CS544 EA

Spring

AOP: JoinPoint

JoinPoint

- Every advice method can optionally receive as its **first argument** a JoinPoint object
  - Not required, but it is usually nice to have

- The JoinPoint object contains info about the method (point) that will be (or was) joined for this call
  - Remember a Pointcut often specifies many points

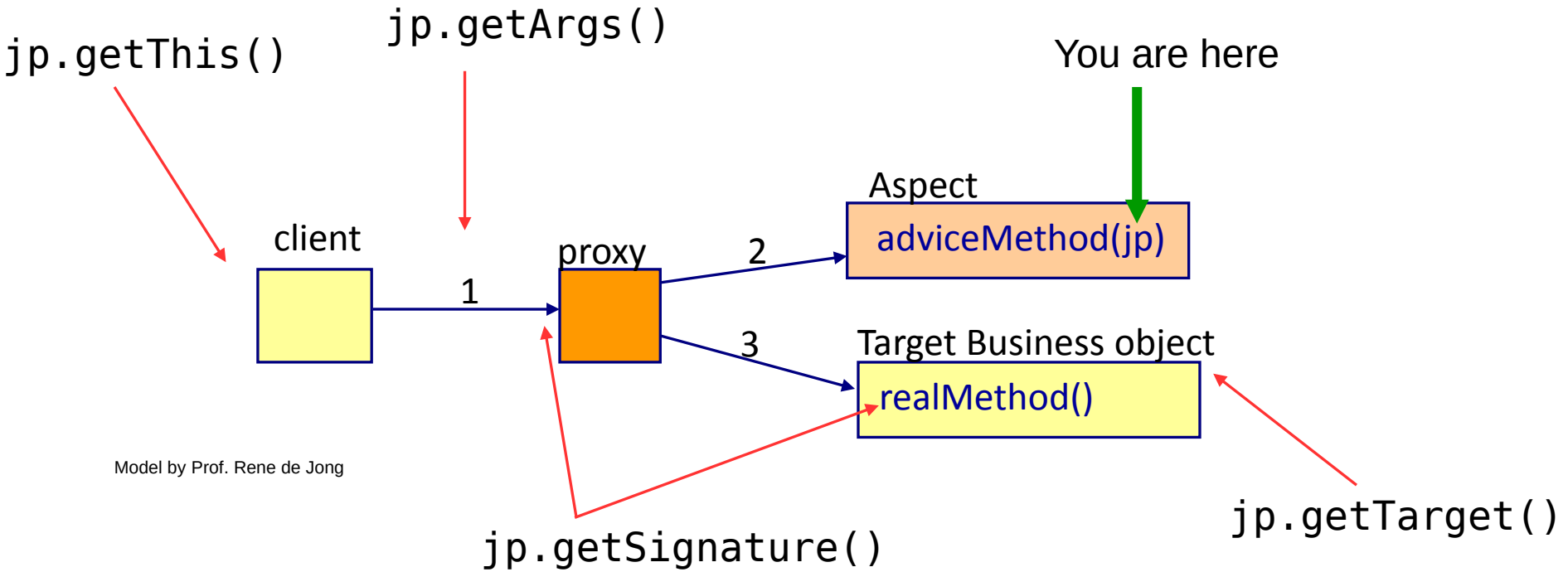
Example Code

```
@Aspect
@Component
public class LogAspect {
     private static final Logger logger = LogManager.getLogger(LogAspect.class.getName());

     @Before("execution(* cs544.spring40.aop.terms.CustomerService.*(..))")
     public void logBefore(JoinPoint joinpoint) {
          logger.warn("About to exec: " + joinpoint.getSignature().getName());
     }

     @After("execution(* cs544.spring40.aop.terms.CustomerService.*(..))")
     public void logAfter(JoinPoint joinpoint) {
          logger.warn("Just execed: " + joinpoint.getSignature().getName());
     }
}
```

JoinPoint API

- The most important methods on a JoinPoint
  - `Signature getSignature()`
    - Returns the method signature of the real method (name, return type, etc)
  - `Object[] getArgs()`
    - Returns the arguments passed to real method as Object[]
  - `Object getTarget()`
    - Returns the Object on which real method was / will be called
  - `Object getThis()`
    - Returns the Object that calls the real method

Example

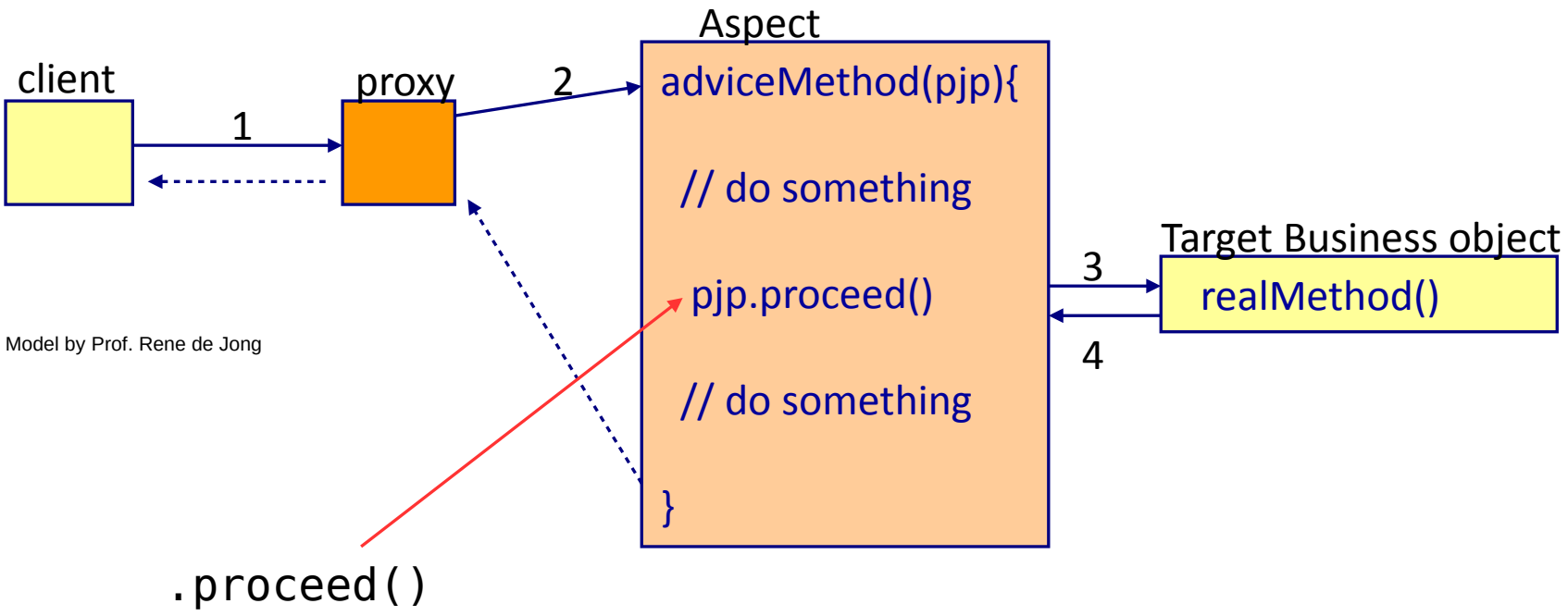

jp.getThis()
jp.getArgs()
You are here
Aspect
adviceMethod(jp)
client
proxy
2
1
3
Target Business object
realMethod()
Model by Prof. Rene de Jong
jp.getSignature()
jp.getTarget()

ProceedingJoinPoint

- ProceedingJoinPoint extends JoinPoint for use inside an @Around advice adding the following overloaded method:
  - `Object proceed()`
  - `Object proceed(Object[] args)`

- Proceed without args causes the real method to be called with the arguments from the client, returning an Object
  - Proceed with args allows you to **give your own version** of the args array

Proceed


client
proxy
1
2
Aspect
adviceMethod(pjp){
// do something
pjp.proceed()
// do something
}
Target Business object
realMethod()
3
4
Model by Prof. Rene de Jong
.proceed()

Not Optional for @Around

- JoinPoint is an optional argument for @Before, @After, @AfterReturning and @AfterThrowing
  - These methods do not need it to function

- ProceedingJoinPoint is **not optional for @Around**
  - Cannot function without it
  - @Around method also has to return Object
  - And declare throws Throwable

Example Code

```
@Around("execution(* cs544.spring41.aop.advices.CustomerService.getName(..))")
public Object around(ProceedingJoinPoint pjp) throws Throwable {
     String m = pjp.getSignature().getName();
     System.out.println("Before " + m);
     Object ret = null;
     try {
          ret = pjp.proceed();
     } catch (Throwable e) {
          e.printStackTrace();
          throw e;
     }
     System.out.println("After " + m + " returned " + ret);
     return ret;
}
```