CS544 EA

Spring

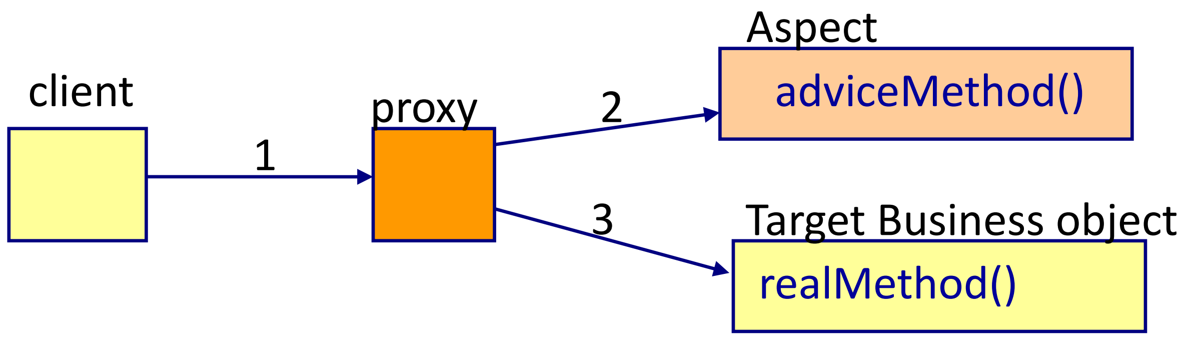
AOP: Types of Advice

5 Types of Advice

- There are **5 types** of advice:
  - @Before
  - @After
  - @AfterReturning (only execs if returns properly)
  - @AfterThrowing (only execs if exception thrown)
  - @Around (single advice method execs before and after)

@Before

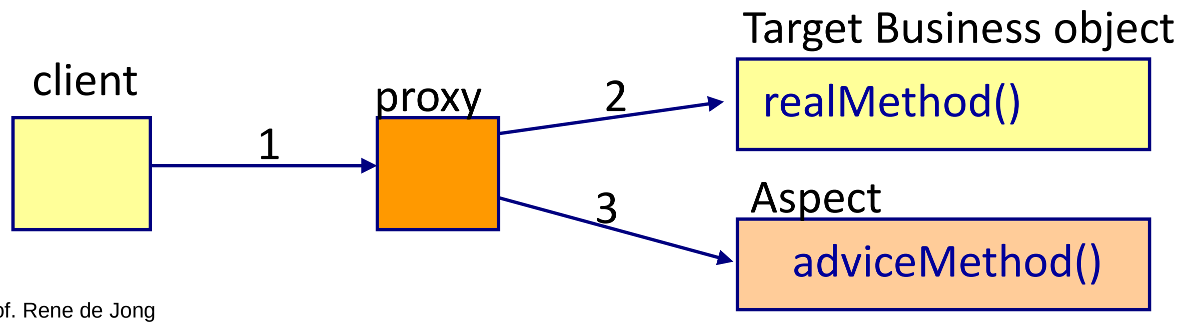
- Calls the advice method before calling the actual method



Model by Prof. Rene de Jong

@After

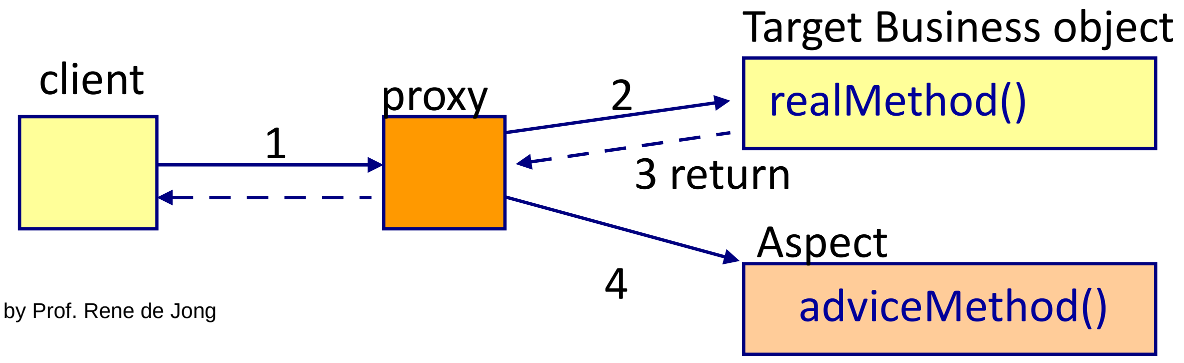
- Calls the advice method (regardless of what happens) after the real method



Model by Prof. Rene de Jong

@AfterReturning

- Calls the advice method only if the real method returned normally
  - Allows the advice to receive the returned value



Model by Prof. Rene de Jong

@AfterReturning

We will go into this in more detail coming up

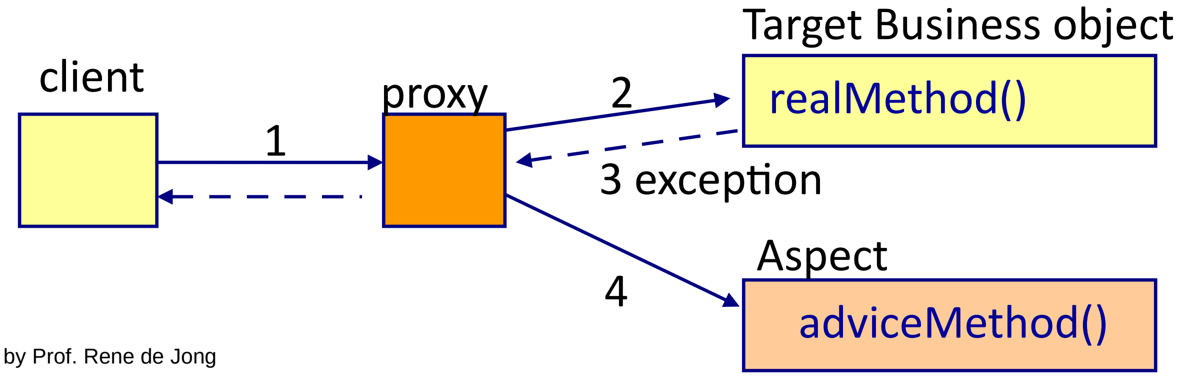
```
package cs544.spring41.aop.advices;

import org.springframework.stereotype.Service;

@Service
public class CustomerService implements ICustomerService {
      public String getName() {
            return "John";
      }
}
```

```
package cs544.spring41.aop.advices;

...

@Aspect
@Component
public class TestAspect {
      @AfterReturning(pointcut="execution(* cs544.spring41.aop.advices.CustomerService.getName(..))", returning="ret")
      public void afterRet(JoinPoint jp, String ret) {
            System.out.println(jp.getSignature().getName() + " returned: " + ret);
      }
}
```

```
getName returned: John
```

@AfterThrowing

- Calls the advice method only if the real method throws and exception
  - Allows the advice to receive the exception



Model by Prof. Rene de Jong

@AfterThrowing

We will go into this in more detail coming up

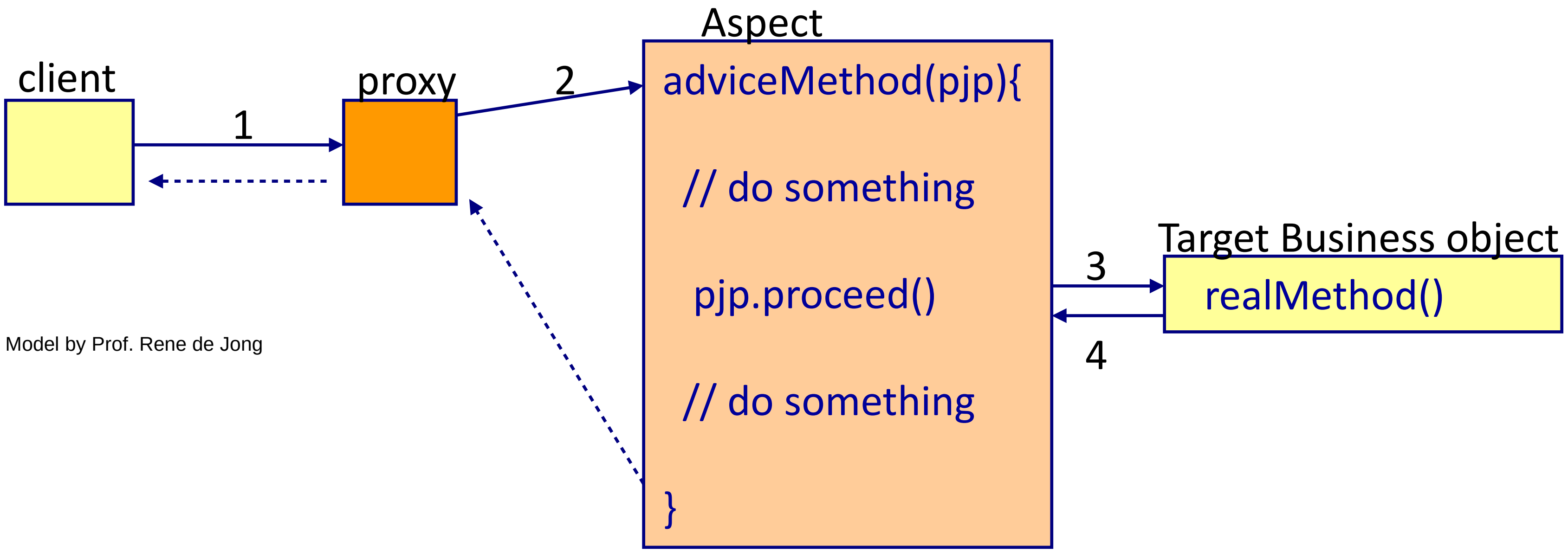
```java
package cs544.spring41.aop.advices;

import org.springframework.stereotype.Service;

@Service
public class CustomerService implements ICustomerService {
	public String getAge() {
		throw new MyException();
	}
}
```

```java
package cs544.spring41.aop.advices;

...

@Aspect
@Component
public class TestAspect {
	@AfterThrowing(pointcut="execution(* cs544.spring41.aop.advices.CustomerService.getAge(..))", throwing="ex")
	public void afterThrow(JoinPoint jp, MyException ex) {
		System.out.println(jp.getSignature().getName() + " threw a: " + ex.getClass().getName());
	}
}
```

```
getAge threw a: cs544.spring41.aop.advices.MyException
```

@Around

- Around has to choose when (and if) it calls the real method.
  - Receives the **parameters** to pass to the real method
  - Receives the **return** from the real method



Model by Prof. Rene de Jong

@Around

We will go into this in more detail coming up

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

```
Before getName
After getName returned John
```

```java
package cs544.spring41.aop.advices;

import org.springframework.stereotype.Service;

@Service
public class CustomerService implements ICustomerService
{
      public String getName() {
            return "John";
      }
}
```