iTPU IT PARK UNIVERSITY

# Instructions

How is it going ? Do you have problems with practical tasks ? Any concerns with learning materials ?

0 responses

IT PARK UNIVERSITY

# Today's agenda

- Total Recall
- Spring Framework
- Quiz
- Questions

Total Recall

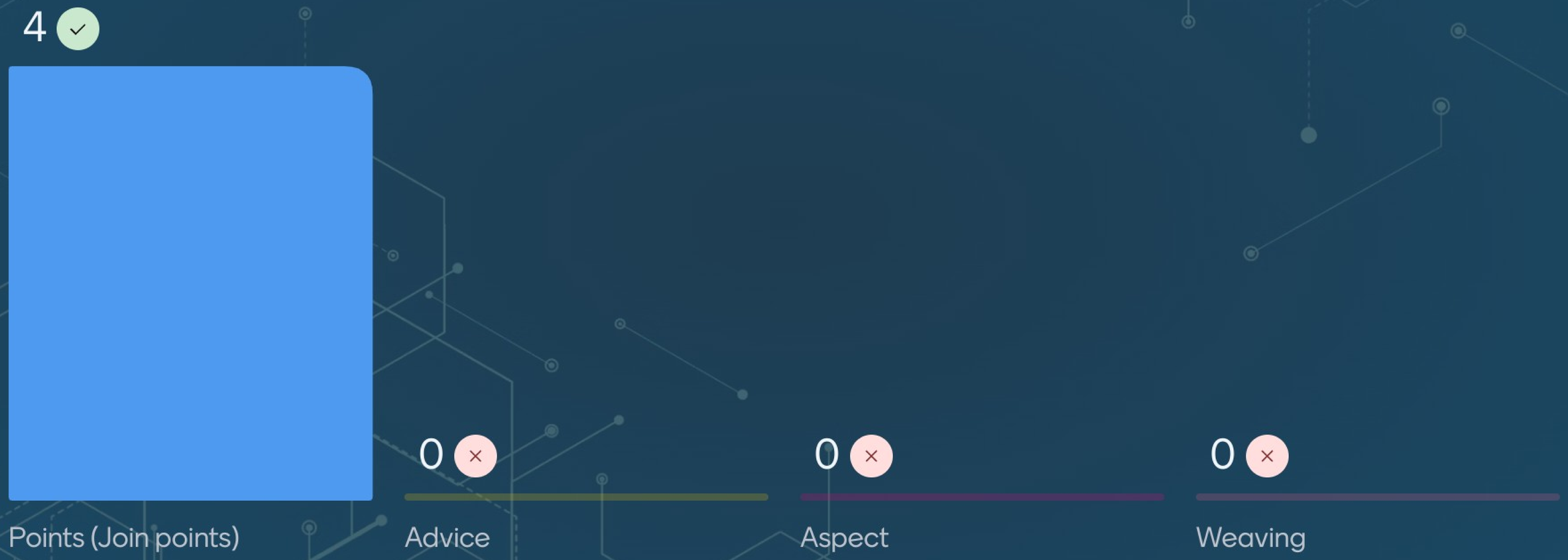
# What do you remember from the previous workshop ?

we discuss main section

1

iTPU IT PARK UNIVERSITY

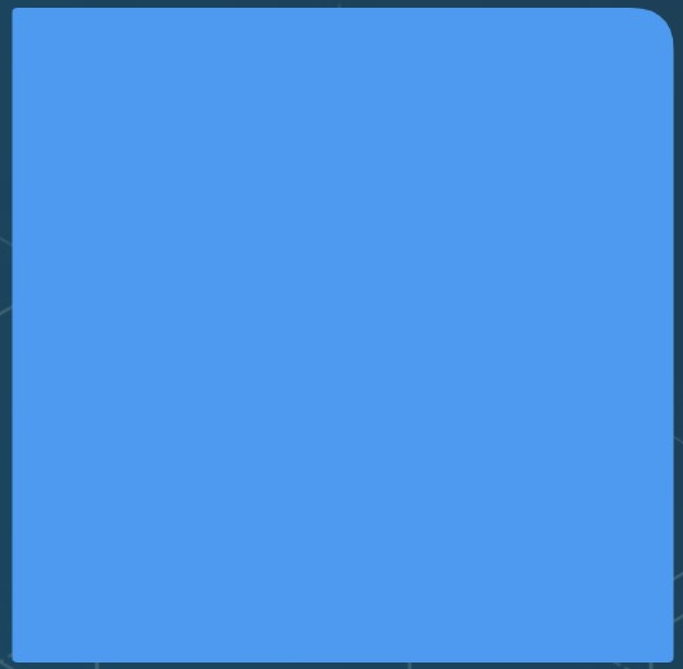
# How places in the code where the aspect is applied are called



iTPU IT PARK UNIVERSITY

# What are the differences between pointcut and join point in Spring AOP?

3



join point is a step while pointcut is an expression

0

no difference

0

join point is an expression while pointcut is a step

1 3

# What are the differences between pointcut and join point in Spring AOP?

A *join point* is a step in the program or application where you apply the AOP aspect. It can also be a specific advice execution instance.

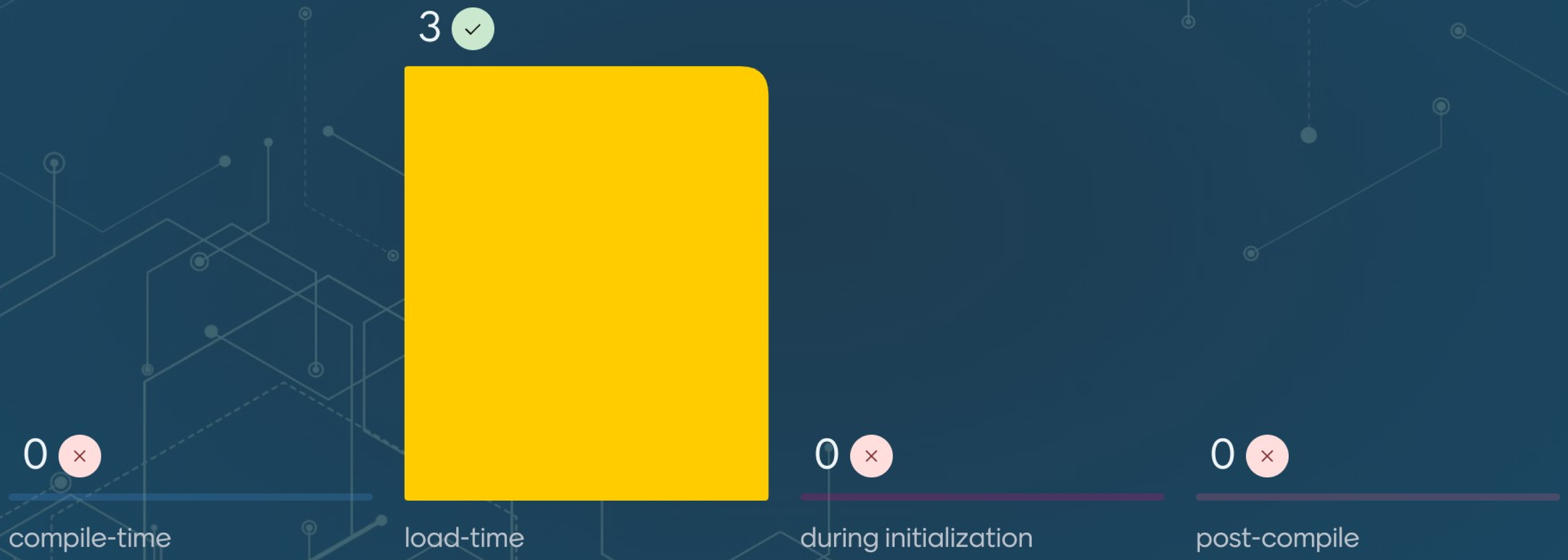
For example, it can be an execution handling, method execution or a change in the object variable's value. In the Spring framework, a join point always denotes a method execution.

A *pointcut* is an expression that uses one or more join points to give advice. You can define a pointcut using patterns or expressions.

For example, if a point cut chooses two join points, it has two expressions.

iTPU IT PARK UNIVERSITY

# What type of weaving does Spring AOP use?




3

# What type of weaving does Spring AOP use?

There are three types of weaving in AOP, including *compile-time, post-compile and load-time weaving*. Spring AOP makes use of runtime weaving, also known as load-time weaving. In runtime weaving, the proxies of the targeted object weave the aspects during the application's execution using dynamic, JDK or CGLIB proxies.

iTPU IT PARK UNIVERSITY

# How to enable advice in Spring AOP?

0 ✓ use annotation

0 ✗ run specific java compile process

0 ✓ use xml configuraion

0 ✗ using vm options

# How to enable advice in Spring AOP?

To enable advice in Spring, you can use the annotations available in AspectJ. Before executing this step, ensure that you implement the @EnableAspectJAutoProxy annotation to the configuration class. This ensures that Spring AOP can handle components denoted with AspectJ's annotations.

# Spring AOP (ws-22)

Aspect-oriented programming

Do you have questions \ uncertainties \ concerns on materials? (If no - it means all clear for you)

No we don't

1

IT PARK UNIVERSITY

# Practical tasks

IT PARK UNIVERSITY

# Spring

practice...

# When should we use AOP?

When cross-cutting concerns break OOP:

- Logging
- Measurements
- Side functionality
- Transactions
- Security

# What are the types of advice supported in Spring AOP?

There are 5 types of advice in the Spring AOP framework. They are *before, after returning, after throwing, after and around advice*.

The before advice executes before a join point, but it cannot prevent execution flow unless it throws an exception. The after-returning advice executes after a joint point finishes normally. The after-throwing advice executes if a method exits by throwing an exception. The after advice executes irrespective of how the join point exits. The around is the most powerful advice and surrounds a join point using a method invocation.

IT PARK UNIVERSITY

# Quiz

# Choose correct

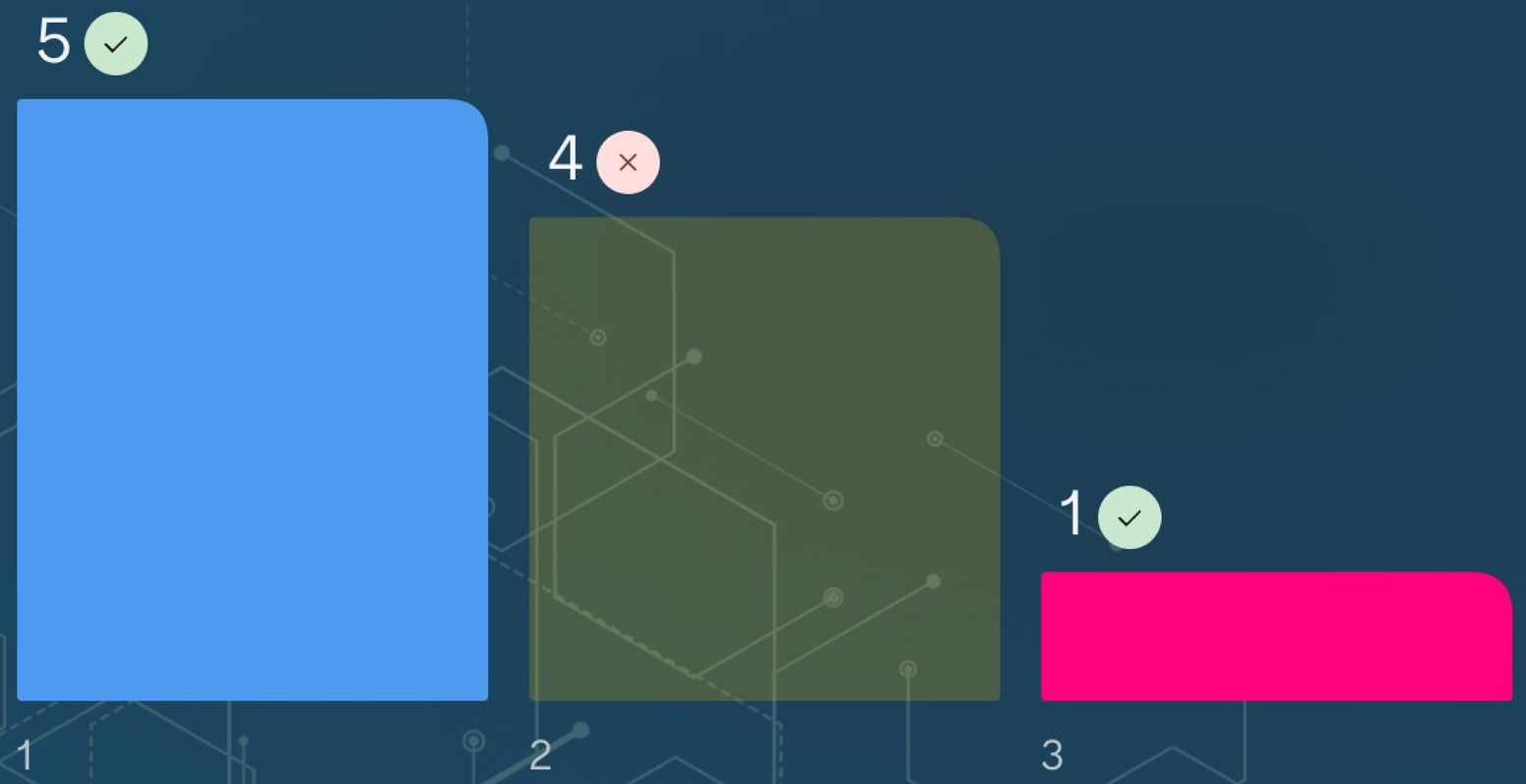
```
@Pointcut("args(String)")
```

- Invalid description
- Any method with String Arguments
- Any method with String Argument

1

6

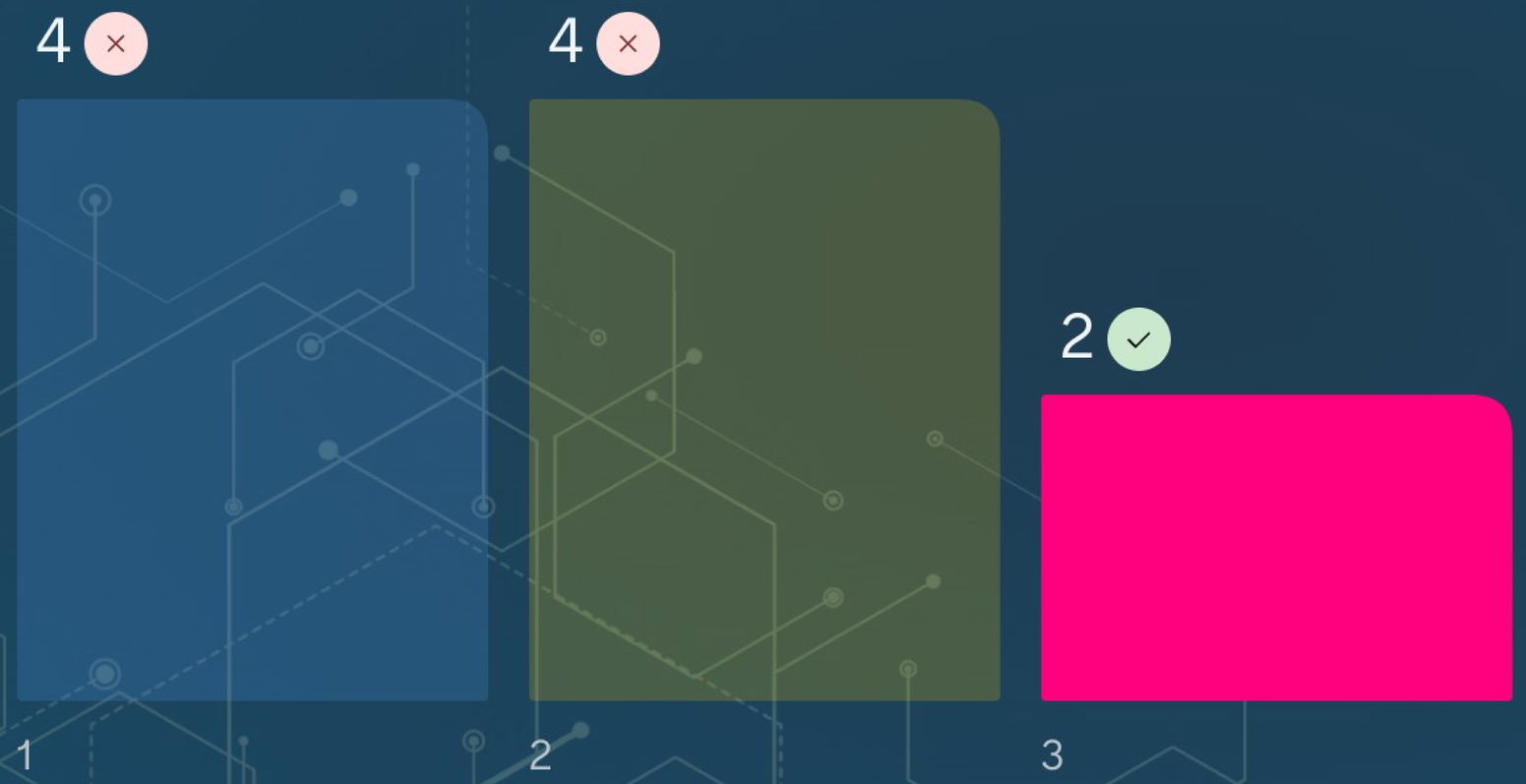
# What methods would trigger described Pointcut?




```
@Pointcut("execution(public * get*())")
_____________________________________

1. public void get()

2. public Object getField(String fieldName)

3. public List<Object> getList()
```

iTPU IT PARK UNIVERSITY

# What methods would trigger described Pointcut?




```
@Pointcut("execution(* * get*())")
_____________________________________

1. public void get()

2. public Object getField(String fieldName)

3. WILL NOT WORK
```

ITPU IT PARK UNIVERSITY

```
@Pointcut("execution(* com.arohau.MyService.*(..))")
```

4 ✗

all methods of all classes in package 'MyService'

3 ✓

all methods of class 'MyService'

3 ✗

all methods of class 'MyService' that HAS params\args

ITPU IT PARK UNIVERSITY

# @Pointcut("execution(* com.arohau.*.*(..))")

5

all methods of all classes in package 'com.arohau'

5

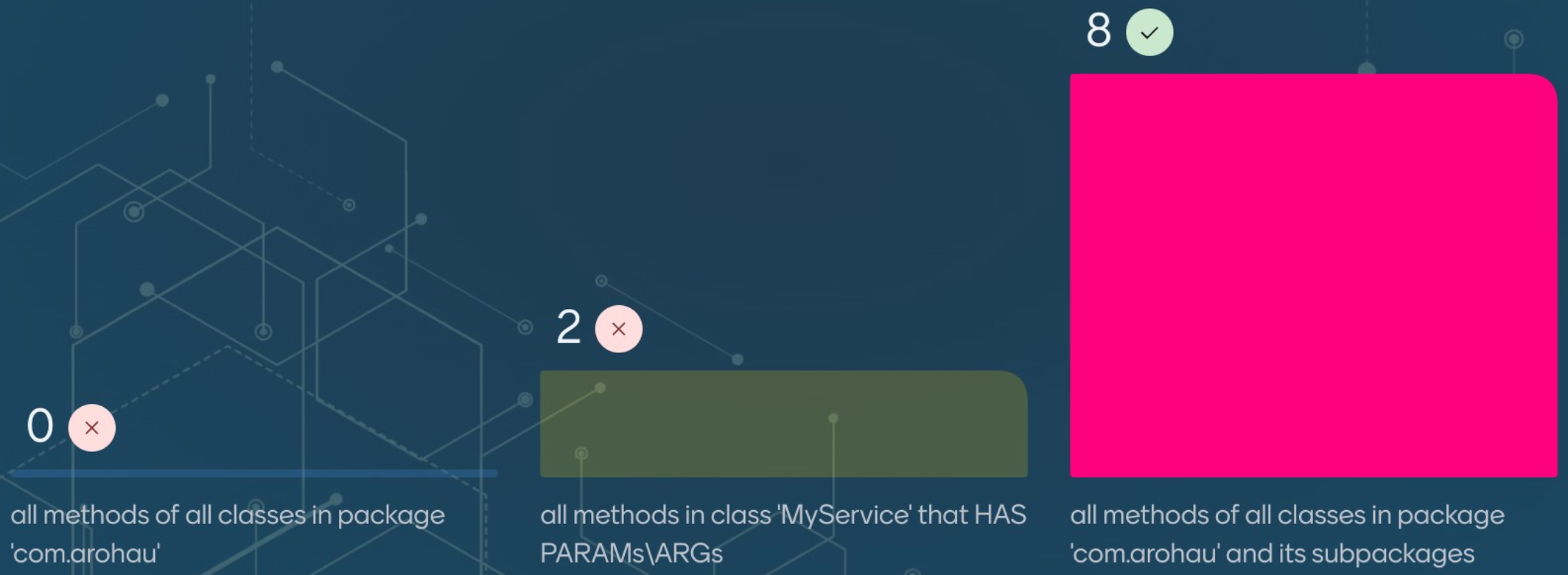
all methods of all classes in package 'com.arohau' and its subpackages

0

all methods in class 'MyService' that HAS PARAMs\ARGs

ITPU IT PARK UNIVERSITY

# @Pointcut("execution(* com.arohau..*.*(..))")



ITPU IT PARK UNIVERSITY

# @Pointcut("within(com.arohau.*Impl)")

2 ✗

invalid expression

3 ✓

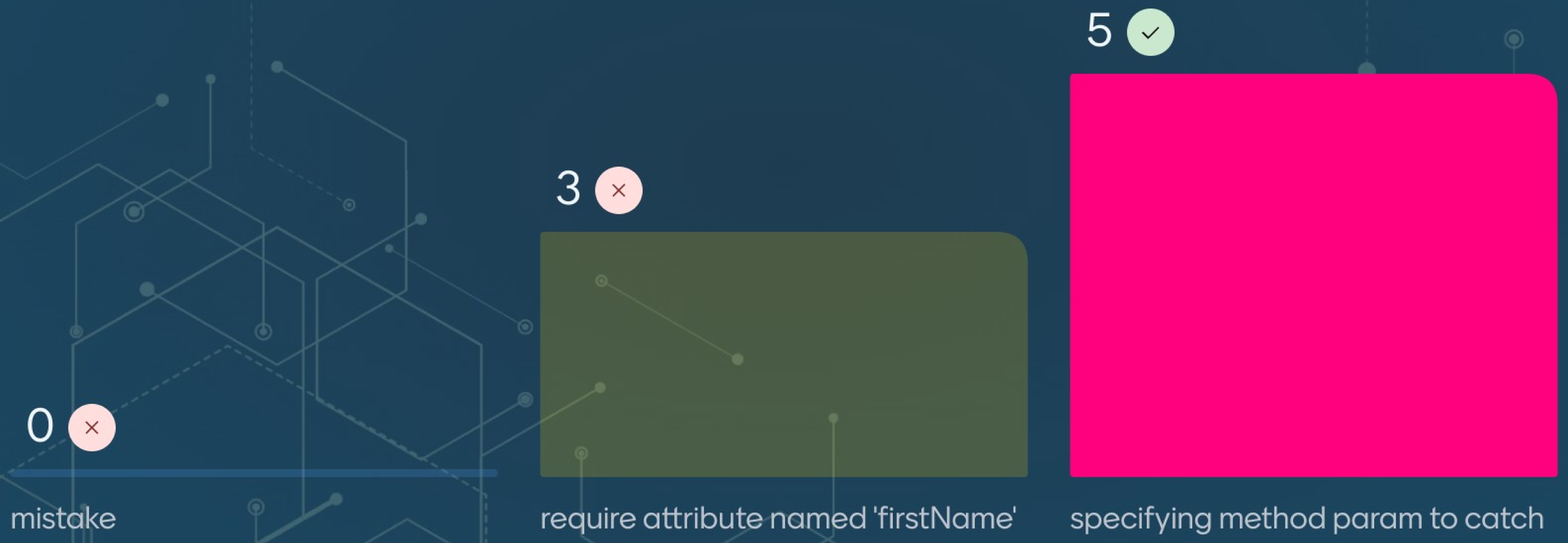
all methods in classes with 'Impl' postfix of 'com.arohau' package

4 ✗

all methods of 'com.arohau' package, which have 'Impl' postfix

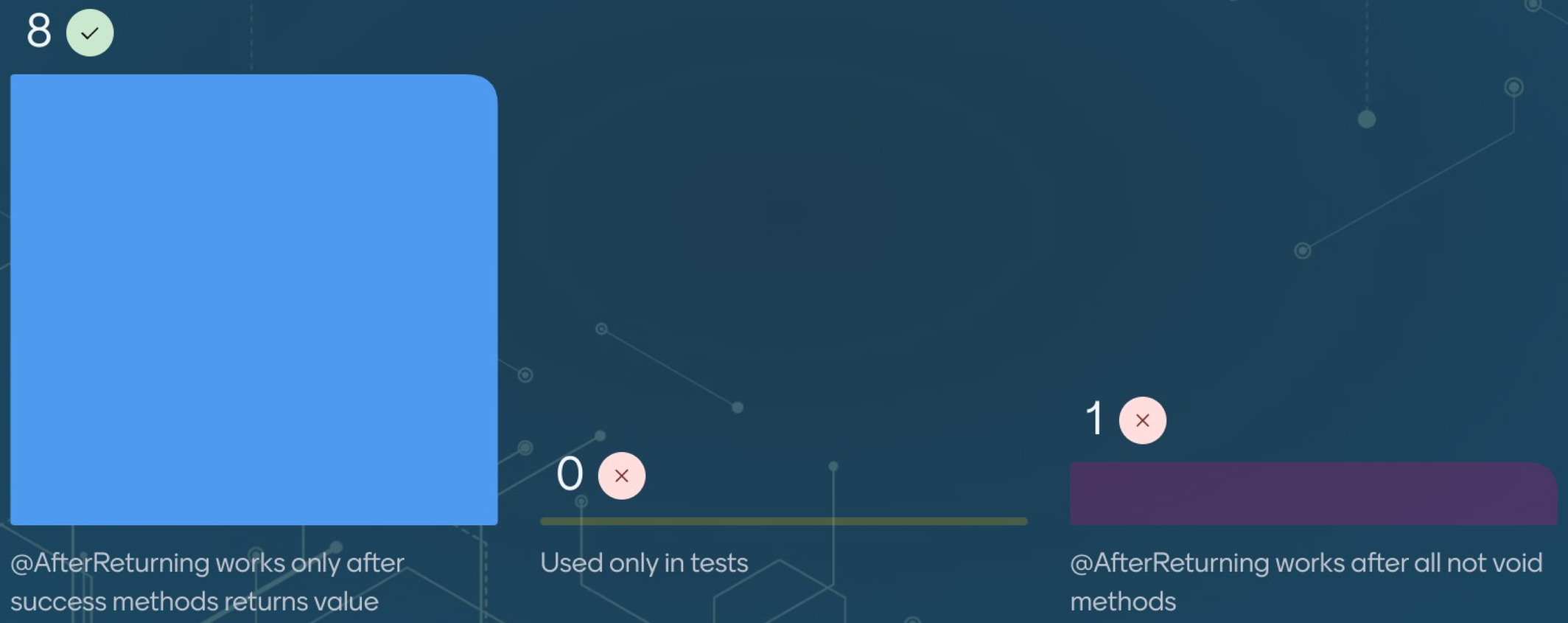
iTPU IT PARK UNIVERSITY

# @Pointcut("args(firstName)") why 'firstName' is used?



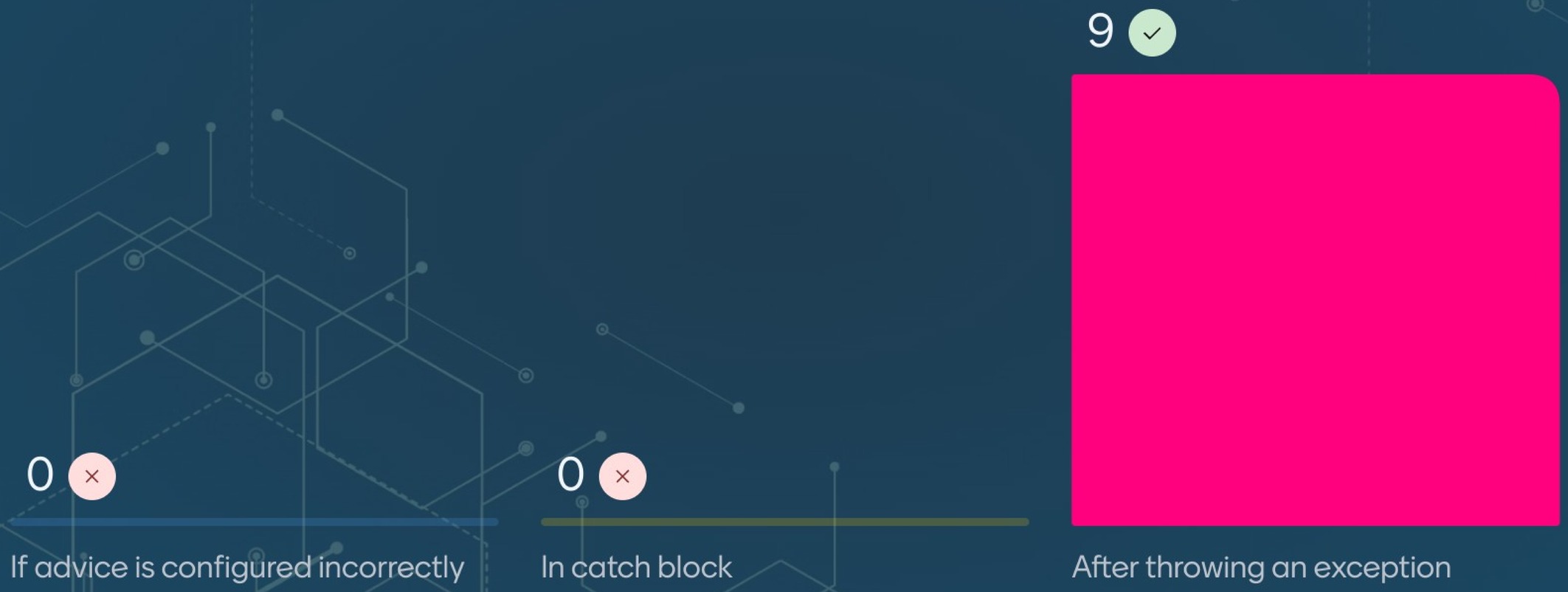

1

# True about @AfterReturning ...



iTPU IT PARK UNIVERSITY

# When @AfterThrowing activates?



# @After is considered as

5

a case of finally block

4

a case happens last after all advices worked

# Choose correct

```
@AfterReturning(pointcut = "args(name)", returning = "returnValue")
public void stringArgMethods(String name, String returnValue) {}
```

difference

```
@AfterReturning(pointcut = "args(name)", returning = "returnValue")
public void stringArgMethods(String name, Object returnValue) {}
```

Same thing

First is incorrect

Second is more abstract

3

8

# Choose correct

```
@AfterThrowing(pointcut = "args(name)", throwing = "ex")
public void stringArgMethods(String name, RuntimeException ex) {}
```

- Any method with String arg and throwing RuntimeException
- Any method with String arg that has finished processing with RuntimeException
- Any method with String and RuntimeException arguments

# Choose correct

```
@Around("execution(public * get*())")
public Object aroundAdvice(ProceedingJoinPoint pjp) {
    Object returnValue = pjp.proceed();
    return returnValue;
}
```

Option to override result 2

Opti to l esult 4

Option to validate result

9

# Quiz leaderboard

# Next time: Spring Security

# Questions?

ITPU IT PARK UNIVERSITY