

Spring AOP

Rajeev Gupta
MTech CS Java
Trainer &
Consultant



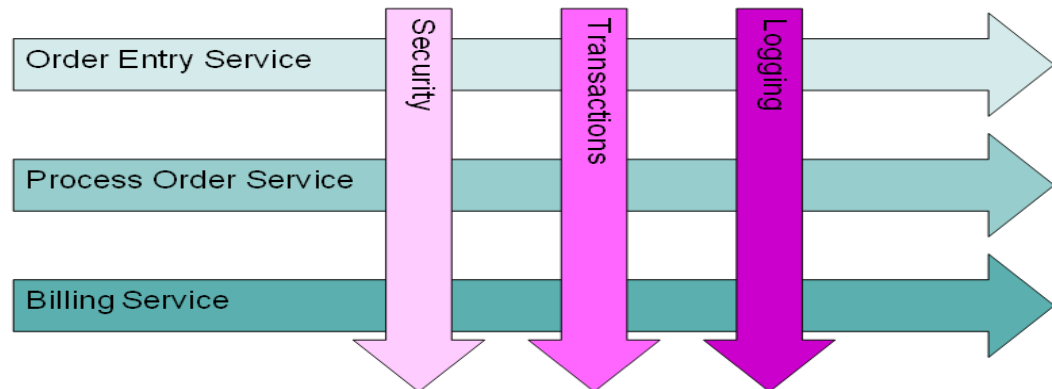
Introduction to AOP

❖ What is AOP?

- AOP is a style of programming, mainly good in separating cross cutting concerns

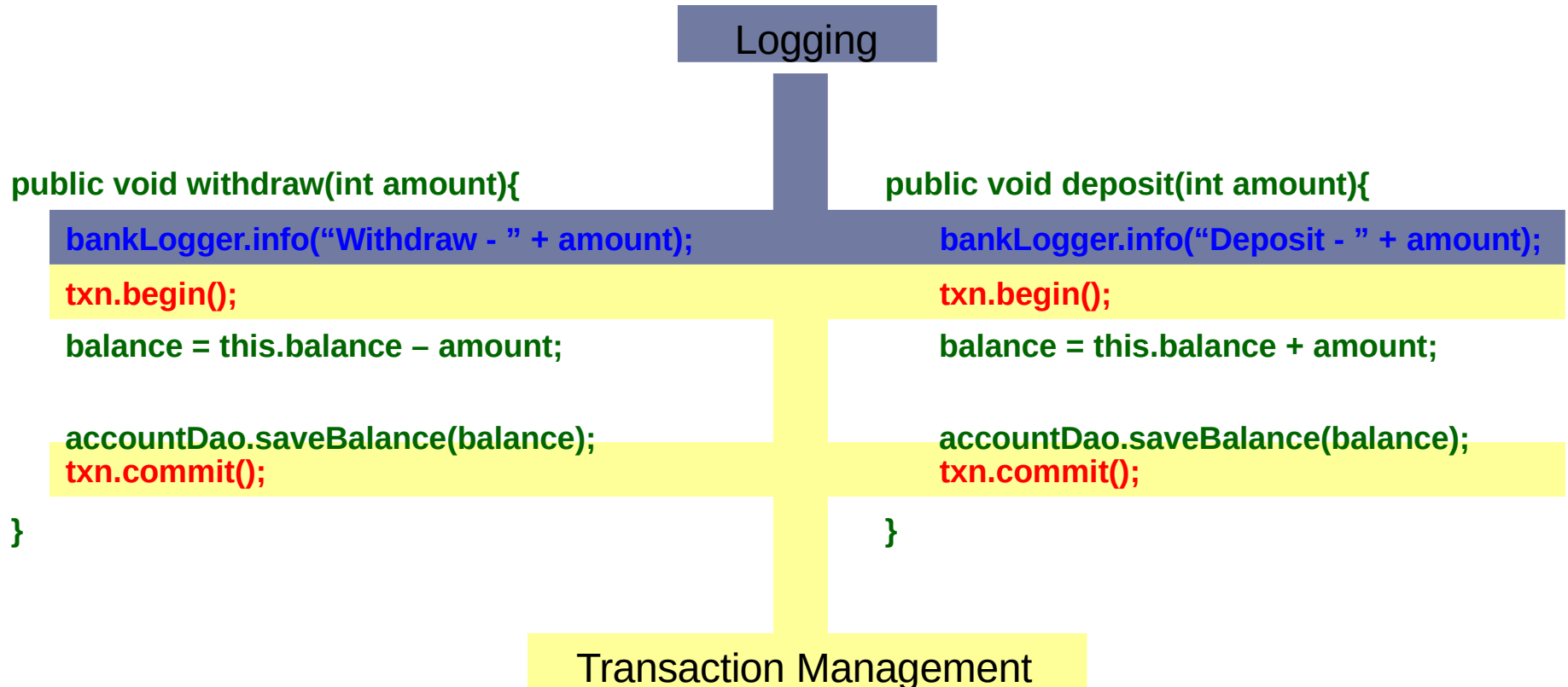
❖ How AOP works?

- Achieved usages Proxy design Pattern to separate CCC's from actual code
- Cross Cutting Concern ?
- Extra code mixed with the actual code is called CCC's
Extra code mixed with code lead to maintenance issues
- **Logging**
- **validations**
- **Auditing**
- **Security**



Crosscutting Concerns

- Eg: Banking Application

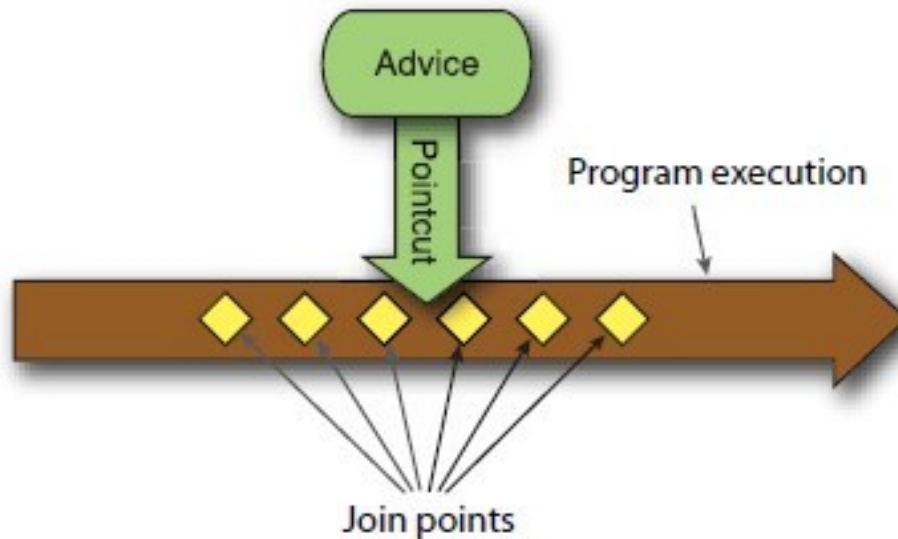


Understanding AOP terminology

Join points : are the options on the menu and
pointcuts : are the items you select

Aspect = Advices + Point Cut

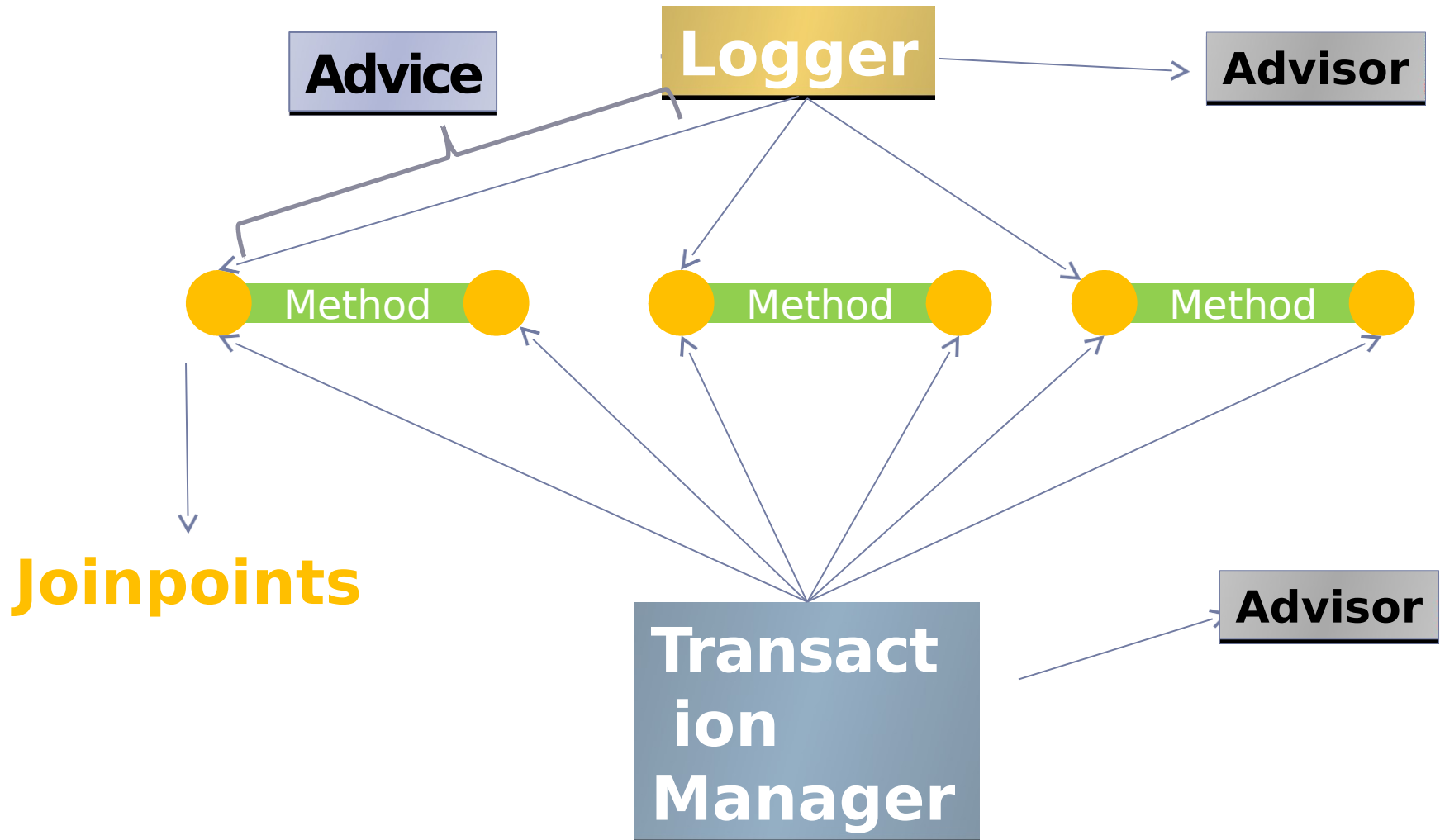
Aspect means
what (extra logic) and where it need to be applied (point cut)



AOP- Definitions.

- **Aspect**
- **Joinpoint**
- **Advice**
- **Pointcut**
- **Target**
- **Object**
- **AOP Proxy**
- **Weaving**

AOP- Definitions.



Advice Types

❑ Before Advice



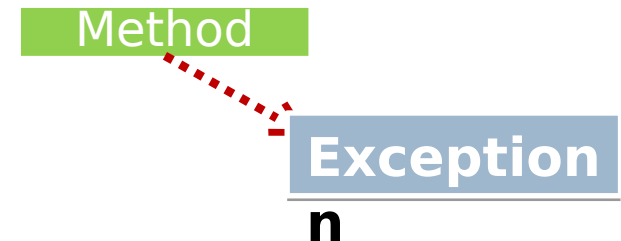
❑ After returning Advice



❑ Around Advice



❑ Throws Advice



n

WEAVING

- Weaving is the process of applying aspects to a target object to create a new proxied object. The aspects are woven into the target object at the specified join points. The weaving can take place at several points in the target object's lifetime:

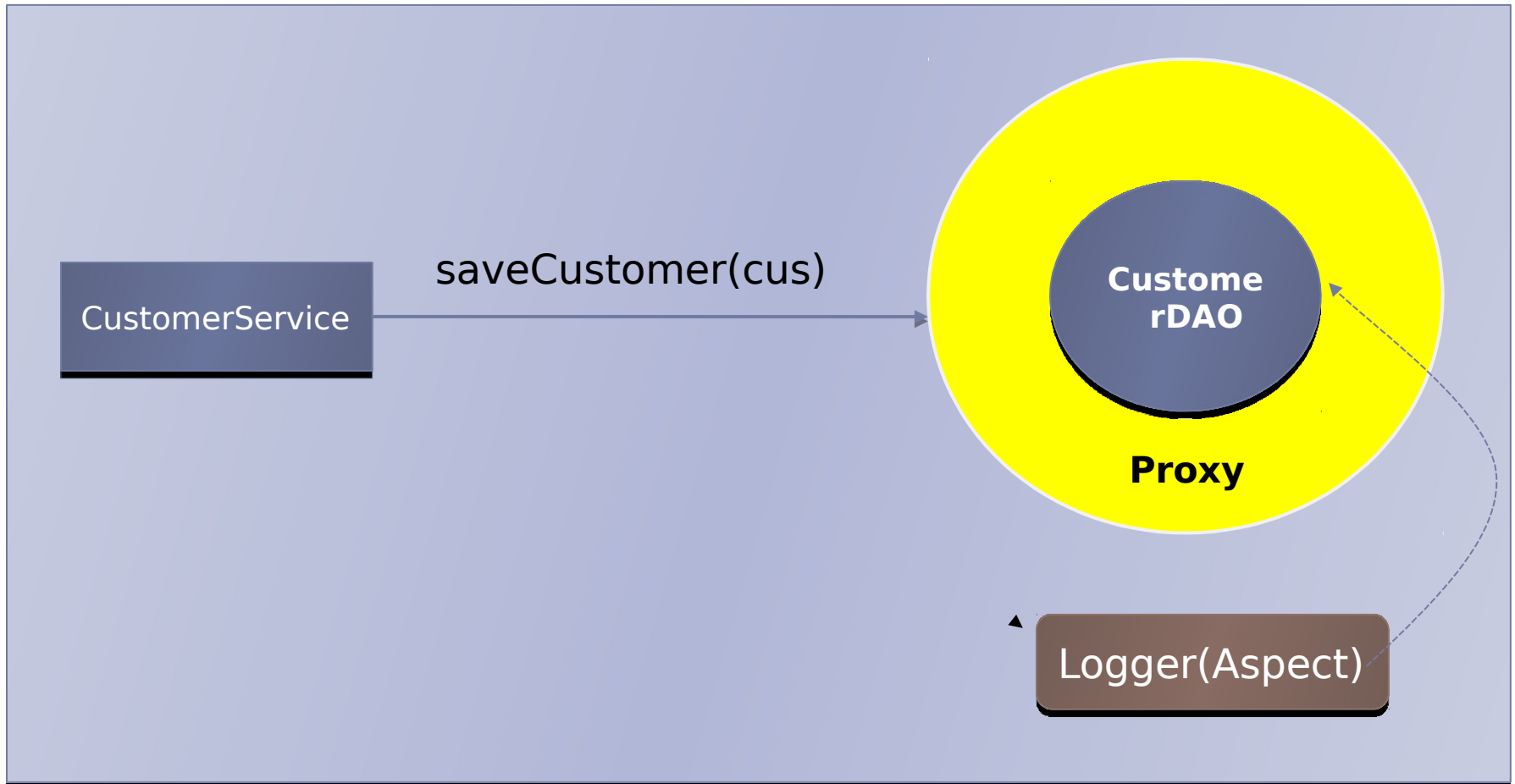
- **Compile time** —Aspects are woven in when the target class is compiled.
- **Classload time** —Aspects are woven in when the target class is loaded

into the JVM.

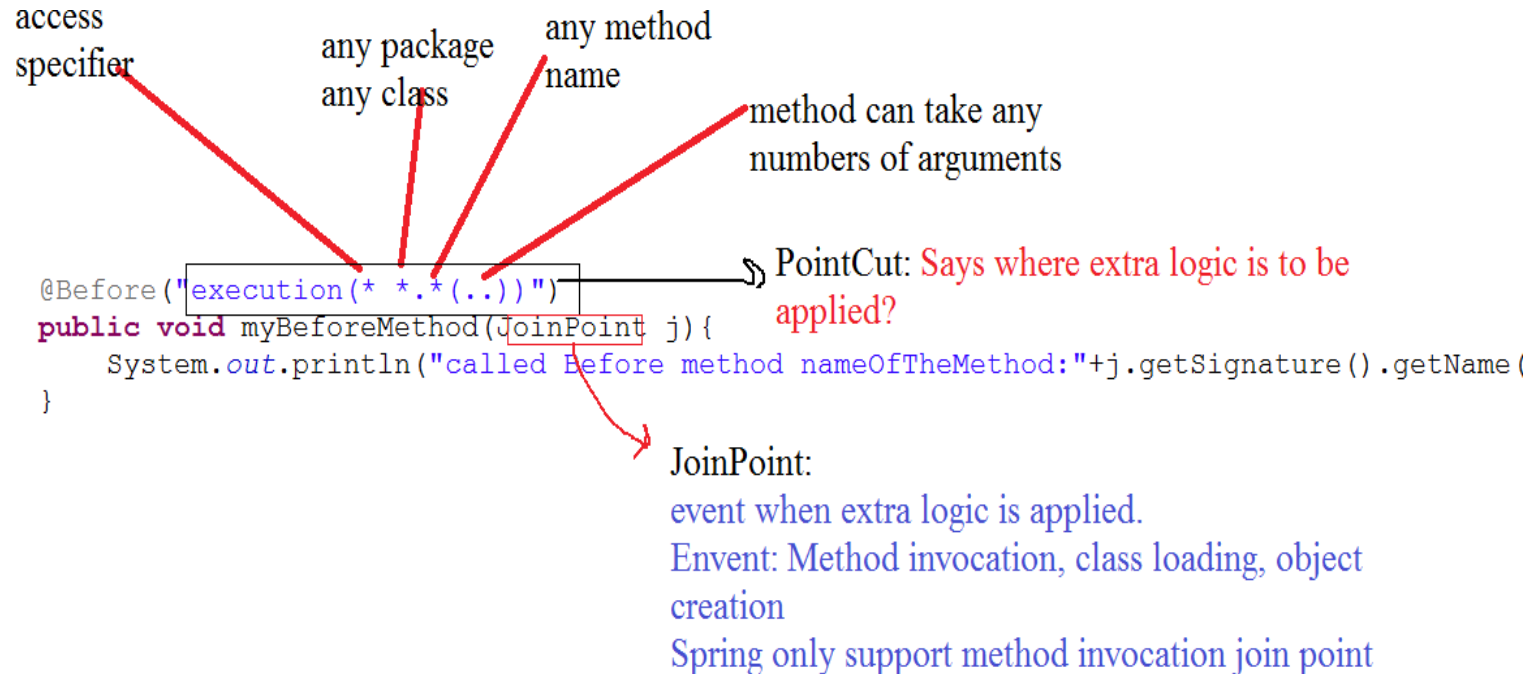
- **Runtime** —Aspects are woven in sometime during the execution of the application. Typically, an AOP container will dynamically generate a proxy object that will delegate to the target object while weaving in the aspects.



AOP Weaving



Understanding Point Cut wildcard



Understanding Point Cut wildcard

- execution is the most used designator

execution(modifiers-pattern? ret-type-pattern declaring-type-pattern?name-pattern(param-pattern) throws-pattern?)

Optional modifier
(public, protected, private)

Return type
* indicates any type

Optional type
(package and class)
ending in .* includes all classes in package
ending in ..* includes classes in sub-packages

Method name
Use . to connect with type
May contain, or just be *

Parameters
Comma separated list
(..) means any parameters
(*) means one param any type
(int, *) = a int and one other type

Optional throws
Comma separated
Optionally include





Any questions?

