BYTECODE ENGINEERING LIBRARY

Su-Hyun Lee
Changwon National University

Contents

- Byte Code Engineering Library
- View Class Information
- Add a Method
- Change a Bytecode

Part 1

INTRODUCTION

3

What is BCEL?

- Byte Code Engineering Library (Apache Commons BCEL)
  - analyze, create, and manipulate Java class file
  - Class file: methods, fields, bytecode instructions
- Functions
  - view and change bytecode
  - read existing file - transform - written to file
  - creation of classes from scratch at run-time
- Application
  - compiler
  - optimizer
  - obfuscator
  - code analysis tool

4

BCEL Packages

- BCEL vs. Reflection
  - Reflection: java.lang.Class
  - BCEL: org.apache.bcel.classfile.JavaClass
- package org.apache.bcel.classfile
  - examining and viewing class file
- import package

```
import org.apache.bcel.classfile.*;
import org.apache.bcel.*;
```

5

Download & Run

- Apache Commons
  - http://commons.apache.org
- BCEL home
  - http://commons.apache.org/proper/commons-bcel/
  - download bcel-5.2.jar
- Compile & Run
  - `javac -cp bcel-5.2.jar <name>.java`
  - `java -cp .;bcel-5.2.jar <name>`

6

Part 2

GETTING CLASS INFO

7

Sample Java Program

- P.java

```
class P {
   //static void willBeAdded(String str) {
   //   System.out.println(str);
   //}
   void existing(String str) {
      int a = 1;
      System.out.println(str);
      System.out.println(a);
   }
}
```

8

P.class

```
명령 프롬프트
C:₩Users₩slee₩Desktop₩대학원13차세대PL₩tools>javap -c P
Compiled from "P.java"
class P {
  P();
    Code:
       0: aload_0
       1: invokespecial #1                  // Method java/lang/Object."<init>":
<>V
       4: return

  void existing(java.lang.String);
    Code:
       0: iconst_1
       1: istore_2
       2: getstatic     #2                  // Field java/lang/System.out:Ljava/
io/PrintStream;
       5: aload_1
       6: invokevirtual #3                  // Method java/io/PrintStream.printl
n:<Ljava/lang/String;>V
       9: getstatic     #2                  // Field java/lang/System.out:Ljava/
io/PrintStream;
      12: iload_2
      13: invokevirtual #4                  // Method java/io/PrintStream.printl
n:<I>V
      16: return
}
```

9

ViewClass.java

```
public class ViewClass {
   public static void main(String[] args) throws Exception {

      JavaClass myClass = Repository.lookupClass("P");

      System.out.println("*******Constant Pool*********");
      System.out.println(myClass.getConstantPool());

      System.out.println("*******Fields*********");
      System.out.println(Arrays.toString(myClass.getFields()));

      System.out.println("*******Methods*********");
      System.out.println(Arrays.toString(myClass.getMethods()));

      for(Method m: myClass.getMethods()){
         System.out.println(m);
         System.out.println(m.getCode());
      }
   }
}
```

10

JavaClass

- JavaClass class
  - represents a Java class
  - data structures, constant pool, fields, methods, commands contained in .class file
- Methods
  - **`String getClassName()`**
  - **`ConstantPool getConstantPool()`**
  - **`Field[] getFields()`**
  - **`Method[] getMethods()`**
  - **`void dump(String <file_name>)`**
    - create file and place the class

11

ViewClass.java

```
public class ViewClass {
   public static void main(String[] args) throws Exception {

      JavaClass myClass = Repository.lookupClass("P");

      System.out.println("*******Constant Pool*********");
      System.out.println(myClass.getConstantPool());

      System.out.println("*******Fields*********");
      System.out.println(Arrays.toString(myClass.getFields()));

      System.out.println("*******Methods*********");
      System.out.println(Arrays.toString(myClass.getMethods()));

      for(Method m: myClass.getMethods()){
         System.out.println(m);
         System.out.println(m.getCode());
      }
   }
}
```

12

JavaClass Constructor

```
JavaClass(  int class_name_index,
            int superclass_name_index,
            String file_name,
            int major,
            int minor,
            int access_flags,
            ConstantPool constant_pool,
            int[] interfaces,
            Field[] fields,
            Method[] methods,
            Attribute[] attributes,
            byte source )
```

13

JavaClass Instance

- Class instance generator
- org.apache.bcel.Repository class
  - **`lookupClass(String <class_name>)`**
    - look for the named class in classpath

14

ViewClass.java

```
public class ViewClass {
   public static void main(String[] args) throws Exception {

      JavaClass myClass = Repository.lookupClass("P");

      System.out.println("*******Constant Pool*********");
      System.out.println(myClass.getConstantPool());

      System.out.println("*******Fields*********");
      System.out.println(Arrays.toString(myClass.getFields()));

      System.out.println("*******Methods*********");
      System.out.println(Arrays.toString(myClass.getMethods()));

      for(Method m: myClass.getMethods()){
         System.out.println(m);
         System.out.println(m.getCode());
      }
   }
}
```

15

Compile & Run (1)

- Compile & Run
  - `javac -cp bcel-5.2.jar ViewClass.java`
  - `java -cp .;bcel-5.2.jar ViewClass`

16

Compile & Run (2)

명령 프롬프트

```
C:₩PL₩tools₩ javac -cp bcel-5.2.jar ViewClass.java

C:₩PL₩tools₩ java -cp .;bcel-5.2.jar ViewClass
*******Constant Pool*********
1)CONSTANT_Methodref[10](class_index = 6, name_and_type_index = 15)
2)CONSTANT_Fieldref[9](class_index = 16, name_and_type_index = 17)
3)CONSTANT_Methodref[10](class_index = 18, name_and_type_index = 19)
4)CONSTANT_Methodref[10](class_index = 18, name_and_type_index = 20)
5)CONSTANT_Class[7](name_index = 21)
6)CONSTANT_Class[7](name_index = 22)
7)CONSTANT_Utf8[1]("<init>")
8)CONSTANT_Utf8[1]("()V")
9)CONSTANT_Utf8[1]("Code")
10)CONSTANT_Utf8[1]("LineNumberTable")
11)CONSTANT_Utf8[1]("existing")
12)CONSTANT_Utf8[1]("(Ljava/lang/String;)V")
13)CONSTANT_Utf8[1]("SourceFile")
14)CONSTANT_Utf8[1]("P.java")
15)CONSTANT_NameAndType[12](name_index = 7, signature_index = 8)
16)CONSTANT_Class[7](name_index = 23)
17)CONSTANT_NameAndType[12](name_index = 24, signature_index = 25)
18)CONSTANT_Class[7](name_index = 26)
19)CONSTANT_NameAndType[12](name_index = 27, signature_index = 12)
20)CONSTANT_NameAndType[12](name_index = 27, signature_index = 28)
21)CONSTANT_Utf8[1]("P")
22)CONSTANT_Utf8[1]("java/lang/Object")
```

17

Compile & Run (3)

명령 프롬프트

```
*******Fields*********
[]

*******Methods*********
[void <init>(), void existing(String arg1)]
void <init>()
Code(max_stack = 1, max_locals = 1, code_length = 5)
0:    aload_0
1:    invokespecial     java.lang.Object.<init> ()V (1)
4:    return

Attribute(s) =
LineNumber(0, 1)

void existing(String arg1)
Code(max_stack = 2, max_locals = 3, code_length = 17)
0:    iconst_1
1:    istore_2
2:    getstatic         java.lang.System.out Ljava/io/PrintStream; (2)
5:    aload_1
6:    invokevirtual     java.io.PrintStream.println (Ljava/lang/String;)V (3)
9:    getstatic         java.lang.System.out Ljava/io/PrintStream; (2)
12:   iload_2
13:   invokevirtual     java.io.PrintStream.println (I)V (4)
16:   return

Attribute(s) =
```

18

Part 3

INSERT NEW METHOD

19

Test Program

- TestBCEL.java

```
public class TestBCEL {
   public static void main(String[] args) {
      P p = new P();
      String str = "~~using BCEL~~";

      System.out.println("calling an existing method...");
      p.existing(str);

      System.out.println("calling the added method...");
      P.willBeAdded(str);
   }
}
```

20

Sample Java Program

- P.java

```
class P {
   static void willBeAdded(String str) {
      System.out.println(str);
   }
   void existing(String str) {
      int a = 1;
      System.out.println(str);
      System.out.println(a);
   }
}
```

```
명령 프롬프트

C:₩PL₩tools₩ javac TestBCEL.java

C:₩PL₩tools₩ java TestBCEL
calling an existing method...
~~using BCEL~~
1
calling the added method...
~~using BCEL~~

C:₩PL₩tools₩ _
```

P.class

- javap –c P

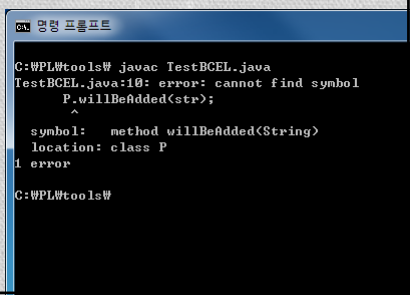
```
명령 프롬프트
C:₩PL₩tools₩ javap -c P
Compiled from "P.java"
class P {
  P();
    Code:
       0: aload_0
       1: invokespecial #1                  // Method java/lang/Object."<init>":
<>V
       4: return

  static void willBeAdded(java.lang.String);
    Code:
       0: getstatic     #2                  // Field java/lang/System.out:Ljava/
io/PrintStream;
       3: aload_0
       4: invokevirtual #3                  // Method java/io/PrintStream.printl
n:<Ljava/lang/String;>V
       7: return

  void existing(java.lang.String);
    Code:
       0: iconst_1
       1: istore_2
       2: getstatic     #2                  // Field java/lang/System.out:Ljava/
io/PrintStream;
       5: aload_1
       6: invokevirtual #3                  // Method java/io/PrintStream.printl
```

22

Sample Java Program

- P.java

```
class P {
   //static void willBeAdded(String str) {
   //   System.out.println(str);
   //}
   void existing(String str) {
      int a = 1;
      System.out.println(str);
      System.out.println(a);
   }
}
```



What To do?

- Add a method to an existing class
  - wilBeAdded(String) to P.class

```
static void willBeAdded(String str) {
  System.out.println(str);
}
```

- Bytecodes

```
0:  getstatic      #2; //Field java/lang/System.out:Ljava/io/P..
3:  aload_0
4:  invokevirtual  #3; //Method java/io/PrintStream.println:(L..
7:  return
```

Changing Classes

- JavaClass and ClassGen
  - same contents, different mutability
- Immutable and mutable
  - JavaClass is immutable, ClassGen is mutable
    - cannot change JavaClass contents
  - also Method and MethodGen
  - Field and FieldGen

25

ClassGen Class

- ClassGen
  - package org.apache.bcel.generic
- Constructor
  - `ClassGen(JavaClass clazz)`
- Methods
  - `JavaClass getJavaClass()`
    - JavaClass from ClassGen
  - `void addMethod(Method m)`
  - `void setMethods(Method[] methods)`

26

Reading and Writing Classes

```
import org.apache.bcel.Repository;
import org.apache.bcel.classfile.JavaClass;
import org.apache.bcel.generic.ClassGen;

public class SomeBCELClass {
  public static void main(String[] a) throws Exception {

    JavaClass myClass = Repository.lookupClass("MyClass");
    ClassGen cg = new ClassGen(myClass);

    //
    //this is where you mess around with the classes
    //

    cg.getJavaClass().dump("MyClass.class");
  }
}
```

27

Method Class

- org.apache.bcel.classfile.Method
  - a method has access flags, a name, a signature and a number of attributes
- Methods
  - `String getName()`
  - `Code getCode()`

28

ViewClass.java

```
public class ViewClass {
   public static void main(String[] args) throws Exception {

      JavaClass myClass = Repository.lookupClass("P");

      System.out.println("*******Constant Pool*********");
      System.out.println(myClass.getConstantPool());

      System.out.println("*******Fields*********");
      System.out.println(Arrays.toString(myClass.getFields()));

      System.out.println("*******Methods*********");
      System.out.println(Arrays.toString(myClass.getMethods()));

      for(Method m: myClass.getMethods()){
         System.out.println(m);
         System.out.println(m.getCode());
      }
   }
}
```

29

Method Class

- Example

```
Method mainMethod = null;
for (Method m: myClass.getMethods()) {
   if (m.getName().equals("main")) {
      mainMethod = m;
      System.out.println("Found main Method");
      break;
   }
}
```

30

MethodGen Class

- org.apache.bcel.generic.MethodGen
- Constructor 1
  - `MethodGen(Method m, String class_name, ConstantPoolGen cp)`
- Example 1

```
ClassGen cg = new ClassGen(myClass);
ConstantPoolGen theCPool = cg.getConstantPool();
MethodGen mainMethodGen =
    new MethodGen(mainMethod, cg.getClassName(), theCPool);
```

31

MethodGen Class

- Constructor 2
  - `MethodGen(int access_flags, Type return_type, Type[] arg_types, String[] arg_names, String method_name, String class_name, InstructionList il, ConstantPoolGen cp)`
- Example 2

```
ClassGen cg = new ClassGen(myClass);
ConstantPoolGen cPool = cg.getConstantPool();
InstructionList iList = new InstructionList();

MethodGen mg = new MethodGen(
            Constants.ACC_STATIC | Constants.ACC_PUBLIC,
            Type.VOID,
            new Type[] { Type.STRING },
            new String[] { "str" },
            "willBeAdded",
            "P",
            iList,
            cPool );
```

```
class P {
   //static void willBeAdded(String str) {
   //   System.out.println(str);
   //}
   void existing(String str) {
      int a = 1;
      System.out.println(str);
      System.out.println(a);
   }
}
```

32

MethodGen Class

- Methods
  - **`Method getMethod()`**
    - get method object
  - **`void setInstructionList(InstructionList il)`**
  - **`void setMaxLocals()`**
    - compute maximum number of local variables
  - **`void setMaxStack()`**
    - computes maximum stack size

33

AddMethod.java

```
public static void appendMethod(ClassGen cg) {
  ConstantPoolGen cPool = cg.getConstantPool();
  InstructionList iList = new InstructionList();
  InstructionFactory iFac = new InstructionFactory(cg, cPool);

  GETSTATIC getstatic = iFac.createGetStatic("java.lang.System", "out", new
ObjectType("java.io.PrintStream") );
  iList.append(getstatic);

  iList.append(new ALOAD(0));

  InvokeInstruction invoke = iFac.createInvoke("java.io.PrintStream",
"println", Type.VOID, new Type[] { Type.STRING }, Constants.INVOKEVIRTUAL);
  iList.append(invoke);

  iList.append(new RETURN());

  MethodGen mg = new MethodGen(Constants.ACC_STATIC | Constants.ACC_PUBLIC,
Type.VOID, new Type[] { Type.STRING }, new String[] { "str" },
"willBeAdded", "P", iList, cPool );

  mg.setMaxLocals();
  mg.setMaxStack();
  cg.addMethod(mg.getMethod());
}
```

34

Instruction Class

- Instruction class
  - abstract class for Java bytecode
  - super class for all bytecodes
    - all instructions actually have their own class
    - ACONST_NULL, ArithmeticInstruction, ArrayInstruction, ARRAYLENGTH, ATHROW, BIPUSH, BranchInstruction, BREAKPOINT, ConversionInstruction, CPInstruction, DCMPG, DCMPL, DCONST, FCMPG, FCMPL, FCONST, ICONST, IMPDEP1, IMPDEP2, LCMP, LCONST, LocalVariableInstruction, MONITORENTER, MONITOREXIT, NEWARRAY, NOP, RET, ReturnInstruction, SIPUSH, StackInstruction
- InstructionList
  - container for list of Instruction objects
  - append, insert, move, delete

35

Instruction Class

- Methods
  - `String getName()`
  - `short getOpcode()`

36

InstructionList Class

- Methods
  - **`Instruction[] getInstructions()`**
  - **`InstructionHandle[] getInstructionHandles()`**
  - **`InstructionHandle append(Instruction i)`**
    - append an instruction to the end of list
  - **`InstructionHandle append(Instruction i, Instruction j)`**
    - append a single instruction j after another instruction i
  - **`InstructionHandle insert(Instruction i, Instruction j)`**
    - insert a single instruction j before another instruction i
  - **`void delete(Instruction i)`**
    - remove instruction from this list.
  - **`byte[] getByteCode()`**
    - when everything is finished, use this method to convert the instruction list into an array of bytes

37

AddMethod.java

```
public static void appendMethod(ClassGen cg) {
  ConstantPoolGen cPool = cg.getConstantPool();
  InstructionList iList = new InstructionList();
  InstructionFactory iFac = new InstructionFactory(cg, cPool);

  GETSTATIC getstatic = iFac.createGetStatic("java.lang.System", "out", new
ObjectType("java.io.PrintStream") );
  iList.append(getstatic);

  iList.append(new ALOAD(0));

  InvokeInstruction invoke = iFac.createInvoke("java.io.PrintStream",
"println", Type.VOID, new Type[] { Type.STRING }, Constants.INVOKEVIRTUAL);
  iList.append(invoke);

  iList.append(new RETURN());

  MethodGen mg = new MethodGen(Constants.ACC_STATIC | Constants.ACC_PUBLIC,
Type.VOID, new Type[] { Type.STRING }, new String[] { "str" },
"willBeAdded", "P", iList, cPool );

  mg.setMaxLocals();
  mg.setMaxStack();
  cg.addMethod(mg.getMethod());
}
```

38

Making an Instruction

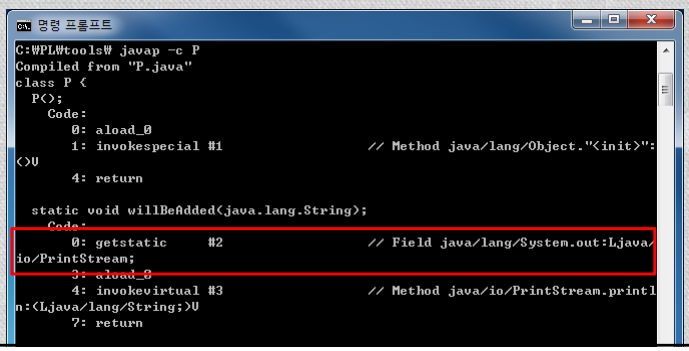
- InstructionFactory class
  - creates introduction
- Methods
  - **`GETSTATIC createGetStatic(String class_name, String name, Type t)`**
  - **`InvokeInstruction createInvoke(String class_name, String name, Type ret_type, Type[] arg_types, short kind)`**
    - Kind: INVOKEINTERFACE, INVOKESTATIC, INVOKEVIRTUAL, INVOKESPECIAL
  - **`Instruction createConstant(Object value)`**
  - and much more…

39

Making an Instruction

- Example

```
InstructionFactory iFac = new InstructionFactory(cg, cPool);
GETSTATIC getstatic = iFac.createGetStatic(
                    "java.lang.System",
                    "out",
                    new ObjectType("java.io.PrintStream") );
```

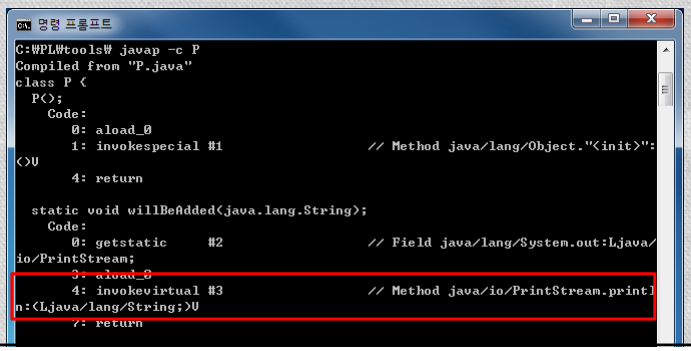


40

Making an Instruction

- Example

```
InstructionFactory iFac = new InstructionFactory(cg, cPool);
InvokeInstruction invoke = iFac.createInvoke(
                    "java.io.PrintStream",
                    "println",
                    Type.VOID,
                    new Type[] { Type.STRING },
                    Constants.INVOKEVIRTUAL);
```



Making an Instruction

- ALOAD constructor
  - **ALOAD(int n)**
- RETURN constructor
  - **RETURN()**

AddMethod.java

```
public static void appendMethod(ClassGen cg) {
  ConstantPoolGen cPool = cg.getConstantPool();
  InstructionList iList = new InstructionList();
  InstructionFactory iFac = new InstructionFactory(cg, cPool);

  GETSTATIC getstatic = iFac.createGetStatic("java.lang.System", "out", new
ObjectType("java.io.PrintStream") );
  iList.append(getstatic);

  iList.append(new ALOAD(0));

  InvokeInstruction invoke = iFac.createInvoke("java.io.PrintStream",
"println", Type.VOID, new Type[] { Type.STRING }, Constants.INVOKEVIRTUAL);
  iList.append(invoke);

  iList.append(new RETURN());

  MethodGen mg = new MethodGen(Constants.ACC_STATIC | Constants.ACC_PUBLIC,
Type.VOID, new Type[] { Type.STRING }, new String[] { "str" },
"willBeAdded", "P", iList, cPool );

  mg.setMaxLocals();
  mg.setMaxStack();
  cg.addMethod(mg.getMethod());
}
```

43

AddMethod.java

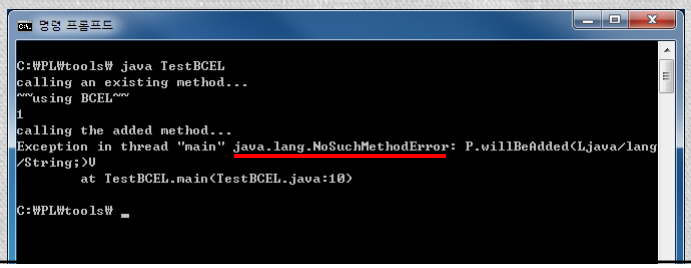
```
import org.apache.bcel.classfile.*;
import org.apache.bcel.generic.*;
import org.apache.bcel.*;

public class AddMethod {
   public static void appendMethod(ClassGen cg) {

   ...

   }

   public static void main(String[] args) throws Exception {
      System.out.println("== 작업을 시작했습니다 ==");

      JavaClass myClass = Repository.lookupClass("P");
      ClassGen cg = new ClassGen(myClass);
      appendMethod(cg);
      cg.getJavaClass().dump("P.class");

      System.out.println("== 작업을 끝냈습니다 ==");
   }
}
```

44

Sample Java Program

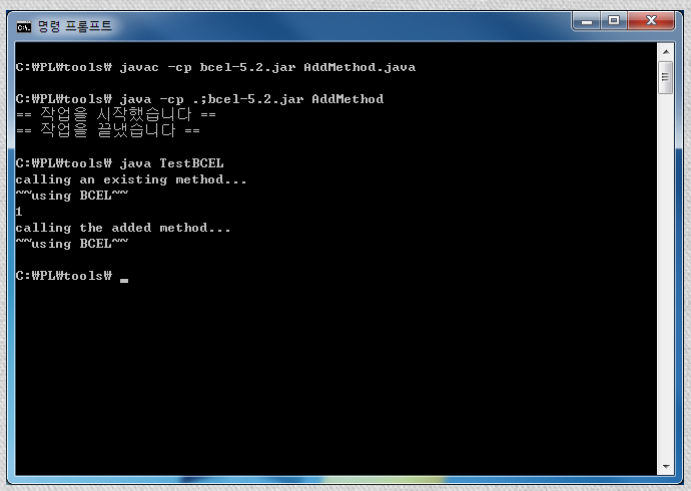
- P.java

```
class P {
   //static void willBeAdded(String str) {
   //   System.out.println(str);
   //}
   void existing(String str) {
      int a = 1;
      System.out.println(str);
      System.out.println(a);
   }
}
```



45

Compile & Run



46

Part 4

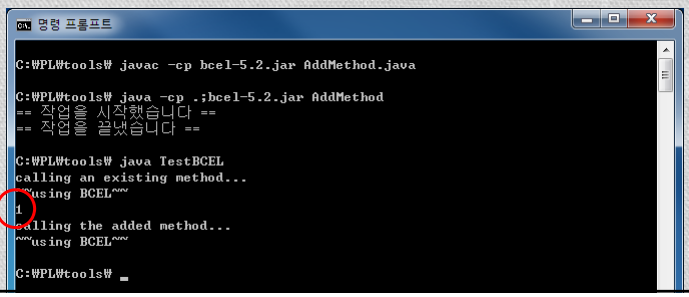
REPLACE INSTRUCTION

47

What To do?

- Change a bytecode
  - iconst_1 -> iconst_2
- P.java

```
void existing(String str) {
  int a = 1;
  System.out.println(str);
  System.out.println(a);
}
```



48

P.class

```
명령 프롬프트
C:₩Users₩slee₩Desktop₩대학원13차세대PL₩tools>javap -c P
Compiled from "P.java"
class P {
  P();
    Code:
       0: aload_0
       1: invokespecial #1                  // Method java/lang/Object."<init>":
<>V
       4: return

  void existing(java.lang.String);
    Code:
       0: iconst_1
       1: istore_2
       2: getstatic     #2                  // Field java/lang/System.out:Ljava/
io/PrintStream;
       5: aload_1
       6: invokevirtual #3                  // Method java/io/PrintStream.printl
n:<Ljava/lang/String;>V
       9: getstatic     #2                  // Field java/lang/System.out:Ljava/
io/PrintStream;
      12: iload_2
      13: invokevirtual #4                  // Method java/io/PrintStream.printl
n:<I>V
      16: return
}
```

49

ChangeCode.java (1)

```
public class ChangeCode {
  public static void changeCode(ClassGen cg) {
    ConstantPoolGen cPool = cg.getConstantPool();
    Method[] m = cg.getMethods();

    for (int i=0; i<m.length; i++) {
      MethodGen mg = new MethodGen(m[i], cg.getClassName(), cPool);
      InstructionList iList = mg.getInstructionList();
      InstructionHandle[] iHand = iList.getInstructionHandles();

      for (int f=0; f<iHand.length; f++) {
        Instruction inst = iHand[f].getInstruction();
        if (inst instanceof ICONST) {
          ICONST ic = (ICONST)inst;
          if (ic.getValue().intValue() == 1) {
            InstructionFactory iFac = new InstructionFactory(cg, cPool);
            Instruction newInst = iFac.createConstant(2);
            try {
              iList.insert(iHand[f+1], newInst);
              iList.delete(iHand[f]);
            } catch(TargetLostException e) { }
          }
        }
      }
```

50

ChangeCode.java (2)

```
        iList.setPositions();
        mg.setInstructionList(iList);
        mg.setMaxStack();
        mg.setMaxLocals();
        mg.removeLineNumbers();

        cg.replaceMethod(m[i], mg.getMethod());
      }
    }

    public static void main(String[] args) throws Exception {
      System.out.println("== 작업을 시작했습니다 ==");

      JavaClass myClass = Repository.lookupClass("P");
      ClassGen cg = new ClassGen(myClass);
      changeCode(cg);
      cg.getJavaClass().dump("P.class");

      System.out.println("== 작업을 끝냈습니다 ==");
    }
  }
```

51

Finding an Instruction

- Example

```
Instruction inst = null;
InstructionHandle[] iHand = iList.getInstructionHandles();
for (int f=0; f<iHandles.length; f++) {
   if (iHand[f].getInstruction() instanceof INVOKEVIRTUAL) {
      inst = iHandles[f].getInstruction();
      System.out.println("found the invoke virtual");
      break;
   }
}
```

52

InstructionHandle Class

- InstructionHandle class
  - reference to an Instruction
- Methods
  - **`Instruction getInstruction()`**

53

1. For Each Method...

```
public class ChangeCode {
  public static void changeCode(ClassGen cg) {
    ConstantPoolGen cPool = cg.getConstantPool();
    Method[] m = cg.getMethods();

    for (int i=0; i<m.length; i++) {
      MethodGen mg = new MethodGen(m[i], cg.getClassName(), cPool);
      InstructionList iList = mg.getInstructionList();
      InstructionHandle[] iHand = iList.getInstructionHandles();

      for (int f=0; f<iHand.length; f++) {
        Instruction inst = iHand[f].getInstruction();
        if (inst instanceof ICONST) {
          ICONST ic = (ICONST)inst;
          if (ic.getValue().intValue() == 1) {
            InstructionFactory iFac = new InstructionFactory(cg, cPool);
            Instruction newInst = iFac.createConstant(2);
            try {
              iList.insert(iHand[f+1], newInst);
              iList.delete(iHand[f]);
            } catch(TargetLostException e) { }
          }
        }
      }
```

54

2. Get Instructions and Handles

```
public class ChangeCode {
  public static void changeCode(ClassGen cg) {
    ConstantPoolGen cPool = cg.getConstantPool();
    Method[] m = cg.getMethods();

    for (int i=0; i<m.length; i++) {
      MethodGen mg = new MethodGen(m[i], cg.getClassName(), cPool);
      InstructionList iList = mg.getInstructionList();
      InstructionHandle[] iHand = iList.getInstructionHandles();

      for (int f=0; f<iHand.length; f++) {
        Instruction inst = iHand[f].getInstruction();
        if (inst instanceof ICONST) {
          ICONST ic = (ICONST)inst;
          if (ic.getValue().intValue() == 1) {
            InstructionFactory iFac = new InstructionFactory(cg, cPool);
            Instruction newInst = iFac.createConstant(2);
            try {
              iList.insert(iHand[f+1], newInst);
              iList.delete(iHand[f]);
            } catch(TargetLostException e) { }
          }
        }
      }
```

55

2. Get Instructions and Handles

```
public class ChangeCode {
  public static void changeCode(ClassGen cg) {
    ConstantPoolGen cPool = cg.getConstantPool();
    Method[] m = cg.getMethods();

    for (int i=0; i<m.length; i++) {
      MethodGen mg = new MethodGen(m[i], cg.getClassName(), cPool);
      InstructionList iList = mg.getInstructionList();
      InstructionHandle[] iHand = iList.getInstructionHandles();

      for (int f=0; f<iHand.length; f++) {
        Instruction inst = iHand[f].getInstruction();
        if (inst instanceof ICONST) {
          ICONST ic = (ICONST)inst;
          if (ic.getValue().intValue() == 1) {
            InstructionFactory iFac = new InstructionFactory(cg, cPool);
            Instruction newInst = iFac.createConstant(2);
            try {
              iList.insert(iHand[f+1], newInst);
              iList.delete(iHand[f]);
            } catch(TargetLostException e) { }
          }
        }
      }
```

56

3. For Each Instruction…

```
public class ChangeCode {
  public static void changeCode(ClassGen cg) {
    ConstantPoolGen cPool = cg.getConstantPool();
    Method[] m = cg.getMethods();

    for (int i=0; i<m.length; i++) {
      MethodGen mg = new MethodGen(m[i], cg.getClassName(), cPool);
      InstructionList iList = mg.getInstructionList();
      InstructionHandle[] iHand = iList.getInstructionHandles();

      for (int f=0; f<iHand.length; f++) {
        Instruction inst = iHand[f].getInstruction();
        if (inst instanceof ICONST) {
          ICONST ic = (ICONST)inst;
          if (ic.getValue().intValue() == 1) {
            InstructionFactory iFac = new InstructionFactory(cg, cPool);
            Instruction newInst = iFac.createConstant(2);
            try {
              iList.insert(iHand[f+1], newInst);
              iList.delete(iHand[f]);
            } catch(TargetLostException e) { }
          }
        }
      }
```

57

4. Finding ICONST

```
public class ChangeCode {
  public static void changeCode(ClassGen cg) {
    ConstantPoolGen cPool = cg.getConstantPool();
    Method[] m = cg.getMethods();

    for (int i=0; i<m.length; i++) {
      MethodGen mg = new MethodGen(m[i], cg.getClassName(), cPool);
      InstructionList iList = mg.getInstructionList();
      InstructionHandle[] iHand = iList.getInstructionHandles();

      for (int f=0; f<iHand.length; f++) {
        Instruction inst = iHand[f].getInstruction();
        if (inst instanceof ICONST) {
          ICONST ic = (ICONST)inst;
          if (ic.getValue().intValue() == 1) {
            InstructionFactory iFac = new InstructionFactory(cg, cPool);
            Instruction newInst = iFac.createConstant(2);
            try {
              iList.insert(iHand[f+1], newInst);
              iList.delete(iHand[f]);
            } catch(TargetLostException e) { }
          }
        }
      }
```

58

4. Finding ICONST_1

```
public class ChangeCode {
  public static void changeCode(ClassGen cg) {
    ConstantPoolGen cPool = cg.getConstantPool();
    Method[] m = cg.getMethods();

    for (int i=0; i<m.length; i++) {
      MethodGen mg = new MethodGen(m[i], cg.getClassName(), cPool);
      InstructionList iList = mg.getInstructionList();
      InstructionHandle[] iHand = iList.getInstructionHandles();

      for (int f=0; f<iHand.length; f++) {
        Instruction inst = iHand[f].getInstruction();
        if (inst instanceof ICONST) {
          ICONST ic = (ICONST)inst;
          if (ic.getValue().intValue() == 1) {
            InstructionFactory iFac = new InstructionFactory(cg, cPool);
            Instruction newInst = iFac.createConstant(2);
            try {
              iList.insert(iHand[f+1], newInst);
              iList.delete(iHand[f]);
            } catch(TargetLostException e) { }
          }
        }
      }
```

59

5. Make ICONST_2

```
public class ChangeCode {
  public static void changeCode(ClassGen cg) {
    ConstantPoolGen cPool = cg.getConstantPool();
    Method[] m = cg.getMethods();

    for (int i=0; i<m.length; i++) {
      MethodGen mg = new MethodGen(m[i], cg.getClassName(), cPool);
      InstructionList iList = mg.getInstructionList();
      InstructionHandle[] iHand = iList.getInstructionHandles();

      for (int f=0; f<iHand.length; f++) {
        Instruction inst = iHand[f].getInstruction();
        if (inst instanceof ICONST) {
          ICONST ic = (ICONST)inst;
          if (ic.getValue().intValue() == 1) {
            InstructionFactory iFac = new InstructionFactory(cg, cPool);
            Instruction newInst = iFac.createConstant(2);
            try {
              iList.insert(iHand[f+1], newInst);
              iList.delete(iHand[f]);
            } catch(TargetLostException e) { }
          }
        }
      }
```

60

InstructionList Class

- Methods
  - **`Instruction[] getInstructions()`**
  - **`InstructionHandle[] getInstructionHandles()`**
  - **`InstructionHandle append(Instruction i, Instruction j)`**
    - append a single instruction j after another instruction i
  - **`InstructionHandle insert(Instruction i, Instruction j)`**
    - insert a single instruction j before another instruction i
  - **`void delete(Instruction i)`**
    - remove instruction from this list.
  - **`void setPositions()`**
    - give all instructions their position number (offset in byte stream)
    - make the list ready to be dumped

61

6. Insert New Instruction

```
public class ChangeCode {
  public static void changeCode(ClassGen cg) {
    ConstantPoolGen cPool = cg.getConstantPool();
    Method[] m = cg.getMethods();

    for (int i=0; i<m.length; i++) {
      MethodGen mg = new MethodGen(m[i], cg.getClassName(), cPool);
      InstructionList iList = mg.getInstructionList();
      InstructionHandle[] iHand = iList.getInstructionHandles();

      for (int f=0; f<iHand.length; f++) {
        Instruction inst = iHand[f].getInstruction();
        if (inst instanceof ICONST) {
          ICONST ic = (ICONST)inst;
          if (ic.getValue().intValue() == 1) {
            InstructionFactory iFac = new InstructionFactory(cg, cPool);
            Instruction newInst = iFac.createConstant(2);
            try {
              iList.insert(iHand[f+1], newInst);
              iList.delete(iHand[f]);
            } catch(TargetLostException e) { }
          }
        }
      }
```

62

7. Delete the Old One

```
public class ChangeCode {
  public static void changeCode(ClassGen cg) {
    ConstantPoolGen cPool = cg.getConstantPool();
    Method[] m = cg.getMethods();

    for (int i=0; i<m.length; i++) {
      MethodGen mg = new MethodGen(m[i], cg.getClassName(), cPool);
      InstructionList iList = mg.getInstructionList();
      InstructionHandle[] iHand = iList.getInstructionHandles();

      for (int f=0; f<iHand.length; f++) {
        Instruction inst = iHand[f].getInstruction();
        if (inst instanceof ICONST) {
          ICONST ic = (ICONST)inst;
          if (ic.getValue().intValue() == 1) {
            InstructionFactory iFac = new InstructionFactory(cg, cPool);
            Instruction newInst = iFac.createConstant(2);
            try {
              iList.insert(iHand[f+1], newInst);
              iList.delete(iHand[f]);
            } catch(TargetLostException e) { }
          }
        }
      }
```

63

8. Rearrange Position Number

```
      iList.setPositions();
      mg.setInstructionList(iList);
      mg.setMaxStack();
      mg.setMaxLocals();
      mg.removeLineNumbers();

      cg.replaceMethod(m[i], mg.getMethod());
    }
  }

  public static void main(String[] args) throws Exception {
    System.out.println("== 작업을 시작했습니다 ==");

    JavaClass myClass = Repository.lookupClass("P");
    ClassGen cg = new ClassGen(myClass);
    changeCode(cg);
    cg.getJavaClass().dump("P.class");

    System.out.println("== 작업을 끝냈습니다 ==");
  }
}
```

64

9. Dump Instructions to the Method

```
        iList.setPositions();
        mg.setInstructionList(iList);
        mg.setMaxStack();
        mg.setMaxLocals();
        mg.removeLineNumbers();

        cg.replaceMethod(m[i], mg.getMethod());
      }
    }

    public static void main(String[] args) throws Exception {
      System.out.println("== 작업을 시작했습니다 ==");

      JavaClass myClass = Repository.lookupClass("P");
      ClassGen cg = new ClassGen(myClass);
      changeCode(cg);
      cg.getJavaClass().dump("P.class");

      System.out.println("== 작업을 끝냈습니다 ==");
    }
  }
```

65

10. Replace the Method

```
        iList.setPositions();
        mg.setInstructionList(iList);
        mg.setMaxStack();
        mg.setMaxLocals();
        mg.removeLineNumbers();

        cg.replaceMethod(m[i], mg.getMethod());
      }
    }

    public static void main(String[] args) throws Exception {
      System.out.println("== 작업을 시작했습니다 ==");

      JavaClass myClass = Repository.lookupClass("P");
      ClassGen cg = new ClassGen(myClass);
      changeCode(cg);
      cg.getJavaClass().dump("P.class");

      System.out.println("== 작업을 끝냈습니다 ==");
    }
  }
```

66

Compile & Run

```
명령 프롬프트
C:₩PL₩tools₩ javac -cp bcel-5.2.jar ChangeCode.java

C:₩PL₩tools₩ java -cp .;bcel-5.2.jar ChangeCode
== 작업을 시작했습니다 ==
== 작업을 끝냈습니다 ==

C:₩PL₩tools₩ java TestBCEL
calling an existing method...
~~~using BCEL~~~
2
calling the added method...
~~~using BCEL~~~

C:₩PL₩tools₩
```

67

ICONST_2

- javap –c P

```
명령 프롬프트
  void existing(java.lang.String);
    Code:
       0: iconst_2
       1: istore_2
       2: getstatic     #2                  // Field java/lang/System.out:Ljava/
io/PrintStream;
       5: aload_1
       6: invokevirtual #3                  // Method java/io/PrintStream.printl
n:(Ljava/lang/String;)V
       9: getstatic     #2                  // Field java/lang/System.out:Ljava/
io/PrintStream;
      12: iload_2
      13: invokevirtual #4                  // Method java/io/PrintStream.printl
n:(I)V
      16: return

  public static void willBeAdded(java.lang.String);
    Code:
       0: getstatic     #2                  // Field java/lang/System.out:Ljava/
io/PrintStream;
       3: aload_0
       4: invokevirtual #3                  // Method java/io/PrintStream.printl
n:(Ljava/lang/String;)V
       7: return
}

C:₩PL₩tools₩ _
```

68