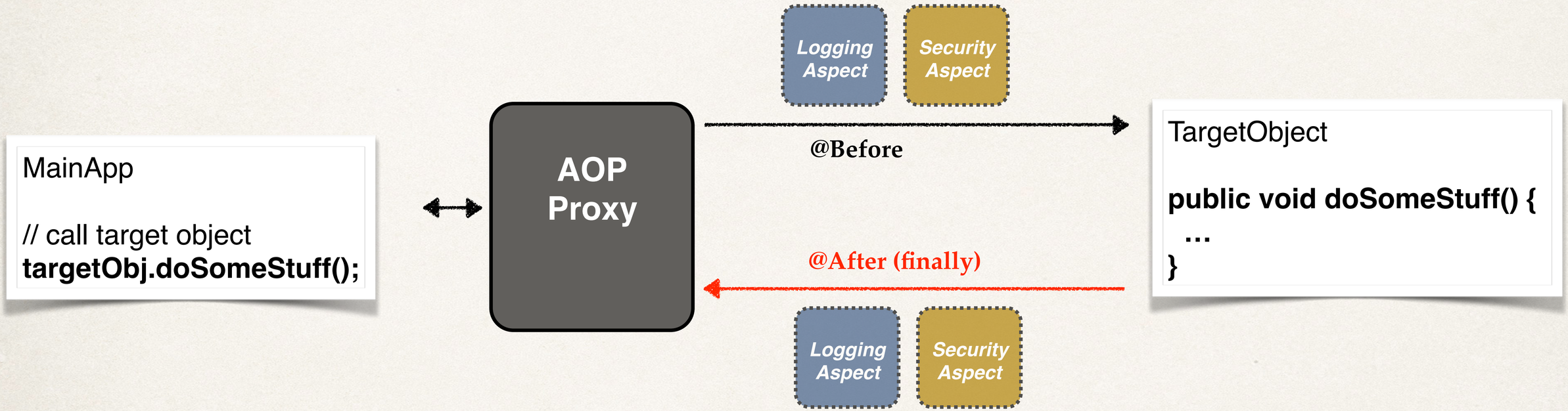
# Aspect-Oriented Programming (AOP)
# @After Advice

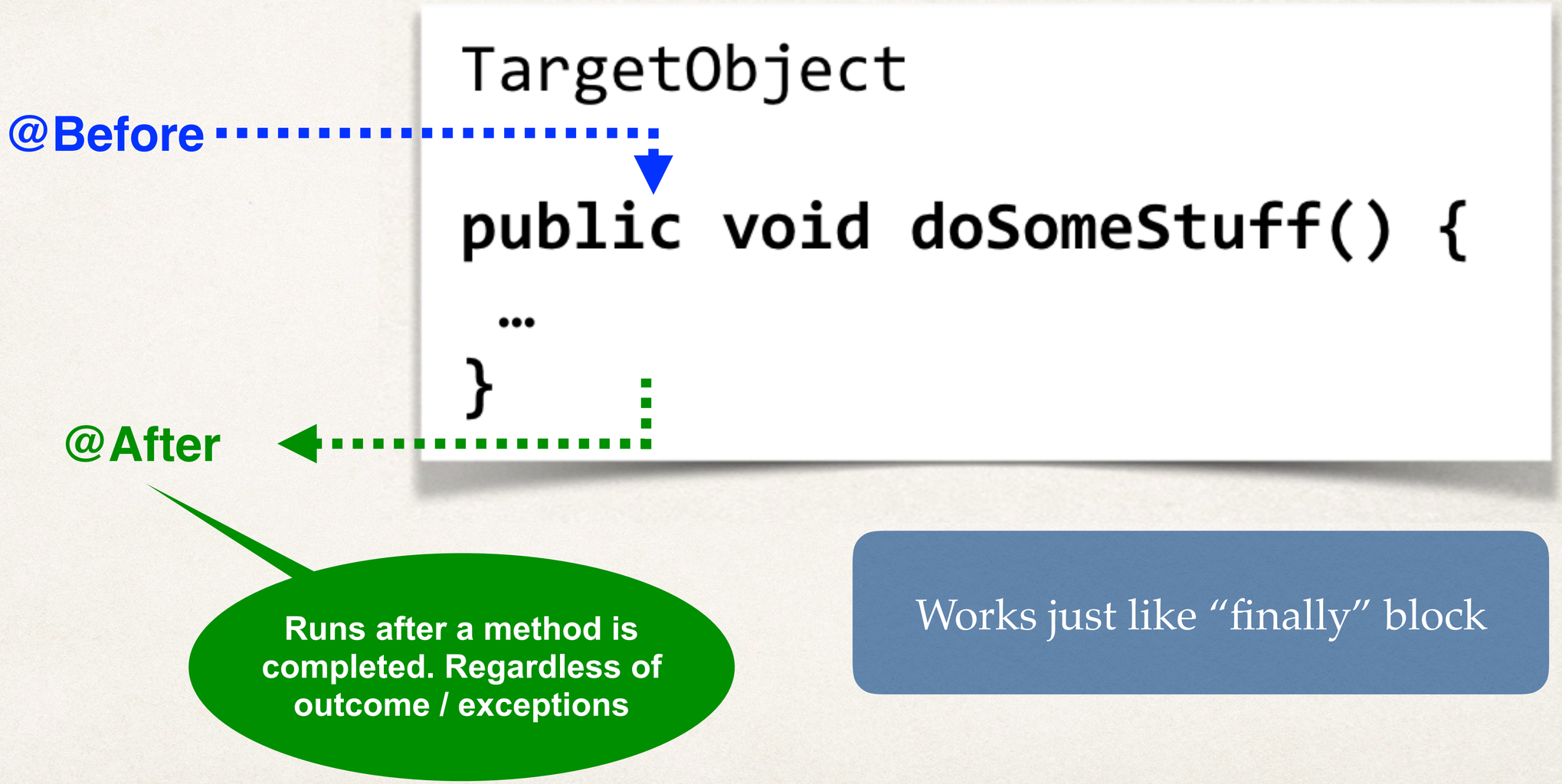
# Advice Types

- **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

# @After (finally) Advice - Interaction

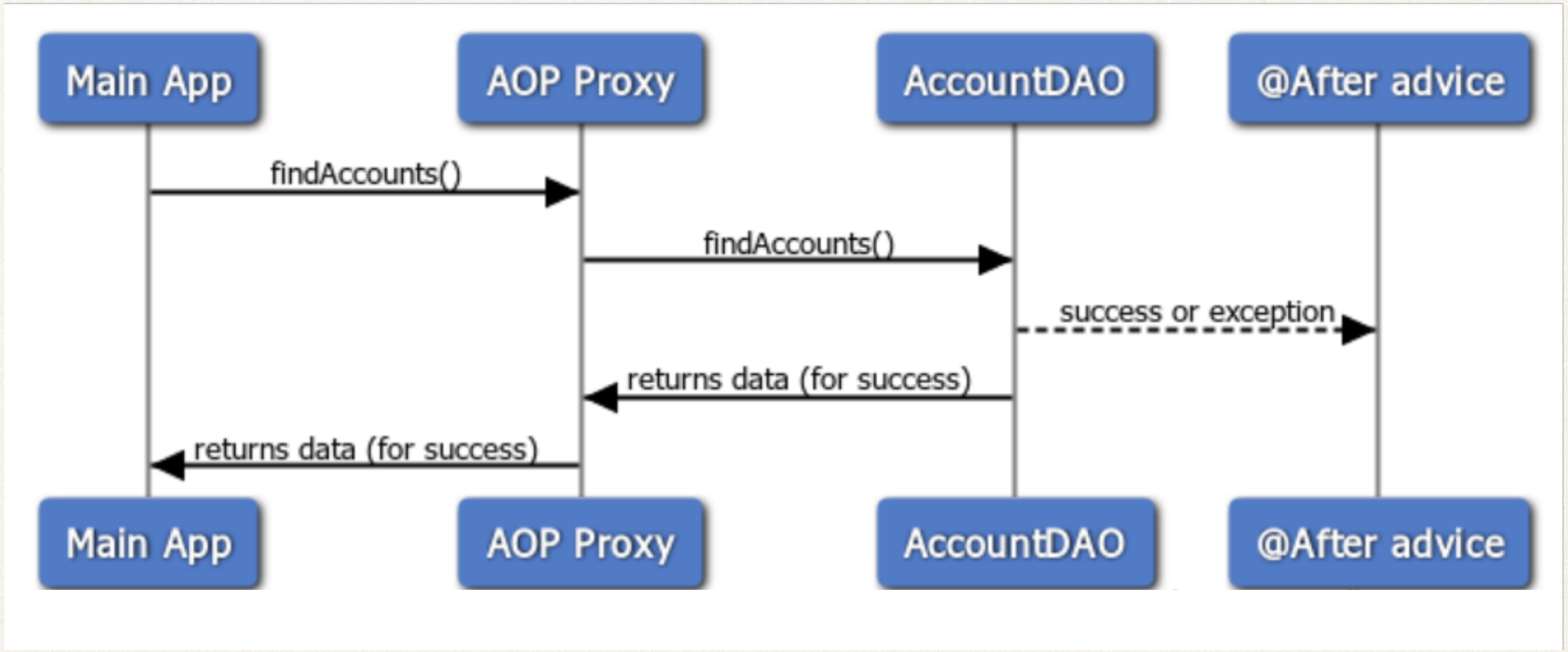


# Advice - Interaction



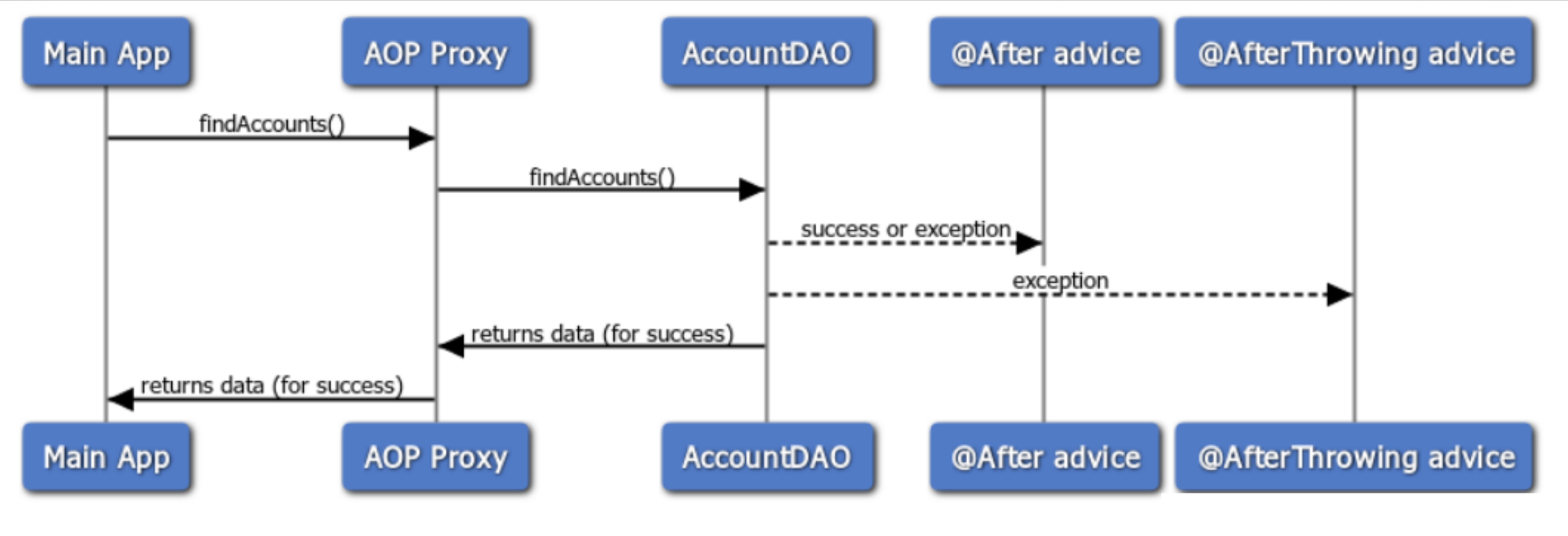

Works just like “finally” block

# Sequence Diagram



# Sequence Diagram

- @After will execute **before** @AfterThrowing



# @After Advice - Use Cases

- Log the exception and/or perform auditing
- Code to run regardless of method outcome
- Encapsulate this functionality in AOP aspect for easy reuse

# Example

- Create an advice to run after the method (finally … success/failure)

# @After Advice

- This advice will run after the method (finally … success/failure)

```
@After("execution(* com.luv2code.aopdemo.dao.AccountDAO.findAccounts(..))")
public void afterFinallyFindAccountsAdvice() {

   System.out.println("Executing @After (finally) advice");

}
```

# @After Advice - Tips

- The @After advice does not have access to the exception
  - If you need exception, then use @AfterThrowing advice

- The @After advice should be able to run in the case of success or error
  - Your code should not depend on happy path or an exception
  - Logging / auditing is the easiest case here