Spring AOP

by Andrés Arcia

Spring AOP

- @Aspect
- @Before
- @AfterReturning
- @AfterThrowing
- @After
- @Around

What problem does AOP attacks?

- Cross-cutting concerns: “Generic functionality that is needed in many places in your application”
  - Logging and Tracing
  - Security
  - Monitoring
  - …
- Leads to two things:
  - Code tangling (coupling concerns)
  - Code scattering (code tangling spread across modules)

Symptom #1: Code Tangling

```
public class RewardNetworkImpl implements RewardNetwork {
    public RewardConfirmation rewardAccountFor(Dining dining) {
        if (!hasPermission(SecurityContext.getPrincipal()) {
            throw new AccessDeniedException();
        }

        Account a = accountRepository.findByCreditCard(…
        Restaurant r = restaurantRepository.findByMerchantNumber(…
        MonetaryAmount amt = r.calculateBenefitFor(account, dining);
        …
    }
}
```

Mixing of concerns

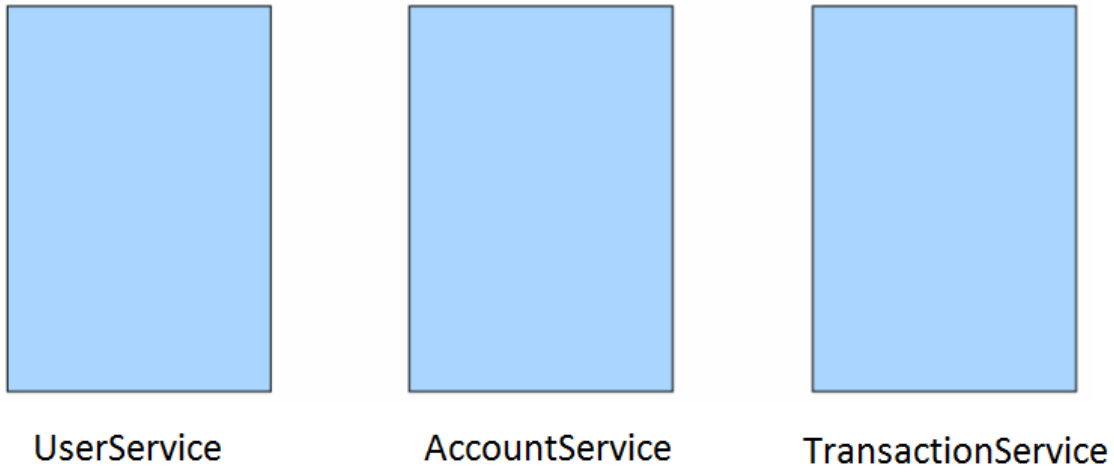
Symptom #2: Code Scattering

```
public class JpaAccountManager implements AccountManager {
    public Account getAccountForEditing(Long id) {
        if (!hasPermission(SecurityContext.getPrincipal()) {
            throw new AccessDeniedException();
        }
        …
```

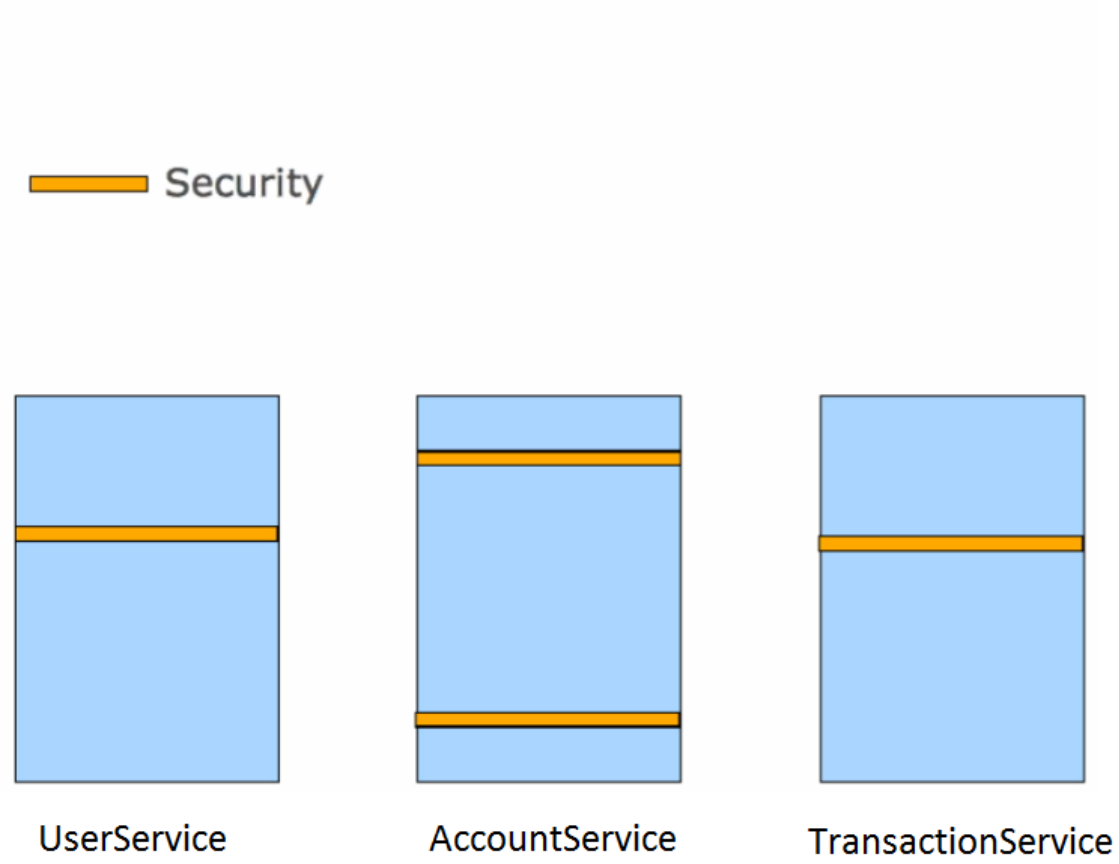
Duplication

```
public class JpaMerchantReportingService
        implements MerchantReportingService {
    public List<DiningSummary> findDinings(String merchantNumber,
                                           DateInterval interval) {
        if (!hasPermission(SecurityContext.getPrincipal()) {
            throw new AccessDeniedException();
        }
        …
```

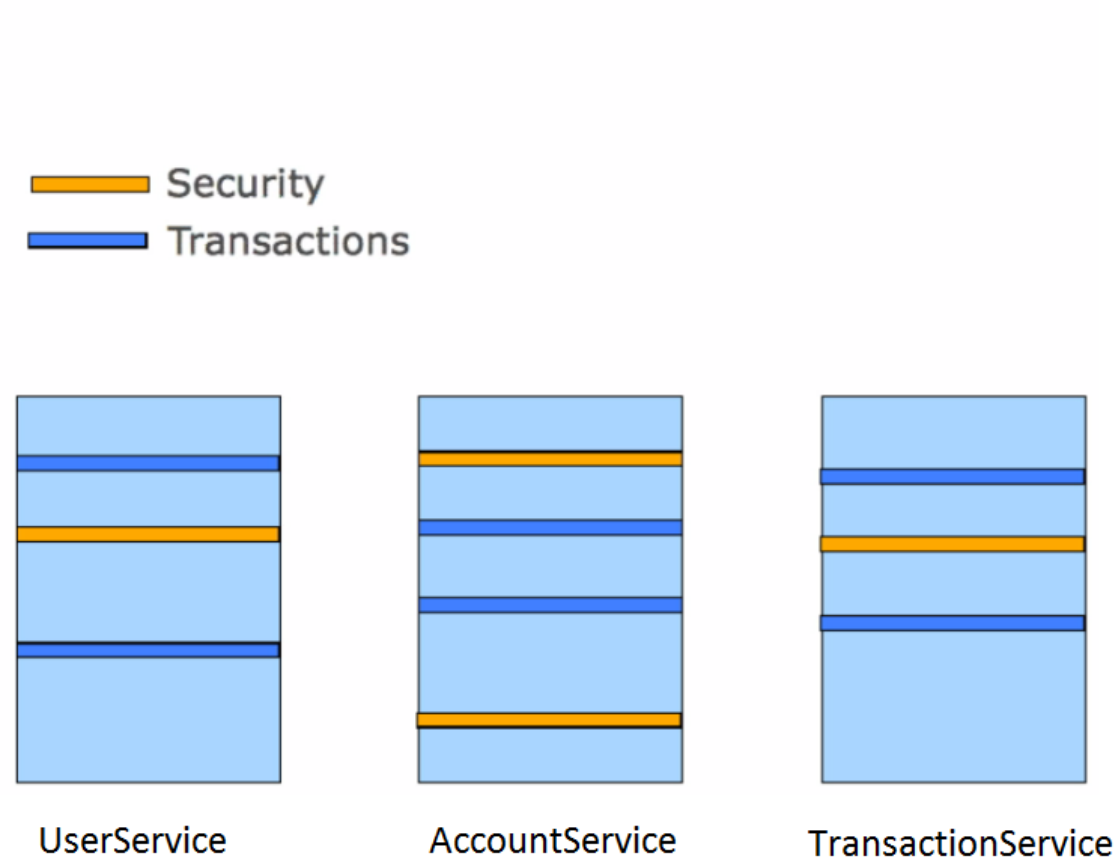
What problem does AOP attacks?



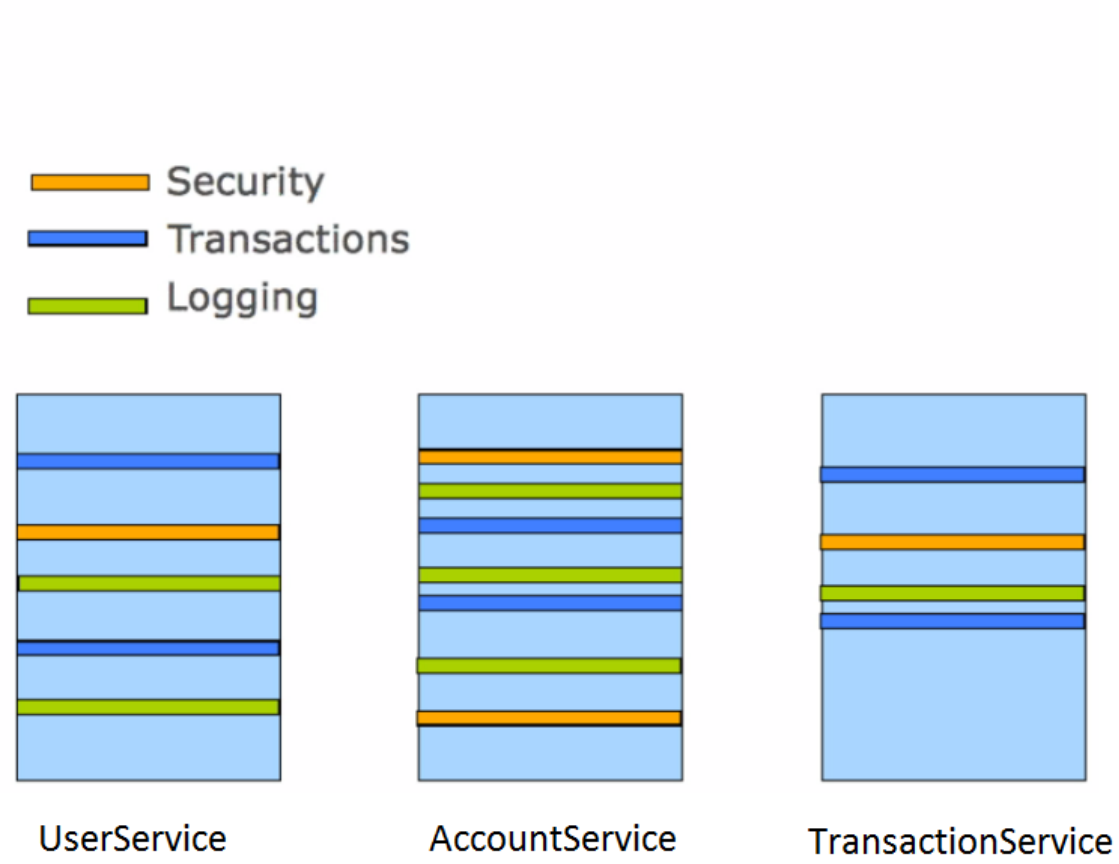
What problem does AOP attacks?



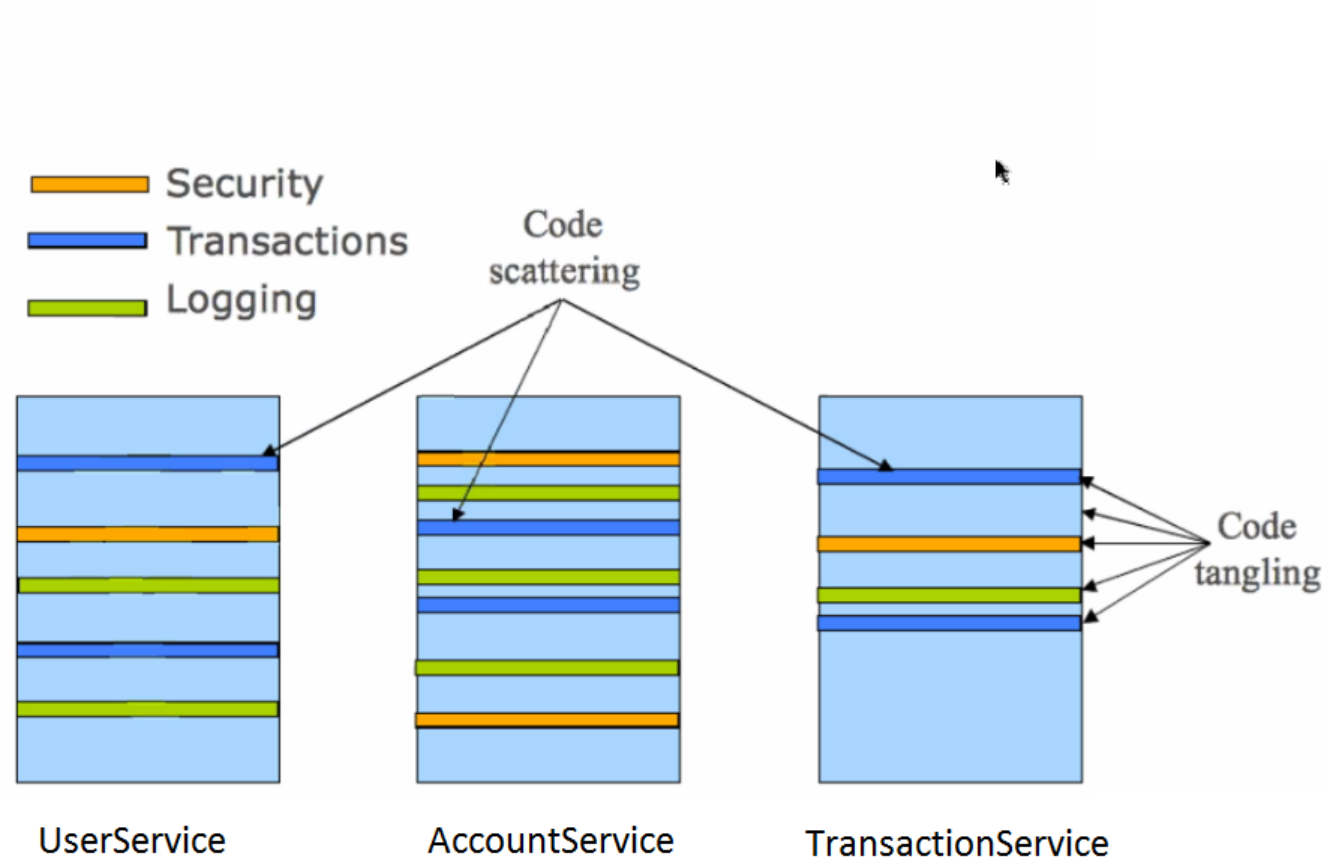
What problem does AOP attacks?



What problem does AOP attacks?



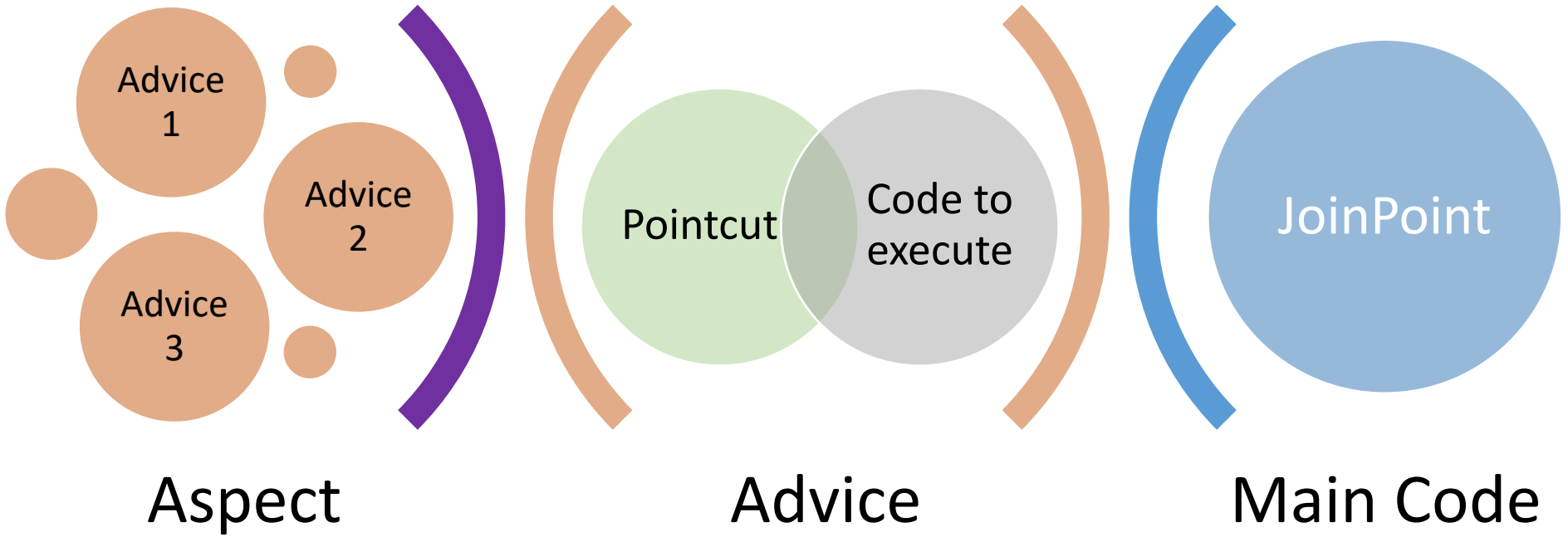
What problem does AOP attacks?



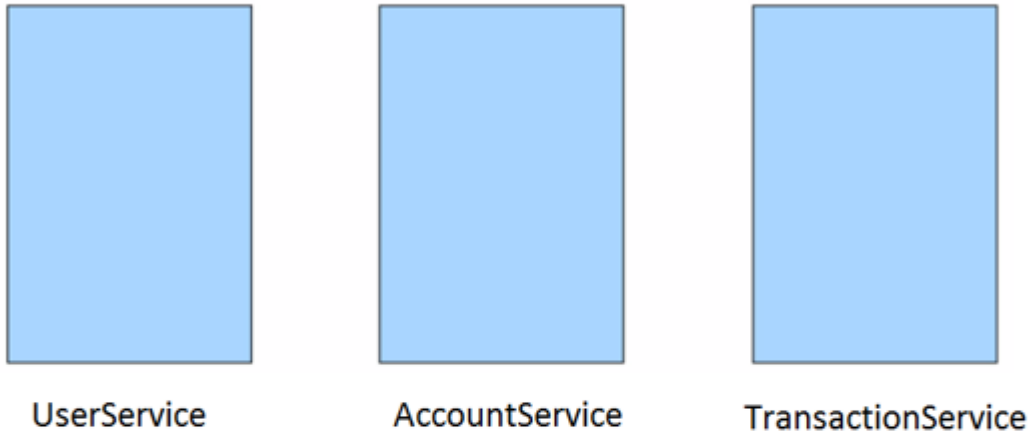
How AOP works?

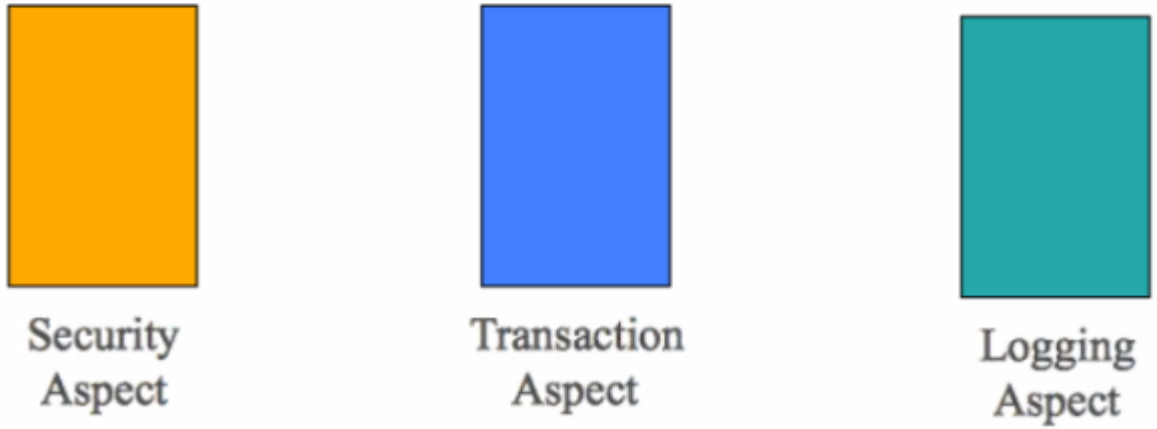
- JoinPoint: A point in the execution of a program where it is intercepted.
- Pointcut: An expression that denotes the intersection.
- Advice: Code to be executed when it is intersected, works with Pointcuts together.
- Aspect: A module that encapsulates many advices.
- Weaving: The technique of using Aspect in the main code

How AOP works?

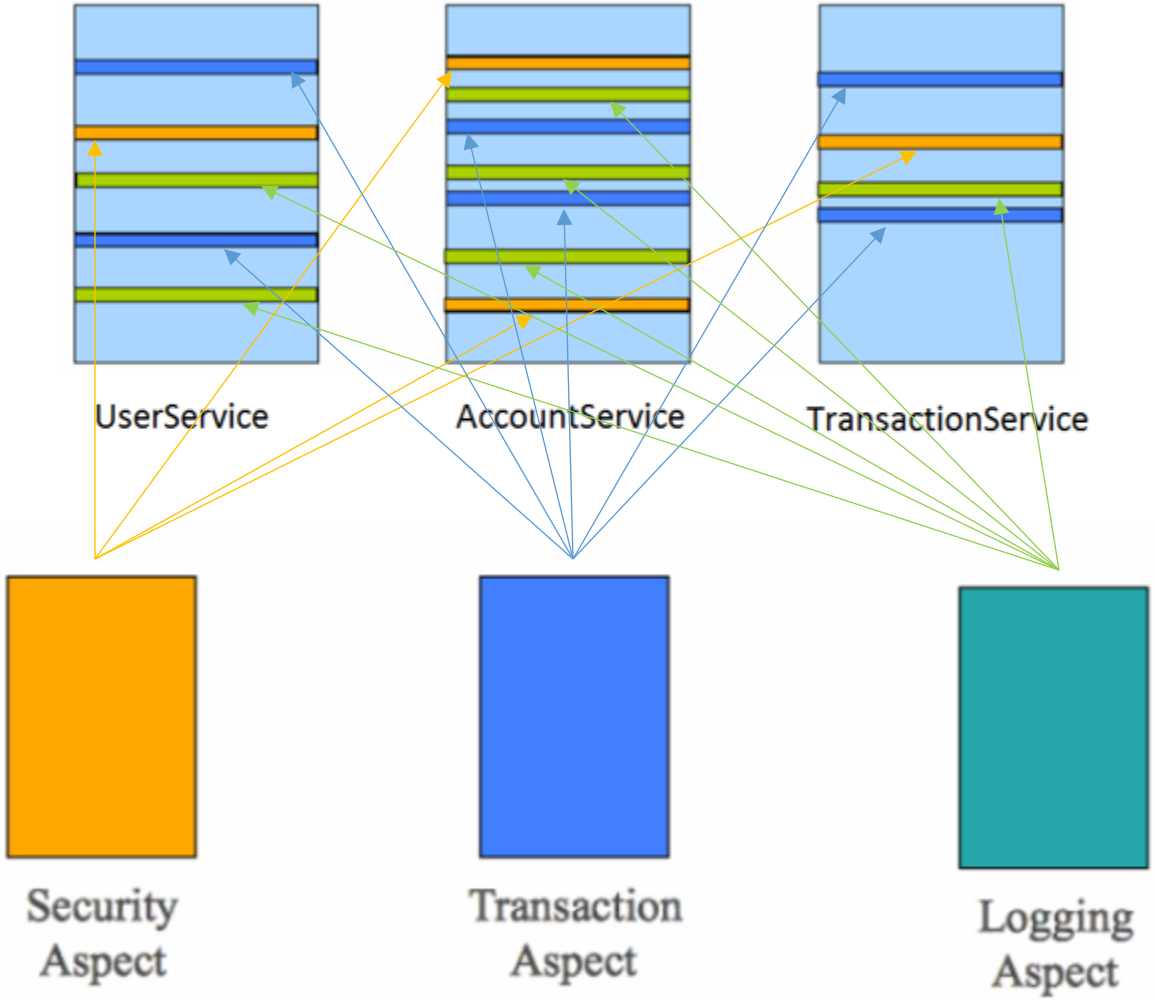


How AOP works?





How AOP works?



How AOP works? - Example

Log a message every time a property is about to change

```
public class SimpleCache implements Cache
{
   private int cacheSize;
   private DataSource dataSource;
   private String name;

   public void setCacheSize(int size) { cacheSize = size; }

   public void setDataSource(DataSource ds) { dataSource = ds; }

   public void setBeanName(String beanName) { name = beanName; }

    ...

    public void toString() { return name; }       // For convenience later
}
```

How AOP works? - Example

```
@Component
public class PropertyChangeTracker {
    private Logger logger = Logger.getLogger(getClass());


    public void trackChange() {
        logger.info("Property about to change…");
    }
}
```

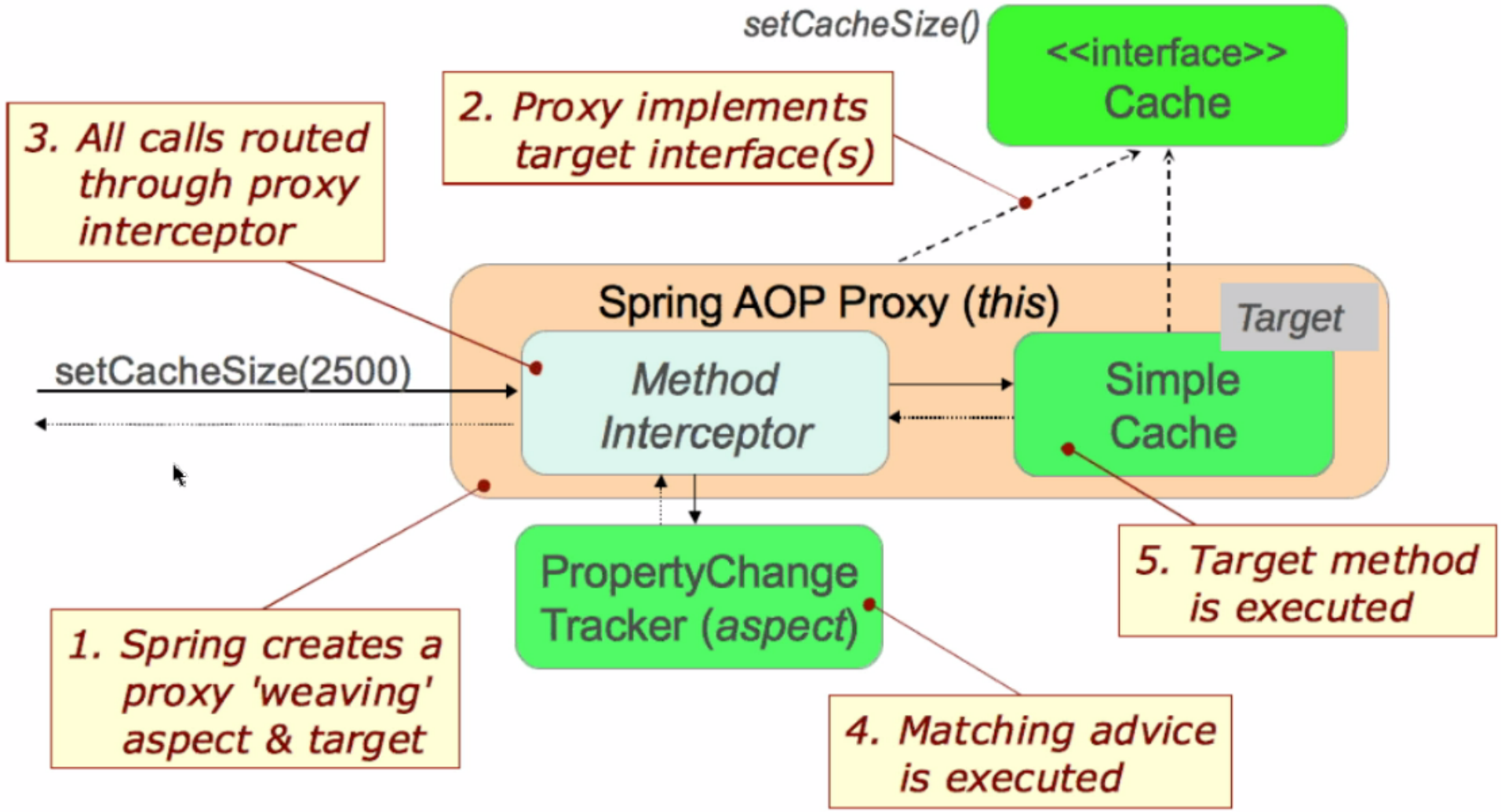
How AOP works? - Example

```
@Aspect
@Component
public class PropertyChangeTracker {
    private Logger logger = Logger.getLogger(getClass());


    public void trackChange() {
        logger.info("Property about to change…");
    }
}
```

How AOP works? - Example

```
@Aspect
@Component
public class PropertyChangeTracker {
    private Logger logger = Logger.getLogger(getClass());

    @Before("execution(void set*(*))")
    public void trackChange() {
        logger.info("Property about to change…");
    }
}
```

How AOP works? - Under the hood



How AOP works? - Other capabilities

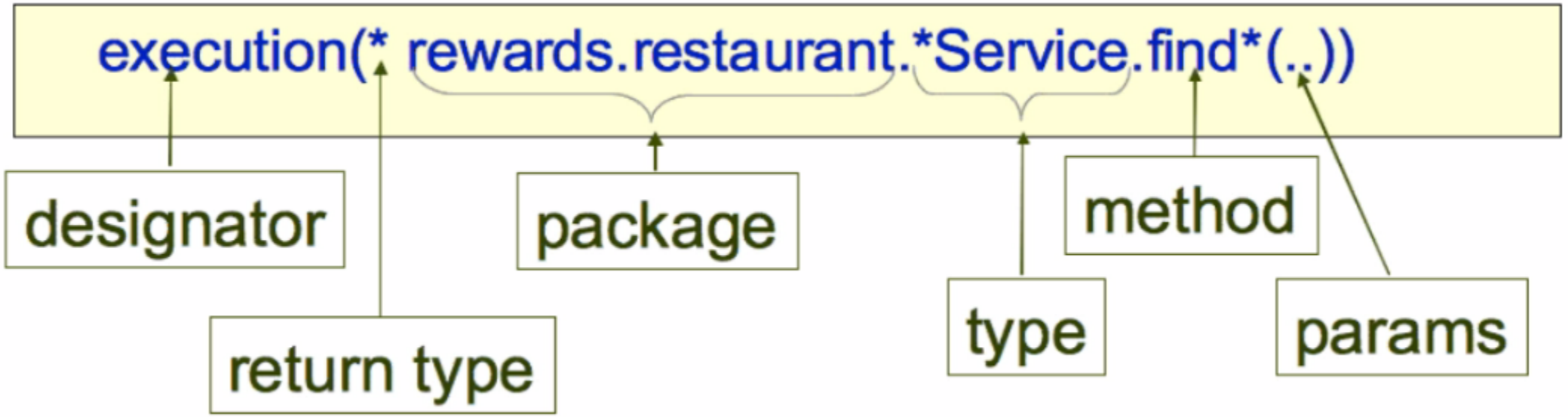
```
@Aspect
public class PropertyChangeTracker {
    private Logger logger = Logger.getLogger(getClass());

    @Before("execution(void set*(*))")
    public void trackChange(JoinPoint point) {
        String name = point.getSignature().getName();
        Object newValue = point.getArgs()[0];
        logger.info(name + " about to change to " + newValue +
            " on " + point.getTarget());

    }
}
```

Context about the intercepted point

toString() returns bean-name

INFO: setCacheSize about to change to 2500 on cache-A

Defining Pointcuts - Signature



Defining Pointcuts - Examples

```
execution(void send*(String))

execution(* send(*))

execution(* send(int, ..))

execution(void example.MessageServiceImpl.*(..))
```

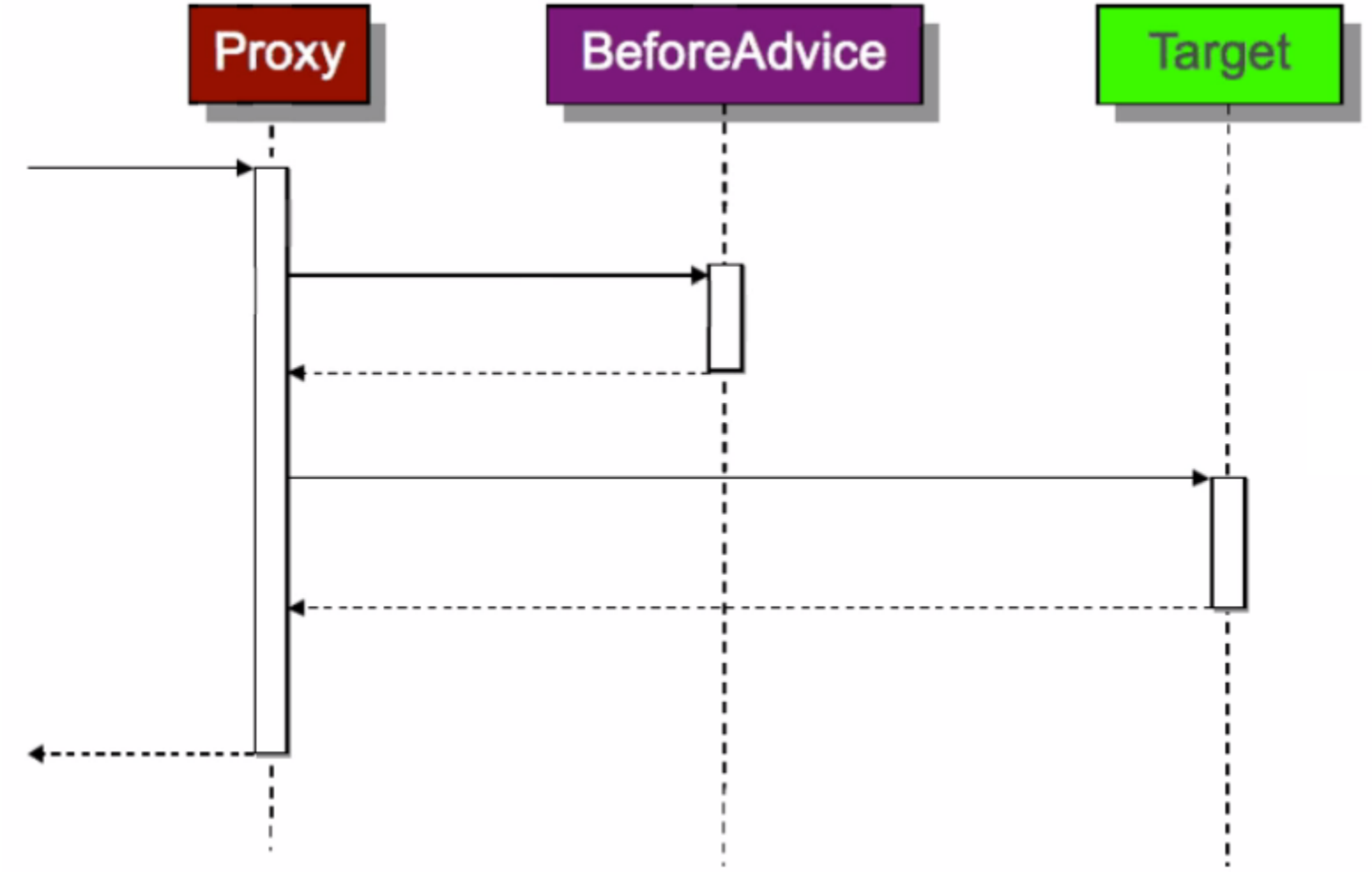
Defining Pointcuts - Examples

```
execution(void example.MessageService.send(*))

execution(* rewards.*.restaurant.*.*(..))

execution(* rewards..restaurant.*.*(..))

execution(* *..restaurant.*.*(..))
```

Advice Types

- @Before
- @AfterReturning
- @AfterThrowing
- @After
- @Around

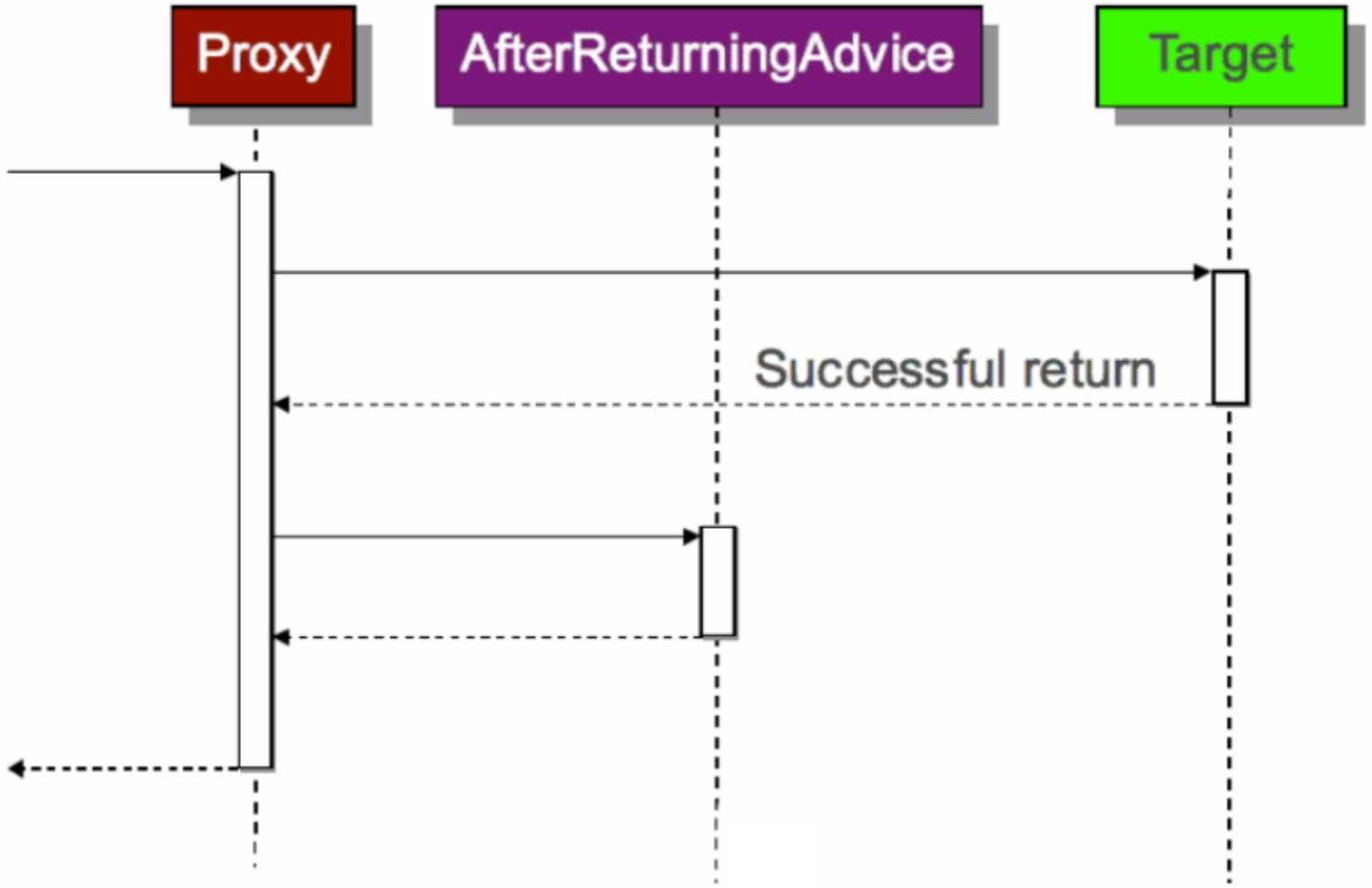
Advice Types: @Before



Advice Types: @Before

```
@Aspect
public class PropertyChangeTracker {
    private Logger logger = Logger.getLogger(getClass());

    @Before("execution(void set*(*))")
    public void trackChange() {
        logger.info("Property about to change…");
    }
}
```

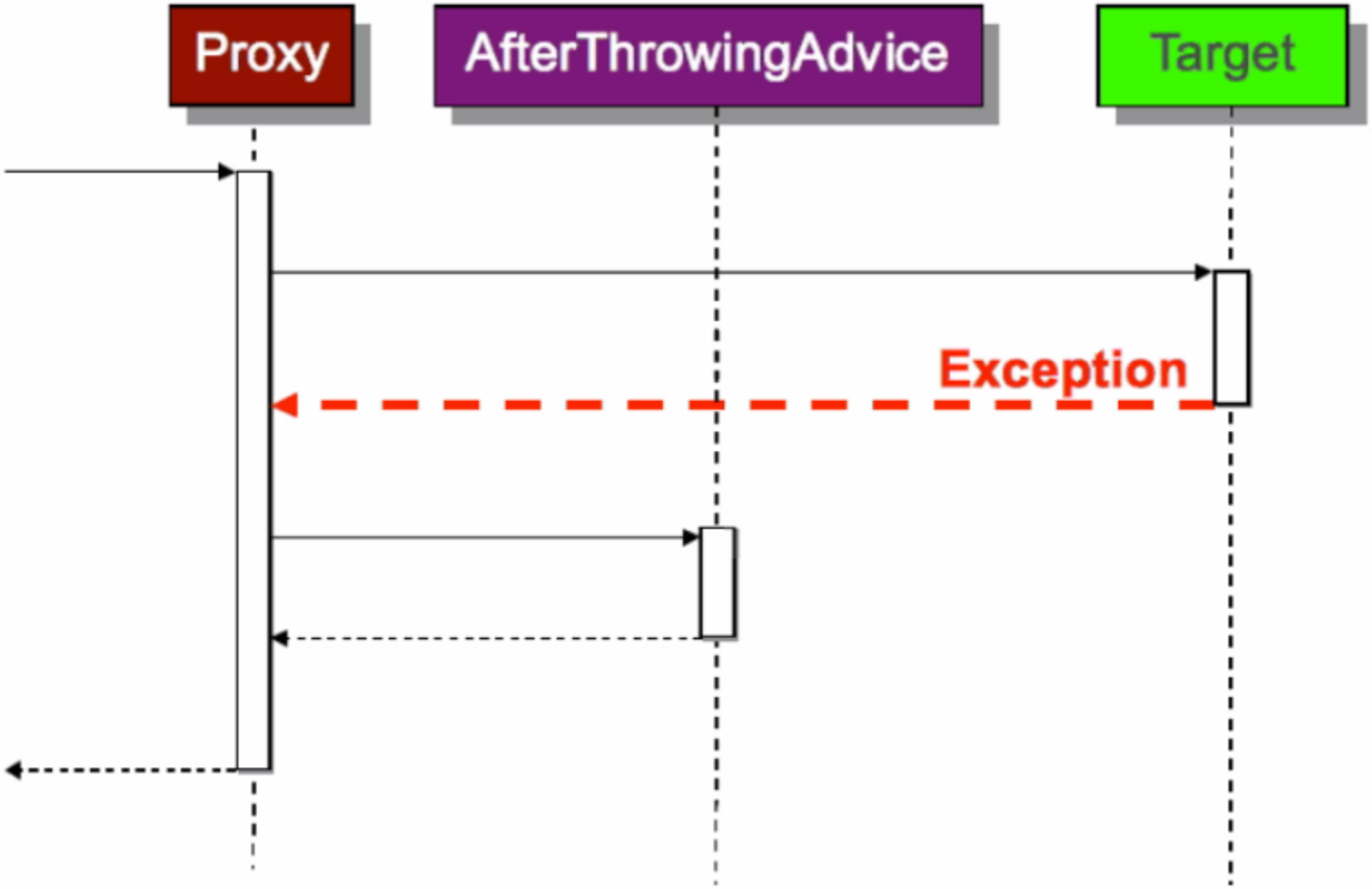
Advice Types: @AfterReturning



Advice Types: @AfterReturning

```
@AfterReturning(value="execution(* service..*.*(..))",
                returning="reward")
public void audit(JoinPoint jp, Reward reward) {
    auditService.logEvent(jp.getSignature() +
        " returns the following reward object :"  + reward.toString() );
}
```

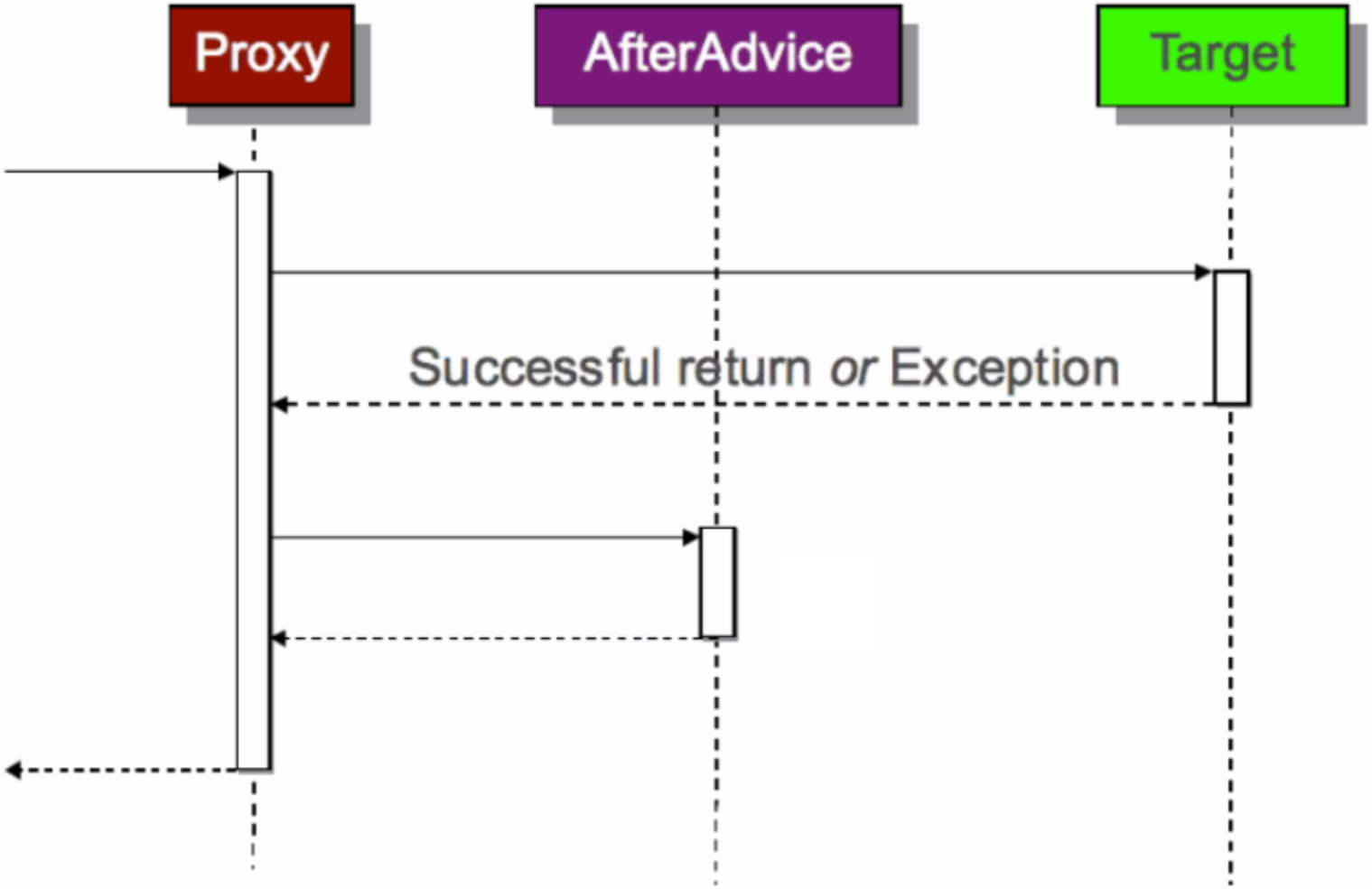
Advice Types: @AfterThrowing



Advice Types: @AfterThrowing

```
@AfterThrowing(value="execution(* *..Repository.*(..))", throwing="e")
public void report(JoinPoint jp, DataAccessException e) {
    mailService.emailFailure("Exception in repository", jp, e);
    throw new RewardsException(e);
}
```

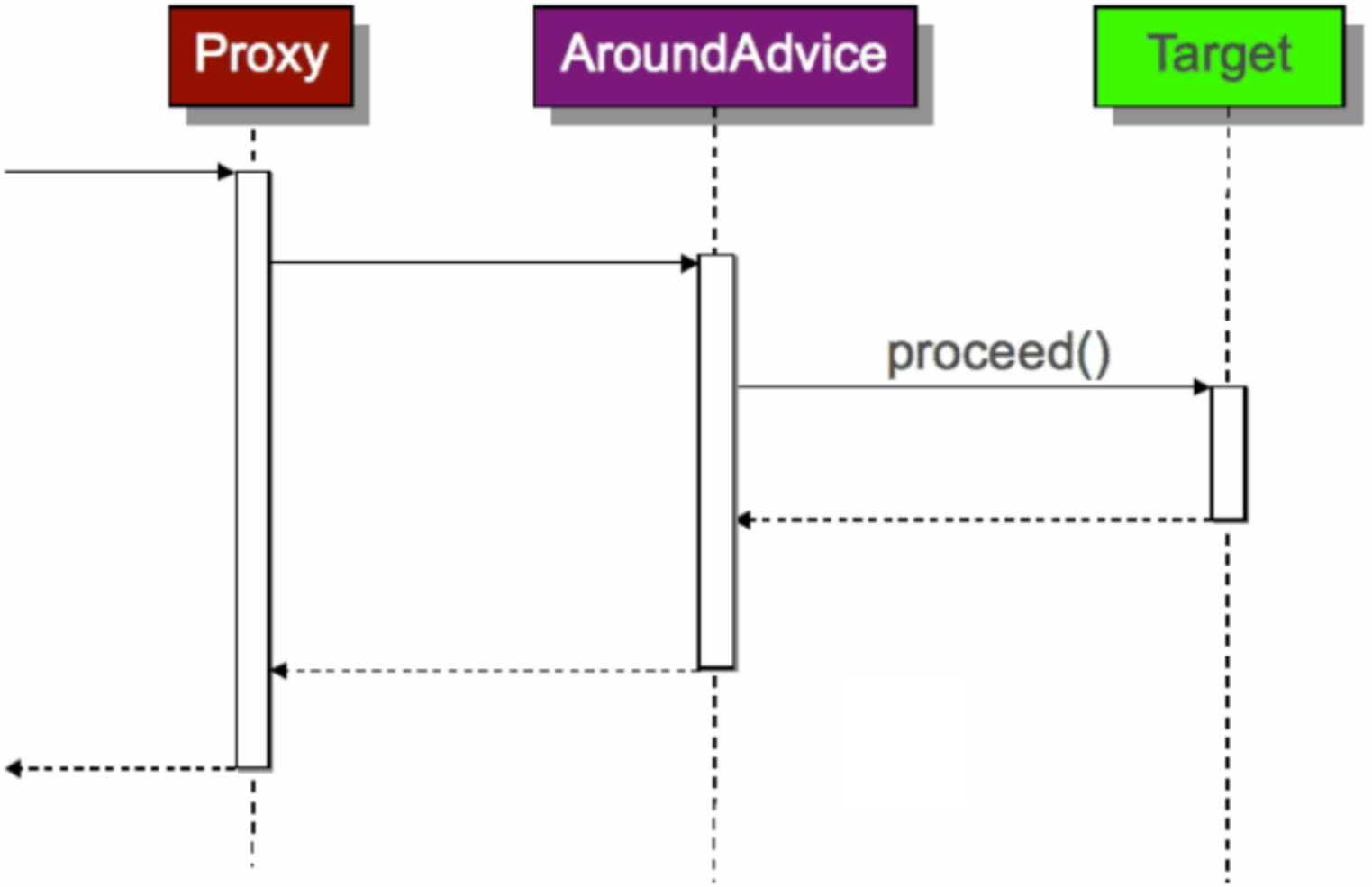
Advice Types: @After



Advice Types: @After

```
@After("execution(void update*(..))")
public void trackUpdate() {
    logger.info("An update has been attempted …");
}
```

Advice Types: @Around



Advice Types: @Around

```
@Around("execution(@example.Cacheable * rewards.service..*.*(..))")
public Object cache(ProceedingJoinPoint point)  throws Throwable {
    Object value = cacheStore.get(cacheKey(point));
    if (value == null) {
        value = point.proceed();
        cacheStore.put(cacheKey(point), value);
    }
    return value;
}
```

Proceed only if not already cached