1. **Wozu dient ein Aspekt? Welche Bestandteile hat er?**

Die Idee eines Aspekts ist die Implementierung eines Crosscutting Concerns, wie bspw. Logging, Security, Caching. Durch einen Aspekt muss ein Anliegen wie Security nicht im Code verstreut implementiert werden, sondern die Implementierung kann an einer zentralen Stelle (im Aspekt) erfolgen.

Ein Aspekt beinhaltet:

* Weaving Rules: „definiert“ WAS (Advice) WOHIN (Pointcut) gewebt wird
* Pointcuts
* Advices
* Statische Crosscutting Concerns (Struktur)

1. **Was ist ein Aspekt Weaver? Welche Strategien gibt es und wodurch unterscheiden sich diese?**

Ein Aspekt Weaver verwebt den Programmcode/Bytecode an den definierten Verwebungspunkten (Joinpoints) mit dem Aspektcode.

Es gibt drei Strategien:

* Source Weaving: Der Programmcode wird mit dem Aspektcode vor der Übersetzung verwebt.
* Binary Weaving: Der Aspektcode wird mit dem bereits übersetzten Bytecode verwebt.
* Load-time Weaving: Der Aspektcode wird mit dem übersetzten Bytecode erst dann verwoben, wenn dieser geladen wird.

1. **Welche 2 logischen Schritte werden in AOP durchgeführt?**

* Weben: Individuelle Belange werden aufgrund der Weaving Regeln kombiniert
* Umwandeln: Die gewebte Information wird in ausführbaren Code umgewandelt

1. **Welche technischen Komponenten hat AspectJ? Nenne weitere AOP Implementierungen?**

* Eigener Aspekt Compiler (ajc – AspectJ Compiler)
* Tool Unterstützung für Eclipse (AspectJ Development Tools)
* Aspekt-Definition: durch eigene Keywords (früher) oder durch Annotationen

Weitere Implementierungen: SpringAOP (Java, .NET), PostSharp (.NET)

1. **Nenne 2 Vorteile und Nachteile von AOP!**

Nachteile: Lernaufwand für AOP, wenige Tools

Vorteile: Sauberere Implementierung (Crosscutting Concerns sind nicht verteilt sondern zentral an einer Stelle definiert), Code wiederverwendbarer

1. **Nenne die wichtigsten Begriffe in AOP!**

**Join Point**: Möglicher Ausführungspunkt für einen Advice

**Pointcut**: Eine Auswahl von **Join** **Points**

**Statisches** **Crosscutting**: Inter-type declarations (betrifft statische Struktur), weave-time declaration

**Dynamisches** **Crosscutting**: before/after/around **advices** (betrifft Methoden)

**Advice**: Hinzuzufügendes/alternatives Verhalten (vor/nach/um JoinPoint)

**Aspect**: zentrale Einheit, welche die Weaving Regeln beinhaltet (**Pointcuts**, **Adivces** (dynamisch), **statisches** **Crosscutting**)

1. **Welche Arten von Advices gibt es? Welche Bestandteile hat ein Advice?**

* Before: Wird vor JoinPoint ausgeführt
* After: Wird nach JoinPoint ausgeführt
* Around: Umgibt JoinPoint oder ersetzt ihn durch alternatives Verhalten

Bestandteile:

* Advice Deklaration: Before, After, Around
* Pointcut Spezifikation: Welche Join Points vom Advice adressiert werden
* Advice Body: Welcher Code beim Join Point ausgeführt wird

1. **Was ist der Unterschied zwischen einem Call und einem Execution Pointcut? Kann überall wo ein Call Pointcut angewendet wird, auch ein Execution Pointcut angewendet werden (Grund)?**

Call Pointcut: Der Advice wird beim Aufruf der Methode verwoben (beim Aufrufenden). Dadurch steht auch Information über den Aufrufenden zur Verfügung.

Execution Pointcut: Der Advice wird in der Methode selbst verwoben.

Nein, beispielsweise können System-Klassen oder Third Party Klassen nicht durch execution Pointcuts verändert werden. Es kann aber sehr wohl auf ihre Aufrufe durch call Pointcuts reagiert werden, da das Verweben im Aufrufer stattfindet.

1. **Was unterscheidet einen Advice von einer Methode? Welche Unterschiede haben before/after Advices zu around Advices?**

* Advices haben keine Namen
* Advices können nicht direkt aufgerufen werden
* Advices haben keine Access Specifier
* Advices haben zusätzliche Variablen (thisJoinPoint, thisJoinPointStaticPart, thisEnclosingJoinPointStaticPart)
* Before/After Advices haben keinen Rückgabewert
* Around Advices können die Methode proceed() aufrufen, um die Zielmethode aufzurufen.
* Around Advices können Rückgabewert haben.

1. **Welche Methoden gibt es für Strukturänderungen?**

* Inter-type declaration (Deklaration für andere Typen, Typhierarchie Änderungen)
* Weave-time error and warning declarations (Findet join-points, zeigt Fehler und Warnungen auf)
* Exception softening (Exceptions werden abgefangen und auf eine Crosscutting Art und Weise verarbeitet)