Program specialization and metaprogramming

Laboratory work No. 10

Using aspects for logging applications in AspectJ

1. Aims

To get acquainted with the advanced concepts of Aspect Oriented programming.

2. Tasks

Analyse and learn how to use advices, joint-points and point-cuts in aspect oriented programs.

3. Work

Download a sample AspectJ application from course Moodle page. Run Eclipse IDE, create AspectJ project and add source files.

1. Now we will define an example that uses a hierarchy of aspects, favoring the reuse of aspects.

o Create the `AbstractLogging` abstract aspect and define in it:

- a `loggedMethods` abstract pointcut
- two advices that print the value of the `thisJoinPoint` variable (contains information about the current join point) before and after the `loggedMethods` pointcut
- note that this class only defines what behavior is associated with logging join points, without defining them, which should be done by a sub-aspect

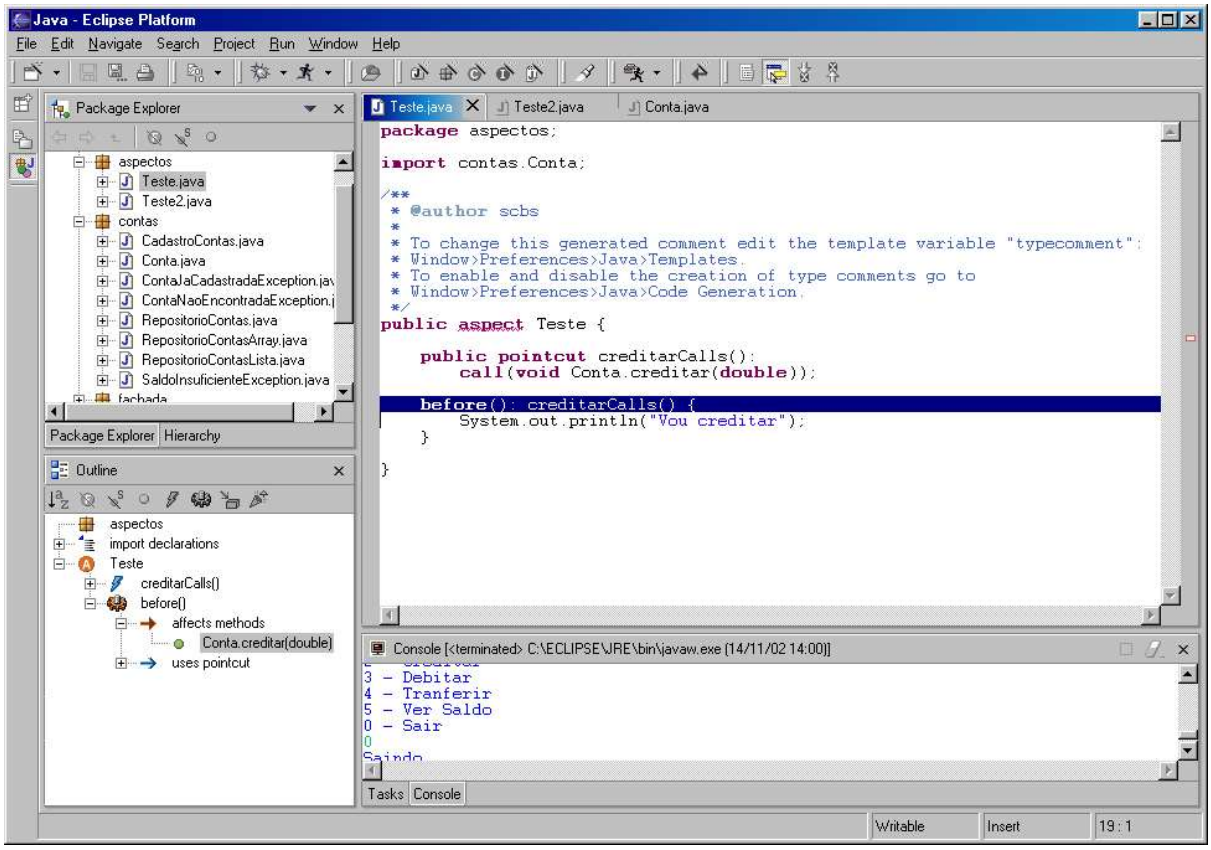
o Now create the abstract aspect `AbstractCallLogging` that inherits from `AbstractLogging` and define the pointcut `loggedMethods` in it, identifying calls to any methods with any type of return, any list of parameters, and any classes.

- Note that this aspect defines the pointcut `loggedMethods` by identifying several join points. The idea is that a third aspect will restrict these join points to a specific system.

o Create the concrete aspect `BancoCallLogging` that inherits from `AbstractCallLogging` and re- define the pointcut `loggedMethods` identifying all the join points identified by the "inherited" pointcut, however restricting it to join points whose target of calls are classes of the `accounts`, `facade`, and `gui` packages.

- Note that this aspect only defines which classes or packages should be logged

o With this structure we can define other abstract sub-aspects of `AbstractCallLogging` to log class method calls from another system. In addition, the hierarchy also allows you to define other sub - aspects of `AbstractLogging`, for example, to define that method executions (`AbstractExecutionLogging`) or access to attributes (`AbstractFieldLogging`) will be logged.



2. Still using the banking application project, we are going to write an aspect to check properties about our application.

o define an aspect called `SystemValidate` that should check if any private attributes are changed outside the set method for it. If so, a warning must be reported at compile time.

o add the aspect in the list of files, generate the new version of the system and observe if in any point of the application the property is violated.

3. Probably one of the points in the system where the property is violated is in the class constructor. As the coding standard suggests this type of programming, change the defined aspect to disregard assignments to private attributes within the class constructor. Generate the application and observe the warnings.

3. Once again, it refines the aspect so as not to consider methods of the business-data interface (`RepositorioContas`), since their implementations (`RepositorioContasArray` and `RepositorioContasLista`) manipulate the attributes in their methods.

4. Regenerate the application. Note that only methods of the `Account` class harm the property. Modify them to change the balance attribute using a `set`, but private, method. Run the application and note that there are no more warnings.

5. Now let's define an aspect to check for a precondition.

o define an aspect to raise the `IllegalArgumentException` exception if any method is called having an invalid reference (`null`) anywhere in the application.

o comment out line 10 of the `Banco` class, add the aspect to the file list, generate and run a new version of the system.

o note the number of classes that your change affects.

6. Implement another precondition that checks whether after making a credit the amount credited has been added to the balance.

7. Run the application and note the results printed in program output window.

What are the program results?

What are the advantages of using aspects for logging?

4. Report

Prepare a report of your work in this lab using a standard word processing application. The report is a single chapter of your final semester report. Final semester report will have to be uploaded to course Moodle page.

The content of report:

1. Title
2. Aims and tasks
3. Work done, described in steps and illustrated by screenshots and written or modified source code. Provide comments what you have done and what were the results (program outputs). Present answers to the questions given in the lab description.
4. Conclusions – what you have learned?