Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik III
Prof. Dr. A. B. Cremers

Aspect-Oriented Software Development
Course in the summer semester 2009
Dr. Günter Kniesel, Daniel Speicher

Exercise Sheet 11

Due: Sunday 12.07.2009, 23:59:59 via SVN

For help, contact aosd-staff@lists.iai.uni-bonn.de (staff only) or aosd-course@lists.iai.uni-bonn.de (staff and participants).

Please start working on the exercises early enough so that you can contact us in time in case of problems. Don't expect us to be available during weekend!

Exercise 1: *"AspectJ/LogicAJ Quickies"* (9 Points)

a) Implement, if possible, a pointcut that matches every execution of `System.out.println(..)`.

b) Your program has a field `public int field`. Name all primitive pointcuts that match a `field++` statement and explain why they do.

c) Is it possible to simulate the `cflow`-pointcut (obviously without using `cflow` or `cflowbelow`)? If yes, describe how you would do it (no need to actually implement it).

d) What is the value of the variable of an `args`-pointcut combined conjunctively with a `set`-pointcut?

e) Which types are returned by the `this`- and `target`-pointcuts of the method calls in `boo()`?

```
class X{
      void foo(){};
}
class Y{
void boo(){
      final X x = new X();
      x.foo();
      boo2();
      new Y(){
            void boo(){
                  x.foo();
                  boo2();
             }
      }.boo();
}
void boo2(){}}
```

f) Method `m` is defined in class `C`. Is there a way to find out if method `m` is called by another method in `C`?
g) Can a pointcut match join points in advices? If it does, is there a way to prohibit this?
h) Which annotation would you prefer and what are your reasons: `@TransmissionOperation` or `@TransmissionLock`?
i) Is the value of `obj` in the following two pointcuts always the same when they match the same `foo(..)`?
```
pointcut p1(Object obj): call(* foo(..)) && target(obj);
pointcut p2(Object obj): execution(* foo(..)) && this(obj);
```
Give a counter-example if this is not the case.
j) Is this a legal pointcut in LogicAJ? Justify your answer.
method(* ?type.foo(..))
&& method(* ?type.boo(..))
k) Explain the `?After` in the LogicAJ substring predicate pointcut
```
subString(?String:string, ?Start:int, ?Length:int, ?After:int,
?Sub:string)
```

Exercise 2: *"Precedence I"* (3 Points)

In your repository you will find the project **ES11_E02_Precedence**. The program calls a begin method that tells you that your working day just started and an end method that tells you that your working day is over.

a) Now your employer got a new face recognition system. Please write an advice in the aspect `A1` that adds "`Identity checked`" to the console. The advice should be applied after `begin()` is executed.
b) The new system also automatically starts your computer. Write another advice, this time in an `aspect A2` that adds "`Your computer is starting`" to the console. Use the same pointcut as in a) to make sure that the same joinpoints are captured.
c) Make sure that the computer only starts after the identity has been checked.

Exercise 3: *"Precedence II"* (4 Points)

We now take the program of exercise 2 and add some more functionality.

a) Write an aspect `DoorHandeling` that adds "`The door is open`" to the console after the call of `begin()` and "`The door is closed`" after the call of `end()`.
b) Write an aspect `LightHandling` that adds "`The light is switched on`" after the call of `begin()` and "`The light is switched off`" to the console after the call of `end()`.
c) After you implemented the aspects try if it is possible to use `precedence` to make sure that the advices are executed in an order so that outputs in the console looks like this:

```
"Begin of your office day."
"The door is open."
"The light is switched on."
"End of your office day."
"The light is switched off."
"The door is closed."
```

What problems do you encounter?

Exercise 4: *"Precedence III"* (4 Points)

To see how the precedence can affect the correctness of the code you now will try several possibilities. In the project **ES11_E04_PrecedenceIII** you will find a class `Base` that does nothing except that it accesses the field `f`. The class `Base` is the common joinpoint for the introductions, and the access to `f` is the common joinpoint for the advices of the `Counter` and `Getter` aspect.
The `counter` aspect introduces the counter `f_count` for the field f to the class `Base` and increments it before each access to `f`.
The `getter` aspect introduces the getter method `getf` for the field `f` to the class `Base` and enforces its use instead of direct accesses to `f`.

- Try the possible orderings of the aspects using precedence.
- Write down for each order, which output you would expect.
- Does AspectJ give you in each case the output you expected?
- In case of differences between the expected and actual output, try to explain what could be the cause of the difference. Do you see any semantic interference, weaving interference or even both?