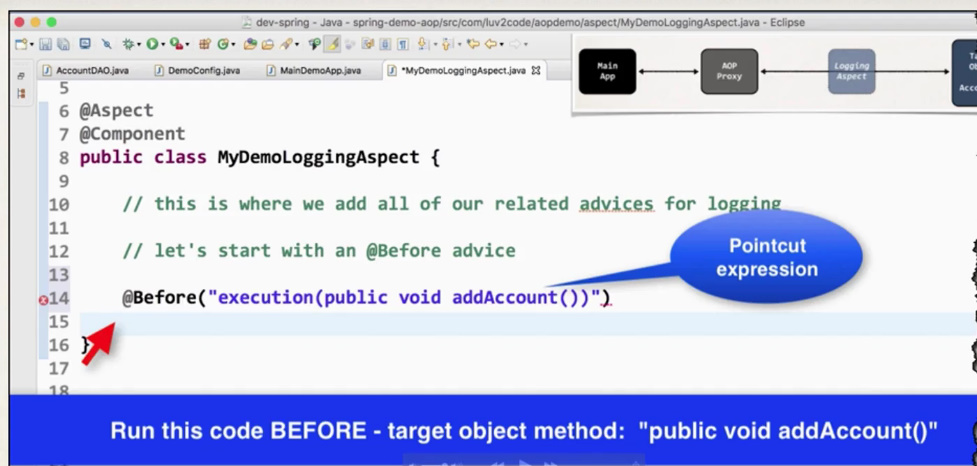
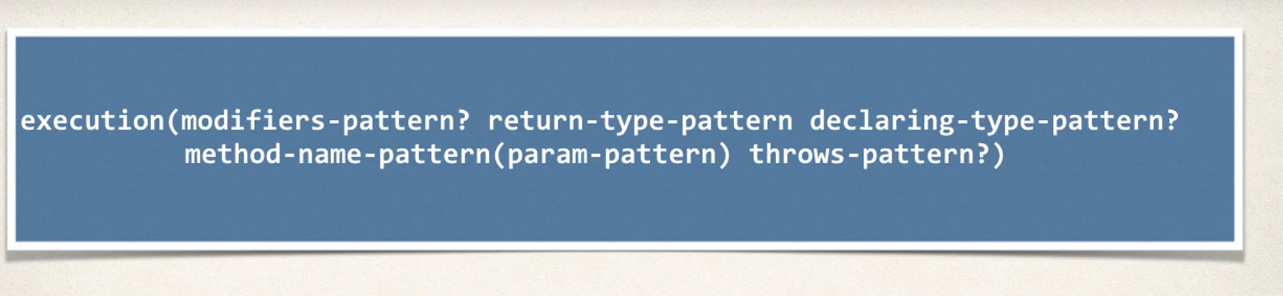
**General**



* Iata de ce spunem ca un PointCut este o expresie predicat care spune unde codul trebuie sa fie executat, adica pentru ce metoda.
* Forma generala pentru execution este:





* **Spring AOP suporta doar metode publice!**
* declaring-type-pattern? – numele clasei unde metoda e cu full path si punem dupa ea .numeMetoda
* throws-pattern? – arata daca metoda are throws dupa ea, adica daca obliga alta metoda ce o apeleaza sa gestioneze vreo exceptie
* pattern cu ? sunt optionale
* Pentru a folosi declaring-type-pattern, vom scrie FullPath.numeClasa.numeMetoda

Asa:

@Before("execution(public void com.java.org.DAO.AccountDAO.addAccount())")

* Daca nu speficam clasa, atunci metoda va fi executata in orice clasa, conteaza doar ca prototipul metodei sa fie ca cel specificat

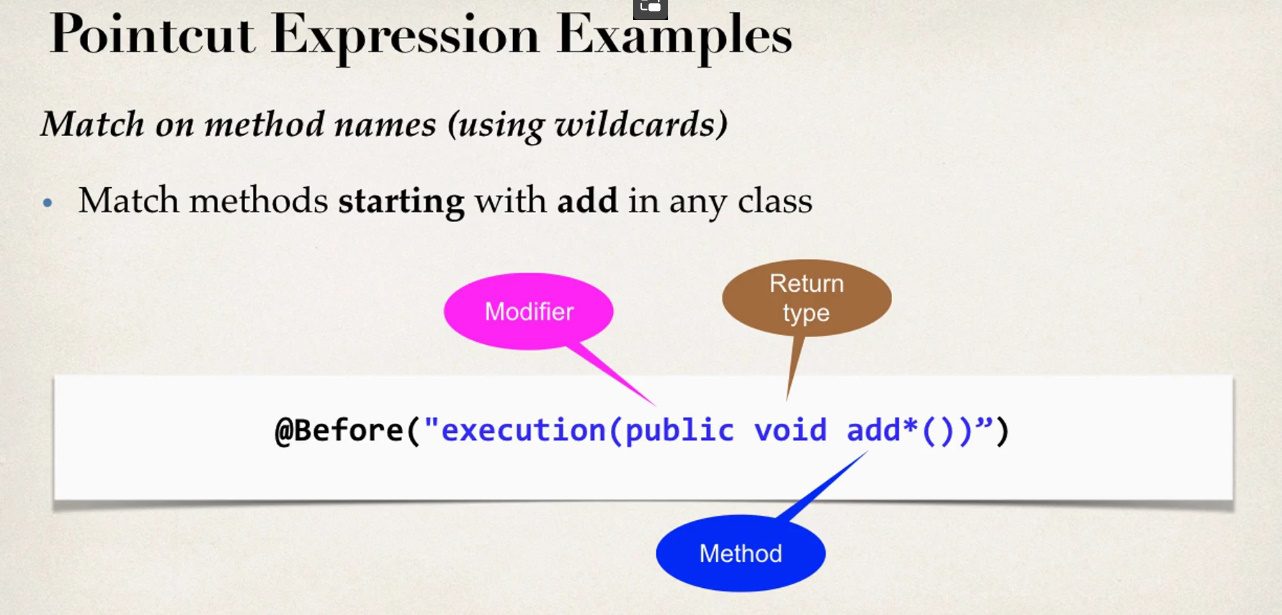
**Cum functioneaza**

Odata ce avem un Aspect, orice metoda ce va fi executata in programul nostru, se va asigura ca nu cumva sa fie pusa undeva prin @Aspect, si daca este, executa si ce este in Advice. Apoi, odata ce se gaseste, se asigura sa execute toate Advices gasite.

Deci, cand abea ruleaza, Proxy AOP va cauta sa nu fie vreun @Before corespunzator, dupa return cauta sa nu fie vreun @AfterReturning si tot asa. Orice metoda, ca vrea sau nu, trece prin acest Proxy.

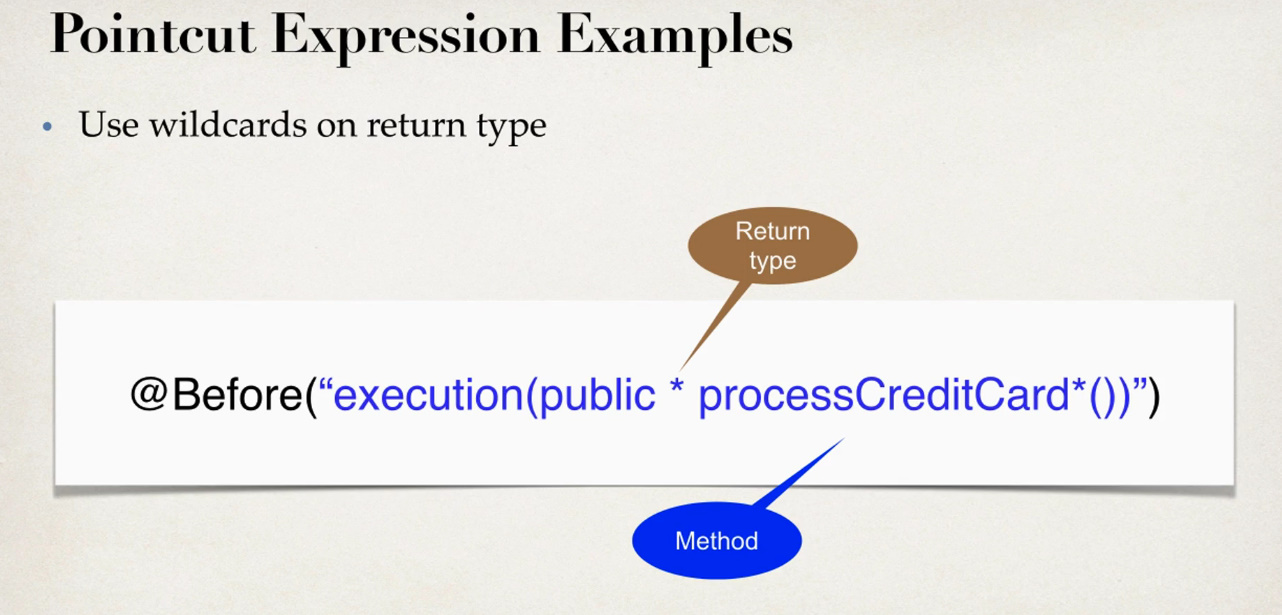
**Steluta(\*)**

* Daca vrem sa spunem ca toate metodele ce incep de ex, cu add, sa fie spionate, folosim add\*, sau orice alt cuvant + \*



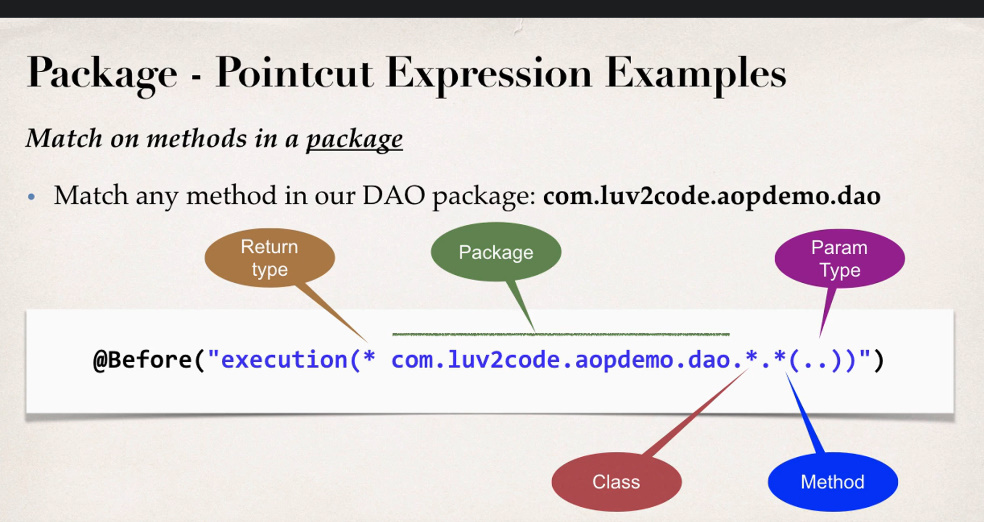
Aici, orice metoda din orice clasa ce incepe cu add va chema advice din aspect

* \* poate fi folosit si la return, pentru a spune ca nu conteaza ce returneaza metoda:



* Daca vrem sa specificam ca intr-un package sa se ia o metoda din orice clasa, punem fullpath.\*.nume()
* Daca orice metoda din orice clasa din package

fullpath.\*.\*()



**Parameters**

() – metoda fara argumente

(\*) – metoda cu un argument de orice tip. Unul, nu 0!

(..) – metoda cu oricate argumente de orice tip, adica 0 sau oricate oricare

Cand vom scrie ce fel de parametru vrem sa avem, daca e vreun obiect creat de noi, trebuie folosit fullpath

@Before("execution(public void add\*(com.java.org.DAO.AccountDAO))")

Nu trebuie numele parametrului, asa cum numele nu conteaza.

* Daca vrem sa avem un parametru de tip Account si apoi dupa el oricate altele, inclusiv si 0, facem asa:

@Before("execution(\* add\*(com.java.org.aopapp.Account, ..))")

putem pune si \* in loc de .. ca sa aratam ca mai vrem un parametru de orice tip