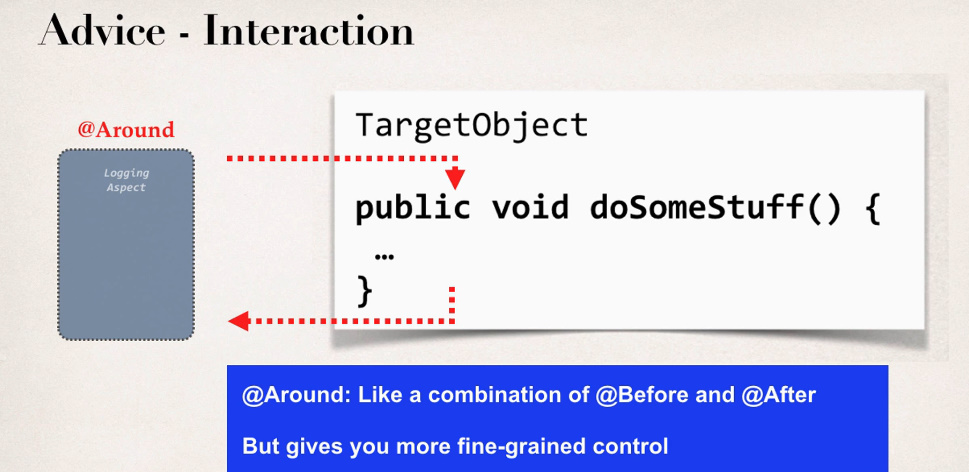
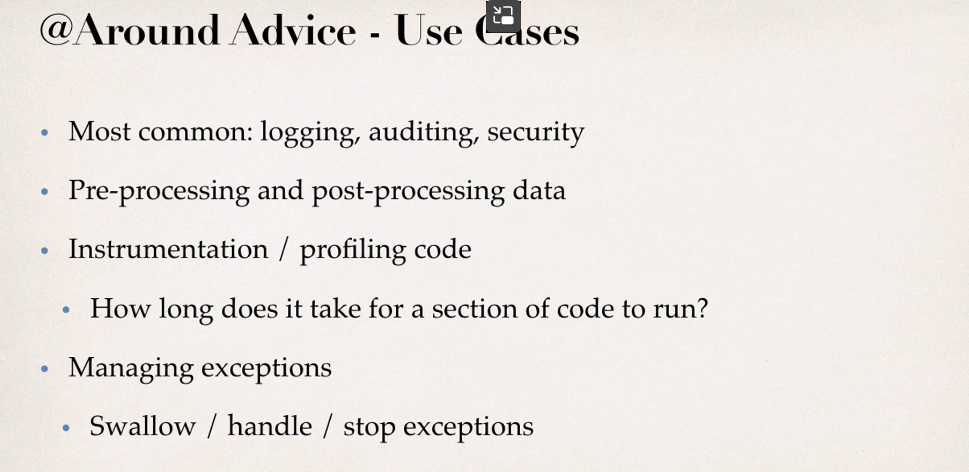
* Il folosim atat pentru a executa vreun cod pana la executarea metodei, cat si dupa executarea ei.



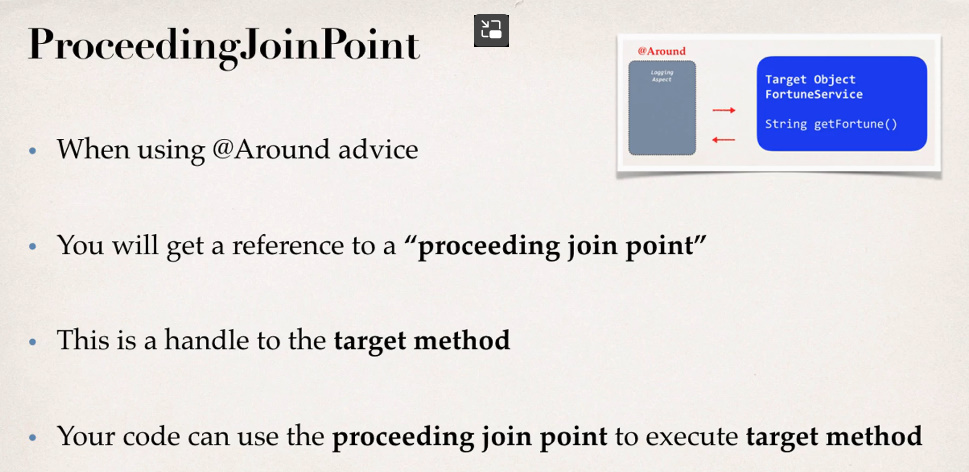
* Este ca o combinatie dintre @Before si @After, doar ca ne ofera mai mult control asupra metodei.

**La ce e util**

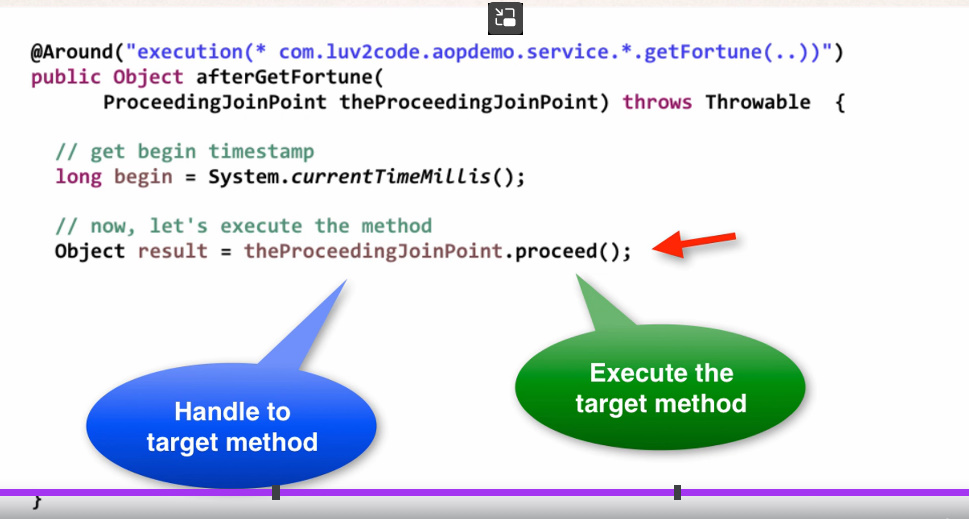


* Deci, el are ceva special ce restul @Advice nu au, si anume ca poate opri propagarea exceptiilor inapoi.

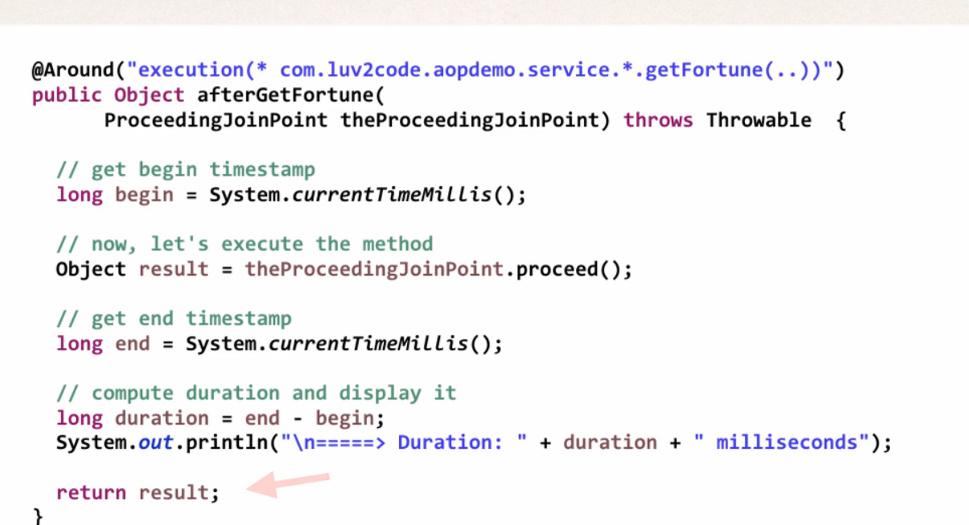
**Proceeding JoinPoint**



**Implementare**



ProceedingJoinPoint creaza ca un fel de referinta catre acea metoda, si metoda sa proceed() o executa direct chiar in Advice, adica in @Around, si deci returneaza ceea ce metoda are ca return.



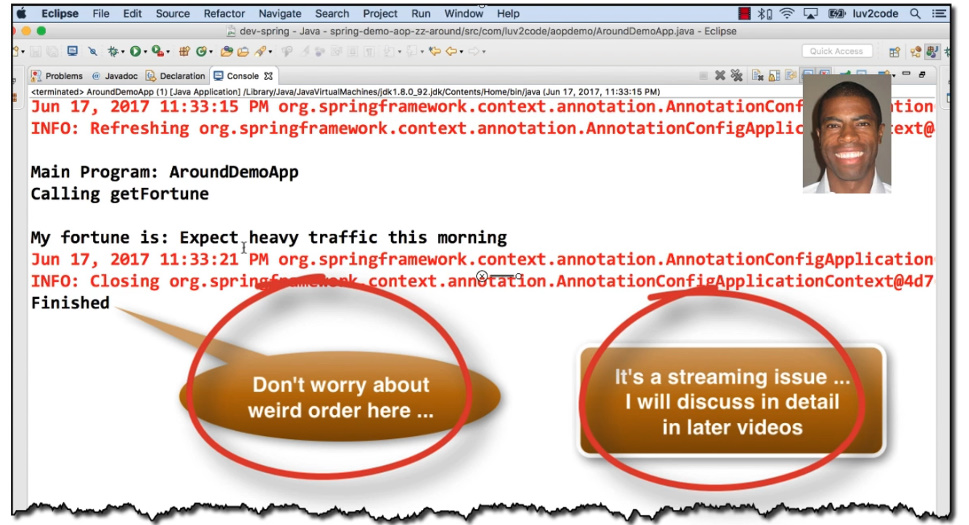
acest return se va asigura sa returneze rezultatul dorit acolo unde metoda a fost apelata.

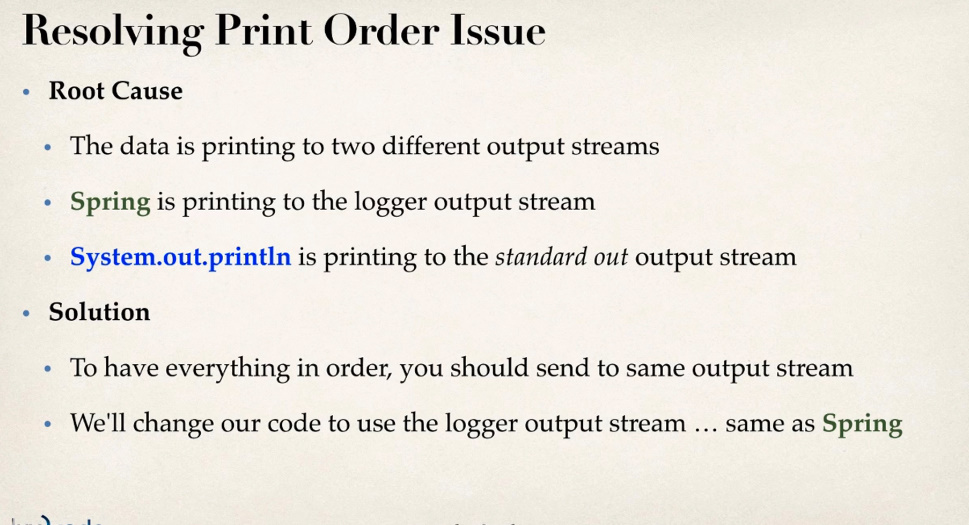
Around preia si responsabilitatea de a returna obiectul in locul metodei, ceea ce @AfterReturning nu face

* Cel mai bine e sa avem pentru metoda proceed() un bloc try catch, nu un throws, ca sa putem cumva sa ne ocupam de exceptia ce ar putea veni din metoda executata spre care e referinta creata.

**Order problem**

Se poate intampla ca mesajele ce sunt afisate in consola sa nu respecte ordinea intocmai.

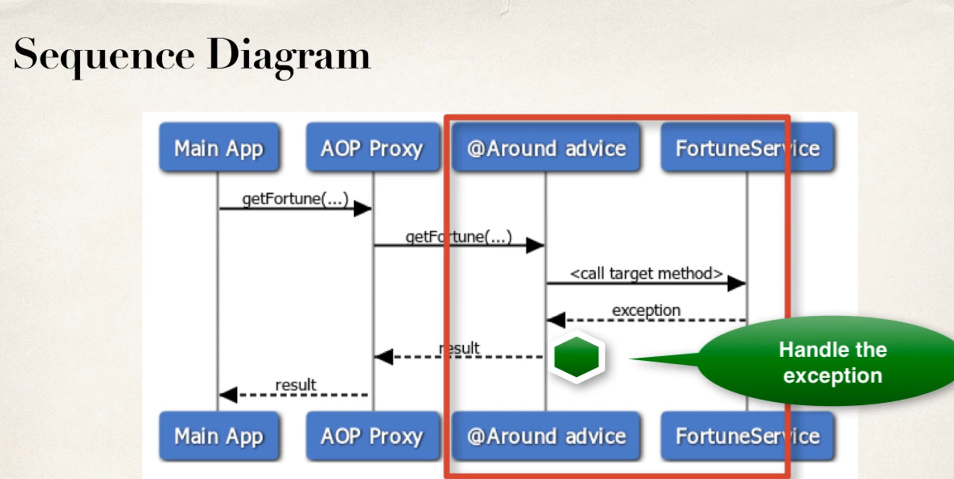




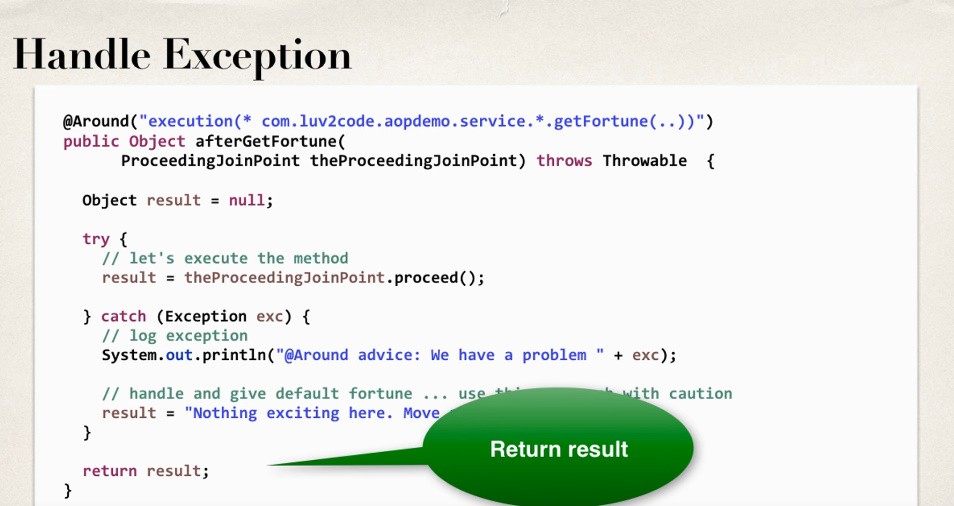
Deci, problema e ca Spring foloseste obiectul logger pentru a afisa mesaje,si deci logger output stream, dar System.out.println foloseste standard output stream, de aceea ele se pot incurca si sa nu fie ordinea buna. Ele doar nu sunt sincronizate

* Solutia este de a nu mai folosi System.out.println, ci sa cream un obiect Logger si sa folosim metoda .info() a lui.

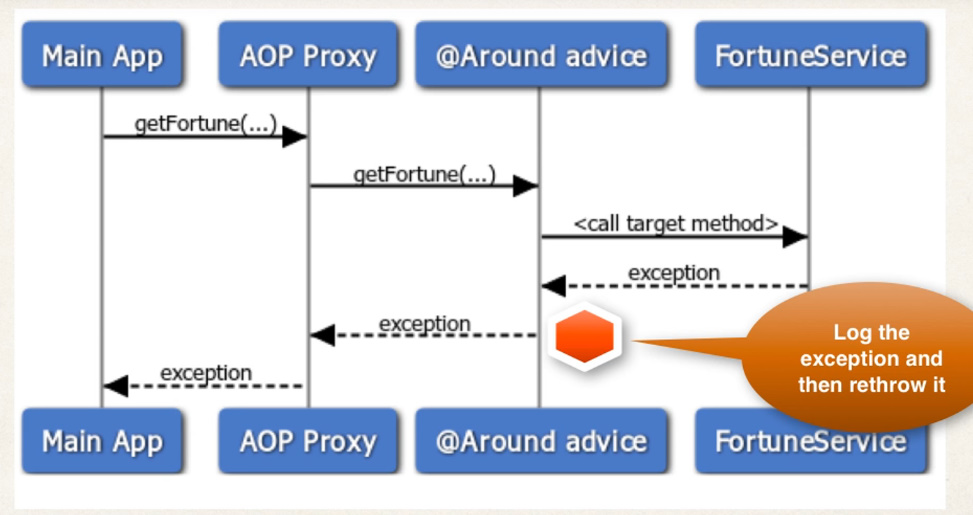
**Handle Exception**



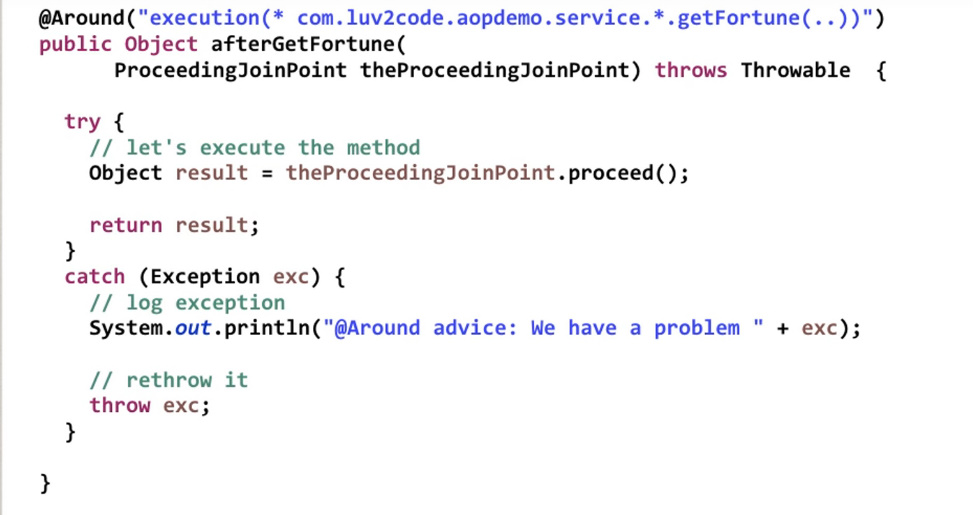
Deci, metoda trece in AOP Proxy, ajunge la @Around, se executa apoi din FortuneService, si daca apare exceptie, ea ajunge la @Around inapoi, se face ceva cu ea acolo, si rezultatul ce trebuie returnat se returneaza de catre @Around

* Ideea e ca in @Around, la aparitia exceptiei, vom decide ce sa facem cu ea si ce sa returnam, asa incat Main App nici nu va sti ca a aparut vreo exceptie.
* Cel mai bine e sa avem pentru metoda proceed() un bloc try catch, nu un throws, ca sa putem cumva sa ne ocupam de exceptia ce ar putea veni de la metoda executata prin referinta.
* 

**Rethrowing exception**



Mare schimbare nu va fi:



Pur si simplu, daca se trece cu bine de metoda proceed(), returnam rezultatul, daca nu, aruncam exceptia aparuta, si return nu mai ajunge niciodata.

Partea cea mai interesanta e ca in main nu vom avea nevoie sa punem throws Throwable, deoarece propriu zis metoda getFortune() nu are nici-un throws, ci arunca un CheckedException, dar in Proxy, adica in @Around, se arunca un obiect de tip Exception, si mai are si throws Throwable, care in mod normal ar obliga metoda apelata sa se ocupe de metoda asta, dar main nu stie de existenta lui Proxy, deci nici nu are de unde sti ca el ar arunca asa exceptie. Deci, cand se arunca exceptia de tip Exception la main, de ea se ocupa compilatorul.