-:- Spring Aop :-

AOP :- Aspect Oriented Programming

→ This concept is used to add External services to Buisness logic without any modification in Buisness Logics.

*** AOP is a Cross - Cutting - Concern.
" It means Adding / Removing External services should not effect the project."

* Here Buisness Logic means Project and External Service Means : Log4J, Security, UnitTesting, Encode & Decode, Cyprography, Email, OTP, Captcha etc..
In simple all external services.

*** → Terminolies Used in AOP :-

(i) Aspect :- It is a class which represent external services.

(ii) Advice :- It is a method defined in Aspect provider service Logic.

(iii) Point Cut :- It is a expression which selects buisness method that needs advice, ~~pro~~ but never provides info. like what advice to be connected.

(IV) JoinPoint :- Combination of Advice and Pointcut. It links buisness methods with one or more advices. (method)

(V) Target :- It is a (Pure) buisness class object.

(VI) Weaving :- It is a process name done by weaver which "Adds advices to Target based on JoinPoints."

(VII) Proxy :- Final Output of Weaver which holds complete Buisness Logic and Selected Advices Logic.

** Weaver is a sub-component (contains) of Spring Container.

Buisness Class

External Service

Point Cut
Select b. methods which need Advice

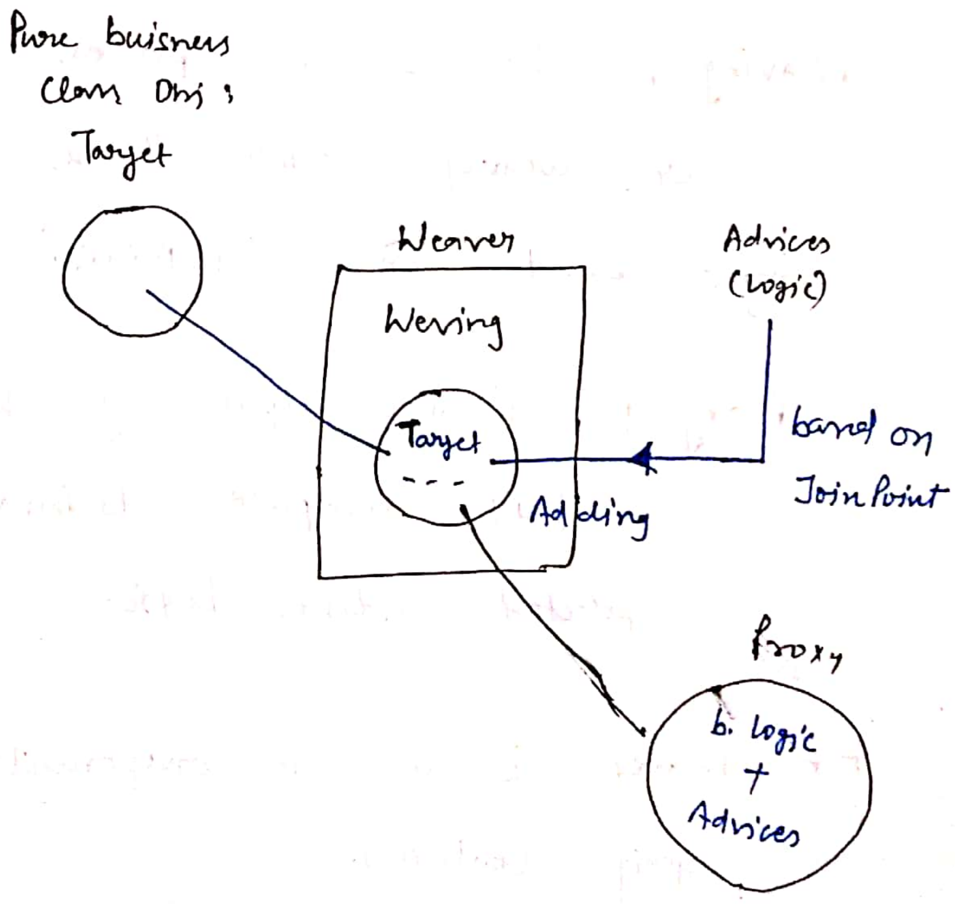
```
class EmpController
{
 ✓ String save() { }
 × String delete () { }
 ✓ String export () { }
}
```

Join Point

buisness methods

```
class Sample  Aspect
{
 void sendMails() { }
 void genOtp() { }
 void findLogs() { }
}
```

Advice (Service Logic) for Aspect



-:- Types of Advices in AOP :-

- Advices are 5 types in Spring AOP. Those are.

1. Before Advice
2. After Advice
3. Around Advice
4. After Returning Advice
5. After throwing Advice

1. Before Advice :- In this case Advice method is executed before buisness method.

-- execution Order --

#i. Advice Method

#ii. Buisness Method

2. After Advice :- In this case Advice Method is exeuted after buisness method completed.

--- execution Order --

(i) Buisness Method

ii. Advice Method

3. Around Advice :- In this case 1st part of Advice is exeuted before Buisness Method and 2nd part of advice is called after Buisness Method.

--- execution Order ---

#i. Advice Method (1st part)

ii Buisness Method

iii Advice Method (2nd part)

*** Use code "proceed ()" to move from Advice to buisness Method.

4. After Returning Advice : In this case only on successful execution of Buisness method advice is executed or not.

-- execution order --

#i Buisness Method (executed Succesfully)

#ii Advice Method

5. After Throwing Advice :- In this case only if buisness method is throwing exception (failure in execution) then advice is executed or not.

---execution Order --

#i Buisness Method (throwing Exception)

#ii Advice Method

:- PointCut Expression :-

PointCut Expressions are used to select business method which needs advices.

Expression Format -

$\boxed{\text{AS} \quad \text{RT} \quad \text{PACK . CLS . METH (PRMTR)}}$

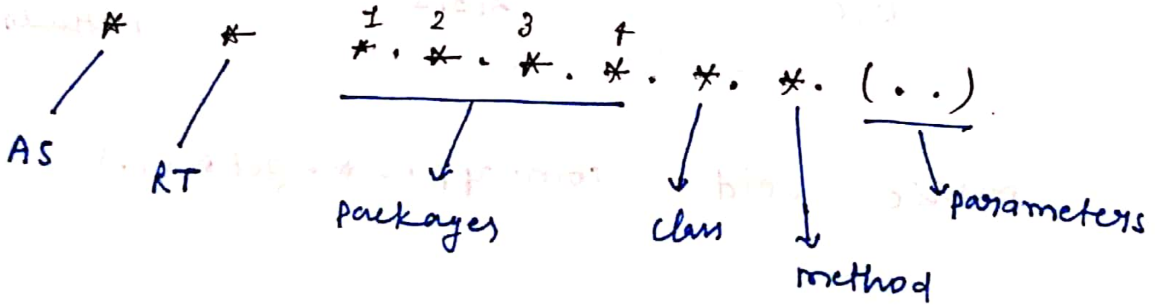
AS = Access Specifier	RT = Return Type
Pack = Package	CLS = Class Name
METH = Method Name	PRMTR = Parameter

a. If package is given as "com.one.abc" then goto "abc" package and select classes from this package only.

b. If two dots [..] provided at package then it indicates current package and all its sub packages are selected.

e.g. com.app.. → means ~~app~~ package classes and all its sub package classes.

e.g.



i.e. 4th level subpackages selected. (all its classes)

→ above expression indicates goto 4th level package and select classes ~~and~~ inside that package.

→ Do not choose super package classes or sub package classes.

*→ Spring AOP uses vendor "AspectJ" which has provided Annotations -

- @Aspect
- @Pointcut ("execution (—)")
- @Before
- @After
- @Around
- @AfterReturning
- @AfterThrowing

and Spring AOP will be enabled using @EnableAspectJAutoProxy

* In pom.xml add below dependencies for AOP programming. Those are -

```
<dependency>
        <groupId> org.aspectj </groupId>
        <artifactId> aspectjrt </artifactId>
        <version>   1.8.7   </version>
</dependency>

<dependency>
        <groupId> org.aspectj </groupId>
        <artifactId> aspectjweaver </~~groupId~~ artifactId>
        version — 1.8.7
```

*** Special Cases :-

Q1. We can define multiple methods with same type of advice in Aspect and those are executed if matched with pointcut expression.

Ans In case of multiple methods matched with Same type then execution order is taken by spring container using Unicode system order.

Unicode System.

Character	Unicode
0	48
A	65
_ (underscore)	95
a	97

Example.

```
@Before ( "p1()")
public void get ()
{ sysout ( "from get Advice");
}
```

```
@Before ("p1()")
public void find()
{
    sysout ("from Advice method");
}

@ Before ("p1()")
public void -3a()
{
    S.O.P. (" ~~aaa~~ from aaa Advice");
}
```

→ If matched then exection order— find(), -3a(), ~~[illegible]~~ get()

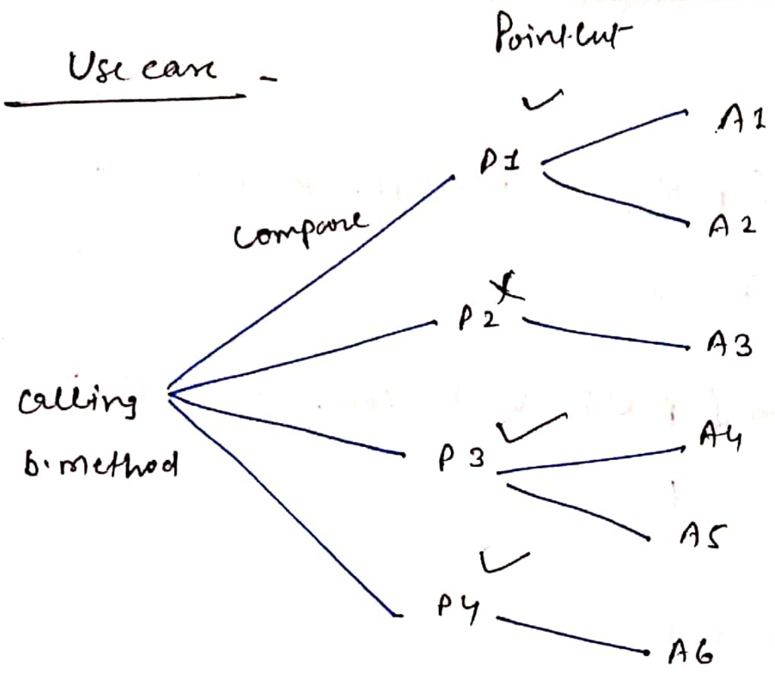
<u>Special case -2</u> → In Aspect we <u>can</u> <u>define multiple pointcuts</u> and b. method will be compared with all pointcuts if matched then all its connected advices will be executed.

<u>code</u> buisness class —

```
package com.app.service;
// import

@Service
public class ProductService
{   public void getData()
        { SOP. (" b.method");
}       }
```

Use case -



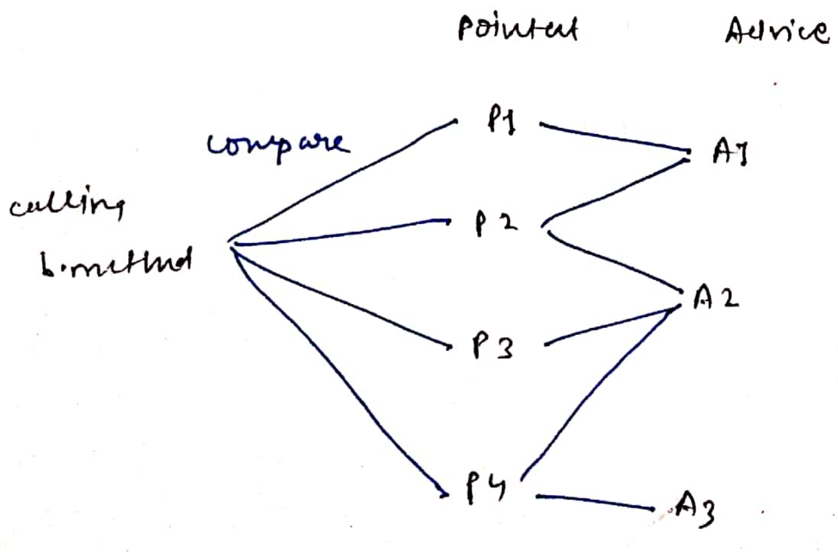
A1, A2, A4, A5, A6 executed in method naming order.

Special Case - 3 :-

* Multiple pointcuts can be connected to logical And (&&), logical OR ("||").

→ At join point code looks as

P1() && P2() || P3() _ _ _.



Logical Operator :

AND(&&) OR (||)

P1	P2	AND	OR
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	F

F = Not Matchin / Not executed

T = matchin / executed