Component-Based Software Engineering	**Exercise Sheet No. 2**
Dipl.-Medieninf. Christian Piechnick	Software Technology Group
INF 2080	Institute for SMT
christian.piechnick@tu-dresden.de	Department of Computer Science
	Technische Universität Dresden
	01062 Dresden

Invasive Software Composition

Task 1: Aspect-oriented Programming (AOP)

1a) Describe the terms *Scattering* and *Tangling*. Why are both effects bad?

1b) What does cross-cutting mean w.r.t. AOP?

1c) Describe the terms *Pointcut*, *Join Point*, *Advice* and *Weaving*. Give an example.

1d) Where is the differenc between static and dynamic Join Points?

1e) Where is the difference between static and dynamic weaving?

1f) Describe the component model, composition technique and composition language of AspectJ.

Task 2: Generic Programming with Generic Components

2a) What is a generic component?

2b) What makes a language *fully generic*? What is consequence?

2c) What is a hook? What is slot? Where is the difference?

2d) Compare Aspect-oriented programming and Generic Programming with Generic Components? Where are the commonalities and differentialities?

Task 3: Invasive Software Composition (ISC)

3a) Try to explain the concept of Invasive Software Composition (ISC)!

3b) What is the component model, composition technique and composition language of ISC?

3c) What composition operators does ISC offer?

3d) Compare ISC to Aspect-oriented Programming! What are the differences? What are the commonalities?

Task 4: AspectJ

In this task your are using AspectJ to write some simple aspects and learn how aspect-oriented software is implemented. AspectJ is a aspect-oriented extension to java. For eclipse there are the AspectJ Development tools available.
In our example we have a very tiny library for representing the structure of houses. The library offers an interface to construct houses. In the current version any combination of ComplexHouseParts is possible (e.g., a level can contain hallways, which can contain levels etc.). Because we want to use our library for any kind of domain, we do not want to put the consistency checking code within the base program. Rather we want to use customer-specific aspects, which can be woven into to base program. Furthermore, rooms can either be clean or not clean. They can be cleaned by calling the `setClean(boolean clean)` method. House parts can be entered and visited. Visiting a house part means entering the part itself and visiting all the subparts. Use AspectJ to implement to following features.

1. Install the AspectJ Development Tools AJDT. (Update site: `http://download.eclipse.org/tools/ajdt/45/dev/update`)
2. Download the source of the core project.
3. Edit the core project to solve the task (add the aspects).

AJDT Website: `http://www.eclipse.org/ajdt/`
AspectJ Documentation: `https://eclipse.org/aspectj/doc/released/progguide/starting.html`
AspectJ Getting Started Tutorials: `http://o7planning.org/web/fe/default/en/document/18642/java-aspect-oriented-programming-tutorial-with-aspectj`

4a) Define an Aspect that logs (prints to the console) when the programm starts and stops running.

4b) Define an Aspect which restricts that levels cannot contain other levels.

4c) Define an Aspect which marks a bathroom as not clean, when it is entered.

4d) Define an Aspect which prints a warning when a room is entered which is not cleaned.

4e) Define an Aspect which prohibits (e.g., throws an exception) any person entering a bathroom which is not cleaned.

4f) In the initialization Script (`buildSimpleHouse()`), a house with a "private room" is created. Define an aspects which prohibits any other person that the owner of the house, to enter the private room.