**Aspect Oriented Programming**

Aspects helps in implementation of crosscutting concerns such as transaction, logging which are not central to business logic without cluttering the code core to its functionality

It helps in avoiding repetitive code

Additional behavior is added using advice to the existing code

**Aspect:** The class which implements the JEE application cross-cutting concerns(transaction, logger etc) is known as the aspect. Annotated with @Aspect. This class contains advices

**Weaving:** The process of linking Aspects with an Advised Object. It can be done at load time, compile time or at runtime time. Spring AOP does weaving at runtime.

**Advice:** The job which is meant to be done by an Aspect or it can be defined as the action taken by the Aspect at a particular point.

There are five types of Advice

1) Before

Runs before the advised method(method on which advice runs) is invoked

2)After

Runs after the advised method completes regardless of the outcome, whether successful or not.

3) AfterReturning

Runs after the advised method successfully completes ie without any runtime exceptions.

4)AfterThrowing : Runs after the advised method throws a Runtime Exception

5)Around

This is the strongest advice among all the advice since it wraps around and runs before and after the advised method

**JoinPoint**

An application has thousands of opportunities or points to apply Advice. These points are known as join points. Spring AOP currently supports only method execution join points

**Pointcut**

Since it is not feasible to apply advice at every point of the code, therefore, the selected join points where advice is finally applied are known as the Pointcut. Often you specify these pointcuts using explicit class and method names or through regular expressions that define a matching class and method name patterns. It helps in reduction of repeating code by writing once and use at multiple points

Different pointcuts Regular expression

Applies after returning on only add

@AfterReturning(value = "execution(\* Calculator.add(..))", returning = "result")

Applies after returning on all methods in class Calculator

@AfterReturning(value = "execution(\* Calculator.\*(..))", returning = "result")

@AfterReturning(value = "execution(\* Calculator.add(int, int)), returning = "result"”

@AfterReturning(value = "execution(public \*\*(..)), returning = "result"”

@AfterReturning(value = "execution(public \*\*(..)), returning = "result"”

applied on all setters

execution(public Employee.set\*(..))

@Pointcut("within(com.company.dao.MyDao)")

@Pointcut("within(com.baeldung..\*)")