CS544 EA

Spring

Aspect Oriented Programming

Aspect Oriented Programming

- Aspect Oriented Programming is a way to provide separation of concerns, creating modularity
  - Certain features are hard to cleanly separate
  - Certain things you need all the time (like logging)
  - Certain code you need before and after (like Transactions)
- These are what AOP calls **cross cutting concerns**
  - You need it across the application

Cross Cutting Concerns

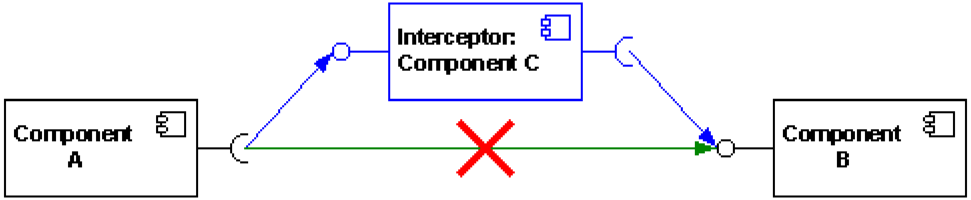
- Cross Cutting Concerns are usually scattered or tangled throughout your code

- Logging is scattered
  - Single thing, copy / pasted everywhere
- Transactions are tangled
  - A part before, a part afterwards every time you use the DB

- How can you **write code like this once** and re-use everywhere?

Interceptor

- AOP with Spring uses the **interceptor pattern**



By UlrichAAB - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=6746767

- Because of IoC and DI, Spring can inject something else (a proxy) in between
  - Proxy then calls the desired code (advice) before and / or after

Proxies

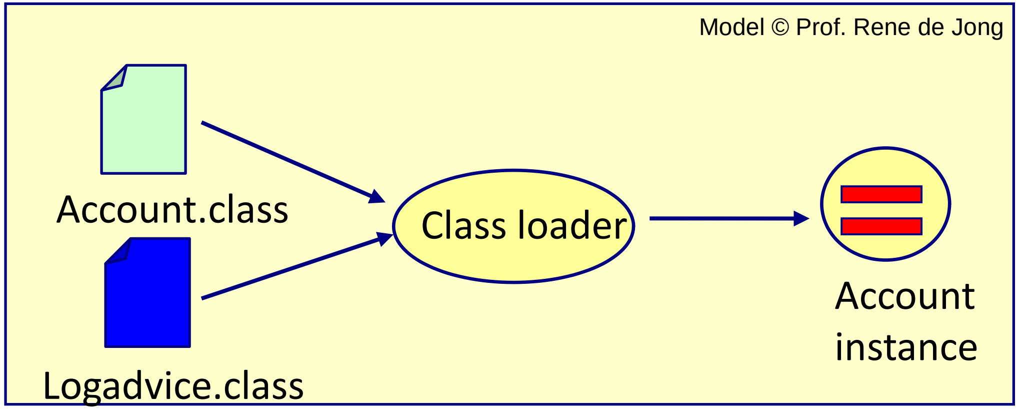
- Spring creates proxies using either JDK dynamic proxies or using CGLIB
  - **JDK proxies** are made by implementing the same interface(s) that your object implements
  - **CGLIB proxies** are made as a subclass
    - Using Objenesis – therefore no super() constructor call!

JDK > CGLIB

- Spring **prefers JDK proxies**
  - Faster to create, and you should **`P2I`** anyway
  - If your class implements 1 or more interfaces
    - Spring will make a JDK proxy based on your interface(s)
    - Leaving out methods not represented on an interface
- If there are no interfaces Spring uses CGLIB
  - You **can force Spring to always use CGLIB** proxies
    - Then you don’t have to write all those interfaces (no P2I)

AOP with ByteCode

- AOP is also possible with **ByteCode Enhancement**
  - Instead of an proxy based interceptor (will not cover in this course)
- The AspectJ project provides tools for this
  - Can be configured to be used with Spring
  - AspectJ combines your .class file with the advice .class file to create a new .class file that contains both


Model © Prof. Rene de Jong
Account.class
Logadvice.class
Class loader
Account
instance