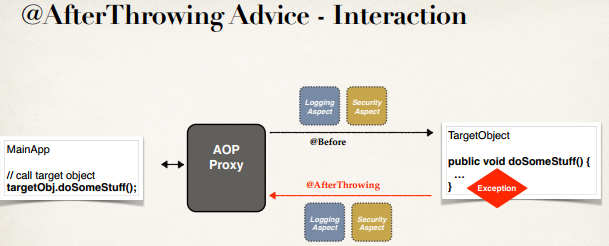
**42.1 AOP @AfterThrowing Overview**

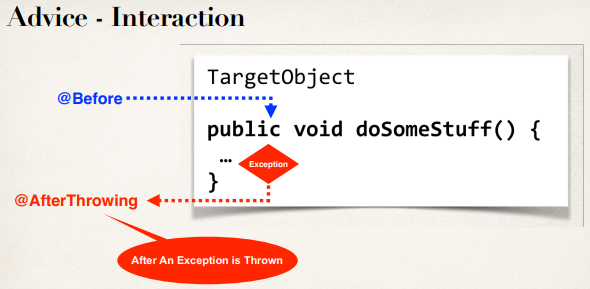
**Advice Types**:

* **Before advice**: run before the method
* **After returning advice**: run after the method (success execution)
* **After throwing advice**: run after method (if exception thrown)
* **After finally advice**: run after the method (finally)
* **Around advice**: run before and after method

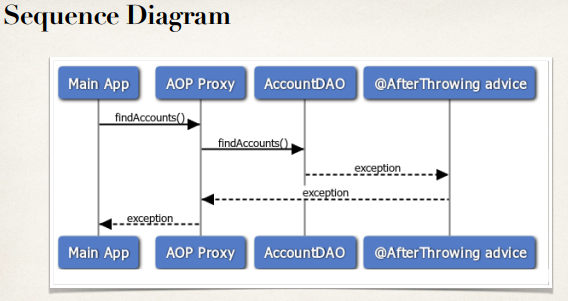
**AfterThrowing Advice – Interaction**:



**Advice – Interaction**:



**Sequence Diagram**:

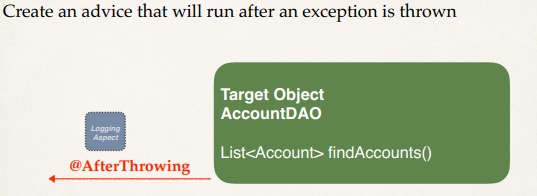


**@AfterThrowing Advice - Use Cases**:

* Log the exception
* Perform auditing on the exception
* Notify DevOps team via email or SMS
* Encapsulate this functionality in AOP aspect for easy reuse

**Example**:

1. Create an advice that will run after an exception is thrown.



1. This advice will run after an exception is thrown

@AfterThrowing("execution(\*com.ruhul.aopdemo.dao.AccountDAO.findAccounts(..))")

**public** **void** afterThrowingFindAccountAdvice() {

System.***out***.println("\n======> Executing @AfterThrowing advice");

}

**Access the Exception**:

@AfterThrowing(

pointcut = "execution(\* com.ruhul.aopdemo.dao.AccountDAO.findAccounts(..))", throwing = "theExe")

**public** **void** afterThrowingFindAccountAdvice(

JoinPoint theJoinPoint, Throwable theExe) {

// print out which method we are advising on

String methodString = theJoinPoint.getSignature().toShortString();

System.***out***.println("\n=> Executing @AfterThrowing on method: " +methodString);

// log the exception

System.***out***.println("\n======> The exception is: " + theExe);

}

**Exception Propagation**:

* At this point, we are only intercepting the exception (reading it)
* However, the exception is still propagated to calling program

42.1 AOP @AfterThrowing Overview