SPRING FRAMEWORK

ASPECT-ORIENTED PROGRAMMING

APRIL, 2015

WHAT IS AOP?

Programming paradigm that aims to increase modularity

by allowing the separation of cross-cutting concerns

WHAT IS AOP?

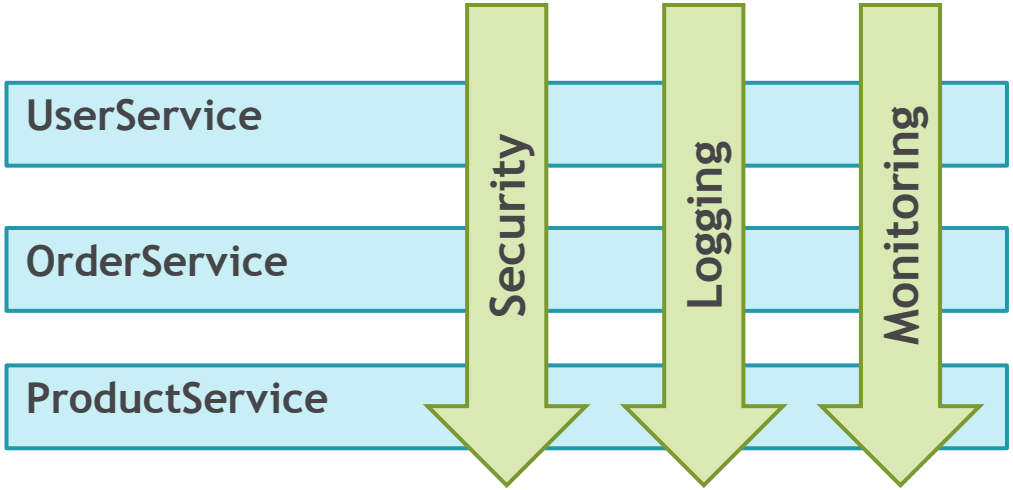
Programming paradigm that aims to increase modularity

by allowing the separation of cross-cutting concerns

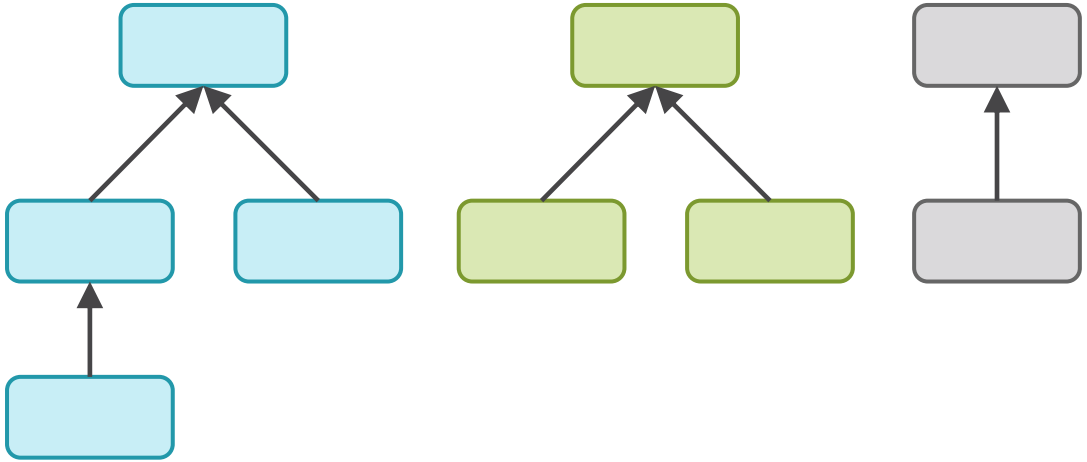
WHAT IS AOP?

Programming paradigm that aims to increase modularity

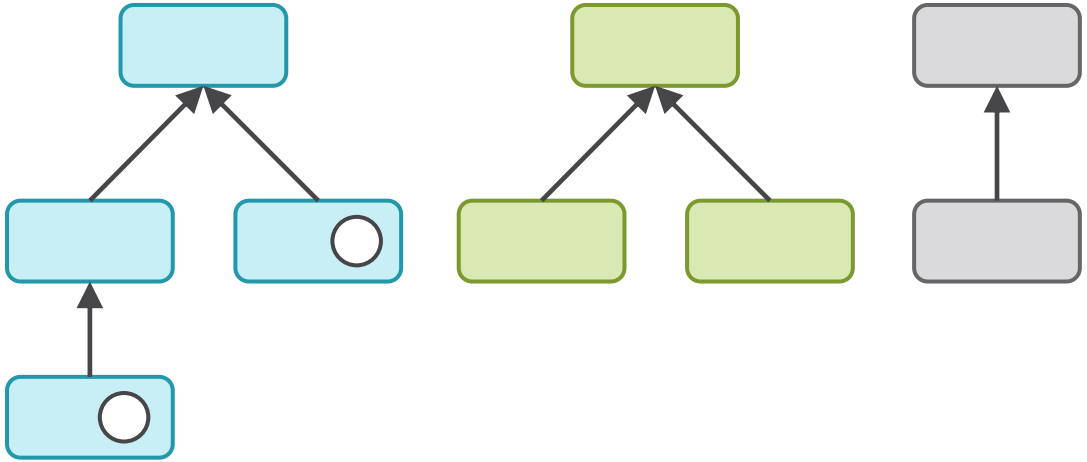
by allowing the separation of cross-cutting concerns



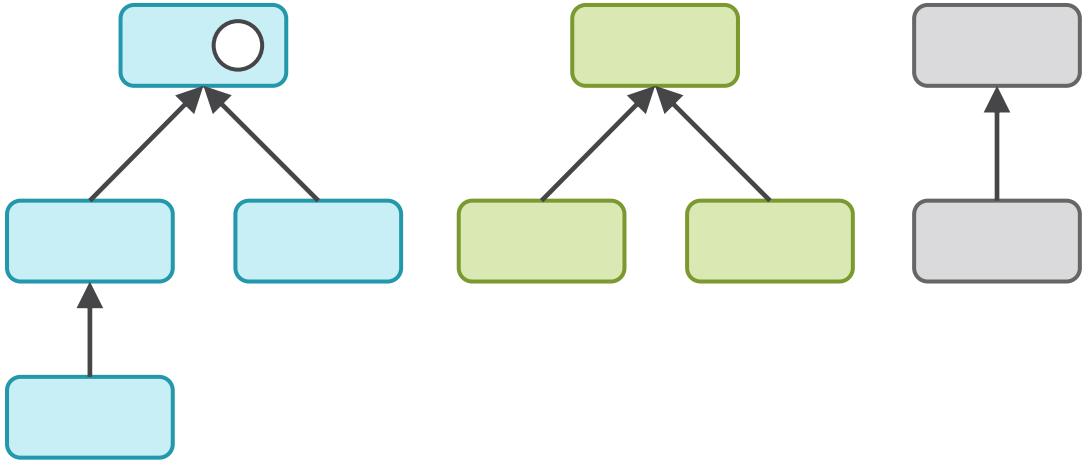
CROSS-CUTTING CONCERNS



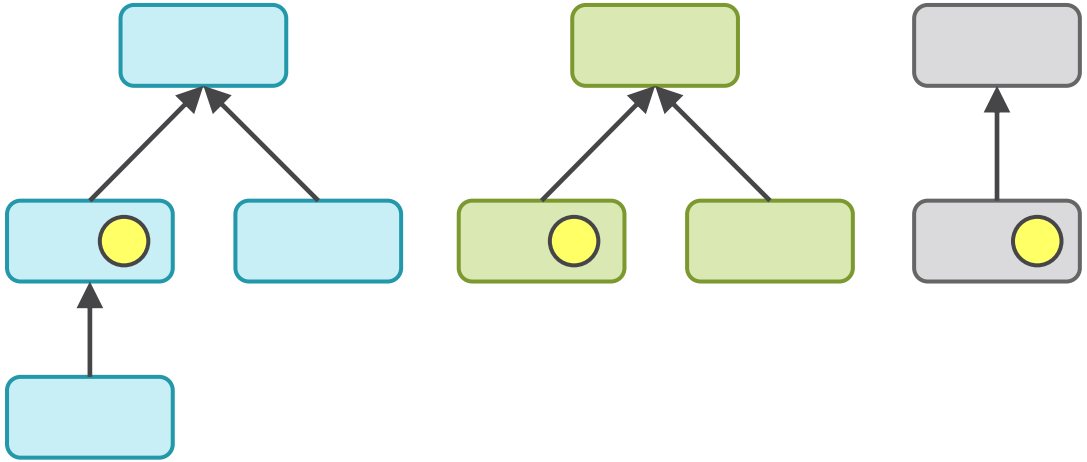
CROSS-CUTTING CONCERNS



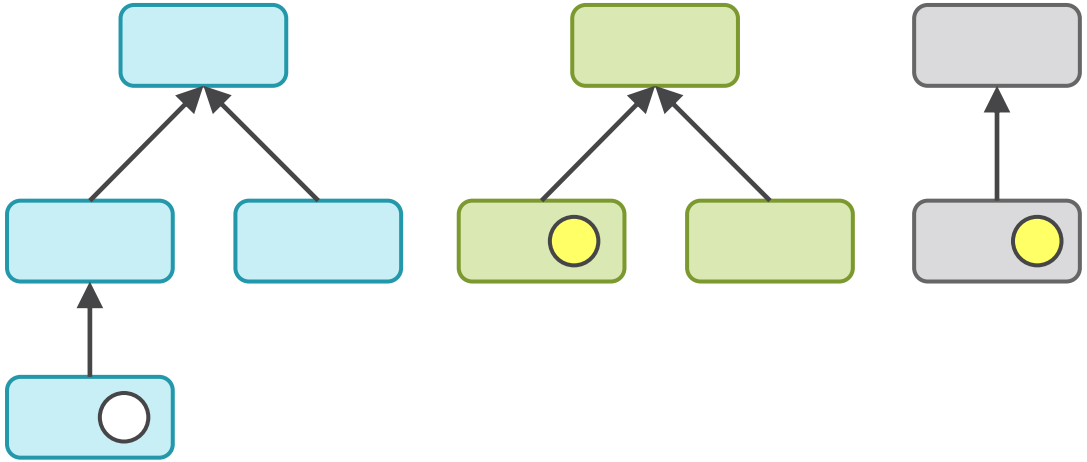
CROSS-CUTTING CONCERNS



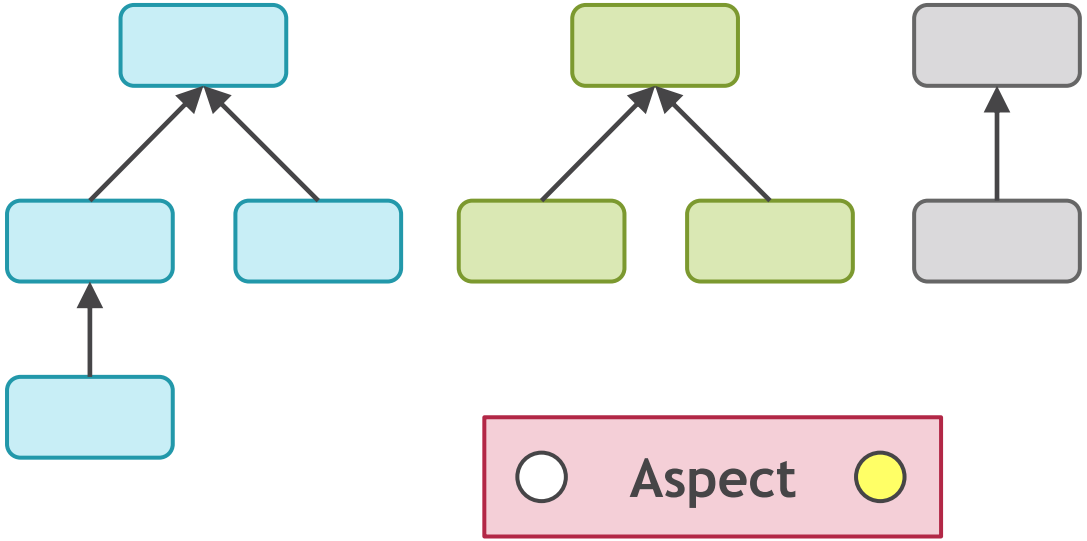
CROSS-CUTTING CONCERNS



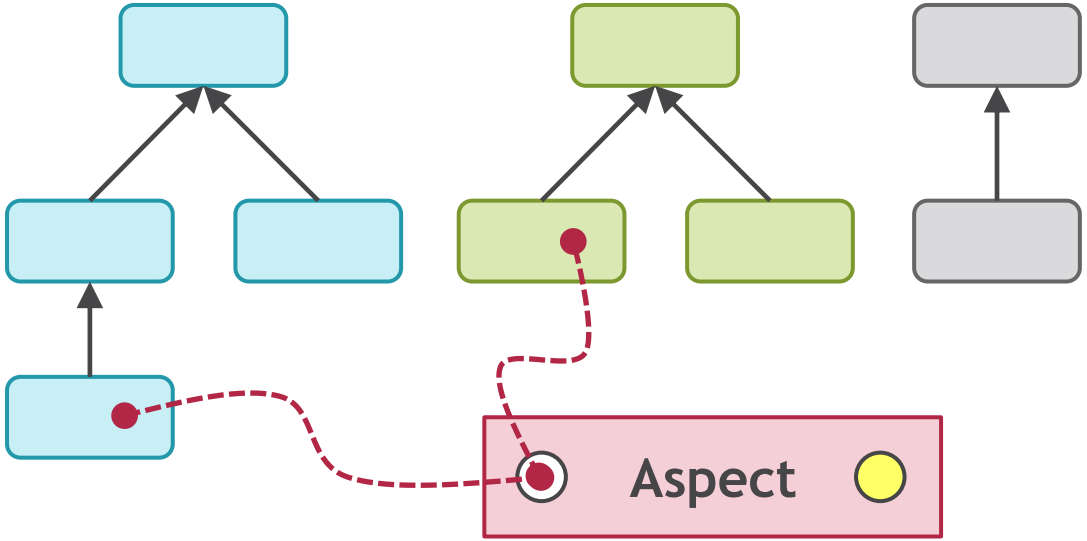
CROSS-CUTTING CONCERNS



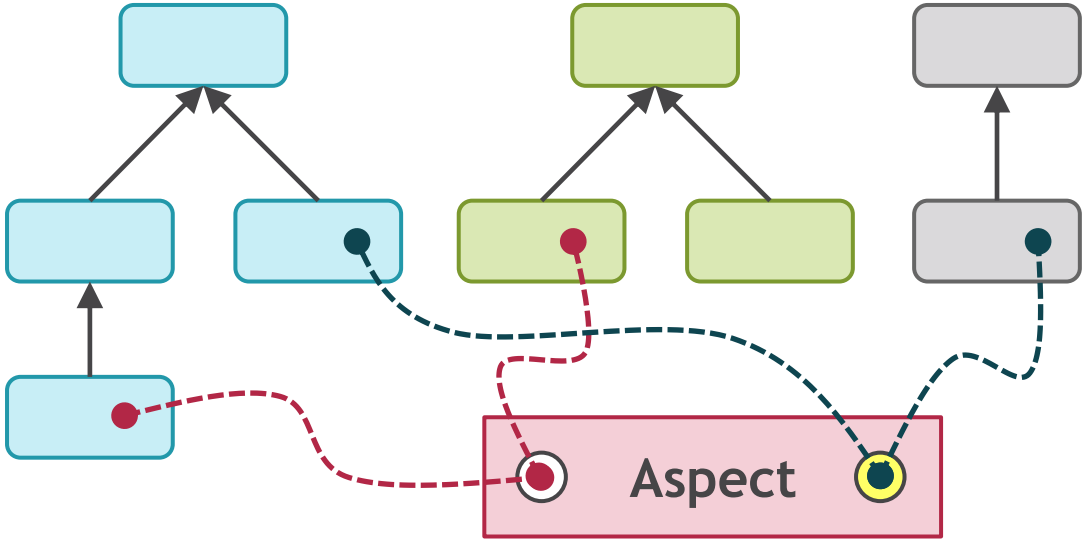
CROSS-CUTTING CONCERNS



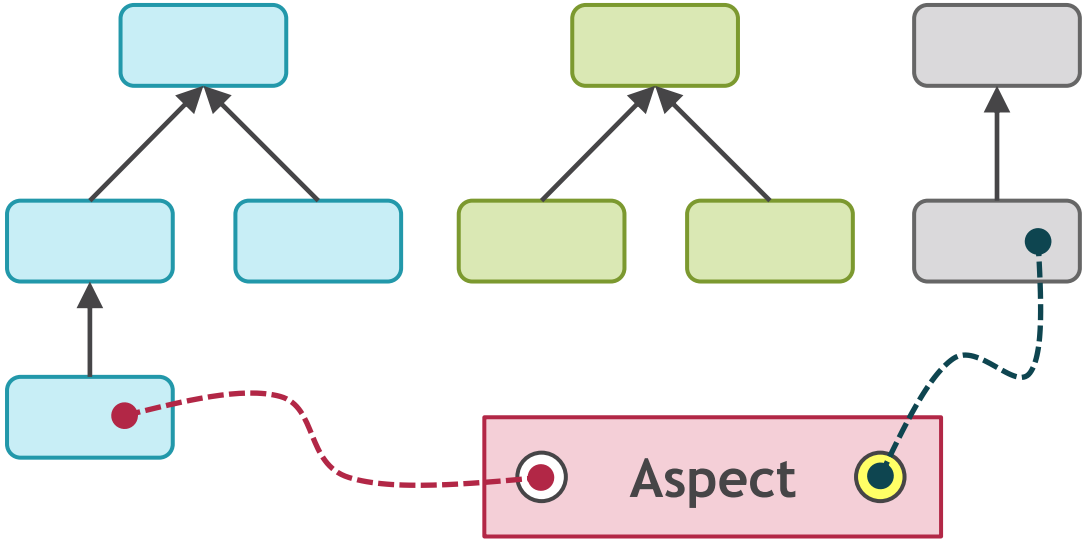
CROSS-CUTTING CONCERNS



CROSS-CUTTING CONCERNS



CROSS-CUTTING CONCERNS



TERMINOLOGY

Aspect - modularization of a concern that cuts across multiple classes

TERMINOLOGY

Aspect - modularization of a concern that cuts across multiple classes

Pointcut - predicate that matches places where aspect is applied

TERMINOLOGY

Aspect - modularization of a concern that cuts across multiple classes

Pointcut - predicate that matches places where aspect is applied

Joint Point - point during the execution of a program in which aspect is applied

TERMINOLOGY

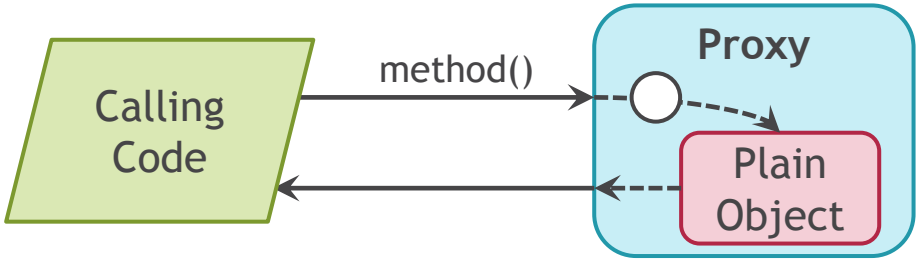
Aspect - modularization of a concern that cuts across multiple classes

Pointcut - predicate that matches places where aspect is applied

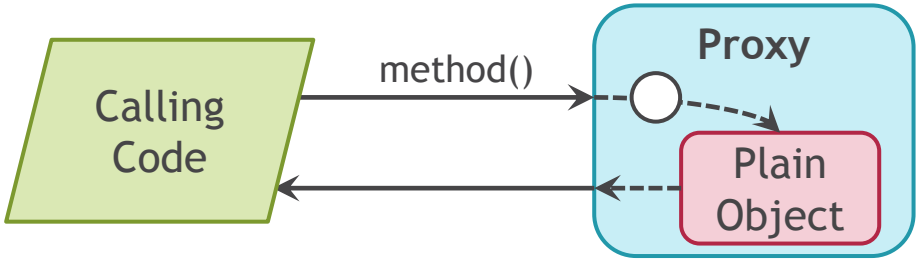
Joint Point - point during the execution of a program in which aspect is applied

Advice - action taken by an aspect at a particular join point

SPRING AOP



SPRING AOP



Currently supports only **method execution** join points (advising the execution of methods on Spring beans)

POINTCUTS

```
designator(args or signature pattern)
```

execution - method execution join points

within - join points within certain types

target - join points where the target object is an instance of the given type

this - join points where the bean reference (proxy) is an instance of the given type

args - join points where the arguments are instances of the given types

@target, ***@within***, ***@args*** – same with annotations

POINTCUTS

```
designator(args or signature pattern)
```

execution - method execution join points

within - join points within certain types

target - join points where the target object is an instance of the given type

this - join points where the bean reference (proxy) is an instance of the given type

args - join points where the arguments are instances of the given types

@target, ***@within***, ***@args*** – same with annotations

AspectJ

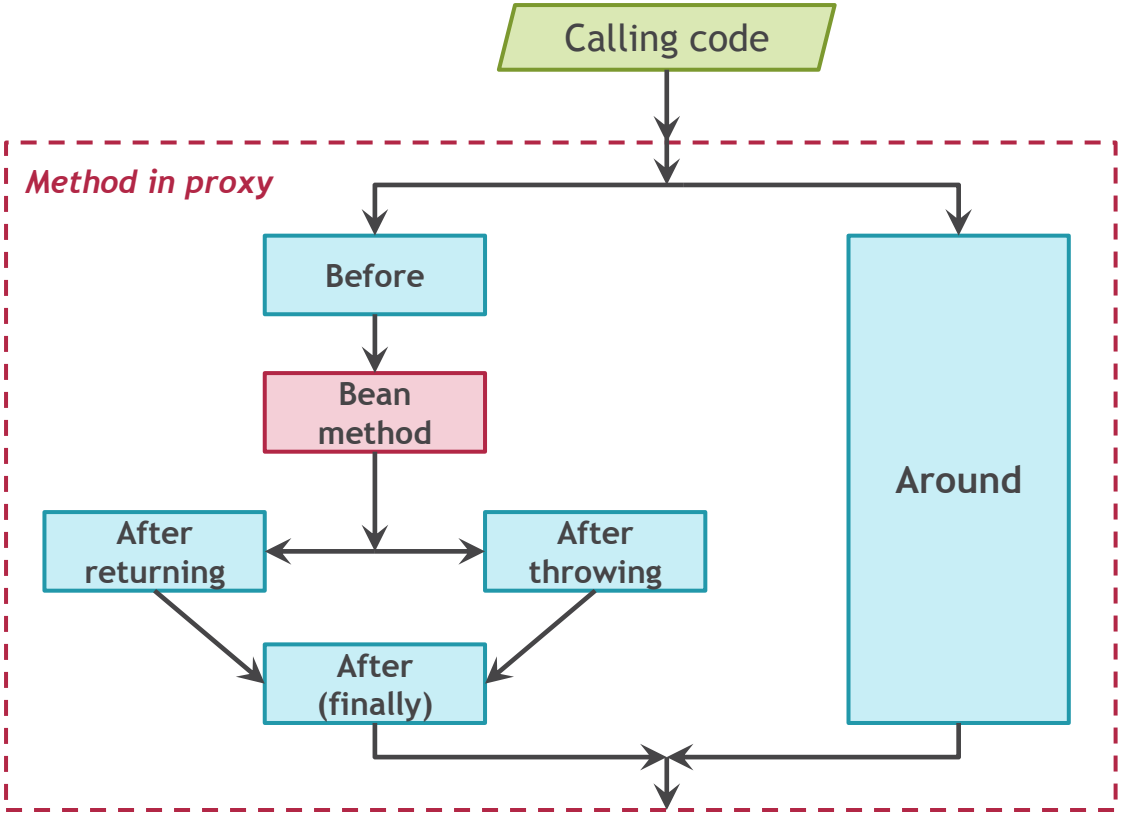
```
call, get, set, preinitialization, staticinitialization,
initialization, handler, adviceexecution, withincode, cflow,
cflowbelow, if, @this, @withincode
```

‘EXECUTION’ POINTCUT

```
execution(modifiers-pattern? ret-type-pattern
 declaring-type-pattern? name-pattern(param-
 pattern) throws-pattern?)
```

```
execution(public * *(..))
execution(* set*(..))
execution(* com.xyz.service.AccountService.*(..))
execution(* com.xyz.service.*.*(..))
```

ADVICE


Calling code
Method in proxy
Before
Bean method
After returning
After throwing
After (finally)
Around

THANK YOU

YURIY_TKACH@EPAM.COM

APRIL, 2015