**1. Сценарий работы процессора внедрения для Target.class и Aspect.class:**

1. Создание ClassNode из Target.class и Aspect из Aspect.class

1a. Если Aspect.class не является аспектом – выдать ошибку

2. Для каждого правила rule\_i из Aspect.class

найти List<MethodNode> список методов в Target.class, соответствующих условию rule\_i

3. Составить список (списков) всех точек внедрения и предложить пользователю.

*«Список найденных точек внедрения сохранен в файл <target.class\_directory>/<target>\_<aspect>\_jp.txt»*

текстовый файл:

---<правило1>

1. <точка внедрения1>

2. <точка внедрения2>

3. …

---<правило2>

4. …

*«Вы можете ввести номера выбранных точек внедрения через запятую или завершить работу (quit)»*

4. Произвести внедрение выбранных точек (1, 3, 4) или завершить работу

* сохранить результат внедрения в *<target.class\_directory>/<target>\_<aspect>/Target.class*
* сохранить *<target>\_<aspect>\_jpweaved.txt* в *<target.class\_directory>/<target>\_<aspect>*

Режим работы command-line:

* weaver.jar <Target.class or directory> <Aspect.class or jar>

**2. Что такое аспект?**

* java класс
* наследует Aspect **??? (NO)**
* содержит *правила* – public static методы, аннотированные AspectAction
* аннотации AspectAction содержат *условия* правил
* Аспект (*имя описание*)-> Правило = Условие + Действие
  + Аспект: *имя:String, описание:String* <- AspectDescription / null, *правила:List<Правило>*
  + Правило: *условие:Условие* <- AspectAction, *действие:Действие, описание: String* <- AspectDescription / null, *аспект:Аспект*
  + Условие: *условие:String, правило:Правило*
  + Действие: *имя:String, desc:String, правило:Правило*

**3.** When using the *%instead* join point kind, please indicate target app’s **static** **(?)** methods only, since the action is also a static method. Otherwise the weaver issues an error message and weaving will be impossible.

* Листинг 4

%instead %call \*BankAccount.withdraw(float) && args(..)

%action

public static float WithdrawWrapper(float amount)

{

BankAccount acc = (BankAccount)TargetObject;

if (isLicenceValid(TargetMemberInfo.Name))

return acc.withdraw(amount);

Console.WriteLine("Withdraw operation is not allowed");

return acc.Balance;

}

**4. Поиск точек внедрения:**

* ClassNode -> MethodNode -> InstNode (MethodInstNode для *%call*) -> maybeJP() ? **fast!**-> проверка по всем правилам (Аспект -> Правило -> Условие.accept(inst)? запомнить пару inst (+methnode?) + rule)

**Внедрение действий:**

* ***Внедрение*** (алгоритм):
  + ! Action.isParametrized()
    - 1. AFTER – сразу после
      2. BEFORE - сразу перед
      3. INSTEAD - для каждого аргумента *disposeValue().*
  + Action.isParametrized().
    - 1. AFTER и BEFORE – вставить DUPx + xSTORE в нужных местах, xLOADs перед вызовом
      2. INSTEAD – для ненужных аргументов *disposeValue()*, остальные переставить

NOTE: для случаев 3, 4, 5 надо в коде, отвечающем за вызов метода выделить части, связанные с каждым из аргументов (*passValue()*)

* *passValue()****:***
* Проблемы алгоритма:
  + POP2, DUP2x: stackCount ? **OK**

Test on previous instr (7359 classfiles):

1. Pop2 – invoke (l)
2. Dup2 – ISUB, iconst\_0, iconst\_1, lor, idiv, lxor, laload, getfield/static(i), putfield/static (i) – [--]++/+=, invoke (l) - assign, iload, irem, lshl, || iconst\_2, iconst\_3, lload || ldc (l) – multi assign, get/putstatic/get/putfield (l) – multi assign,

l/dstore (l) - --/++ in multi assign

100 7 ISUB

3 31 iconst\_0

4 7 iconst\_1

5 2 iconst\_2

6 2 iconst\_3

129 6 lor

108 1 idiv

131 89 lxor

47 57 laload

184 18 invokestatic

185 1 invokeinterface

21 81 iload

22 1 lload

112 1 irem

182 1 invokevirtual

180 112 getfield

121 12 lshl

181 3 putfield

-- ‘+2’ (1word) – 247 times, ‘+1’ (2word) - 185

1. Dup2\_x1 – lload, lconst\_0, laload, get/putfield/static (l), ldc (l) – multi assign, l/dstore (l) - --/++ in multi assign
2. Dup2\_x2 – lconst\_0
   * Наличие Jumps:
3. *a ? b : c*
4. *a <, >, ==, <> b* (boolean)
5. Идея: FRAMES. In fact these nodes must(\*) be inserted just before any instruction node i that follows an unconditionnal branch instruction such as GOTO or THROW, that is the target of a jump instruction, or that starts an exception handler block. The stack map frame types must describe the values of the local variables and of the operand stack elements just before i is executed. (\*) this is mandatory only for classes whose version is greater than or equal to [V1\_6](http://asm.ow2.org/asm31/javadoc/user/org/objectweb/asm/Opcodes.html#V1_6).

* *disposeValue()*:
* Аргументы метода (инструкции):
* \*\*\*\*\*\*\*\*\*\*\* PUSHING INSTR \*\*\*\*\*\*\*\*\*\*\*\*\*

1. Константа (LDC, …) : …
2. Локальная пер (ALOAD, ILOAD, FLOAD, …) : …
3. Поле объекта (GETFIELD) : o
4. Статическое поле (GETSTATIC) : …
5. Вызов метода (INVOKESTATIC, INVOKEVIRTUAL, …) : <o>, v1 , ... , vn
6. \*o Оператор NEW (invokespecial …<init>…) : <o>, v1 , ... , vn
7. Array[i] : o , i
8. IFEQ (*a ? b : c*)
9. \*a Arithmetic and logic **(**xADD, xSUB, xMUL, xDIV, …) : a, <b>
10. \*a INSTANCEOF : o
11. \*o NEWARRAY, ANEWARRAY : n
12. \*o MULTIANEWARRAY: i1 , ... , in
13. \*a ARRAYLENGTH : o

* \*\*\*\*\*\*\*\*\*\*\* SPECIAL CASES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. DUP + xSTORE (PUTx)
2. *method*(**new** **int**[]{0, 1, 2});
3. *method*(**this**.l++);
4. Integer j = 0;

*method*(j++);

* \*\*\*\*\*\*\*\*\*\*\* MODIFYING INSTR \*\*\*\*\*\*\*\*\*\*\*\*\*

1. Casts : v
2. IINC : …

* FRAMEs и MAXLOCAL & MAXSTACK:
  + Maxlocal & maxstack:
    - ClassWriter(ClassWriter.COMPUTE\_MAXS) (*10% slower*)
  + Frames:
  + With new ClassWriter(ClassWriter.COMPUTE\_FRAMES) everything is

computed automatically. You don’t have to call visitFrame, but you must still call visitMaxs (arguments will be ignored and recomputed). (*two times* slower)

* + Java 6 class contains only the frames for the instructions that correspond to jump targets or exception handlers, or that follow unconditional jump instructions (GOTO, ATHROW). These frames can be represented just before their associated bytecode instruction.
  + What is the situation when frames *can be* *not* updated? 1) the inserted code leaves the operand stack unchanged, 2) the inserted code does not contain jump instructions and 3) the jump instructions – or, more formally, the control flow graph – of the original code is not modified.
  + Note that the original frames associated to the method *become invalid* when the local variables are renumbered, and a fortiori when new local variables are inserted. Hopefully it is possible to avoid recomputing these frames from scratch: indeed no frames must be added or removed, and it “suffices” to reorder the content of local variables in the original frames to get the frames of the transformed method. **LocalVariablesSorter** takes care of that automatically.

**5.**

WeavingRuleConditionClause = WeavingContext JoinPointKind Pattern [&& ArgsInfo] **[&&** Restrictions]

WeavingContext =

**%before | %after | %instead .**

JoinPointKind =

**%call .**

Pattern = MethodPattern | …

MethodPattern = MethodNameFilter **[**  MethodArgumentTypes ] .

MethodArgumentTypes = **' ('** **'..'** **|** JavaType **['**, **'** JavaType**]\* [‘, ..’]** **') ' .**

MethodNameFilter =

[**static**] [JavaType| \*] MethodNameWildCard **.**

ArgsInfo = MethodArgsInfo | …

MethodArgsInfo =

%**args '(' arg'['** IntLiteral **']'** [ **', '** **arg'['** IntLiteral **']' ]** \* **')' |**

**%args '(..)'**

Restrictions = Restriction **[** **'**&&**'** Restriction  **]** \* *.*

Restriction =

**%[ '!' ]within '('** TypeNameWildCard**')' |**

**%[ '!' ]withincode '('** MethodNameWildCard **')' .**

**JavaType =** TypeNameWildCard | JavaPrimitiveType| ArrayType

ArrayType = JavaType[]

**6.** Образцы:

* MethodPattern = MethodNameFilter **[**  MethodArgumentTypes ] .

MethodNameFilter =

[**static**] [JavaType| \*] MethodNameWildCard **.**

MethodArgumentTypes = **' ('** **'..'** **|** JavaType **['**, **'** JavaType**]\* [‘, ..’]** **') ' .**

* MethodNameWildCard: [packageWC.][classWC.]methodWC
* TypeNameWildCard: [packageWC.]classWC

Детали:

* String pattern -> Pattern pattern ? YES
* класс EntityWildCard: packageWC:Pattern, classWC:Pattern, methodWC:Pattern, hasPackage:boolean, hasClass:boolean
* CallCondClause inner класс MethodPattern: nameFilter: MethodNameFilter, argTypes: MethodArgTypes, hasArgTypes:boolean
* MethodPattern inner класс MethodNameFilter: access:int, hasAccess:boolean, hasStatic:boolean, retType: EntityWildCard**,** hasRetType:boolean, methodNameWC:EntityWildCard
* MethodPattern inner класс MethodArgTypes: массив JavaType + ‘..’
* [ **public** **|** **private** **|** **protected** | \*/**default**] – пока не поддерживается
* Преобразования при accepts:
  + internal -> normal (just “L” ?)
  + primitive Z, D, … -> “boolean”, “double”
  + V -> “void”
  + Array ([[SomeType -> SomeType[][])
  + ‘;’ ???

**7.** Test:

1. new RuleCondition ***PASS***
2. RuleCondition.accepts()
   1. Pattern
      1. [**static**] [JavaType| \*] MethodNameWildCard ***PASS***
      2. + **' ('** **'..'** **|** JavaType **['**, **'** JavaType**]\*** **') ' *PASS***
   2. Pattern + Restrictions ***PASS***
3. NoArgs: ***PASS***
   1. BEFORE & AFTER ***PASS***
      1. General ***PASS***
      2. Frames ***PASS***
   2. INSTEAD ***PASS***
      1. General ***PASS***
      2. Removing linenumbers ***PASS***
      3. Frames ***PASS***
      4. method(new Object()); POP ? ***PASS***
4. Args: ***PASS***
   1. BEFORE & AFTER
      1. General ***PASS***
      2. Frames ***PASS***
   2. INSTEAD ***PASS***

**8.** Bugs in Aspect.NET:

**9.** Исследование:

* INSTEAD
  + *disposeValue():* Methods – *del*, IINC, NEW ? **OK** (оставить + обсудить)
* AFTER
  + Throw case **OK** (нет проблемы)
* Interface & inherited methods in patterns ??? **OK** (no supported)

**10.** Список вопросов к науч. рук.:

* Удаление аргументов для INSTEAD
* [ **public** **|** **private** **|** **protected** | \*/**default**]
  + Private = INVOKESPECIAL + not <init> + INVOKESTATICown
  + INVOKEINTERFACE=public…

**11. Aspect.Java Framework** **OK**

* “Add existing aspects JARs/folders” – implement adding folders **OK**
* “Reload JAR/folder” **OK**
* Joinpoints:
  + Closing of target project event -> clear joinpoint tab(weave button) **OK**
  + Choosing another target project event -> clear joinpoint tab(weave button) **OK**
  + Изн-но выбраны все **OK**
* Dialogs – “No joinpoints found.” & “The aspects are weaved into <target\_project> successfully. The weaved project is located in aspects-bin directory of <target\_project>.” **OK**
* AspectContentProvider – NullPointerException in inputChanged **OK**
* Изменить иконки на свободные (взять из org.eclipse.jdt.ui.ISharedImages) **OK**
* Default packages **OK**
* Случай нескольких instead для одной инструкции **OK**
* Возможность внедрять несколько раз подряд без перепоиска **OK**
* Сохранение списка точек внедрения **OK**
  + Путь к целевому проекту
  + Пути к контейнерам тв
  + Список внедренных тв

Предполагаемый сценарий работы с Eclipse Aspect.Java Plug-in:

* Выбор целевого проекта
* Добавление интересующих аспектов: проекты, пакеты, jars – from workspace **OK**.
  + TODO отображать пакеты **OK**
  + TODO возможность временно убрать/добавить из списка правила/аспекты/пакеты (checkboxes – изн-но выбраны все) **NO**
* Выбор точек внедрения
  + TODO взаимодействие с исходными кодами **OK**
  + TODO добавить пакеты **OK**
* “Weave Aspects” button -> выбор поддиректории целевого проекта для сохранения
* Сохранение проекта с внедренными аспектами и списка внедренных точек xml
  + TODO добавить в joinpoints.xml информацию о директориях/jar-файлах с аспектами (или создать отдельный aspects.xml) **OK**

**13.** Bugs in Aspect.Java:

* Case when target project has no output (example: imported plug-in)
* Add target project classpath entries (all except ‘/bin’) to classpath.txt

Приложение:

* BufferedReader br = **new** BufferedReader(

**new** InputStreamReader(System.*in*));

L11

LINENUMBER 24 L11

NEW BufferedReader

DUP

L12

LINENUMBER 25 L12

NEW InputStreamReader

DUP

GETSTATIC System.in : InputStream

INVOKESPECIAL InputStreamReader.<init>(InputStream) : void

L13

LINENUMBER 24 L13

INVOKESPECIAL BufferedReader.<init>(Reader) : void

ASTORE 3

* *someMeth*(s = str.replace(".", ".."), 5);

L2

LINENUMBER 9 L2

ALOAD 2

LDC "."

LDC ".."

INVOKEVIRTUAL java/lang/String.replace(Ljava/lang/CharSequence;Ljava/lang/CharSequence;)Ljava/lang/String;

DUP

ASTORE 1

LDC 5.0

INVOKESTATIC pkg/Target.someMeth(Ljava/lang/String;D)[[D

POP

* getPreviousInstr(arg instanceof FieldInsnNode? arg : null);

L2

LINENUMBER 58 L2

ALOAD 0: this

ALOAD 1: arg

INSTANCEOF FieldInsnNode

IFEQ L3

ALOAD 1: arg

GOTO L4

L3

FRAME SAME1 SingleThreadWeaver

ACONST\_NULL

L4

FRAME FULL [SingleThreadWeaver AbstractInsnNode] [SingleThreadWeaver AbstractInsnNode]

INVOKESPECIAL SingleThreadWeaver.getPreviousInstr(AbstractInsnNode) : AbstractInsnNode

* *method*(**new** **int**[]{0, 1, 2});

L33

LINENUMBER 114 L33

ICONST\_3

NEWARRAY T\_INT

DUP

ICONST\_1

ICONST\_1

IASTORE

DUP

ICONST\_2

ICONST\_2

IASTORE

INVOKESTATIC edu/spsu/aj/weaver/SingleThreadWeaver.method([I)V

* *method*(**this**.l++);

L38

LINENUMBER 123 L38

ALOAD 0

DUP

GETFIELD edu/spsu/aj/weaver/SingleThreadWeaver.l : I

DUP\_X1

ICONST\_1

IADD

PUTFIELD edu/spsu/aj/weaver/SingleThreadWeaver.l : I

INVOKESTATIC edu/spsu/aj/weaver/SingleThreadWeaver.method(I)V

* Integer j = 0;

*method*(j++);

L45

LINENUMBER 131 L45

ALOAD 4

DUP

INVOKEVIRTUAL java/lang/Integer.intValue()I

ICONST\_1

IADD

INVOKESTATIC java/lang/Integer.valueOf(I)Ljava/lang/Integer;

ASTORE 4

INVOKESTATIC edu/spsu/aj/weaver/SingleThreadWeaver.method(Ljava/lang/Integer;)V

* + DUP ?
  + ?