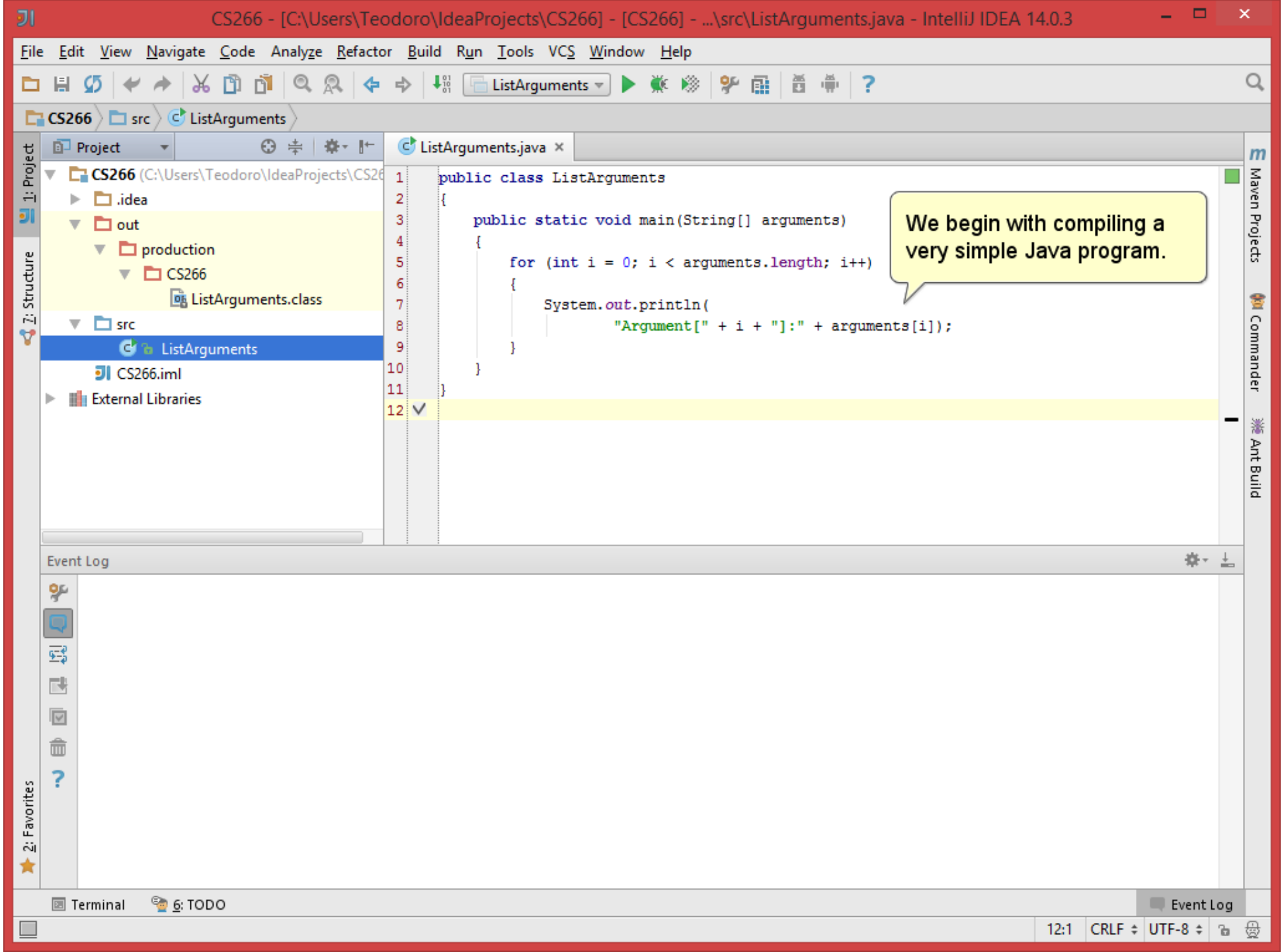
- Machine code is stored in files with varying extensions (**.exe*, **.dll, *.so,..,);* extensions are dependent upon the operating system.
- On the contrary…
  - Java bytecode is *always* stored in files that have a **.class* extension.
- The Java Language Specification allows at most one top-level public class to be defined per **.java* source file and requires that the bytecode be stored in a file with whose name matches the pattern *TopLevelClassName.class*.
- Collections of Java classes, such as those for an application or class library, are stored together in an archive file with a **.jar* extension.

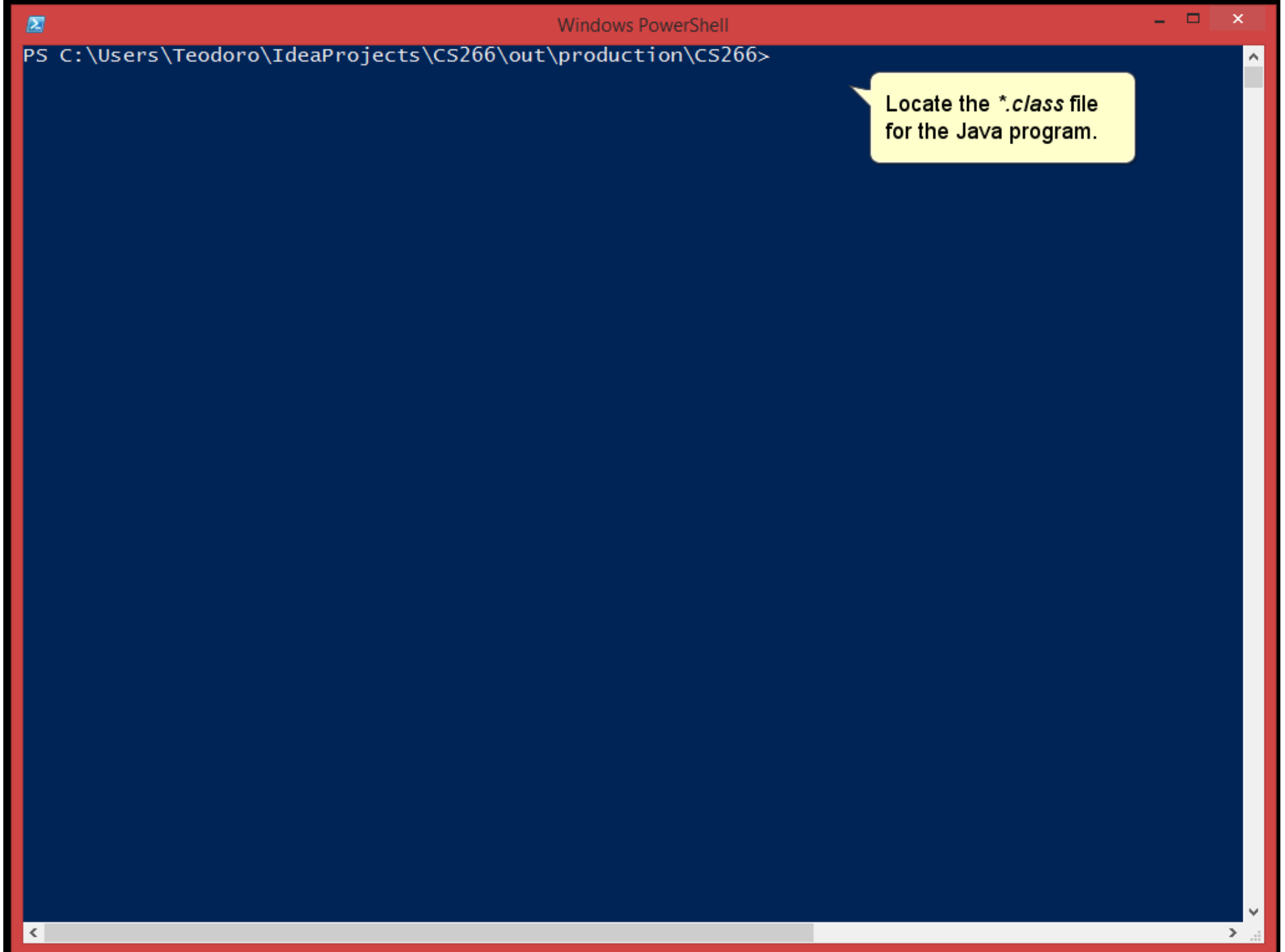
Reversing and Patching Java Bytecode

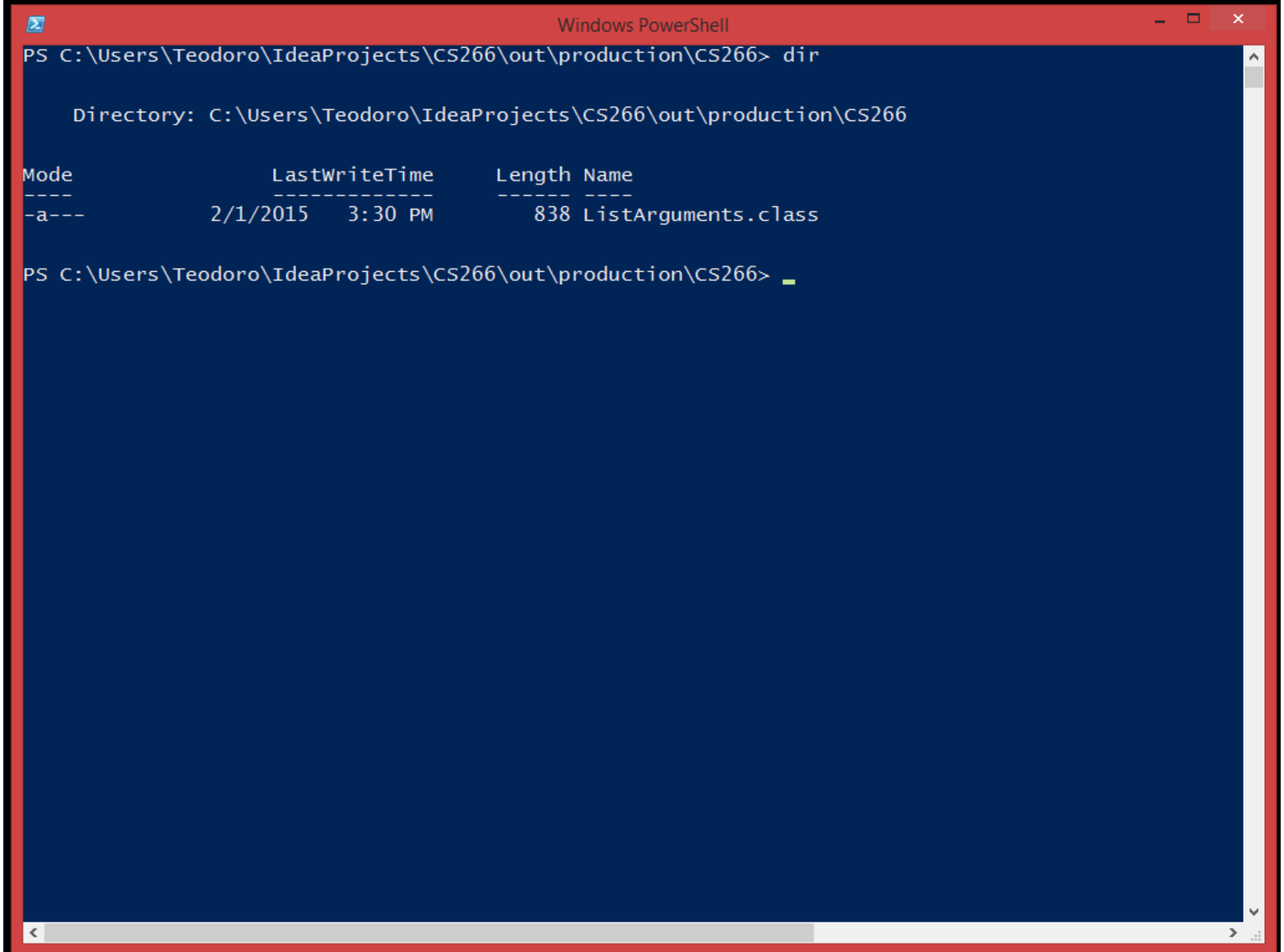
Decompiling and Disassembling Java Bytecode

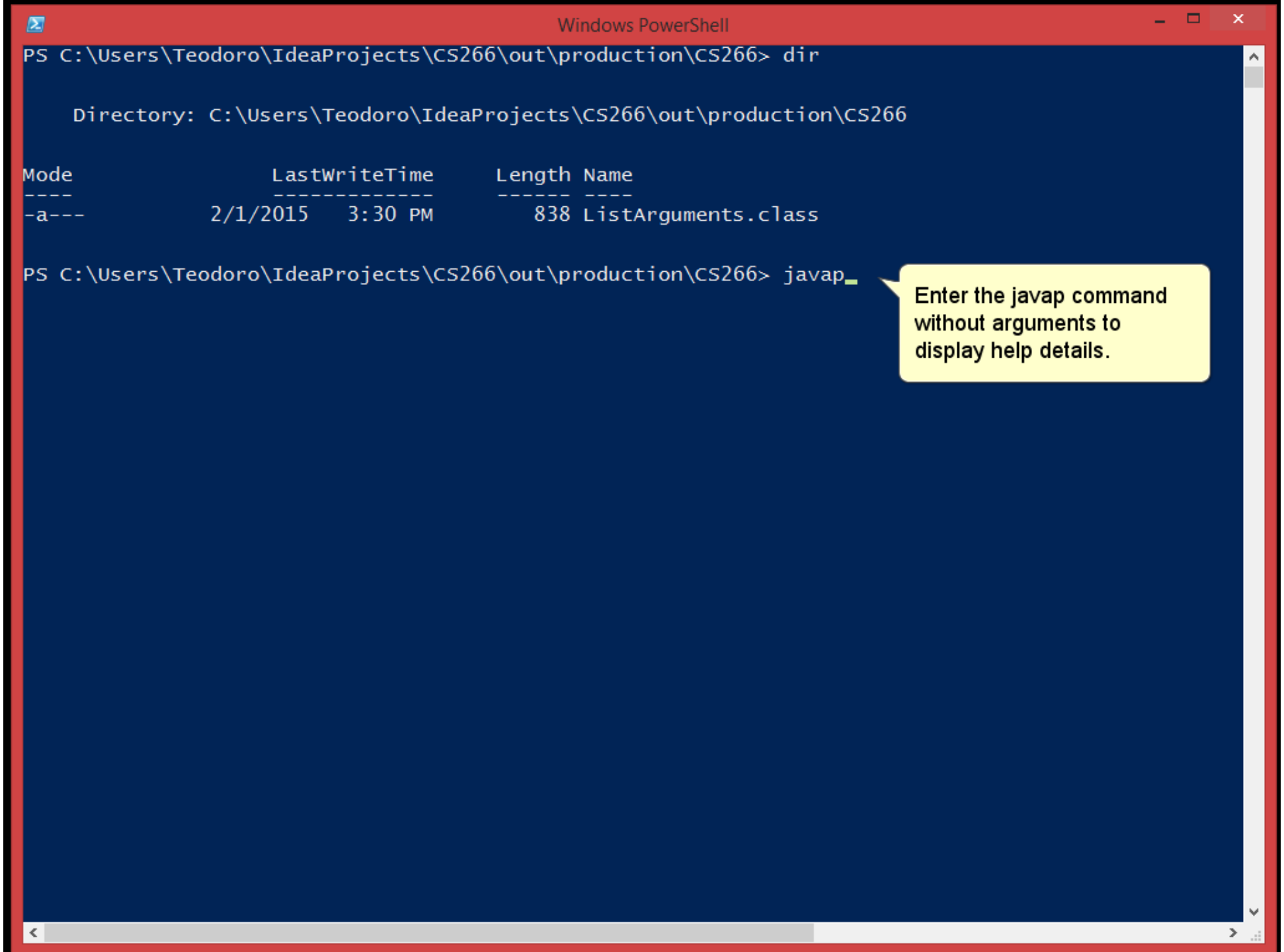
- Bytecode is stored in a binary format that is not human-readable and therefore must be "disassembled" in order to be read.
- Oracle's Java Development Toolkit (JDK) comes with javap, a command-line tool for "disassembling" Java bytecode.
- To say that javap disassembles bytecode is a misnomer because the output of javap is unstructured text which cannot be compiled back to bytecode.
  - The assumption is you already have the **.class* file.
- The output of javap is nonetheless useful as a debugging and performance tuning aid since one can see which JVM instructions are generated from high-level Java language statements.

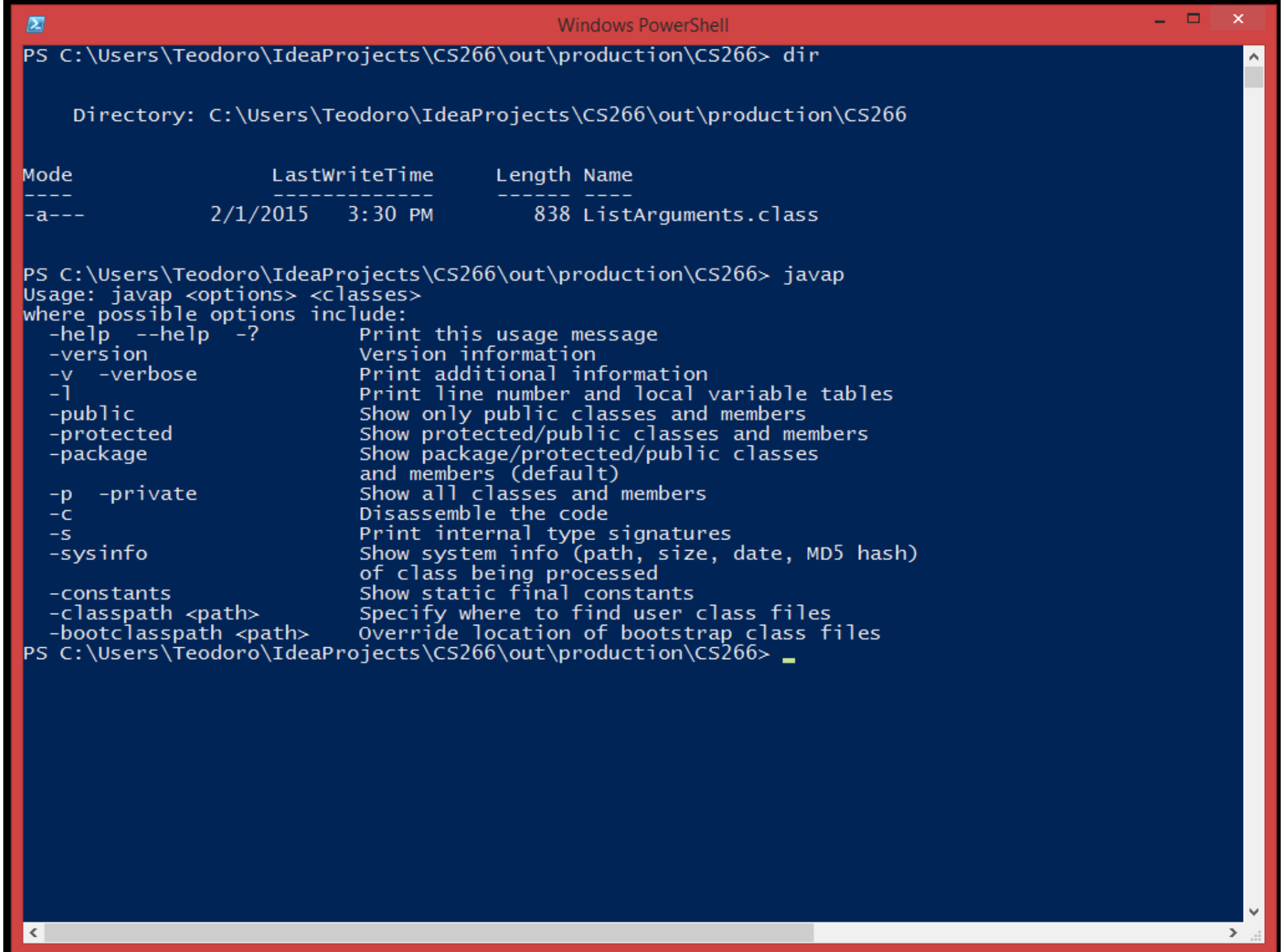
javap demo

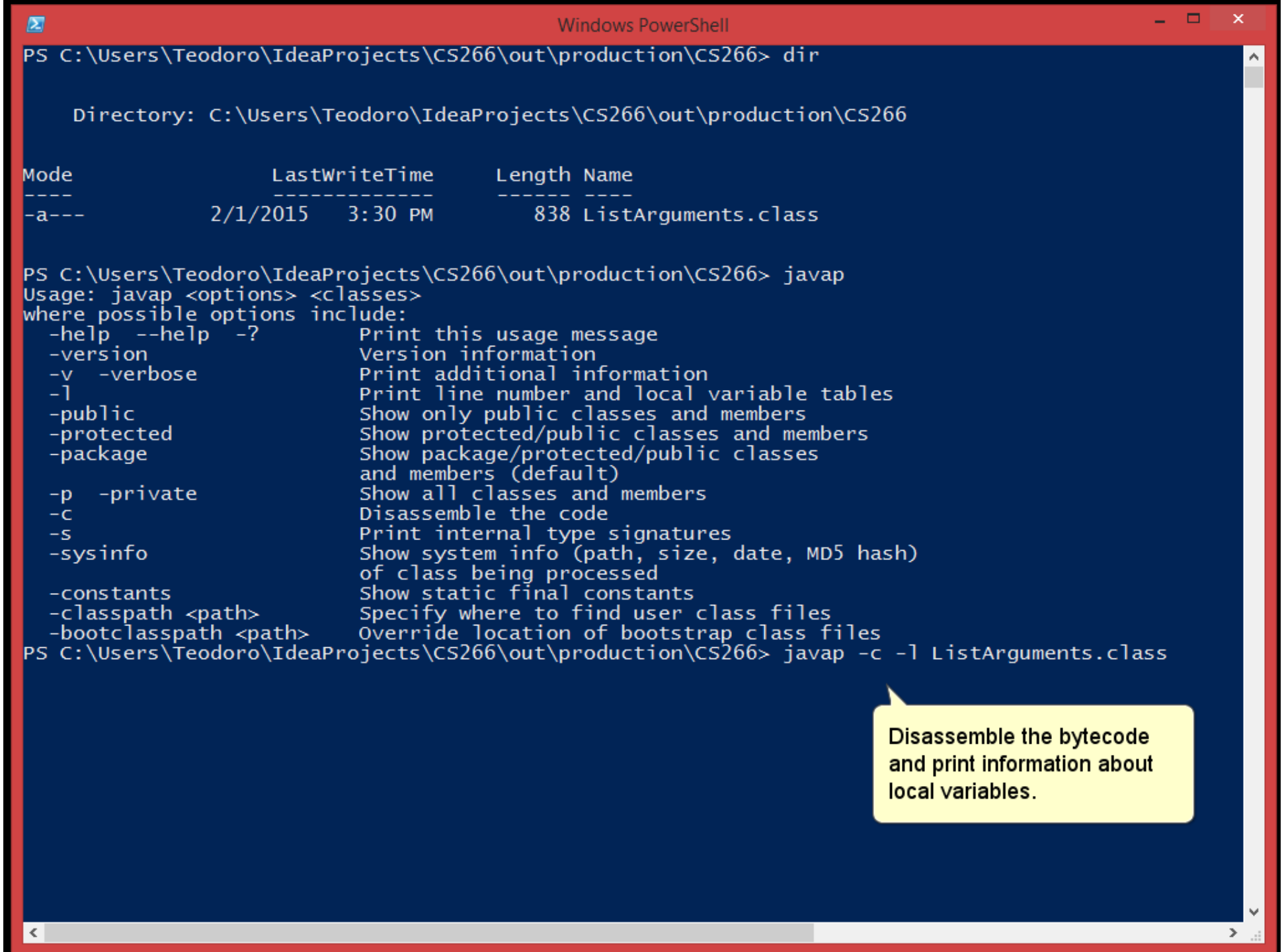

CS266 - [C:\Users\Teodoro\IdeaProjects\CS266] - [CS266] - ...\src\ListArguments.java - IntelliJ IDEA 14.0.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ListArguments
CS266 src ListArguments
Project
CS266 (C:\Users\Teodoro\IdeaProjects\CS26
.idea
out
production
CS266
ListArguments.class
src
ListArguments
CS266.iml
External Libraries
ListArguments.java
1 public class ListArguments
2 {
3 public static void main(String[] arguments)
4 {
5 for (int i = 0; i < arguments.length; i++)
6 {
7 System.out.println(
8 "Argument[" + i + "]:" + arguments[i]);
9 }
10 }
11 }
12
We begin with compiling a very simple Java program.
Event Log
1: Project
7: Structure
2: Favorites
Maven Projects
Commander
Ant Build
Terminal
6: TODO
Event Log
12:1
CRLF
UTF-8


Windows PowerShell
PS C:\Users\Teodoro\IdeaProjects\CS266\out\production\CS266>
Locate the *.class file for the Java program.


Windows PowerShell
PS C:\Users\Teodoro\IdeaProjects\CS266\out\production\CS266> dir
Directory: C:\Users\Teodoro\IdeaProjects\CS266\out\production\CS266
Mode LastWriteTime Length Name
-a--- 2/1/2015 3:30 PM 838 ListArguments.class
PS C:\Users\Teodoro\IdeaProjects\CS266\out\production\CS266>


Windows PowerShell
PS C:\Users\Teodoro\IdeaProjects\CS266\out\production\CS266> dir
Directory: C:\Users\Teodoro\IdeaProjects\CS266\out\production\CS266
Mode LastWriteTime Length Name
-a--- 2/1/2015 3:30 PM 838 ListArguments.class
PS C:\Users\Teodoro\IdeaProjects\CS266\out\production\CS266> javap
Enter the javap command without arguments to display help details.


Windows PowerShell
PS C:\Users\Teodoro\IdeaProjects\CS266\out\production\CS266> dir
Directory: C:\Users\Teodoro\IdeaProjects\CS266\out\production\CS266
Mode LastWriteTime Length Name
-a--- 2/1/2015 3:30 PM 838 ListArguments.class
PS C:\Users\Teodoro\IdeaProjects\CS266\out\production\CS266> javap
Usage: javap <options> <classes>
where possible options include:
-help --help -? Print this usage message
-version Version information
-v -verbose Print additional information
-l Print line number and local variable tables
-public Show only public classes and members
-protected Show protected/public classes and members
-package Show package/protected/public classes
and members (default)
-p -private Show all classes and members
-c Disassemble the code
-s Print internal type signatures
-sysinfo Show system info (path, size, date, MD5 hash)
of class being processed
-constants Show static final constants
-classpath <path> Specify where to find user class files
-bootclasspath <path> Override location of bootstrap class files
PS C:\Users\Teodoro\IdeaProjects\CS266\out\production\CS266>

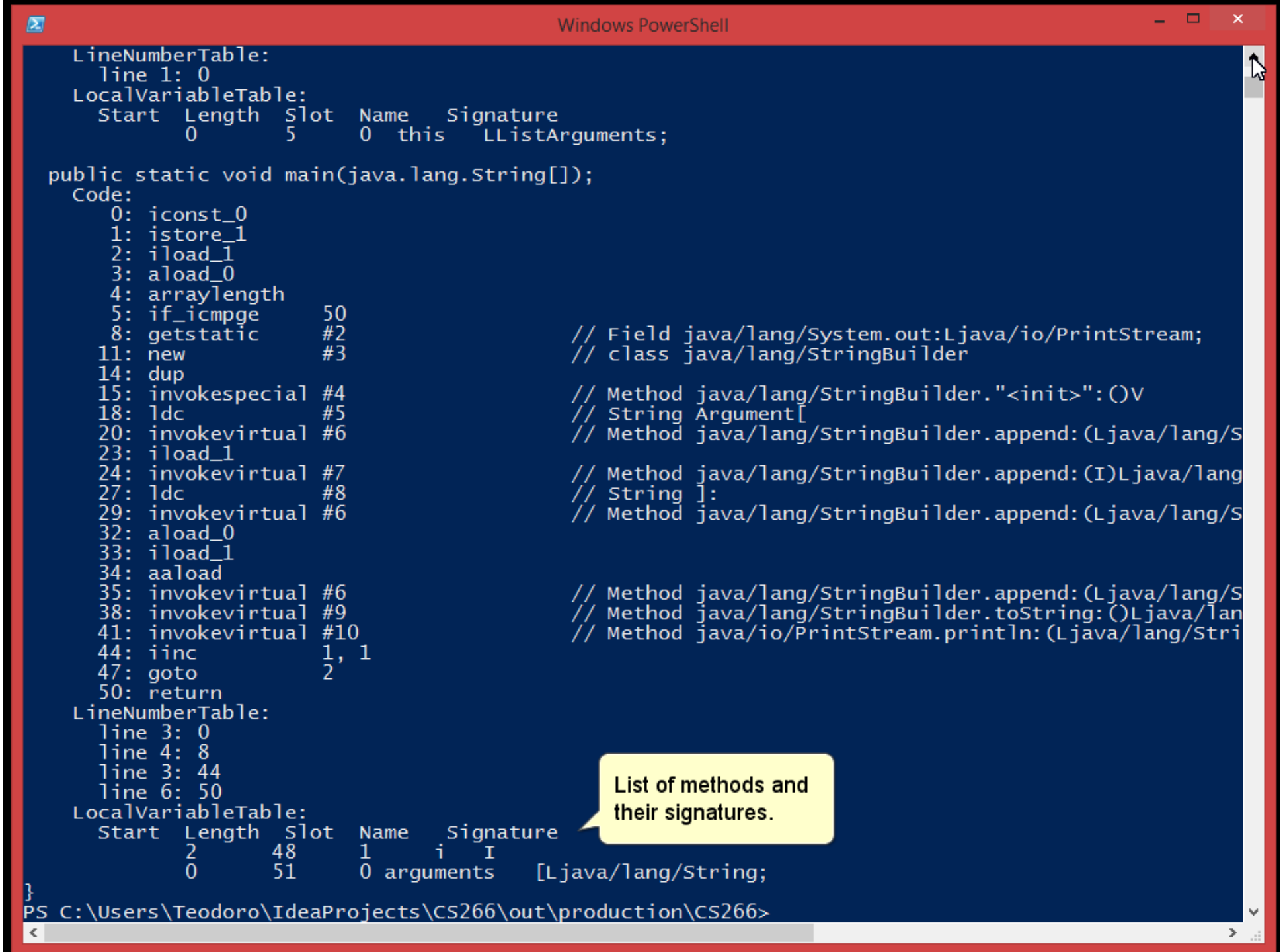

Windows PowerShell
PS C:\Users\Teodoro\IdeaProjects\CS266\out\production\CS266> dir
Directory: C:\Users\Teodoro\IdeaProjects\CS266\out\production\CS266
Mode LastWriteTime Length Name
-a--- 2/1/2015 3:30 PM 838 ListArguments.class
PS C:\Users\Teodoro\IdeaProjects\CS266\out\production\CS266> javap
Usage: javap <options> <classes>
where possible options include:
-help --help -? Print this usage message
-version Version information
-v -verbose Print additional information
-l Print line number and local variable tables
-public Show only public classes and members
-protected Show protected/public classes and members
-package Show package/protected/public classes
and members (default)
-p -private Show all classes and members
-c Disassemble the code
-s Print internal type signatures
-sysinfo Show system info (path, size, date, MD5 hash)
of class being processed
-constants Show static final constants
-classpath <path> Specify where to find user class files
-bootclasspath <path> Override location of bootstrap class files
PS C:\Users\Teodoro\IdeaProjects\CS266\out\production\CS266> javap -c -l ListArguments.class
Disassemble the bytecode and print information about local variables.

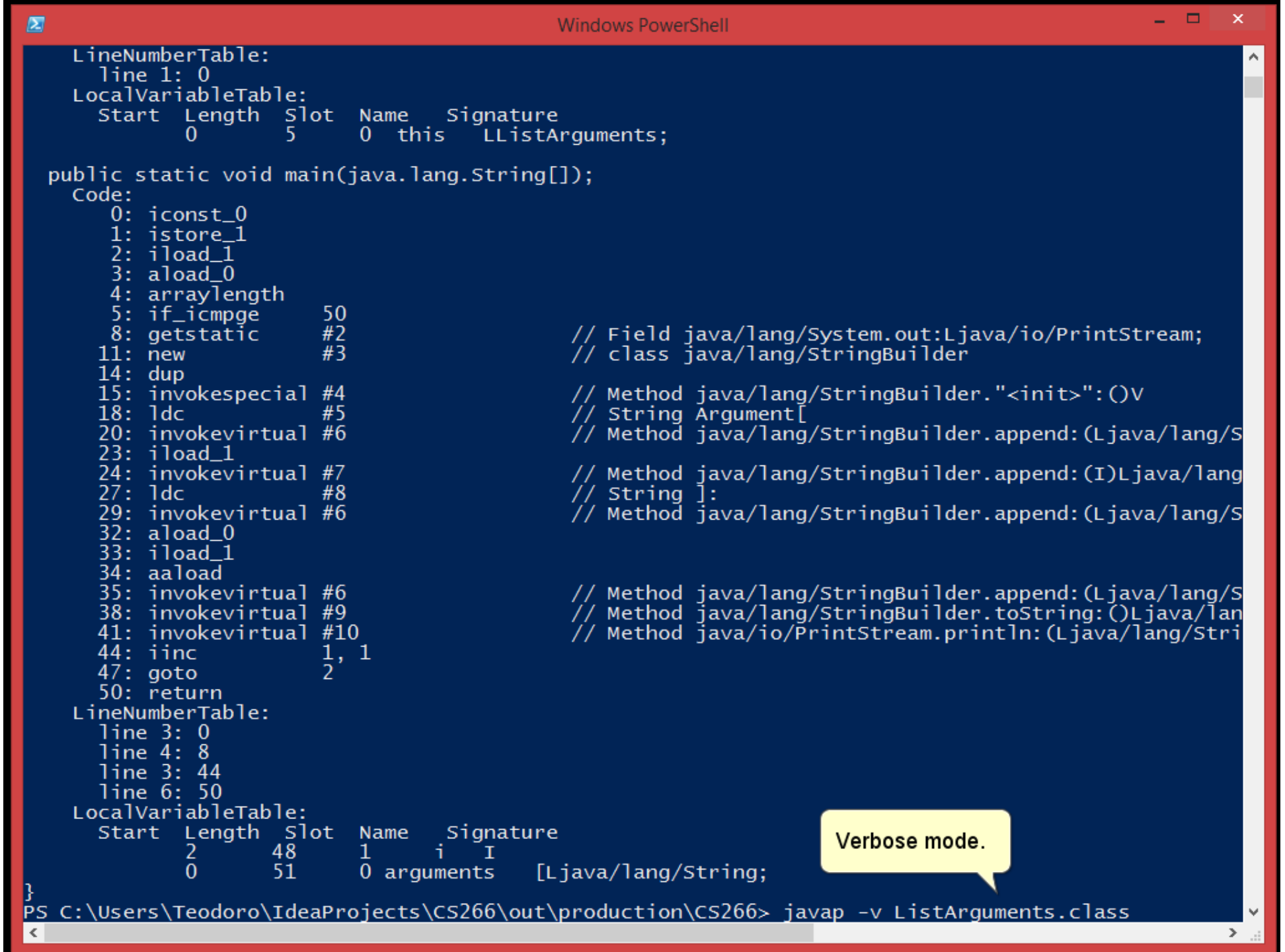
Windows PowerShell

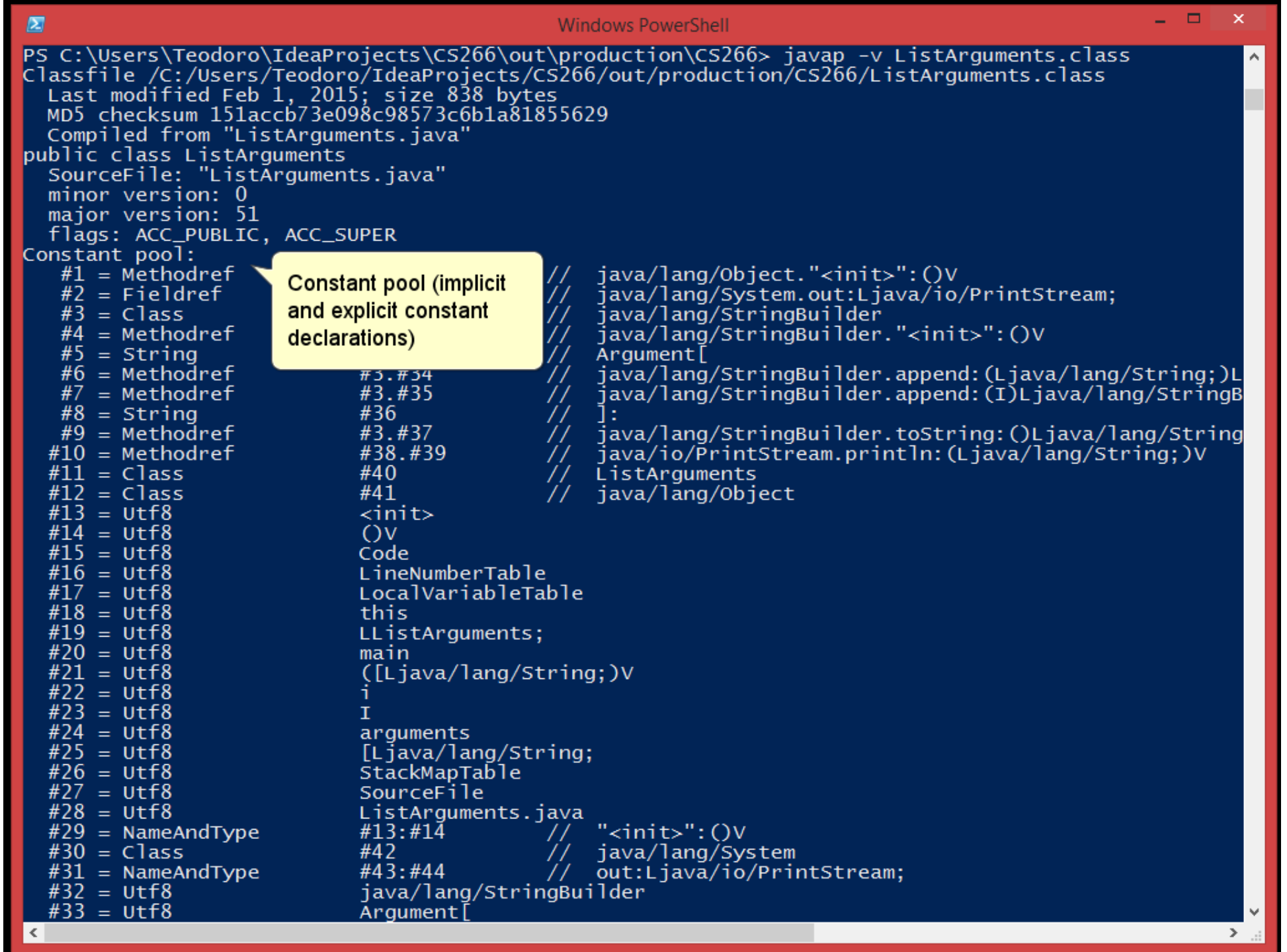
```
PS C:\Users\Teodoro\IdeaProjects\CS266\out\production\CS266> javap -c -l ListArguments.class
Compiled from "ListArguments.java"
public class ListArguments {
  public ListArguments();
    Code:
       0: aload_0
       1: invokespecial #1                  // Method java/lang/Object."<init>":()V
       4: return
    LineNumberTable:
      line 1: 0
    LocalVariableTable:
      Start  Length  Slot  Name   Signature
          0       5     0  this   LListArguments;

  public static void main(java.lang.String[]);
    Code:
       0: iconst_0
       1: istore_1
       2: iload_1
       3: aload_0
       4: arraylength
       5: if_icmpge     50
       8: getstatic     #2                  // Field java/lang/System.out:Ljava/io/PrintStream;
      11: new           #3                  // class java/lang/StringBuilder
      14: dup
      15: invokespecial #4                  // Method java/lang/StringBuilder."<init>":()V
      18: ldc           #5                  // String Argument[
      20: invokevirtual #6                  // Method java/lang/StringBuilder.append:(Ljava/lang/S
      23: iload_1
      24: invokevirtual #7                  // Method java/lang/StringBuilder.append:(I)Ljava/lang
      27: ldc           #8                  // String ]:
      29: invokevirtual #6                  // Method java/lang/StringBuilder.append:(Ljava/lang/S
      32: aload_0
      33: iload_1
      34: aaload
      35: invokevirtual #6                  // Method java/lang/StringBuilder.append:(Ljava/lang/S
      38: invokevirtual #9                  // Method java/lang/StringBuilder.toString:()Ljava/lan
      41: invokevirtual #10                 // Method java/io/PrintStream.println:(Ljava/lang/Stri
      44: iinc          1, 1
      47: goto          2
      50: return
    LineNumberTable:
      line 3: 0
      line 4: 8
```

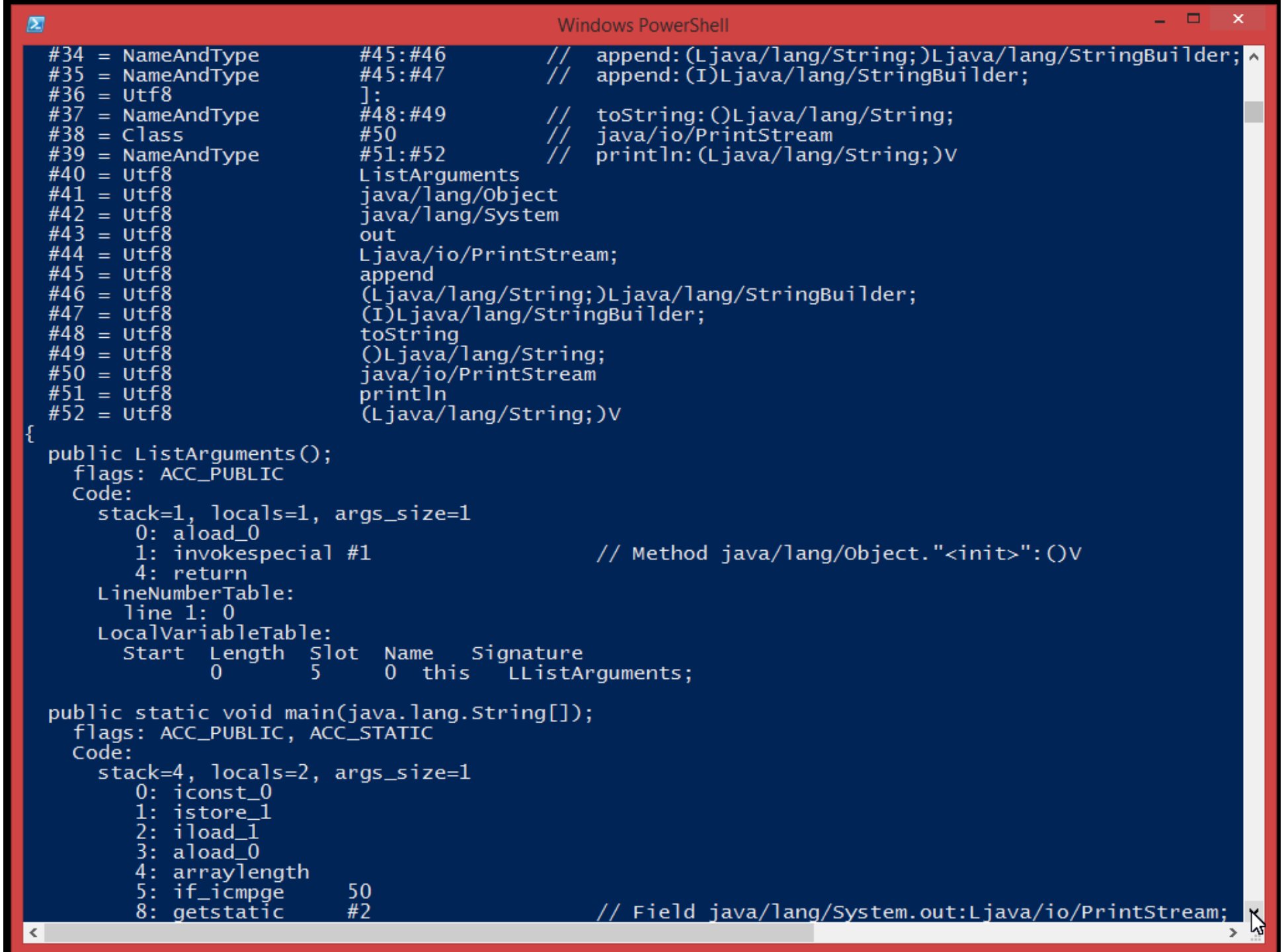
The constructor.

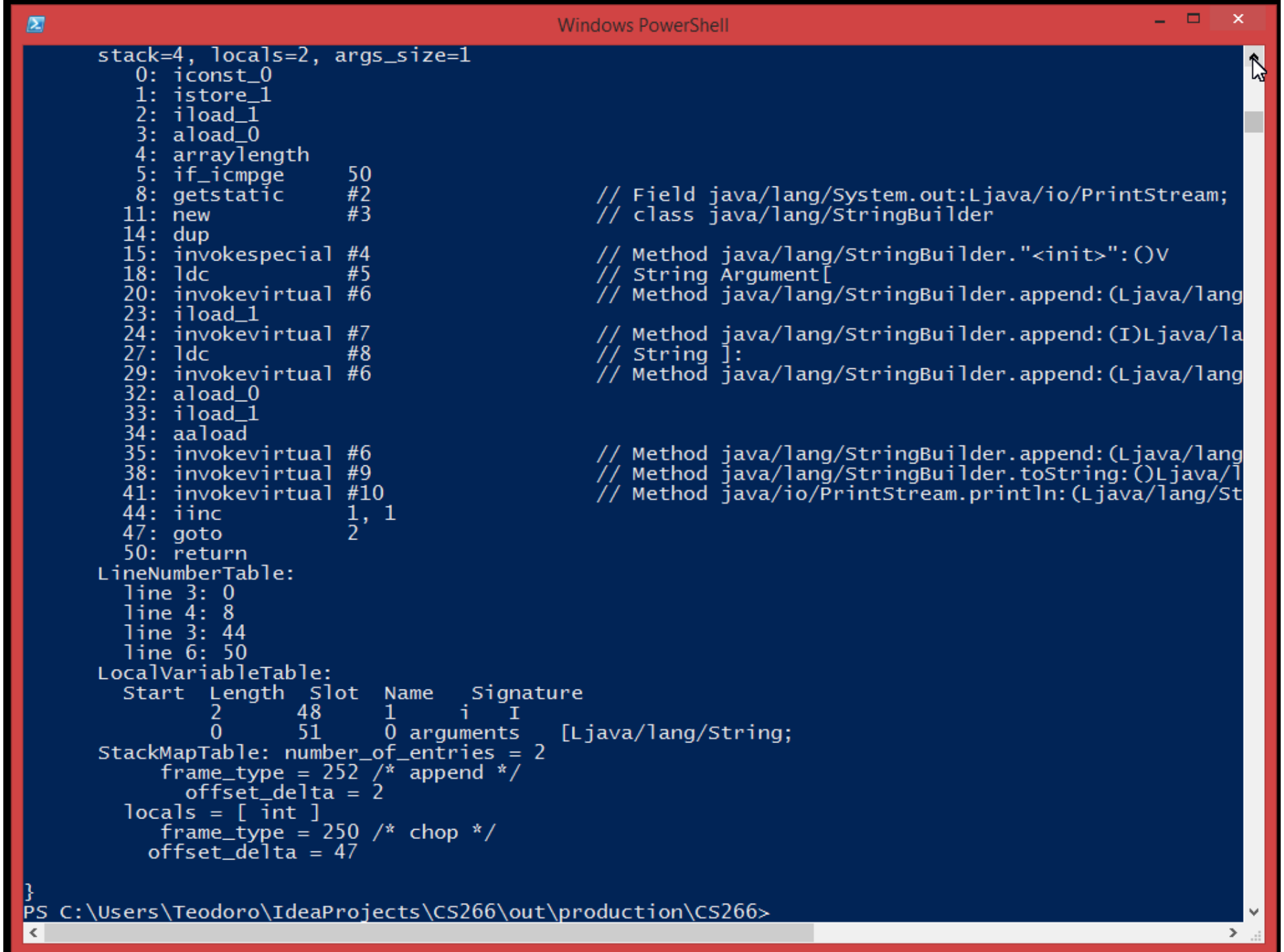
The main method.


Windows PowerShell
LineNumberTable:
line 1: 0
LocalVariableTable:
Start Length Slot Name Signature
0 5 0 this LListArguments;
public static void main(java.lang.String[]);
Code:
0: iconst_0
1: istore_1
2: iload_1
3: aload_0
4: arraylength
5: if_icmpge 50
8: getstatic #2 // Field java/lang/System.out:Ljava/io/PrintStream;
11: new #3 // class java/lang/StringBuilder
14: dup
15: invokespecial #4 // Method java/lang/StringBuilder."<init>":()V
18: ldc #5 // String Argument[
20: invokevirtual #6 // Method java/lang/StringBuilder.append:(Ljava/lang/S
23: iload_1
24: invokevirtual #7 // Method java/lang/StringBuilder.append:(I)Ljava/lang
27: ldc #8 // String]:
29: invokevirtual #6 // Method java/lang/StringBuilder.append:(Ljava/lang/S
32: aload_0
33: iload_1
34: aaload
35: invokevirtual #6 // Method java/lang/StringBuilder.append:(Ljava/lang/S
38: invokevirtual #9 // Method java/lang/StringBuilder.toString:()Ljava/lan
41: invokevirtual #10 // Method java/io/PrintStream.println:(Ljava/lang/Stri
44: iinc 1, 1
47: goto 2
50: return
LineNumberTable:
line 3: 0
line 4: 8
line 3: 44
line 6: 50
LocalVariableTable:
Start Length Slot Name Signature
2 48 1 i I
0 51 0 arguments [Ljava/lang/String;
}
PS C:\Users\Teodoro\IdeaProjects\CS266\out\production\CS266>
List of methods and their signatures.


Windows PowerShell
LineNumberTable:
line 1: 0
LocalVariableTable:
Start Length Slot Name Signature
0 5 0 this LListArguments;
public static void main(java.lang.String[]);
Code:
0: iconst_0
1: istore_1
2: iload_1
3: aload_0
4: arraylength
5: if_icmpge 50
8: getstatic #2 // Field java/lang/System.out:Ljava/io/PrintStream;
11: new #3 // class java/lang/StringBuilder
14: dup
15: invokespecial #4 // Method java/lang/StringBuilder."<init>":()V
18: ldc #5 // String Argument[
20: invokevirtual #6 // Method java/lang/StringBuilder.append:(Ljava/lang/S
23: iload_1
24: invokevirtual #7 // Method java/lang/StringBuilder.append:(I)Ljava/lang
27: ldc #8 // String]:
29: invokevirtual #6 // Method java/lang/StringBuilder.append:(Ljava/lang/S
32: aload_0
33: iload_1
34: aaload
35: invokevirtual #6 // Method java/lang/StringBuilder.append:(Ljava/lang/S
38: invokevirtual #9 // Method java/lang/StringBuilder.toString:()Ljava/lan
41: invokevirtual #10 // Method java/io/PrintStream.println:(Ljava/lang/Stri
44: iinc 1, 1
47: goto 2
50: return
LineNumberTable:
line 3: 0
line 4: 8
line 3: 44
line 6: 50
LocalVariableTable:
Start Length Slot Name Signature
2 48 1 i I
0 51 0 arguments [Ljava/lang/String;
}
PS C:\Users\Teodoro\IdeaProjects\CS266\out\production\CS266> javap -v ListArguments.class
Verbose mode.


Windows PowerShell
PS C:\Users\Teodoro\IdeaProjects\CS266\out\production\CS266> javap -v ListArguments.class
Classfile /C:/Users/Teodoro/IdeaProjects/CS266/out/production/CS266/ListArguments.class
Last modified Feb 1, 2015; size 838 bytes
MD5 checksum 151accb73e098c98573c6b1a81855629
Compiled from "ListArguments.java"
public class ListArguments
SourceFile: "ListArguments.java"
minor version: 0
major version: 51
flags: ACC_PUBLIC, ACC_SUPER
Constant pool:
Constant pool (implicit and explicit constant declarations)
#1 = Methodref // java/lang/Object."<init>":()V
#2 = Fieldref // java/lang/System.out:Ljava/io/PrintStream;
#3 = Class // java/lang/StringBuilder
#4 = Methodref // java/lang/StringBuilder."<init>":()V
#5 = String // Argument[
#6 = Methodref #3.#34 // java/lang/StringBuilder.append:(Ljava/lang/String;)L
#7 = Methodref #3.#35 // java/lang/StringBuilder.append:(I)Ljava/lang/StringB
#8 = String #36 //]:
#9 = Methodref #3.#37 // java/lang/StringBuilder.toString:()Ljava/lang/String
#10 = Methodref #38.#39 // java/io/PrintStream.println:(Ljava/lang/String;)V
#11 = Class #40 // ListArguments
#12 = Class #41 // java/lang/Object
#13 = Utf8 <init>
#14 = Utf8 ()V
#15 = Utf8 Code
#16 = Utf8 LineNumberTable
#17 = Utf8 LocalVariableTable
#18 = Utf8 this
#19 = Utf8 LListArguments;
#20 = Utf8 main
#21 = Utf8 ([Ljava/lang/String;)V
#22 = Utf8 i
#23 = Utf8 I
#24 = Utf8 arguments
#25 = Utf8 [Ljava/lang/String;
#26 = Utf8 StackMapTable
#27 = Utf8 SourceFile
#28 = Utf8 ListArguments.java
#29 = NameAndType #13:#14 // "<init>":()V
#30 = Class #42 // java/lang/System
#31 = NameAndType #43:#44 // out:Ljava/io/PrintStream;
#32 = Utf8 java/lang/StringBuilder
#33 = Utf8 Argument[


Windows PowerShell
#34 = NameAndType #45:#46 // append:(Ljava/lang/String;)Ljava/lang/StringBuilder;
#35 = NameAndType #45:#47 // append:(I)Ljava/lang/StringBuilder;
#36 = Utf8]:
#37 = NameAndType #48:#49 // toString:()Ljava/lang/String;
#38 = Class #50 // java/io/PrintStream
#39 = NameAndType #51:#52 // println:(Ljava/lang/String;)V
#40 = Utf8 ListArguments
#41 = Utf8 java/lang/Object
#42 = Utf8 java/lang/System
#43 = Utf8 out
#44 = Utf8 Ljava/io/PrintStream;
#45 = Utf8 append
#46 = Utf8 (Ljava/lang/String;)Ljava/lang/StringBuilder;
#47 = Utf8 (I)Ljava/lang/StringBuilder;
#48 = Utf8 toString
#49 = Utf8 ()Ljava/lang/String;
#50 = Utf8 java/io/PrintStream
#51 = Utf8 println
#52 = Utf8 (Ljava/lang/String;)V
{
public ListArguments();
flags: ACC_PUBLIC
Code:
stack=1, locals=1, args_size=1
0: aload_0
1: invokespecial #1 // Method java/lang/Object."<init>":()V
4: return
LineNumberTable:
line 1: 0
LocalVariableTable:
Start Length Slot Name Signature
0 5 0 this LListArguments;
public static void main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=4, locals=2, args_size=1
0: iconst_0
1: istore_1
2: iload_1
3: aload_0
4: arraylength
5: if_icmpge 50
8: getstatic #2 // Field java/lang/System.out:Ljava/io/PrintStream;


Windows PowerShell
stack=4, locals=2, args_size=1
0: iconst_0
1: istore_1
2: iload_1
3: aload_0
4: arraylength
5: if_icmpge 50
8: getstatic #2 // Field java/lang/System.out:Ljava/io/PrintStream;
11: new #3 // class java/lang/StringBuilder
14: dup
15: invokespecial #4 // Method java/lang/StringBuilder."<init>":()V
18: ldc #5 // String Argument[
20: invokevirtual #6 // Method java/lang/StringBuilder.append:(Ljava/lang
23: iload_1
24: invokevirtual #7 // Method java/lang/StringBuilder.append:(I)Ljava/la
27: ldc #8 // String]:
29: invokevirtual #6 // Method java/lang/StringBuilder.append:(Ljava/lang
32: aload_0
33: iload_1
34: aaload
35: invokevirtual #6 // Method java/lang/StringBuilder.append:(Ljava/lang
38: invokevirtual #9 // Method java/lang/StringBuilder.toString:()Ljava/l
41: invokevirtual #10 // Method java/io/PrintStream.println:(Ljava/lang/St
44: iinc 1, 1
47: goto 2
50: return
LineNumberTable:
line 3: 0
line 4: 8
line 3: 44
line 6: 50
LocalVariableTable:
Start Length Slot Name Signature
2 48 1 i I
0 51 0 arguments [Ljava/lang/String;
StackMapTable: number_of_entries = 2
frame_type = 252 /* append */
offset_delta = 2
locals = [int]
frame_type = 250 /* chop */
offset_delta = 47
}
PS C:\Users\Teodoro\IdeaProjects\CS266\out\production\CS266>

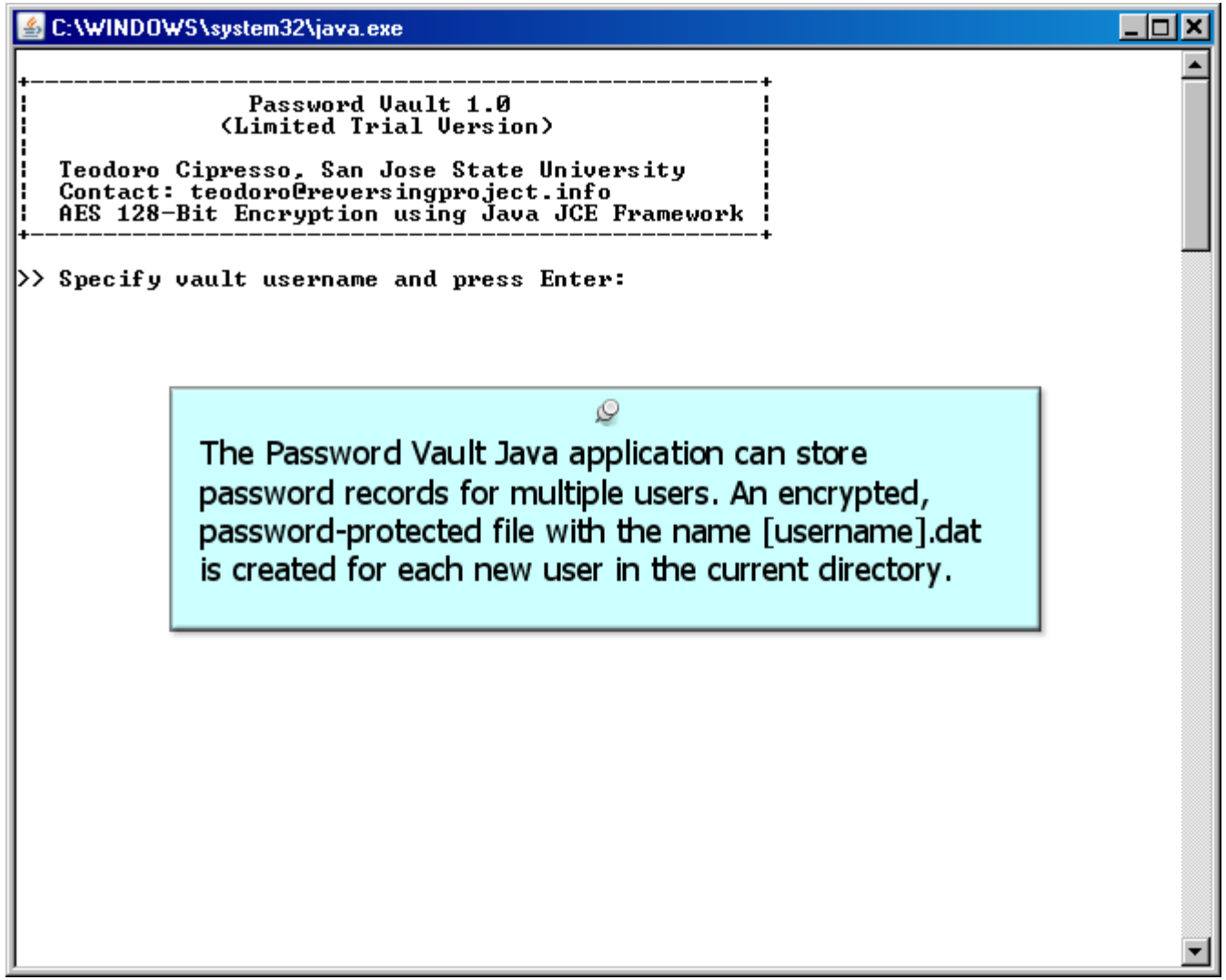
Reversing and Patching Java Bytecode

Decompiling and Disassembling Java Bytecode

- The result of disassembling Java bytecode is a pseudo-assembly language, a language that cannot be compiled or assembled but serves to provide a more abstract, readable representation of the bytecode.
- While it may seem possible to use a hex editor directly modify Java bytecode in a **.class* file, this is similar in difficulty to editing machine code in a hex editor.
  - However, libraries exist for deserializing and serializing bytecode using in-memory Java object models. This allows programmatic modification.
- Decompiling then recompiling Java bytecode after making modifications is perhaps the most straightforward way to reverse and patch Java applications.

Reversing and Patching Java Bytecode

Java Bytecode Reversing and Patching Exercise



Reversing and Patching Java Bytecode

Java Bytecode Reversing and Patching Exercise

PasswordVault.jar

PasswordVault.jar

```
C:\WINDOWS\system32\java.exe
(+) Name: record01
    (-) Username : username01
    (-) Password: password01
    (-) Description: description01

(+) Name: record02
    (-) Username : username02
    (-) Password: password02
    (-) Description: description02

(+) Name: record03
    (-) Username : username03
    (-) Password: password03
    (-) Description: description03

(+) Name: record04
    (-) Username : username04
    (-) Password: password04
    (-) Description: description04

(+) Name: record05
    (-) Username : username05
    (-) Password: password05
    (-) Description: description05

+-------------------------------------------------+
|              Password Vault 1.0                 |
|            (Limited Trial Version)              |
|                                                 |
|  Teodoro Cipresso, San Jose State University    |
|  Contact: teodoro@reversingproject.info         |
|  AES 128-Bit Encryption using Java JCE Framework|
+-------------------------------------------------+

(1) Display Password Records
(2) Create a Password Record
(3) Edit a Password Record
(4) Delete a Password Record
(5) Change the Vault Password
(6) Save Records and Quit

>> Specify an option number and press Enter: 2
```

Specify option (2) to attempt to create a sixth password record.

Reversing and Patching Java Bytecode

Java Bytecode Reversing and Patching Exercise

PasswordVault.jar

PasswordVault.jar

```
C:\WINDOWS\system32\java.exe
    (-) Username : username05
    (-) Password: password05
    (-) Description: description05

+-------------------------------------------------+
|              Password Vault 1.0                 |
|            (Limited Trial Version)              |
|                                                 |
|  Teodoro Cipresso, San Jose State University    |
|  Contact: teodoro@reversingproject.info         |
|  AES 128-Bit Encryption using Java JCE Framework|
+-------------------------------------------------+

(1) Display Password Records
(2) Create a Password Record
(3) Edit a Password Record
(4) Delete a Password Record
(5) Change the Vault Password
(6) Save Records and Quit

>> Specify an option number and press Enter: 2

[Error] Thank you for trying Password Vault! You have reached the maximum number
 of records allowed in this trial version.

+-------------------------------------------------+
|              Password Vault 1.0                 |
|            (Limited Trial Version)              |
|                                                 |
|  Teodoro Cipresso, San Jose State University    |
|  Contact: teodoro@reversingproject.info         |
|  AES 128-Bit Encryption using Java JCE Framework|
+-------------------------------------------------+

(1) Display Password Records
(2) Create a Password Record
(3) Edit a Password Record
(4) Delete a Password Record
(5) Change the Vault Password
(6) Save Records and Quit

>> Specify an option number and press Enter: _
```

A message is displayed indicating that the trial limitation has been reached.

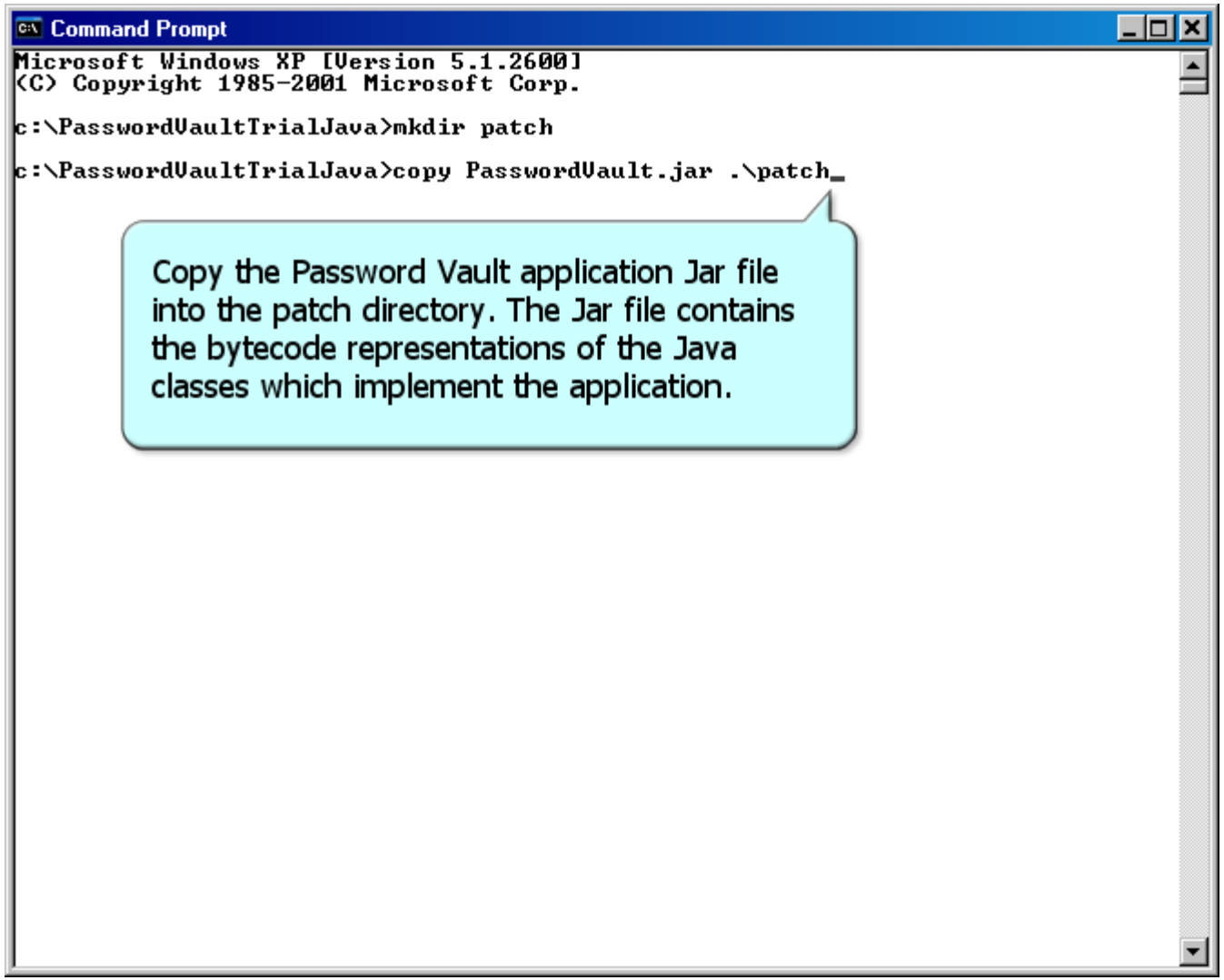
Make note of this note of this error message as it's an artifact related to the program's check for whether the trial limitation has been reached.

Reversing and Patching Java Bytecode

Java Bytecode Reversing and Patching Exercise

PasswordVault.jar

PasswordVault.jar

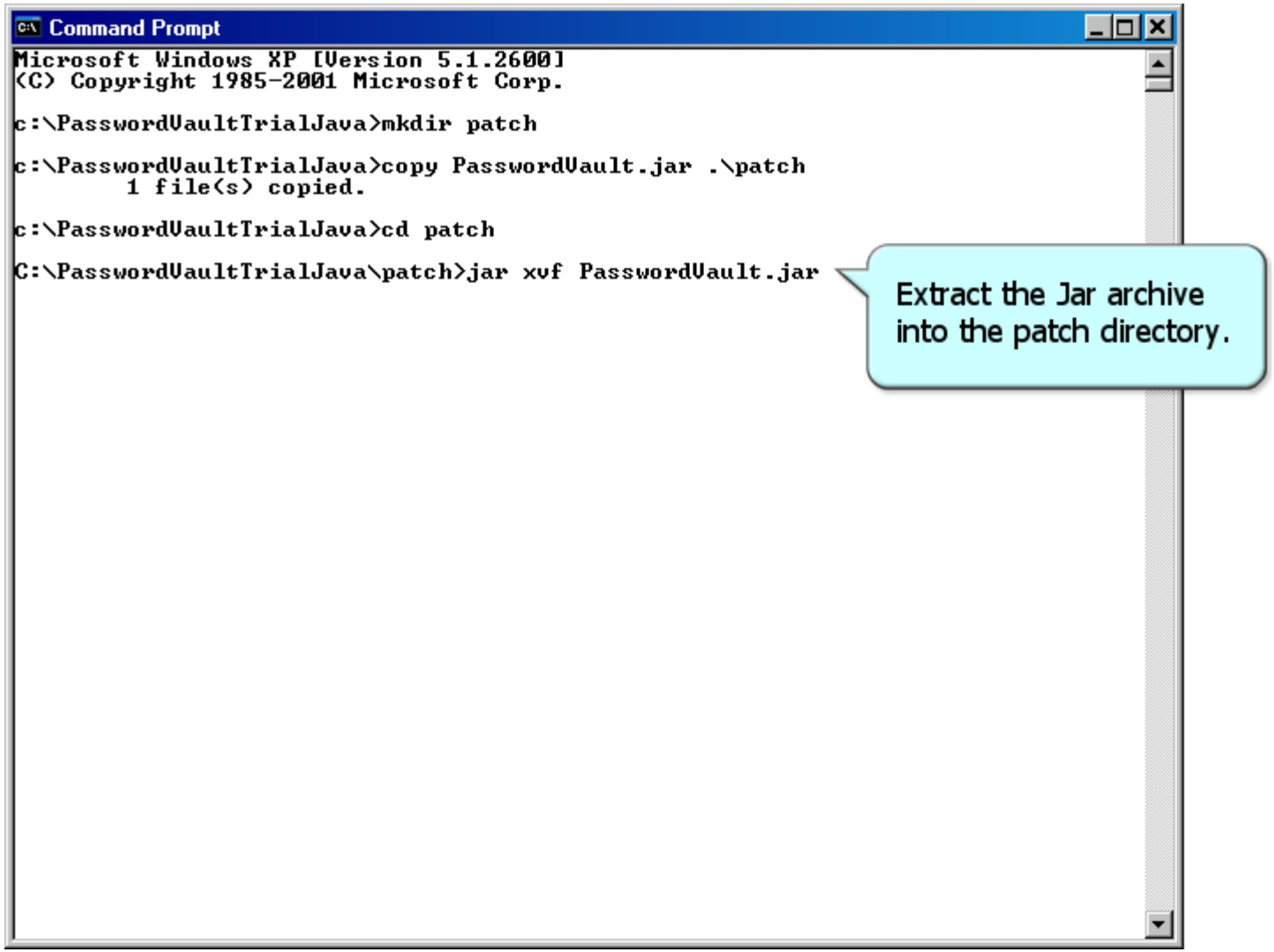


Reversing and Patching Java Bytecode

Java Bytecode Reversing and Patching Exercise

PasswordVault.jar

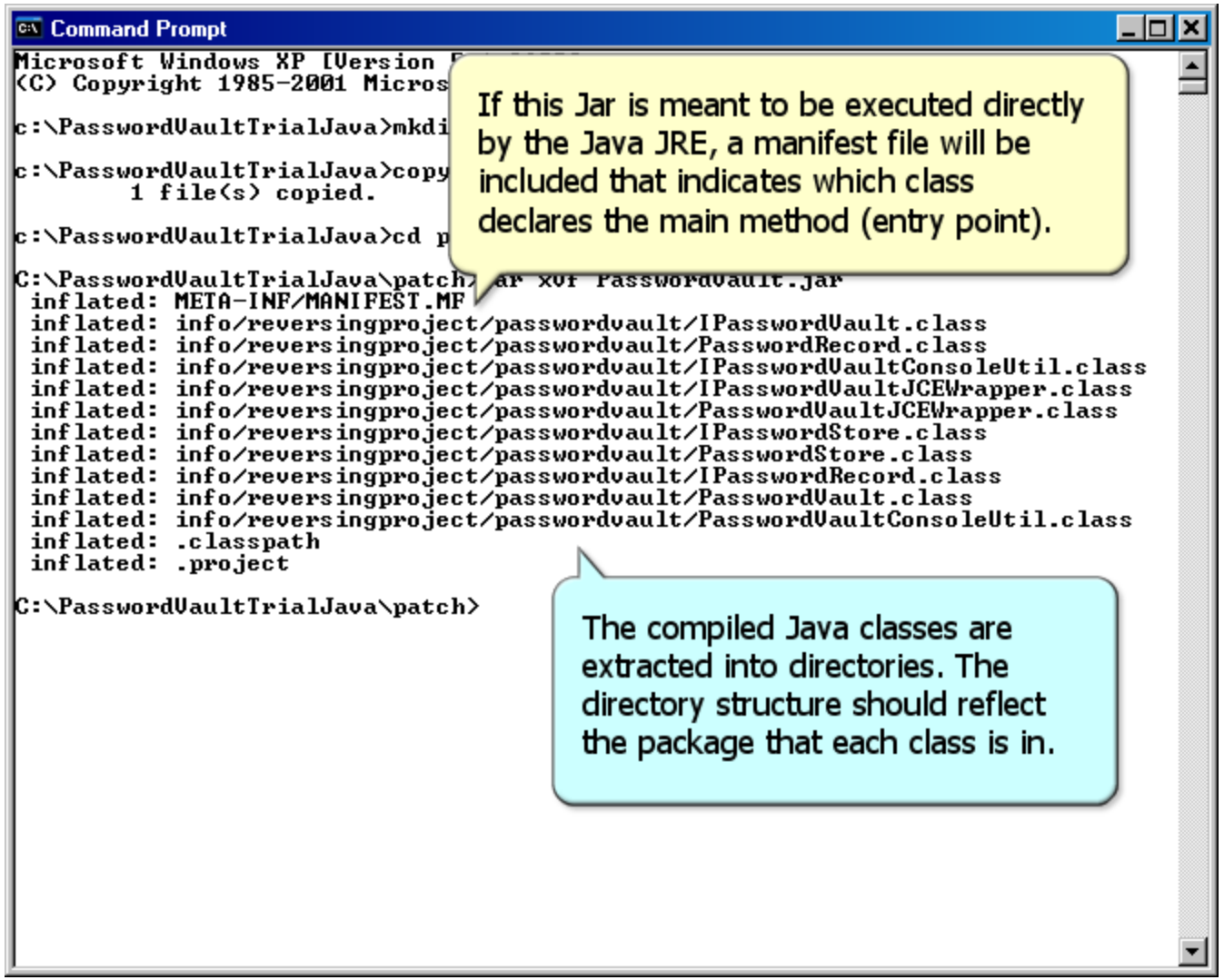
PasswordVault.jar



Reversing and Patching Java Bytecode

Java Bytecode Reversing and Patching Exercise

PasswordVault.jar

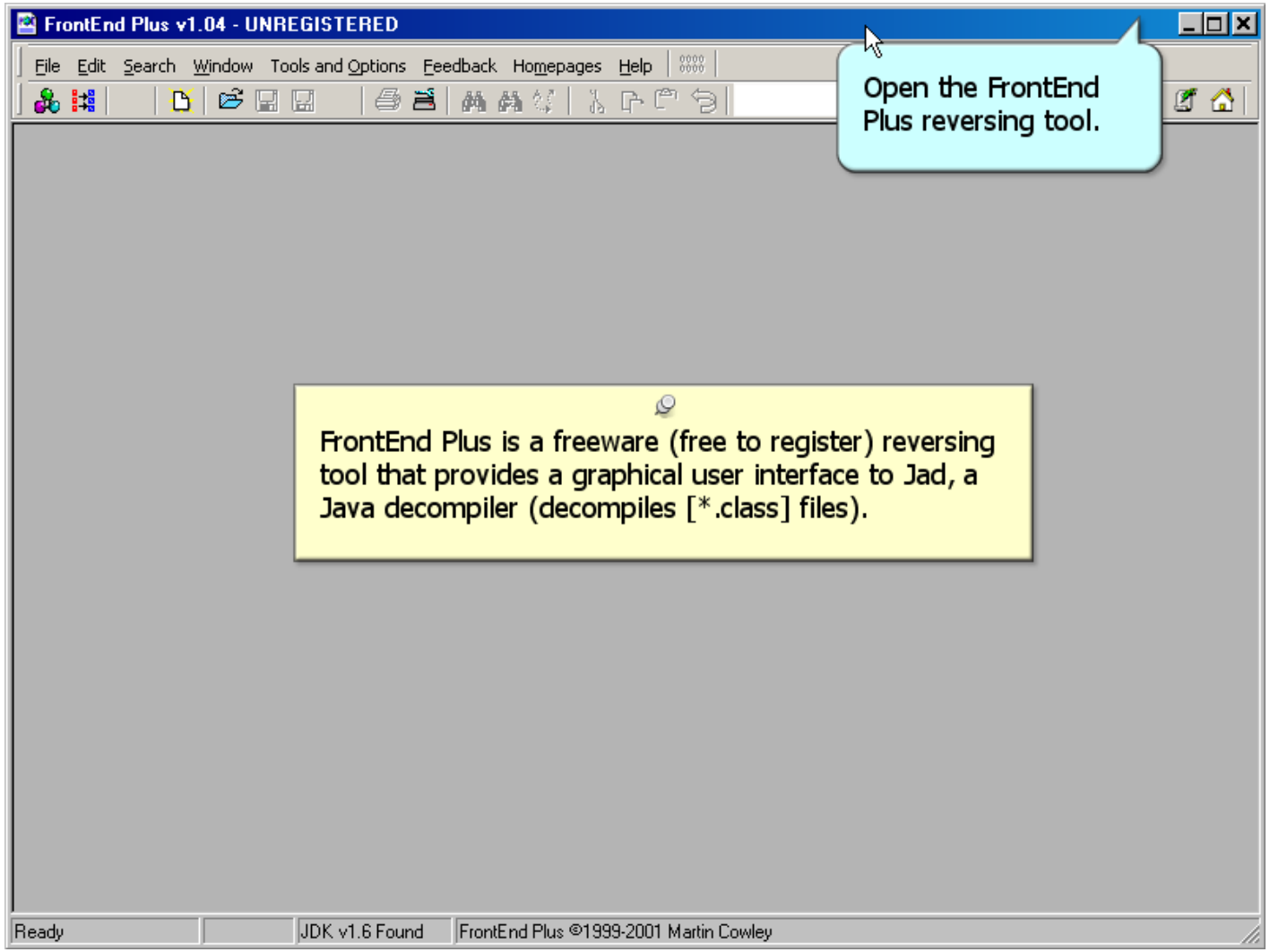


Reversing and Patching Java Bytecode

Java Bytecode Reversing and Patching Exercise

FrontEndPlusV1andV2.zip

FrontEndPlusV1andV2.zip

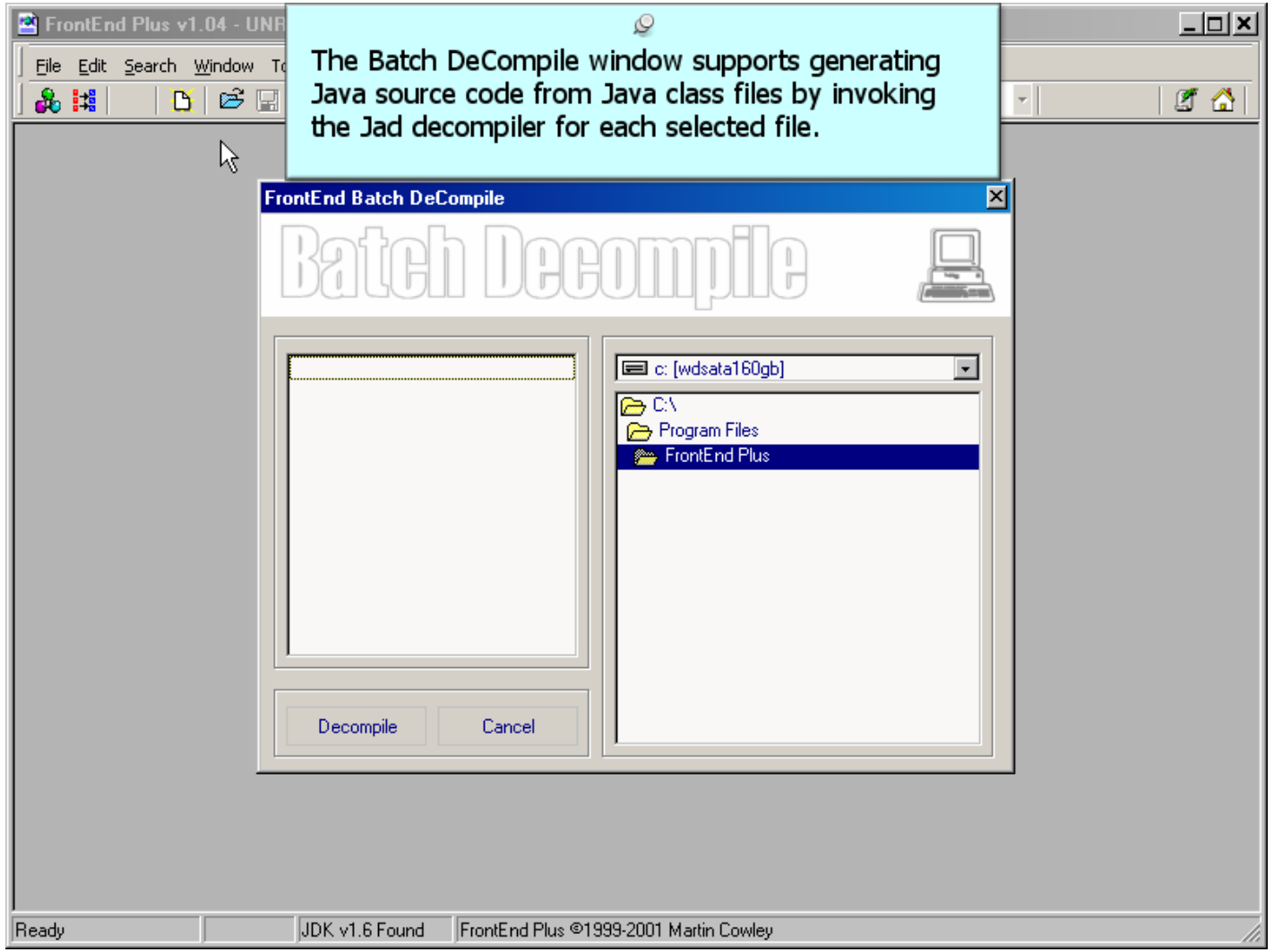


Reversing and Patching Java Bytecode

Java Bytecode Reversing and Patching Exercise

FrontEndPlusV1andV2.zip

FrontEndPlusV1andV2.zip

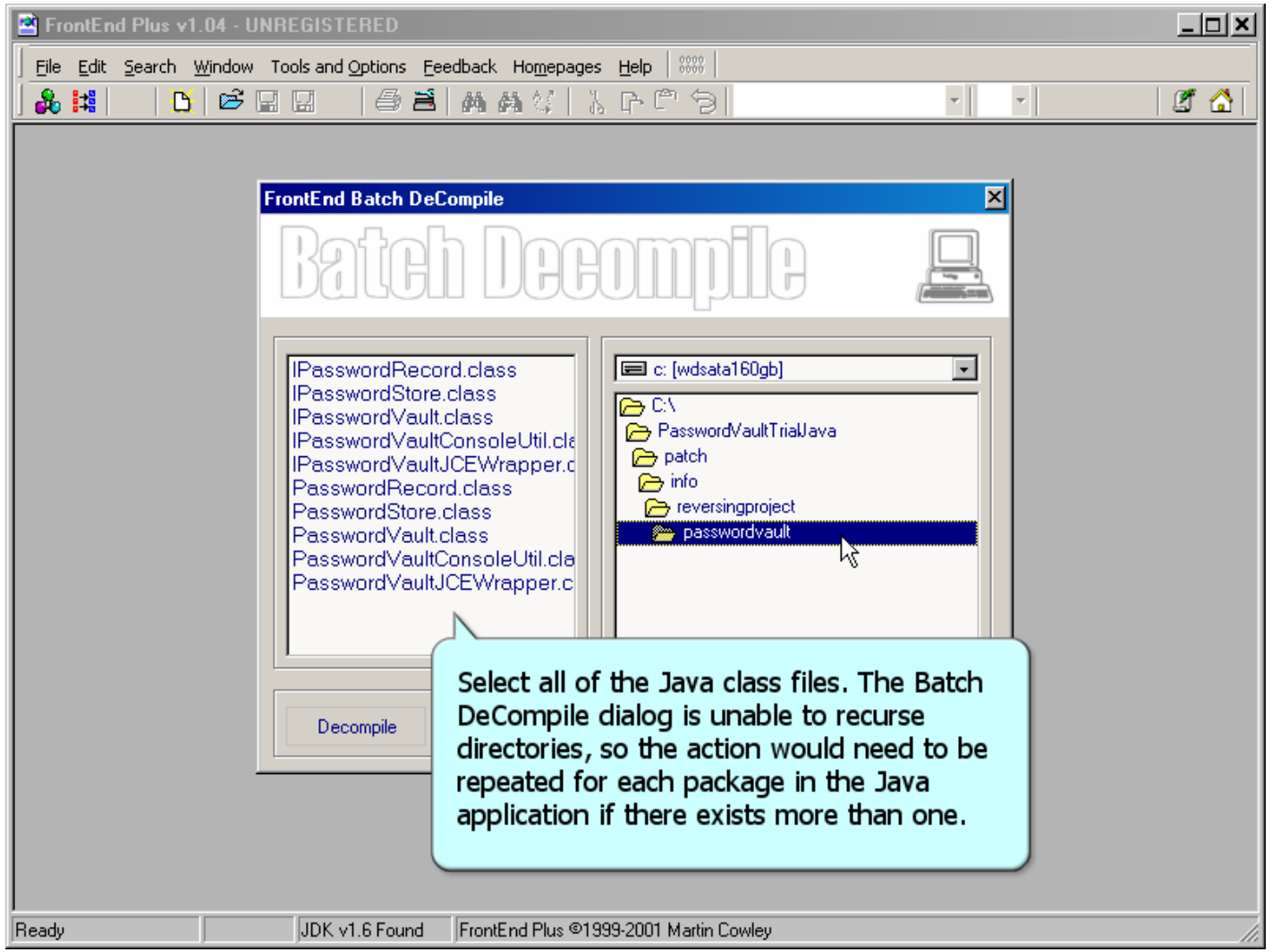


Reversing and Patching Java Bytecode

Java Bytecode Reversing and Patching Exercise

FrontEndPlusV1andV2.zip

FrontEndPlusV1andV2.zip

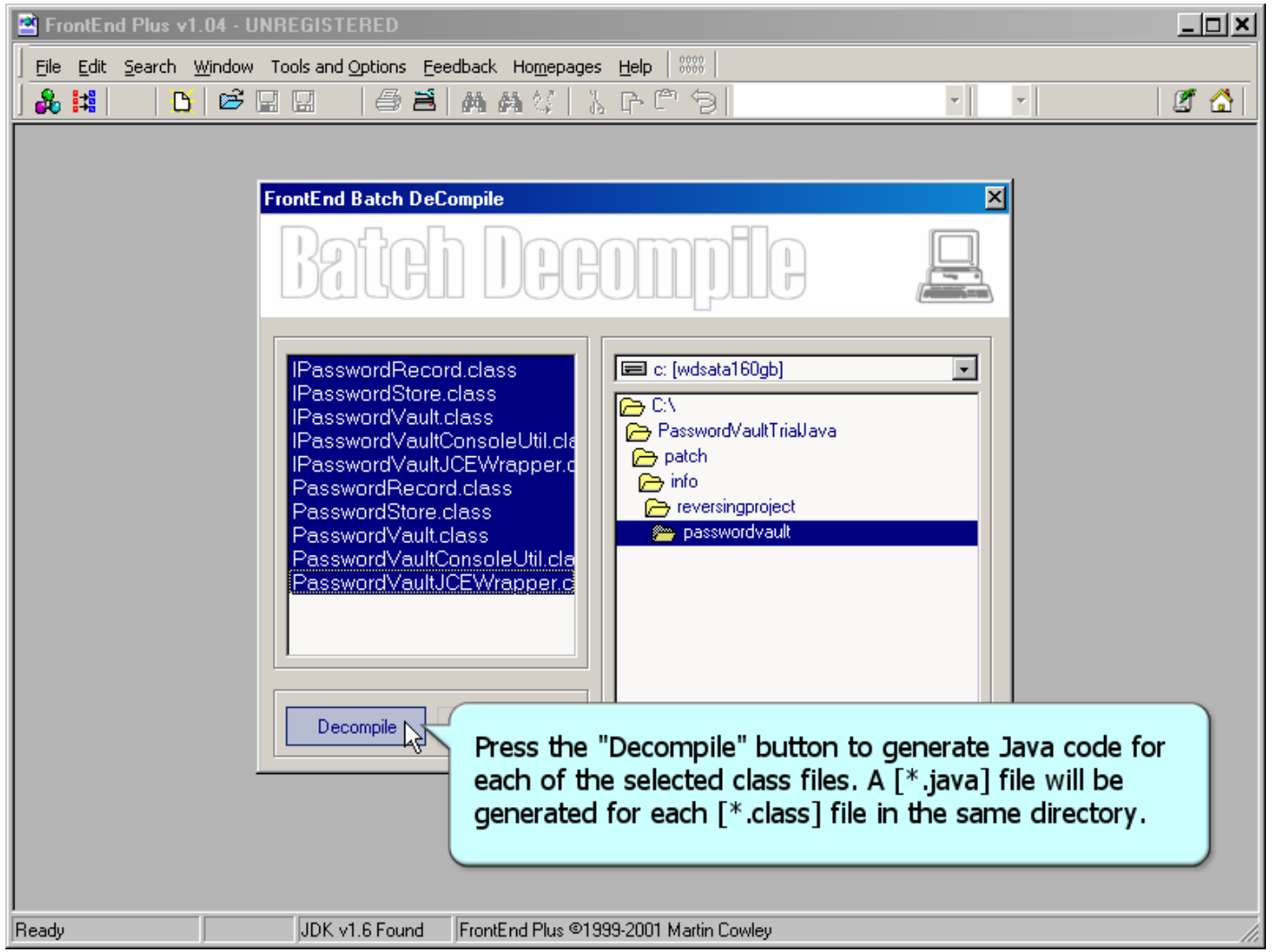


Reversing and Patching Java Bytecode

Java Bytecode Reversing and Patching Exercise

FrontEndPlusV1andV2.zip

FrontEndPlusV1andV2.zip

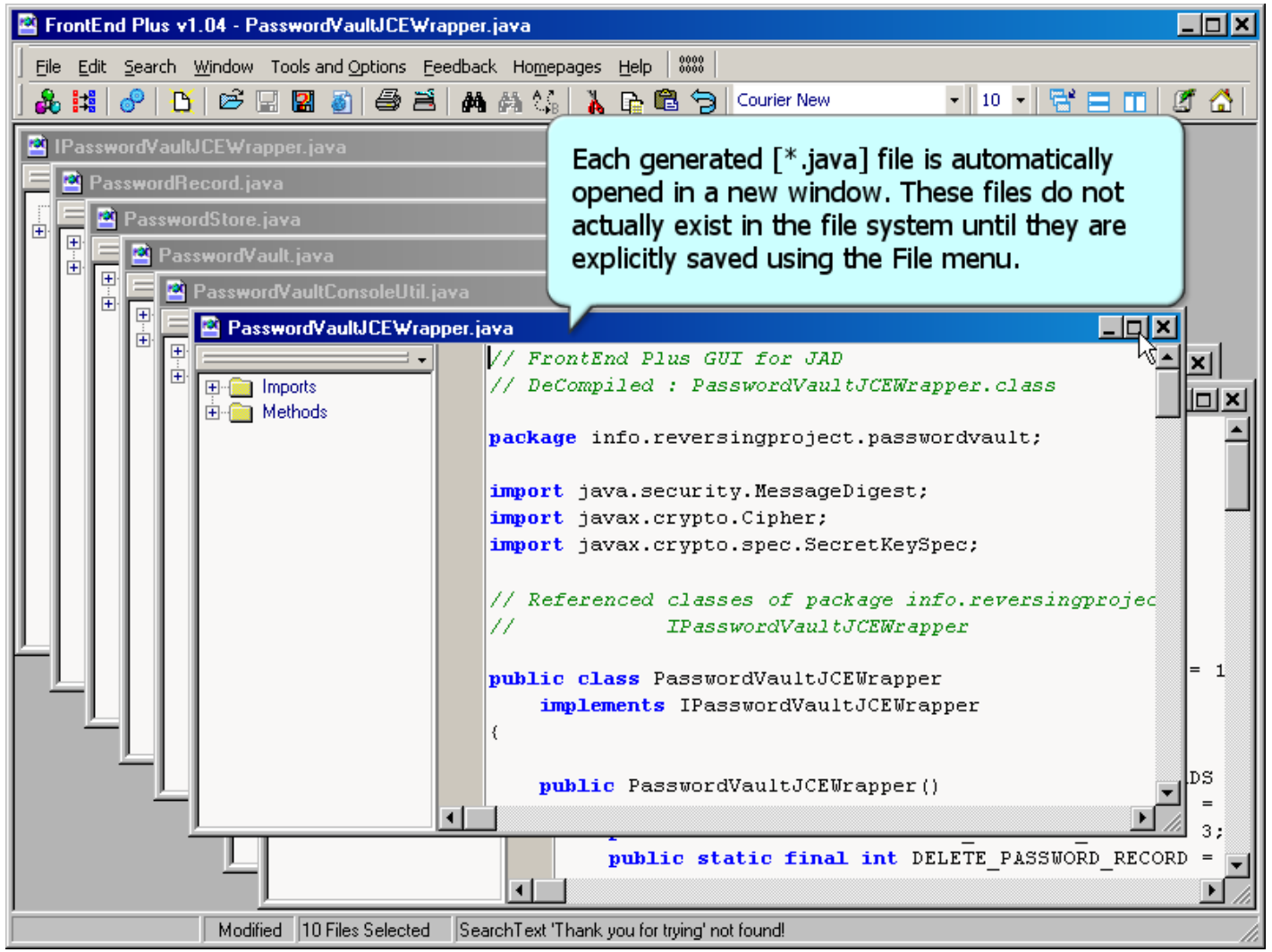


Reversing and Patching Java Bytecode

Java Bytecode Reversing and Patching Exercise

FrontEndPlusV1andV2.zip

FrontEndPlusV1andV2.zip

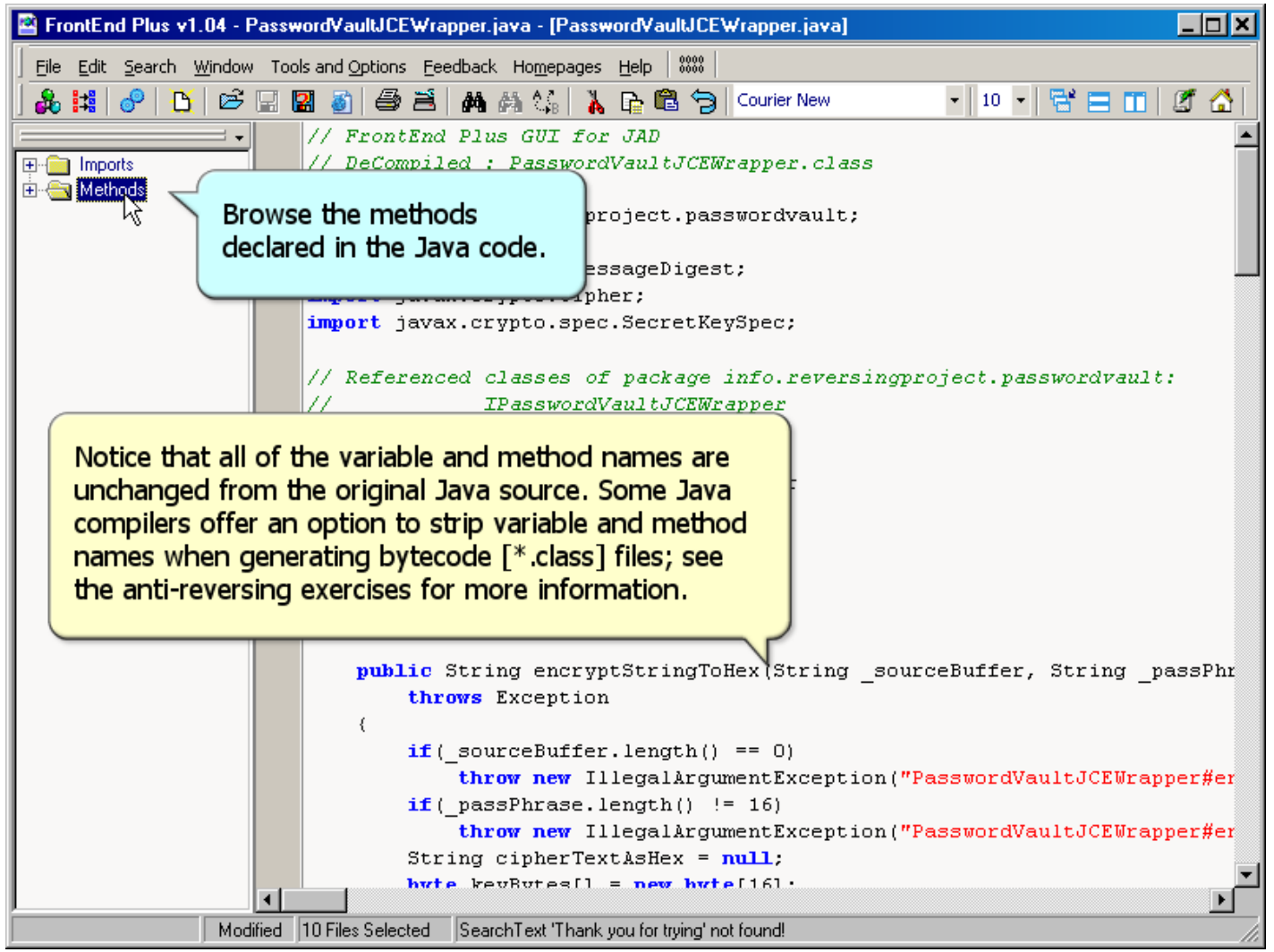


Reversing and Patching Java Bytecode

Java Bytecode Reversing and Patching Exercise

FrontEndPlusV1andV2.zip

FrontEndPlusV1andV2.zip

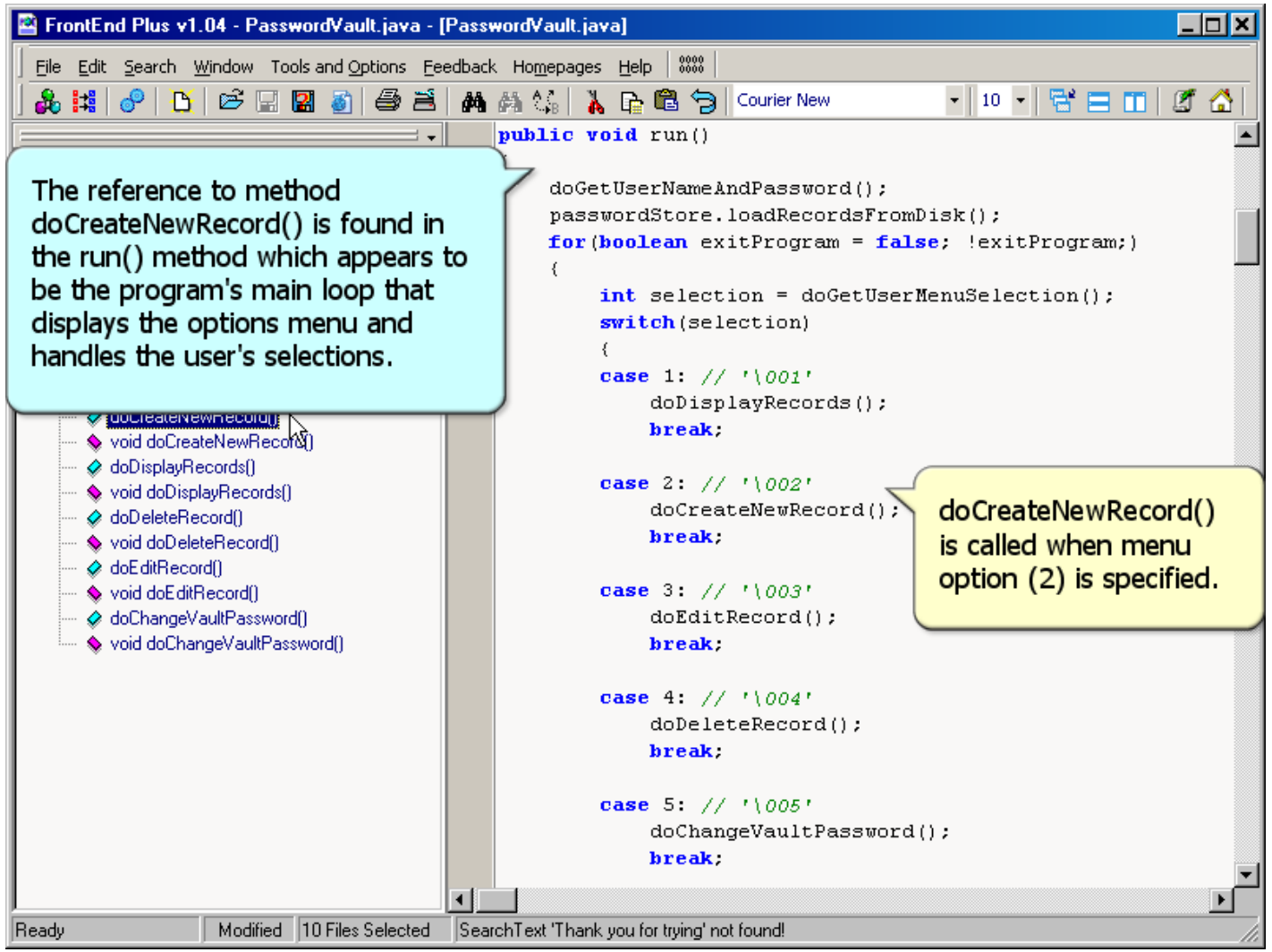


Reversing and Patching Java Bytecode

Java Bytecode Reversing and Patching Exercise

FrontEndPlusV1andV2.zip

FrontEndPlusV1andV2.zip

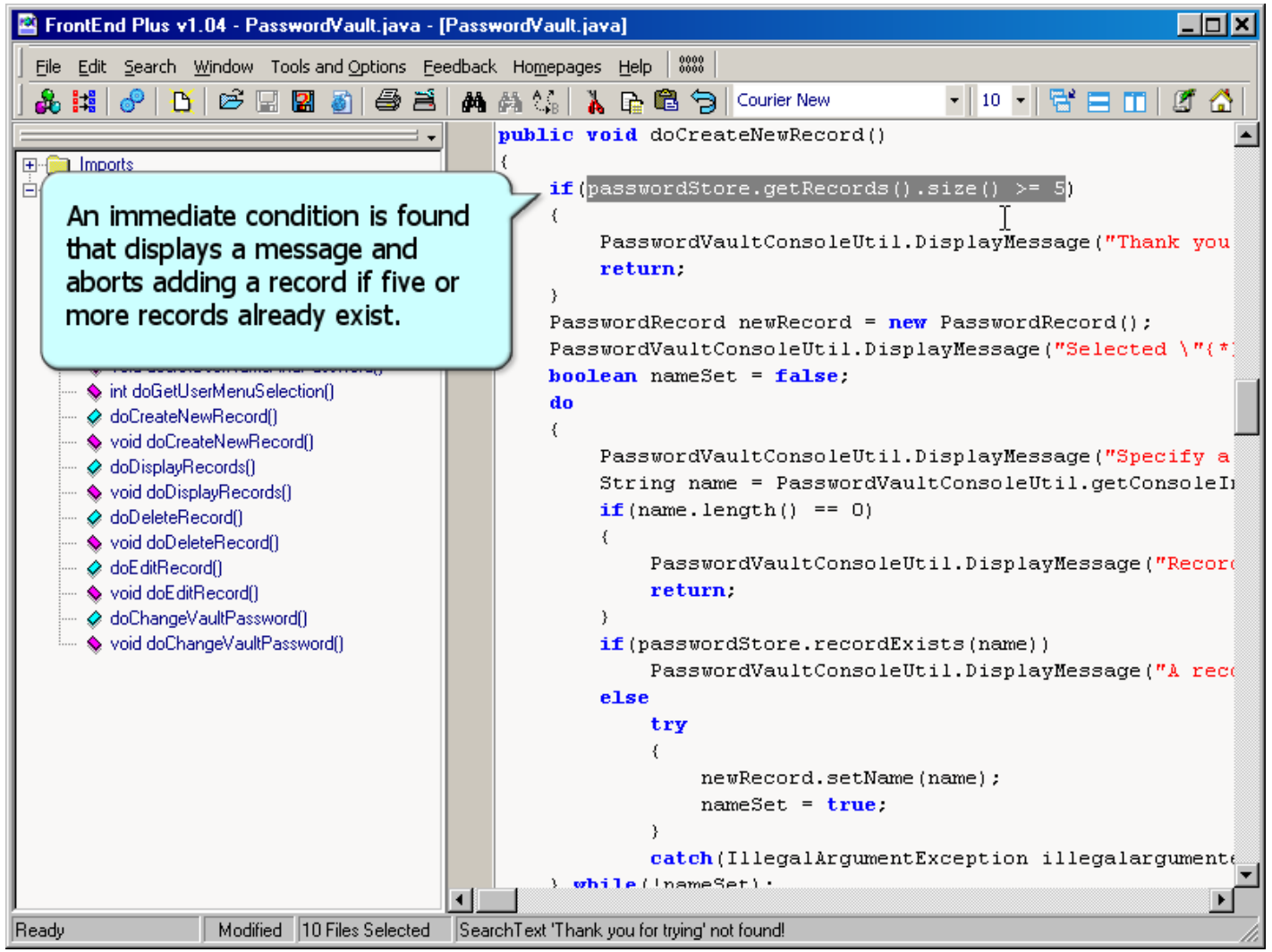


Reversing and Patching Java Bytecode

Java Bytecode Reversing and Patching Exercise

FrontEndPlusV1andV2.zip

FrontEndPlusV1andV2.zip

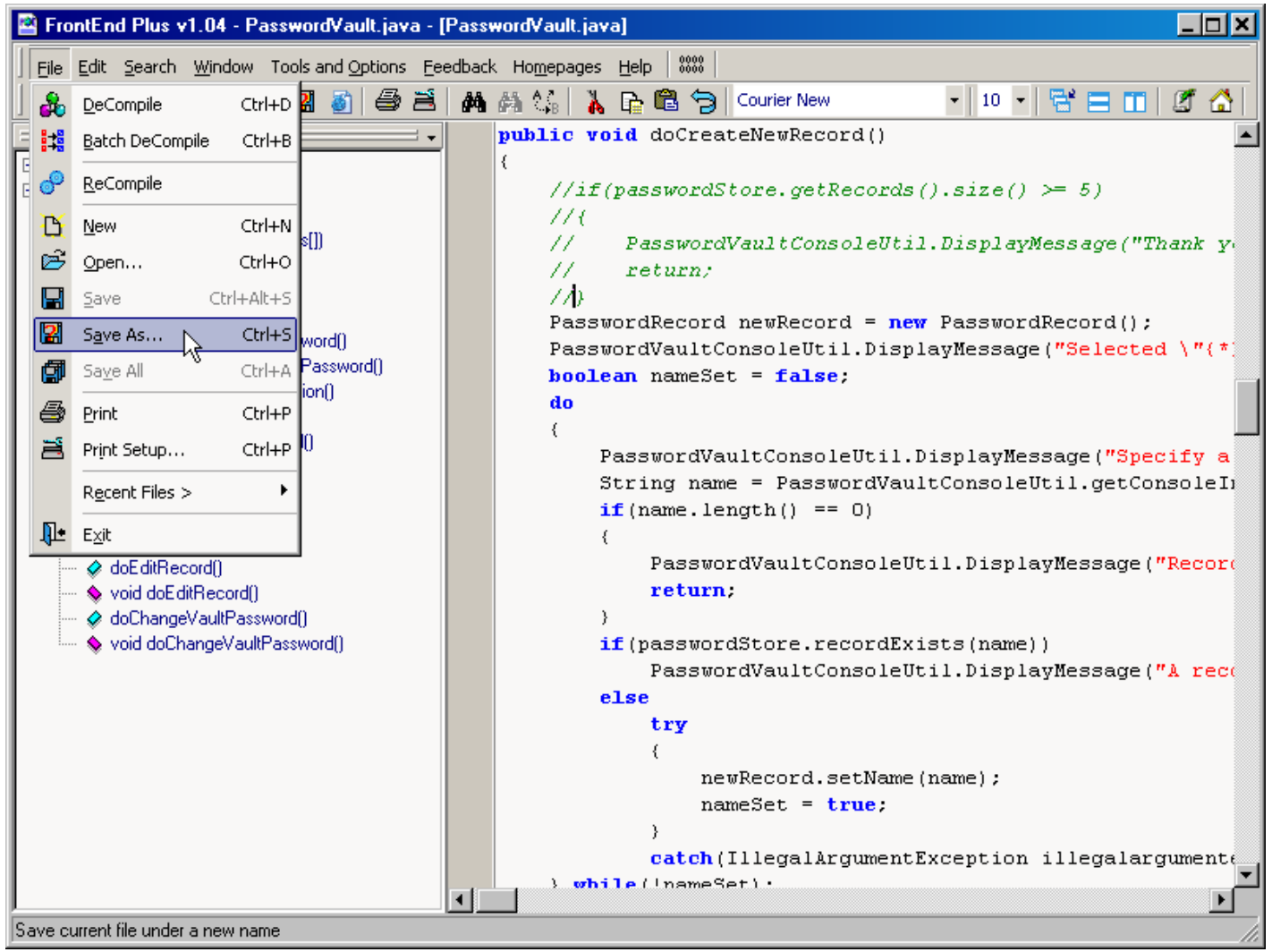


Reversing and Patching Java Bytecode

Java Bytecode Reversing and Patching Exercise

FrontEndPlusV1andV2.zip

FrontEndPlusV1andV2.zip



Reversing and Patching Java Bytecode

Java Bytecode Reversing and Patching Exercise

PasswordVault.jar

PasswordVault.jar

```
Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

c:\PasswordVaultTrialJava>mkdir patch

c:\PasswordVaultTrialJava>copy PasswordVault.jar .\patch
        1 file(s) copied.

c:\PasswordVaultTrialJava>cd patch

C:\PasswordVaultTrialJava\patch>jar xvf PasswordVault.jar
 inflated: META-INF/MANIFEST.MF
 inflated: info/reversingproject/passwordvault/IPasswordVault.class
 inflated: info/reversingproject/passwordvault/PasswordRecord.class
 inflated: info/reversingproject/passwordvault/IPasswordVaultConsoleUtil.class
 inflated: info/reversingproject/passwordvault/IPasswordVaultJCEWrapper.class
 inflated: info/reversingproject/passwordvault/PasswordVaultJCEWrapper.class
 inflated: info/reversingproject/passwordvault/IPasswordStore.class
 inflated: info/reversingproject/passwordvault/PasswordStore.class
 inflated: info/reversingproject/passwordvault/IPasswordRecord.class
 inflated: info/reversingproject/passwordvault/PasswordVault.class
 inflated: info/reversingproject/passwordvault/PasswordVaultConsoleUtil.class
 inflated: .classpath
 inflated: .project

C:\PasswordVaultTrialJava\patch>javac info\reversingproject\passwordvault\Passwo
rdVault.java_
```

Compile the modified, generated Java source for the PasswordVault.class file. This step will overwrite the original, extracted class file.

Reversing and Patching Java Bytecode

Java Bytecode Reversing and Patching Exercise

PasswordVault.jar

PasswordVault.jar

```
Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

c:\PasswordVaultTrialJava>mkdir patch

c:\PasswordVaultTrialJava>copy PasswordVault.jar .\patch
        1 file(s) copied.

c:\PasswordVaultTrialJava>cd patch

C:\PasswordVaultTrialJava\patch>jar xvf PasswordVault.jar
 inflated: META-INF/MANIFEST.MF
 inflated: info/reversingproject/passwordvault/IPasswordVault.class
 inflated: info/reversingproject/passwordvault/PasswordRecord.class
 inflated: info/reversingproject/passwordvault/IPasswordVaultConsoleUtil.class
 inflated: info/reversingproject/passwordvault/IPasswordVaultJCEWrapper.class
 inflated: info/reversingproject/passwordvault/PasswordVaultJCEWrapper.class
 inflated: info/reversingproject/passwordvault/IPasswordStore.class
 inflated: info/reversingproject/passwordvault/PasswordStore.class
 inflated: info/reversingproject/passwordvault/IPasswordRecord.class
 inflated: info/reversingproject/passwordvault/PasswordVault.class
 inflated: info/reversingproject/passwordvault/PasswordVaultConsoleUtil.class
 inflated: .classpath
 inflated: .project

C:\PasswordVaultTrialJava\patch>javac info\reversingproject\passwordvault\Passwo
rdVault.java

C:\PasswordVaultTrialJava\patch>jar uvf PasswordVault.jar info\reversingproject\
passwordvault_
```

Update the copy of the Jar file from the extracted class files--including the recreated PasswordVault.class file.

Reversing and Patching Java Bytecode

Java Bytecode Reversing and Patching Exercise

PasswordVault.jar

PasswordVault.jar

```
Command Prompt
 inflated: info/reversingproject/passwordvault/PasswordRecord.class
 inflated: info/reversingproject/passwordvault/IPasswordVaultConsoleUtil.class
 inflated: info/reversingproject/passwordvault/IPasswordVaultJCEWrapper.class
 inflated: info/reversingproject/passwordvault/PasswordVaultJCEWrapper.class
 inflated: info/reversingproject/passwordvault/IPasswordStore.class
 inflated: info/reversingproject/passwordvault/PasswordStore.class
 inflated: info/reversingproject/passwordvault/IPasswordRecord.class
 inflated: info/reversingproject/passwordvault/PasswordVault.class
 inflated: info/reversingproject/passwordvault/PasswordVaultConsoleUtil.class
 inflated: .classpath
 inflated: .project

C:\PasswordVaultTrialJava\patch>javac info\reversingproject\passwordvault\Passwo
rdVault.java

C:\PasswordVaultTrialJava\patch>jar uvf PasswordVault.jar info\reversingproject\
passwordvault
adding: info/reversingproject/passwordvault/IPasswordVault.class(in = 436) (out=
 293)(deflated 32%)
adding: info/reversingproject/passwordvault/PasswordRecord.class(in = 3239) (out
= 1366)(deflated 57%)
adding: info/reversingproject/passwordvault/IPasswordVaultConsoleUtil.class(in =
 6204) (out= 2405)(deflated 61%)
adding: info/reversingproject/passwordvault/IPasswordVaultJCEWrapper.class(in =
419) (out= 233)(deflated 44%)
adding: info/reversingproject/passwordvault/PasswordVaultJCEWrapper.class(in = 4
191) (out= 2024)(deflated 51%)
adding: info/reversingproject/passwordvault/IPasswordStore.class(in = 941) (out=
 483)(deflated 48%)
adding: info/reversingproject/passwordvault/PasswordStore.class(in = 8470) (out=
 4076)(deflated 51%)
adding: info/reversingproject/passwordvault/IPasswordRecord.class(in = 581) (out
= 336)(deflated 42%)
adding: info/reversingproject/passwordvault/PasswordVault.class(in = 6549) (out=
 3265)(deflated 50%)
adding: info/reversingproject/passwordvault/PasswordVaultConsoleUtil.class(in =
3589) (out= 1673)(deflated 53%)
adding: info/reversingproject/passwordvault/(in = 0) (out= 0)(stored 0%)
adding: info/reversingproject/passwordvault/PasswordVault.java(in = 13207) (out=
 2105)(deflated 84%)

C:\PasswordVaultTrialJava\patch>
```

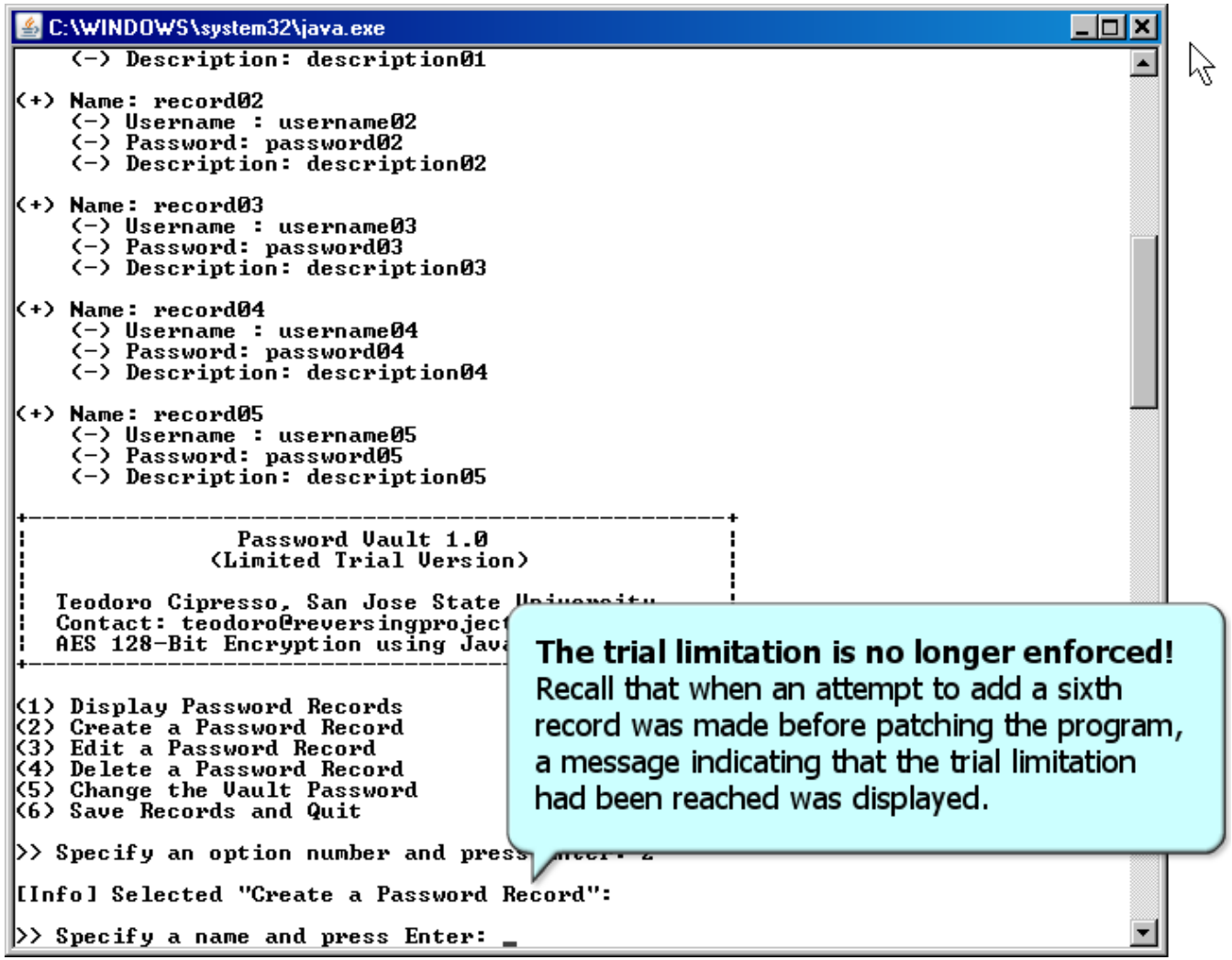
The Jar file is updated with the Java resources located at .\info\reversingproject\passwordvault.

Reversing and Patching Java Bytecode

Java Bytecode Reversing and Patching Exercise

PasswordVault.jar

PasswordVault.jar



The trial limitation is no longer enforced! Recall that when an attempt to add a sixth record was made before patching the program, a message indicating that the trial limitation had been reached was displayed.

Reversing and Patching Java Bytecode

Byte Code Engineering Library (BCEL)

- BCEL provides a convenient way to analyze, create, and manipulate Java Bytecode (Java *.class files).
- Perform static analysis, dynamically create or transform Java bytecode.
- High level of abstraction of the Java class file format relieves the programmer from internal details of the Java class file format.
- BCEL is used in projects such as compilers, optimizers, obfuscators, code generators and analysis tools.

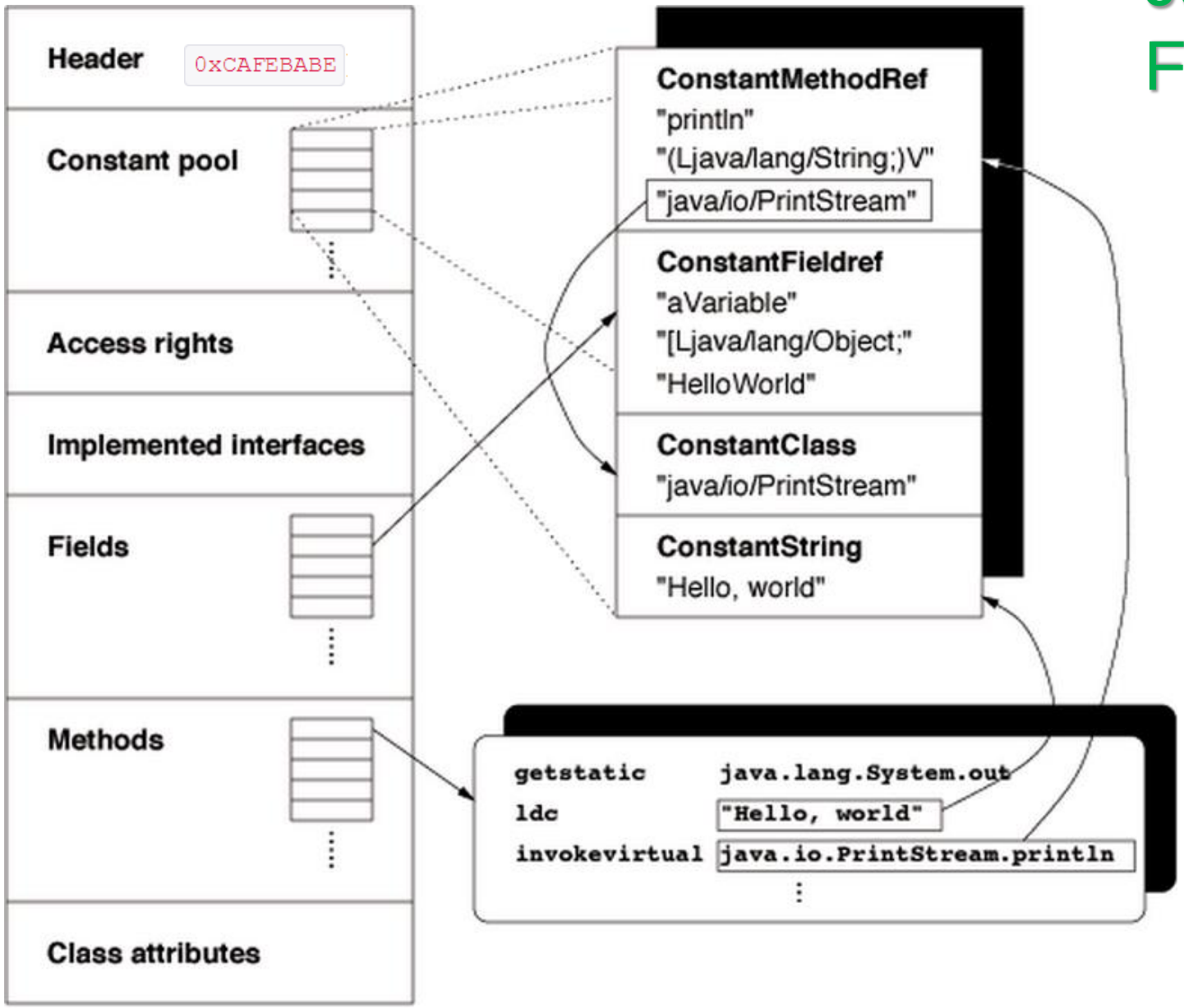
Apache Commons BCEL ™

Reversing and Patching Java Bytecode

Byte Code Engineering Library (BCEL) API

- The BCEL API abstracts the Java Virtual Machine and reading and writing binary Java class files. The API consists mainly of three parts:
  - A package that may be used to read and write class files from or to a file. The main data structure is JavaClass.
    - Usage: analyze Java classes without having the source files at hand.
  - A package to dynamically generate or modify JavaClass or Method objects.
    - Usage: insert analysis code, strip information, implement a code generator back-end of a Java compiler.
  - Various code examples and utilities such as a class file viewer.

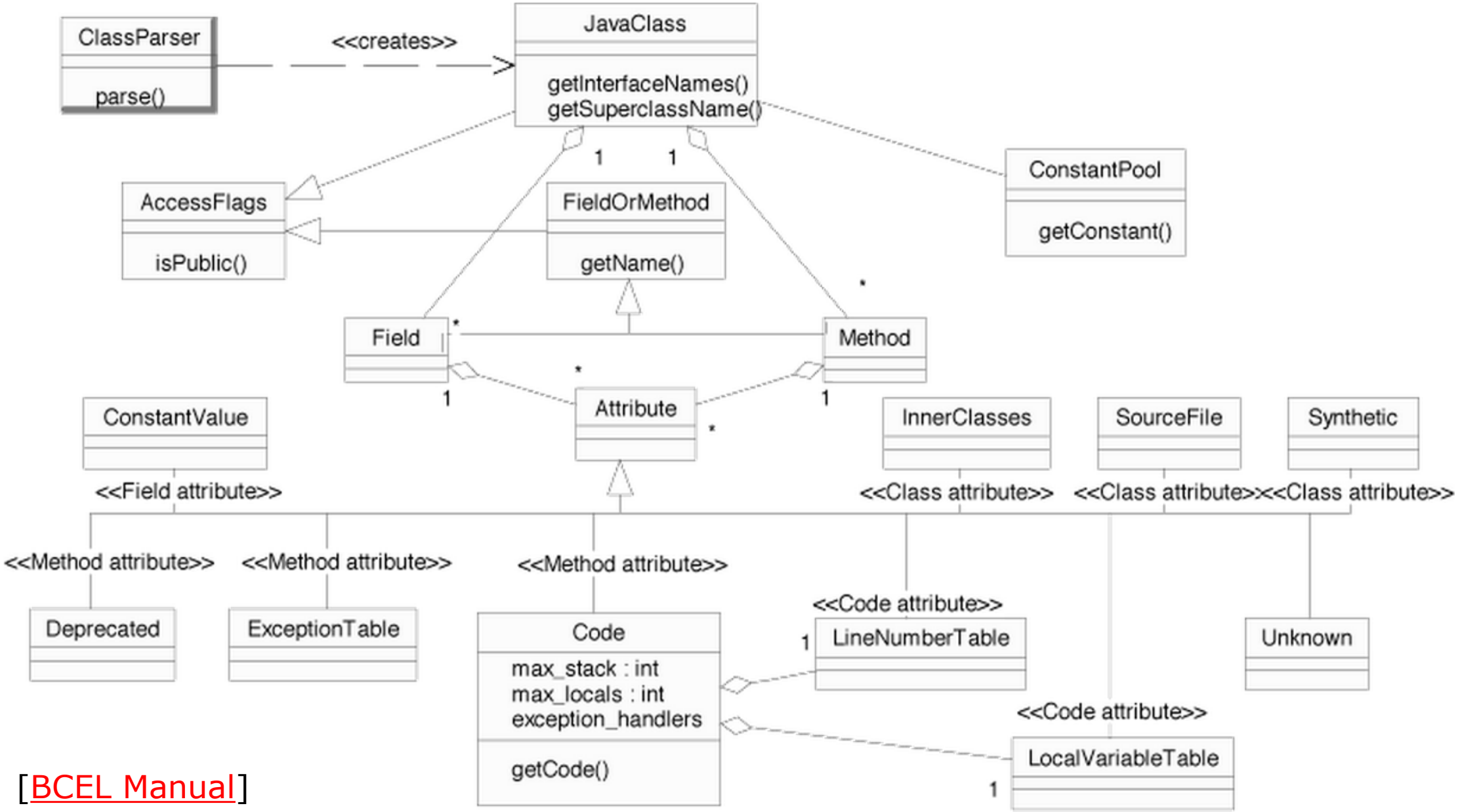
Java Class File Format



HelloWorld.class

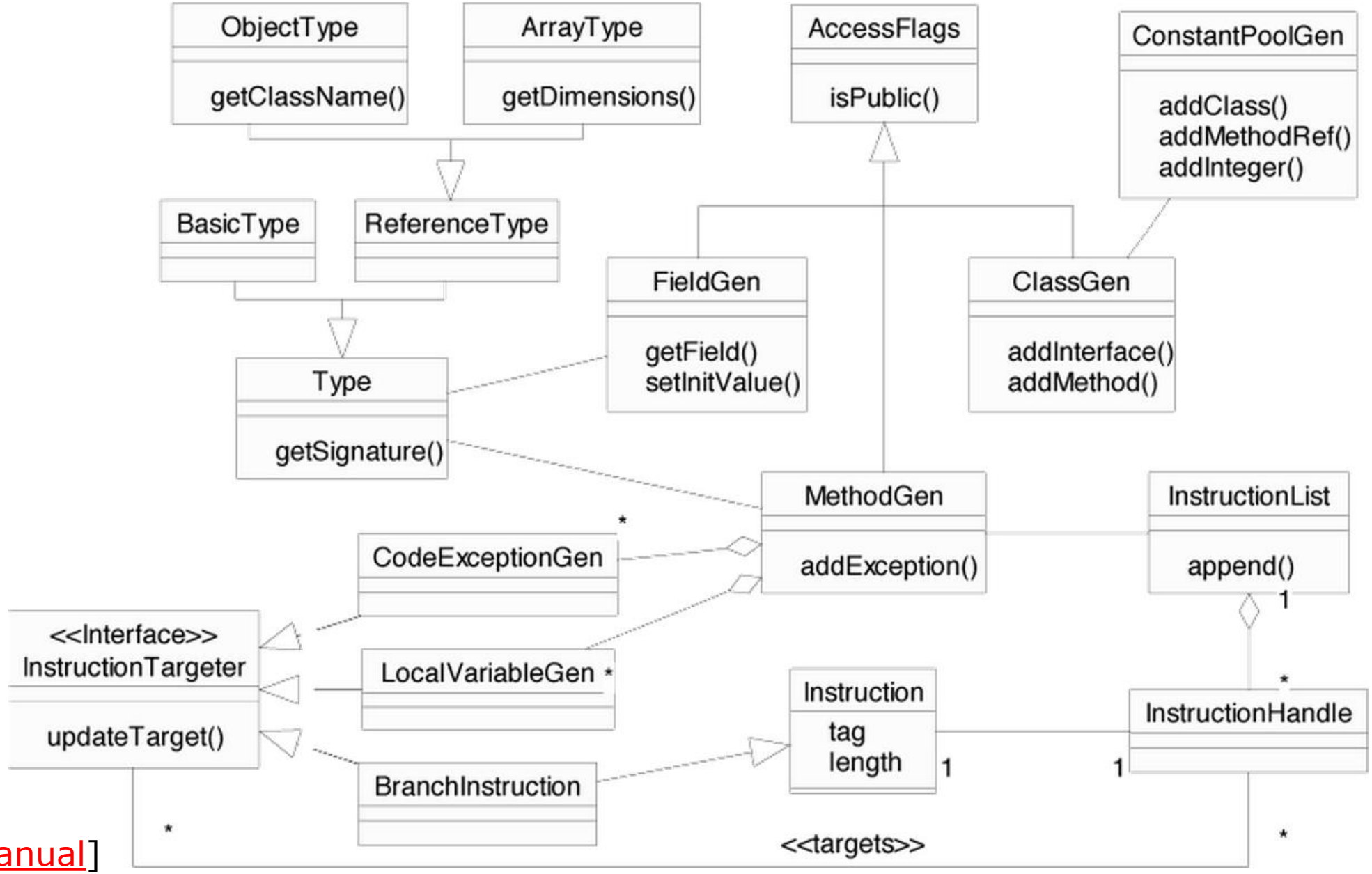
[BCEL Manual]

JavaClass



[BCEL Manual]

ClassGen



[BCEL Manual]

BCEL demo

End