Pleiad

Programming Languages and Environments for Intelligent, Adaptable and Distributed systems

Departamento de Ciencias de la Computación
UNIVERSIDAD DE CHILE

FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE

A Semantics for Execution Levels with Exceptions

Ismael Figueroa
Éric Tanter

Pleiad Laboratory
Department of Computer Science
University of Chile - Chile

2011

Aspects and Exceptions

Researchers have studied exception handling as a crosscutting concern

Lippert and Lopez (ICSE '00)

Castor *et.al.* (SIGSOFT '06, ICSM '07, SPE '09)

Coelho *et.al.* (PLoP '08, ECOOP '08)

Pleiad

Confusion with Exceptions and Aspects

```
class A {
  public void foo() {
    Integer configValue;
    try {
       configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```

Confusion with Exceptions and Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig()){
    log.append("getConfig");
    return proceed();}}
```

```
class A {
  public void foo() {
    Integer configValue;
    try {
      configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```

Pleiad

Confusion with Exception-Handling Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig())
{   log.append("getConfig'');
    return proceed();}}
```

```
class A {
  public void foo() {
    Integer configValue;
    try {
       configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```

Pleiad

Confusion with Exception-Handling Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig())
{   log.append("getConfig'');
    return proceed();}}
```

```
class A {
  public void foo() {
    Integer configValue;
    try {
       configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```

Confusion with Exception-Handling Aspects

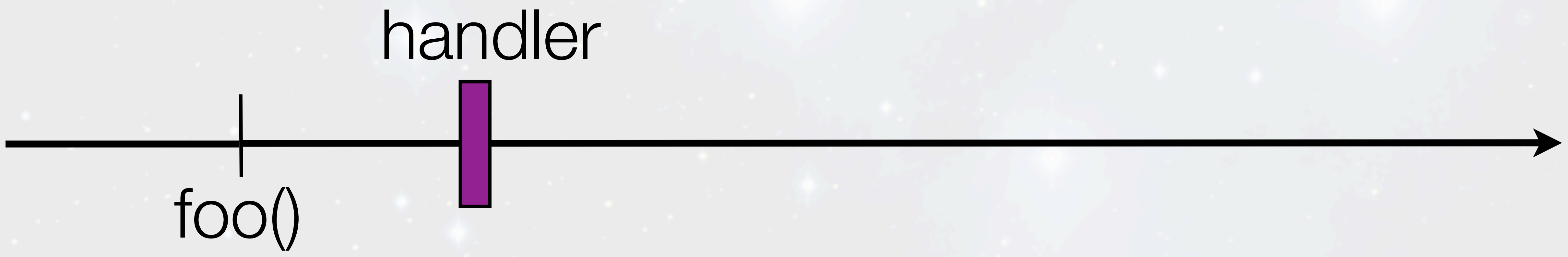
```
aspect Logging {
  Object around() : call(Integer getConfig())
{   log.append("getConfig");
    return proceed();}}
```

```
class A {
  public void foo() {
    Integer configValue;
    try {
       configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```

Confusion with Exception-Handling Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig())
{   log.append("getConfig");
    return proceed();}}
```

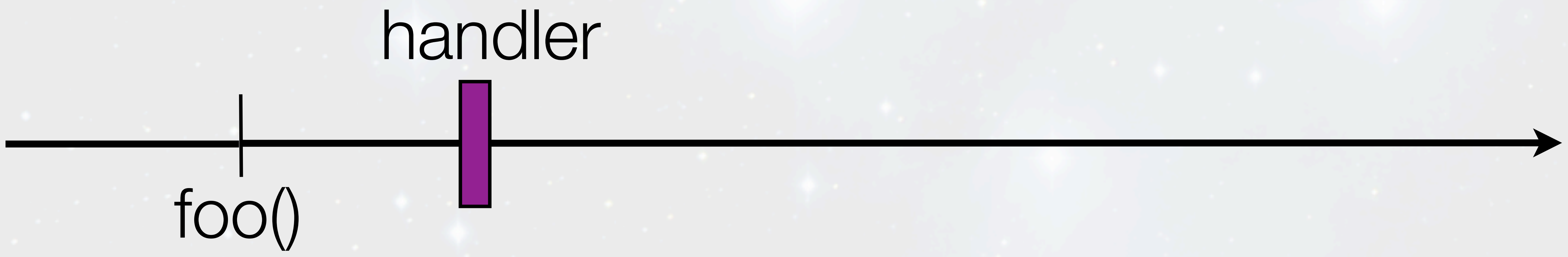
```
class A {
  public void foo() {
    Integer configValue;
    try {
      configValue = getConfig();
    } catch(Exception ex) {
     configValue = DEFAULT; }}}
```



Confusion with Exception-Handling Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig())
{   log.append("getConfig");
    return proceed();}}
```

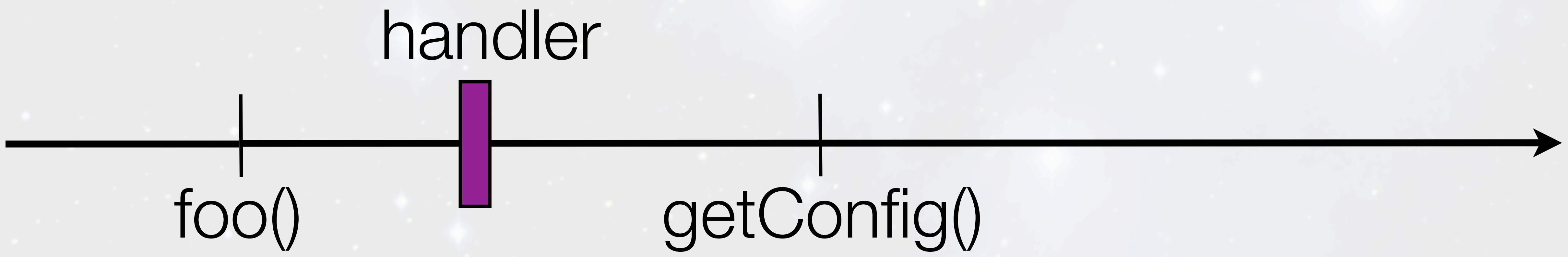
```
class A {
  public void foo() {
    Integer configValue;
    try {
      configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```



Confusion with Exception-Handling Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig())
{   log.append("getConfig");
    return proceed();}}
```

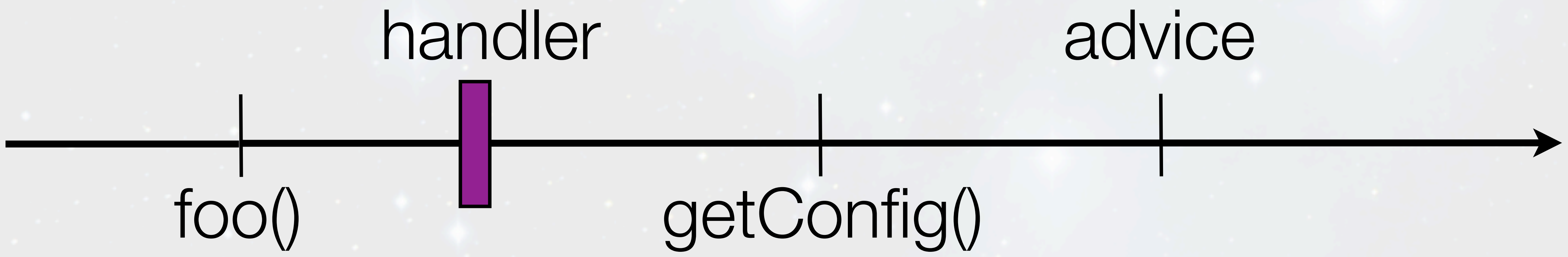
```
class A {
  public void foo() {
    Integer configValue;
    try {
      configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```



Confusion with Exception-Handling Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig())
{   log.append("getConfig");
    return proceed();}}
```

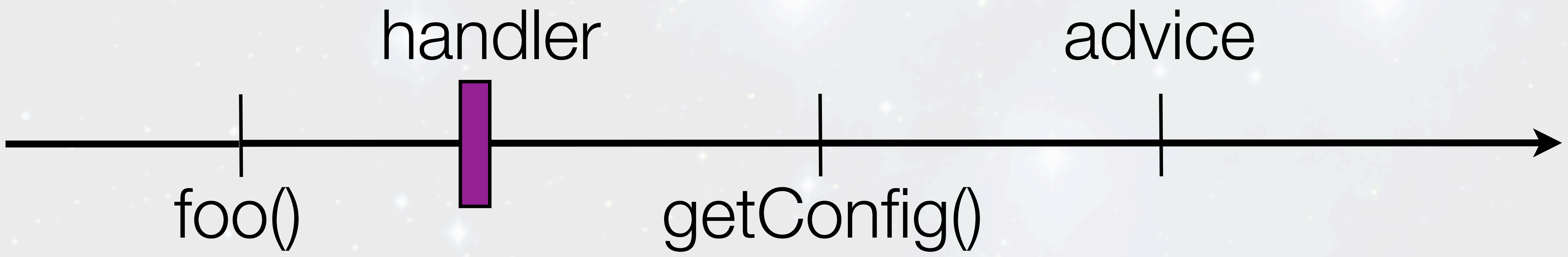
```
class A {
  public void foo() {
    Integer configValue;
    try {
      configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```



Confusion with Exception-Handling Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig())
{   log.append("getConfig");
    return proceed();}}
```

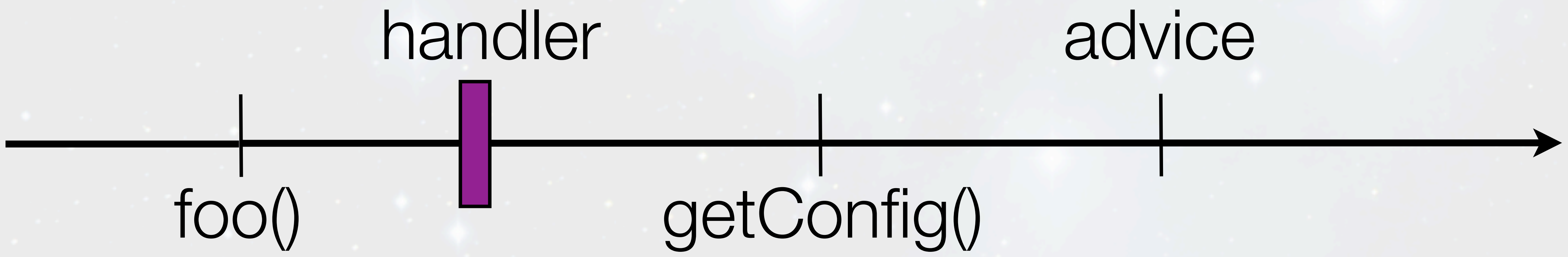
```
class A {
  public void foo() {
    Integer configValue;
    try {
      configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```



Confusion with Exception-Handling Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig())
{   log.append("getConfig");
    return proceed();}}
```

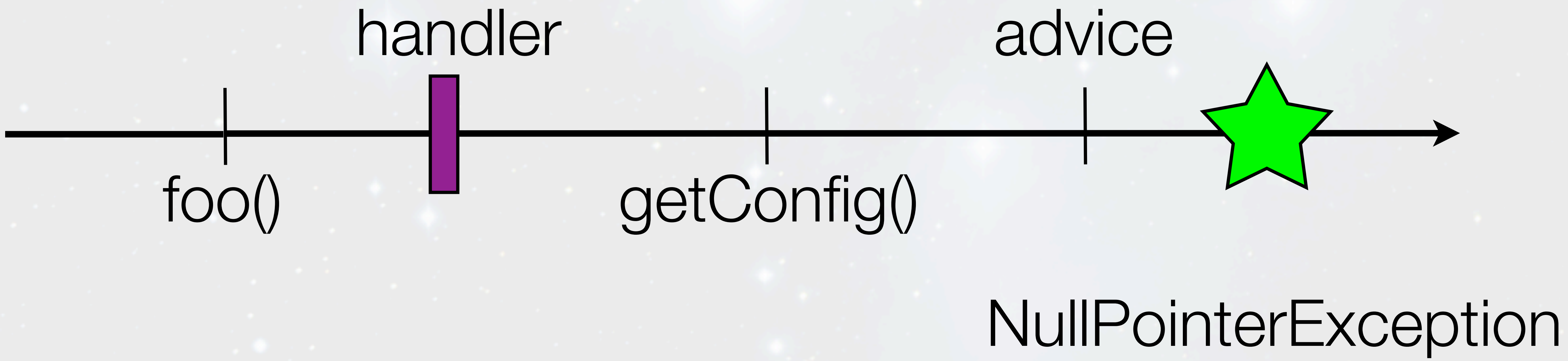
```
class A {
  public void foo() {
    Integer configValue;
    try {
      configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```



Confusion with Exception-Handling Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig())
{   log.append("getConfig");
    return proceed();}}
```

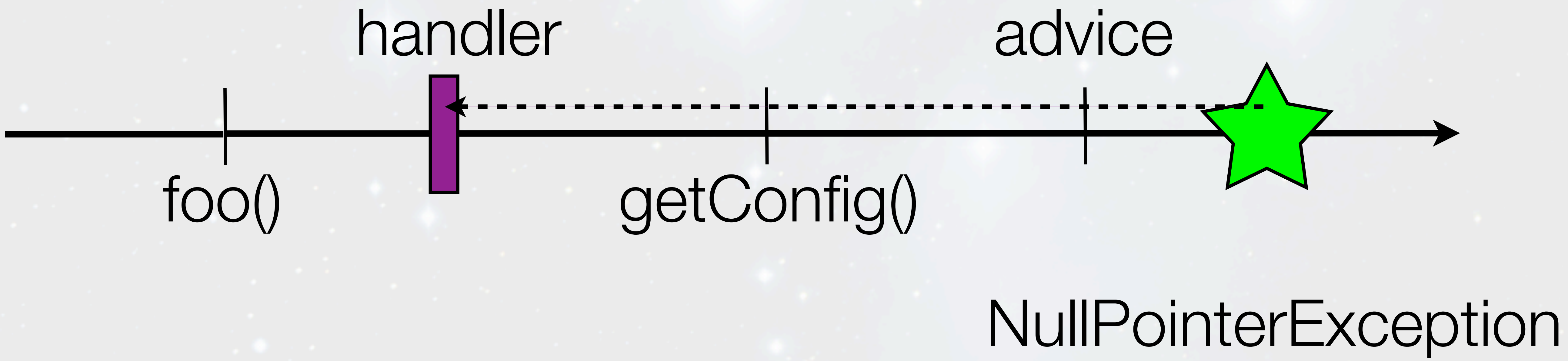
```
class A {
  public void foo() {
    Integer configValue;
    try {
      configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```



Confusion with Exception-Handling Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig())
{   log.append("getConfig");
    return proceed();}}
```

```
class A {
  public void foo() {
    Integer configValue;
    try {
      configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```

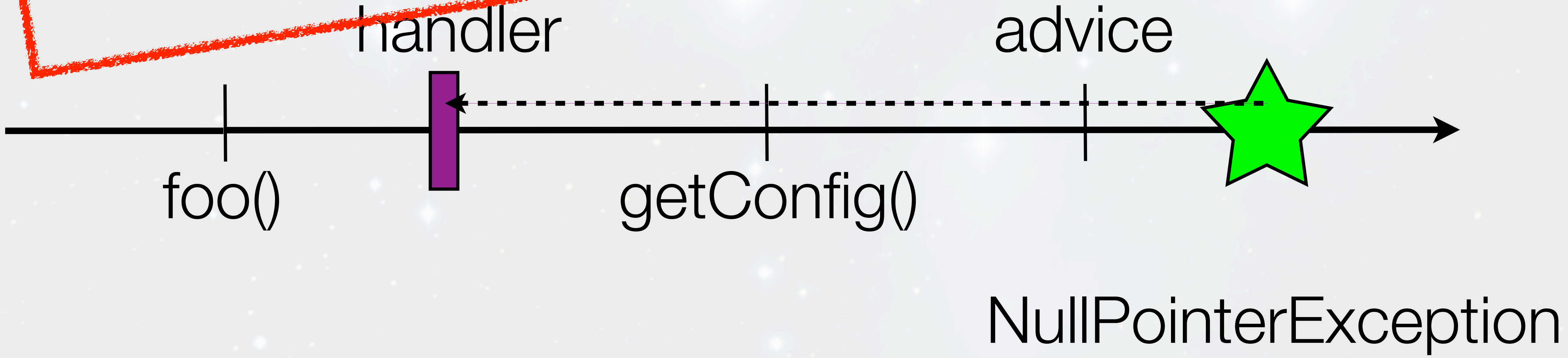


Confusion with Exception-Handling Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig())
{  log.append("getConfig");
   return proceed();}}
```

```
class A {
  public void foo() {
    Integer configValue;
    try {
      configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```

Default value used! even if getConfig returns OK!



Confusion with Exception-Handling Aspects

Aspect exception was handled as a base exception

Confusion with Exceptions and Aspects

```
class A {
  public void foo() {
    Integer configValue;
    try {
       configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```

Pleiad

Confusion with Exceptions and Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig()){
    try {
      log.append("getConfig'');
      return proceed();
    } catch (Exception ex) { System.exit(-1); }}}
```

```
class A {
  public void foo() {
    Integer configValue;
    try {
      configValue = getConfig();
    } catch(Exception ex) {
     configValue = DEFAULT; }}}
```

Confusion with Exception-Handling Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig()){
    try {
      log.append("getConfig");
      return proceed();
    } catch (Exception ex) {
      System.exit(-1);
    }}}
```

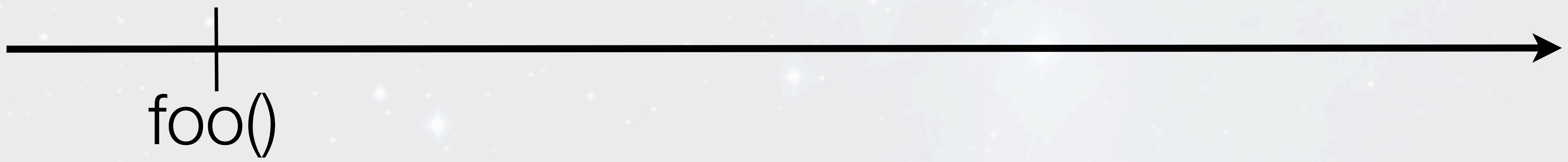
```
class A {
  public void foo() {
    Integer configValue;
    try {
      configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```

Pleiad

Confusion with Exception-Handling Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig()){
    try {
      log.append("getConfig");
      return proceed();
    } catch (Exception ex) {
      System.exit(-1);
    }}}
```

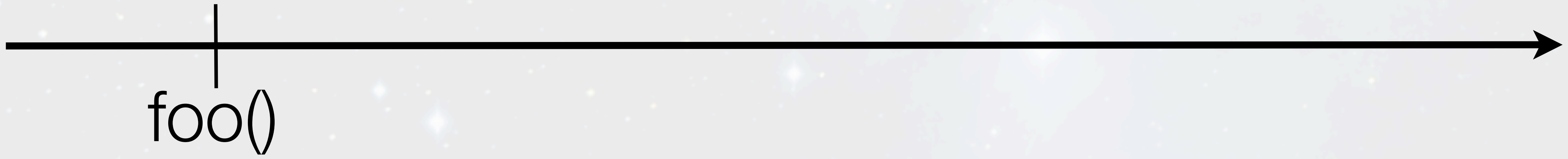
```
class A {
  public void foo() {
    Integer configValue;
    try {
       configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```



Confusion with Exception-Handling Aspects

```
aspect Logging {
 Object around() : call(Integer getConfig()){
  try {
   log.append("getConfig");
   return proceed();
  } catch (Exception ex) {
   System.exit(-1);
  }}}
```

```
class A {
 public void foo() {
  Integer configValue;
  try {
    configValue = getConfig();
  } catch(Exception ex) {
   configValue = DEFAULT; }}}
```

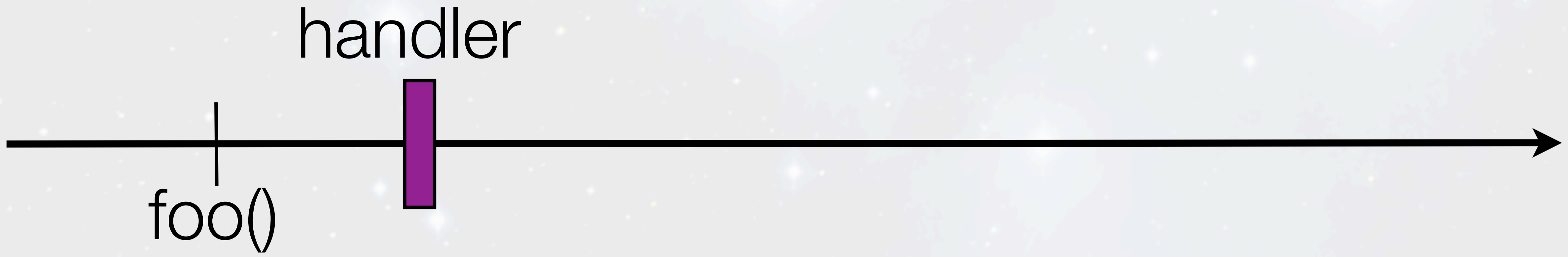


Pleiad

Confusion with Exception-Handling Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig()){
    try {
      log.append("getConfig'');
      return proceed();
    } catch (Exception ex) {
      System.exit(-1);
    }}}
```

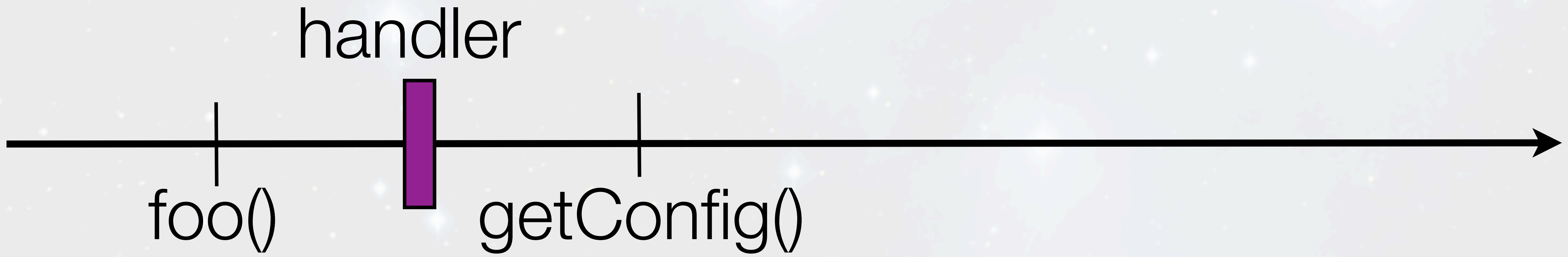
```
class A {
  public void foo() {
    Integer configValue;
    try {
      configValue = getConfig();
    } catch(Exception ex) {
     configValue = DEFAULT; }}}
```



Confusion with Exception-Handling Aspects

```
aspect Logging {
 Object around() : call(Integer getConfig()){
  try {
   log.append("getConfig");
   return proceed();
  } catch (Exception ex) {
   System.exit(-1);
  }}}
```

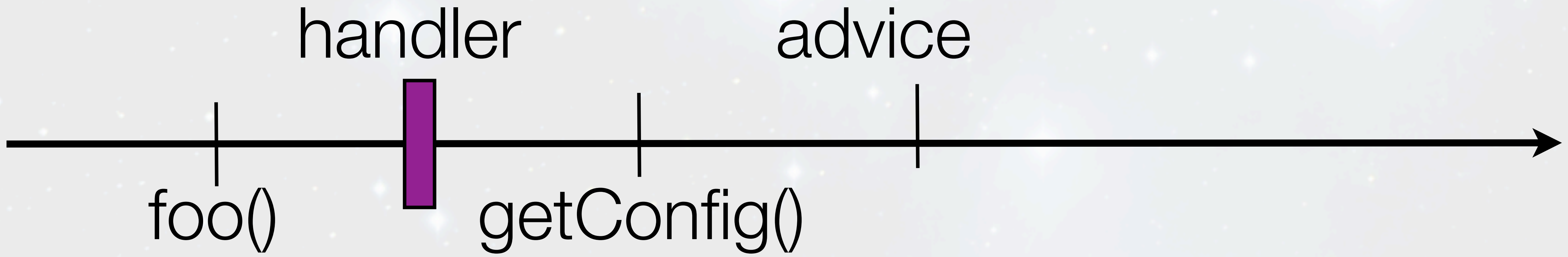
```
class A {
 public void foo() {
  Integer configValue;
  try {
    configValue = getConfig();
  } catch(Exception ex) {
   configValue = DEFAULT; }}}
```



Confusion with Exception-Handling Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig()){
    try {
      log.append("getConfig'');
      return proceed();
    } catch (Exception ex) {
      System.exit(-1);
    }}}
```

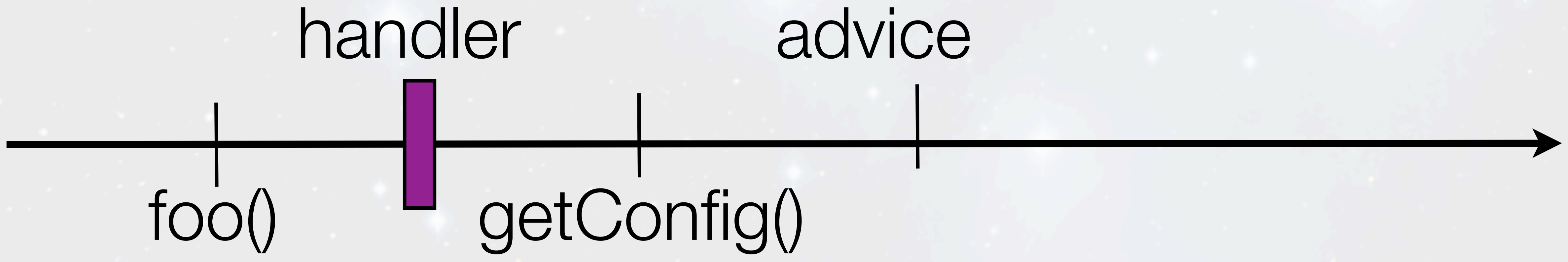
```
class A {
  public void foo() {
    Integer configValue;
    try {
      configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```



Confusion with Exception-Handling Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig()){
   try {
     log.append("getConfig'');
     return proceed();
   } catch (Exception ex) {
     System.exit(-1);
   }}}
```

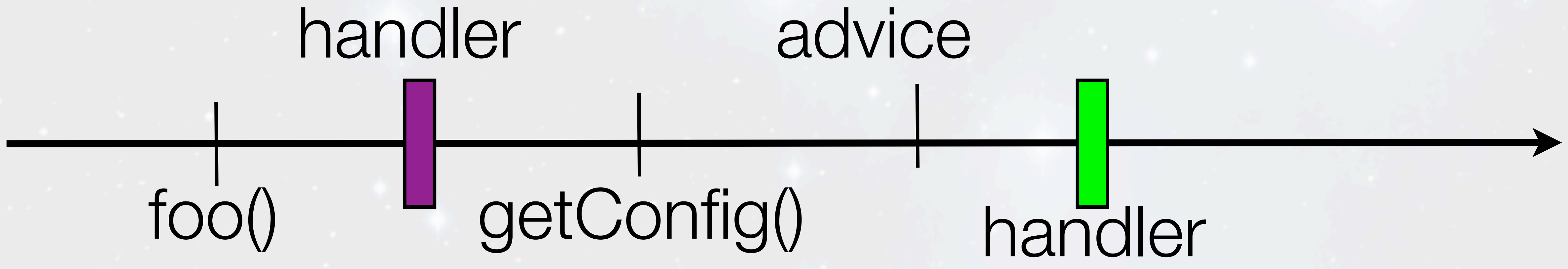
```
class A {
  public void foo() {
    Integer configValue;
    try {
       configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```



Confusion with Exception-Handling Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig()){
    try {
      log.append("getConfig");
      return proceed();
    } catch (Exception ex) {
      System.exit(-1);
    }}}
```

```
class A {
  public void foo() {
    Integer configValue;
    try {
      configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```

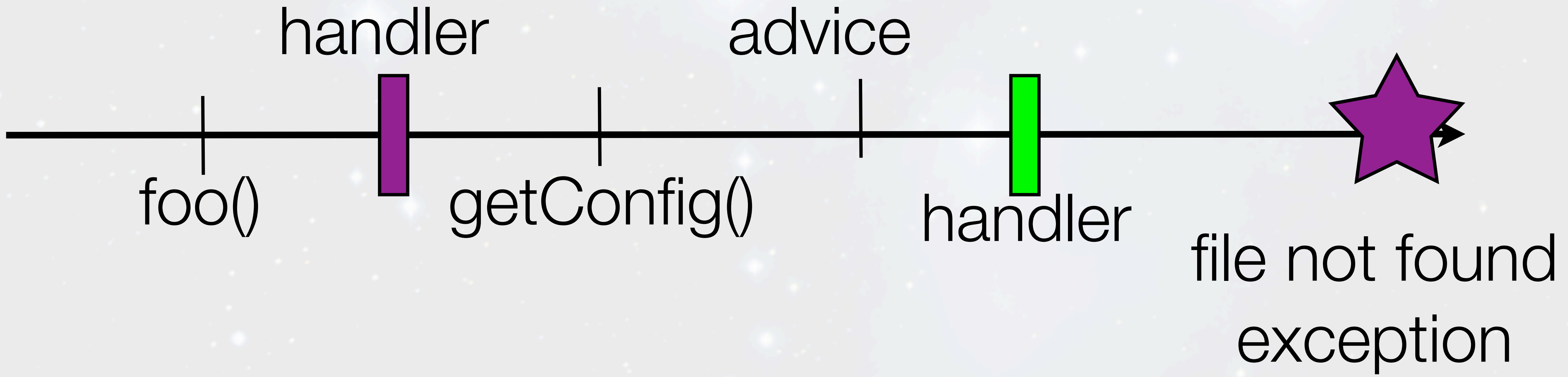


Pleiad

Confusion with Exception-Handling Aspects

```
aspect Logging {
 Object around() : call(Integer getConfig()){
  try {
    log.append("getConfig");
    return proceed();
  } catch (Exception ex) {
    System.exit(-1);
  }}}
```

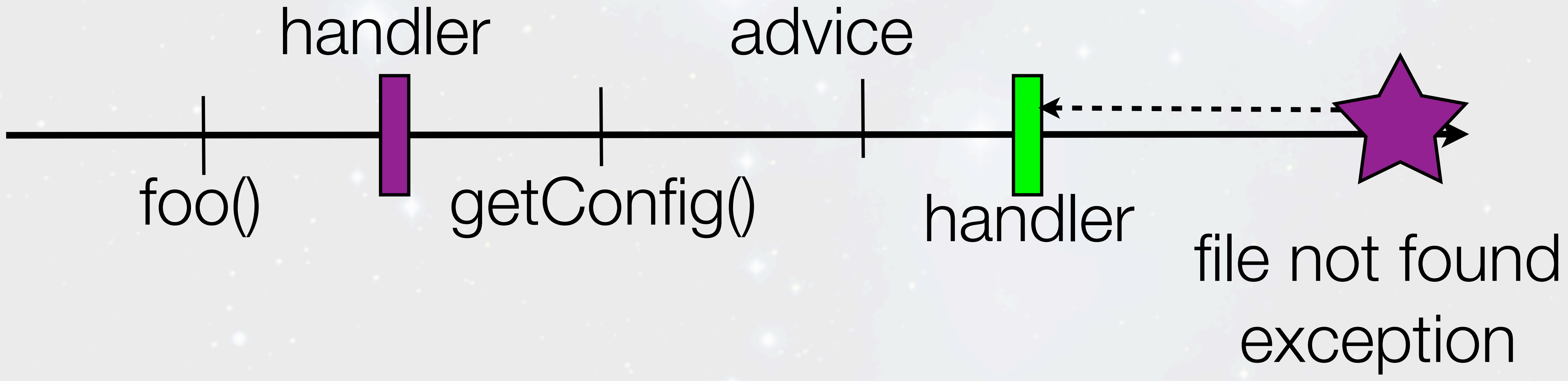
```
class A {
 public void foo() {
  Integer configValue;
  try {
    configValue = getConfig();
  } catch(Exception ex) {
   configValue = DEFAULT; }}}
```



Confusion with Exception-Handling Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig()){
   try {
     log.append("getConfig");
     return proceed();
   } catch (Exception ex) {
     System.exit(-1);
   }}}
```

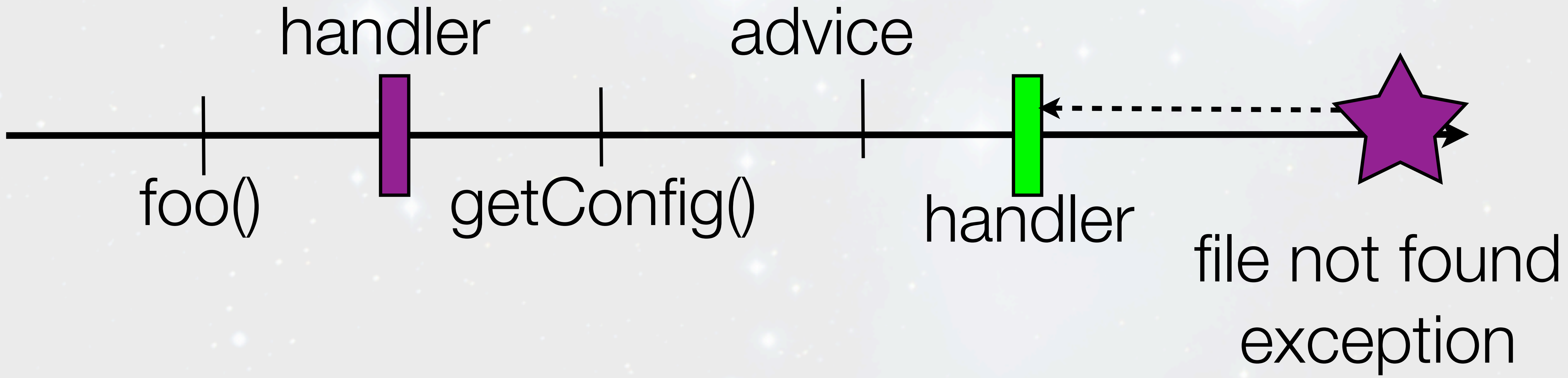
```
class A {
  public void foo() {
    Integer configValue;
    try {
      configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```



Confusion with Exception-Handling Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig()){
    try {
      log.append("getConfig'');
      return proceed();
    } catch (Exception ex) {
      System.exit(-1);
    }}}
```

```
class A {
  public void foo() {
    Integer configValue;
    try {
      configValue = getConfig();
    } catch(Exception ex) {
      configValue = DEFAULT; }}}
```

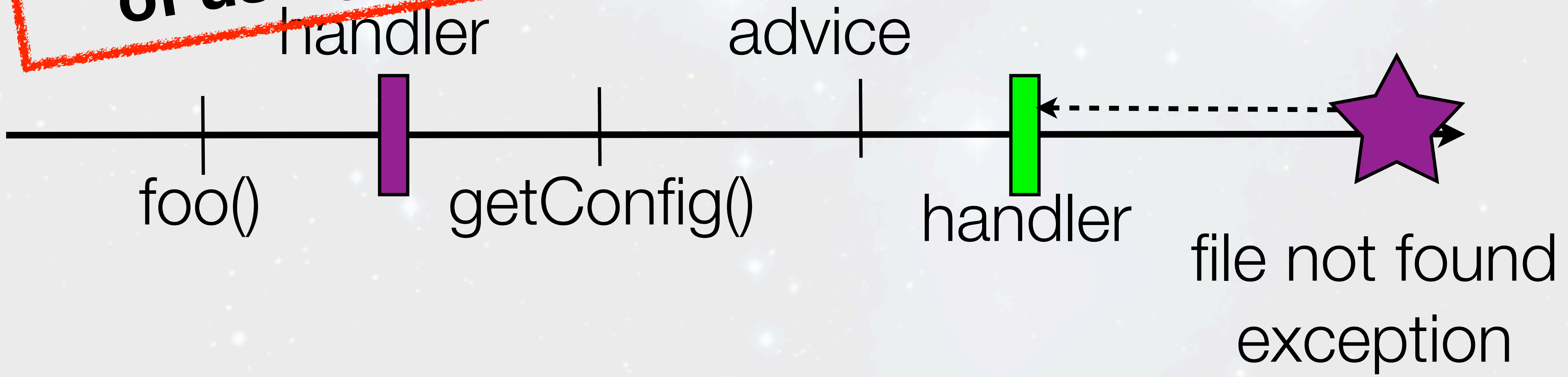


Confusion with Exception-Handling Aspects

```
aspect Logging {
  Object around() : call(Integer getConfig()){
   try {
     log.append("getConfig");
     return proceed();
   } catch (Exception ex) {
     System.exit(-1);
   }}}
```

```
class A {
  public void foo() {
   Integer configValue;
   try {
     configValue = getConfig();
   } catch(Exception ex) {
     configValue = DEFAULT; }}}
```

Program aborted instead of using default value!



Confusion with Exception-Handling Aspects

Base exception was handled as an aspect exception

Exception Conflation Problem

Exception Conflation Problem

Exception handling mechanism merges aspect and base handlers and exceptions in a flat control flow

Exception Conflation Problem

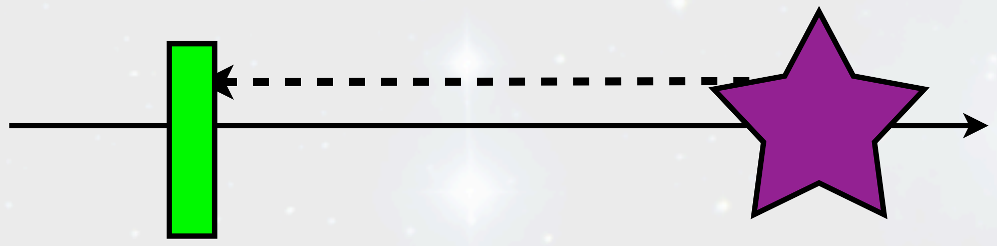
Exception handling mechanism merges aspect and base handlers and exceptions in a flat control flow

Based on Coelho *et.al.* bug patterns (PLoP ’08)

Exception Conflation Problem

Exception handling mechanism merges aspect and base handlers and exceptions in a flat control flow

Based on Coelho *et.al.* bug patterns (PLoP ’08)

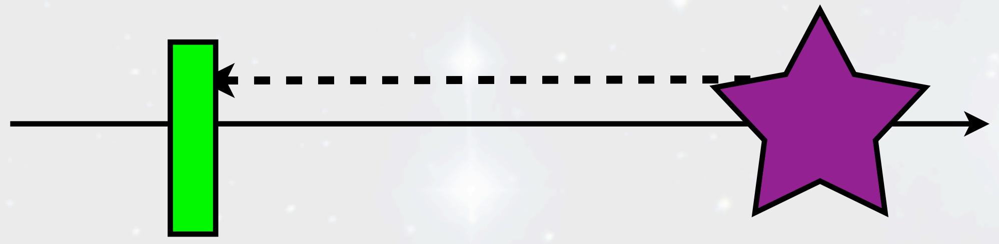


base exception/aspect handler

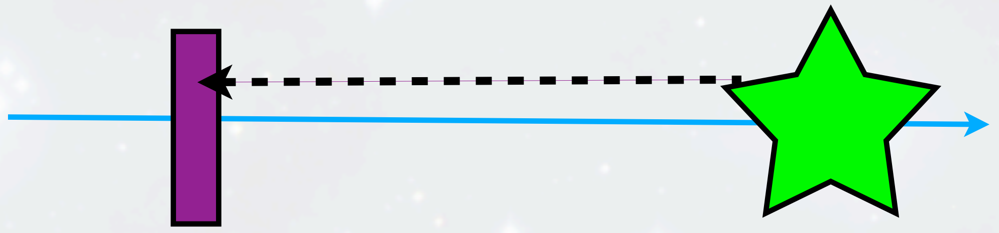
Exception Conflation Problem

Exception handling mechanism merges aspect and base handlers and exceptions in a flat control flow

Based on Coelho *et.al.* bug patterns (PLoP ’08)



base exception/aspect handler

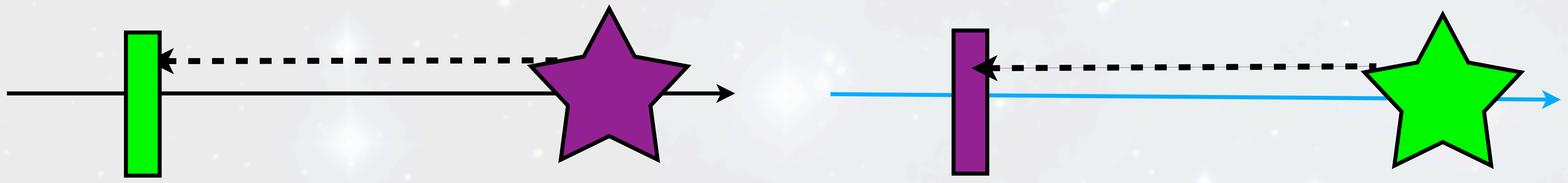


aspect exception/base handler

Exception Conflation Problem

Exception handling mechanism merges aspect and base handlers and exceptions in a flat control flow

Based on Coelho *et.al.* bug patterns (PLoP ’08)



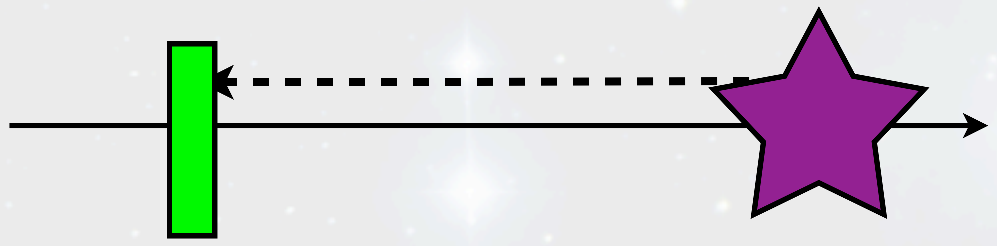
base exception/aspect handler **aspect exception/base handler**

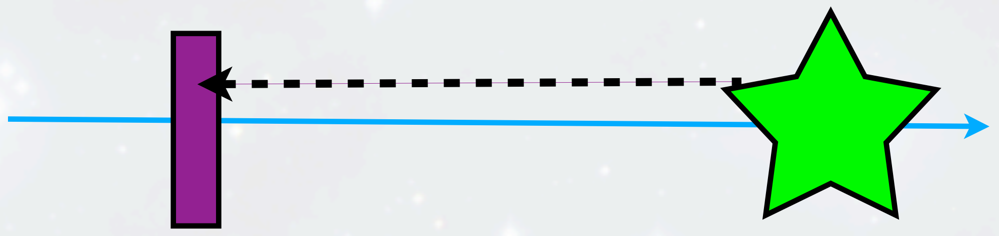
Inadvertent execution of unintended handlers

Exception Conflation Problem

Exception handling mechanism merges aspect and base handlers and exceptions in a flat control flow

Based on Coelho *et.al.* bug patterns (PLoP '08)





base exception/aspect handler **aspect exception/base handler**

Inadvertent execution of unintended handlers

Programmers can not state the desired interaction

Pleiad

Our Solution

Use Execution Levels to control Exception Conflation

Execution Levels in a Nutshell

Tanter (AOSD ’10)

Execution Levels in a Nutshell

Tanter (AOSD '10)

Level 2

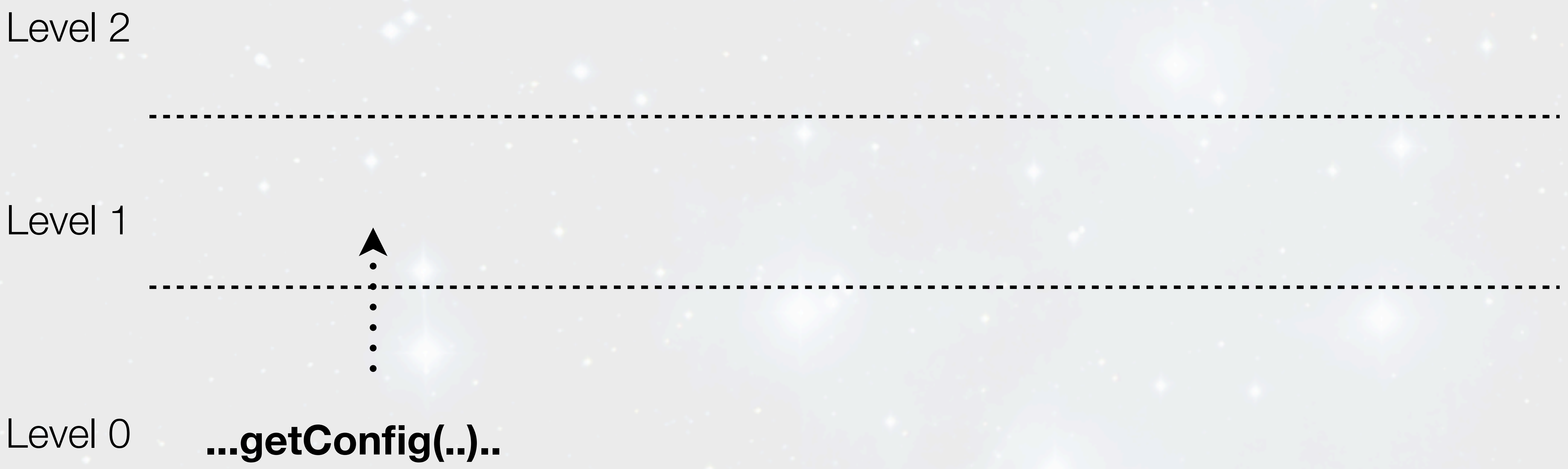
Level 1

Level 0 **...getConfig(..)..**

Pleiad 11

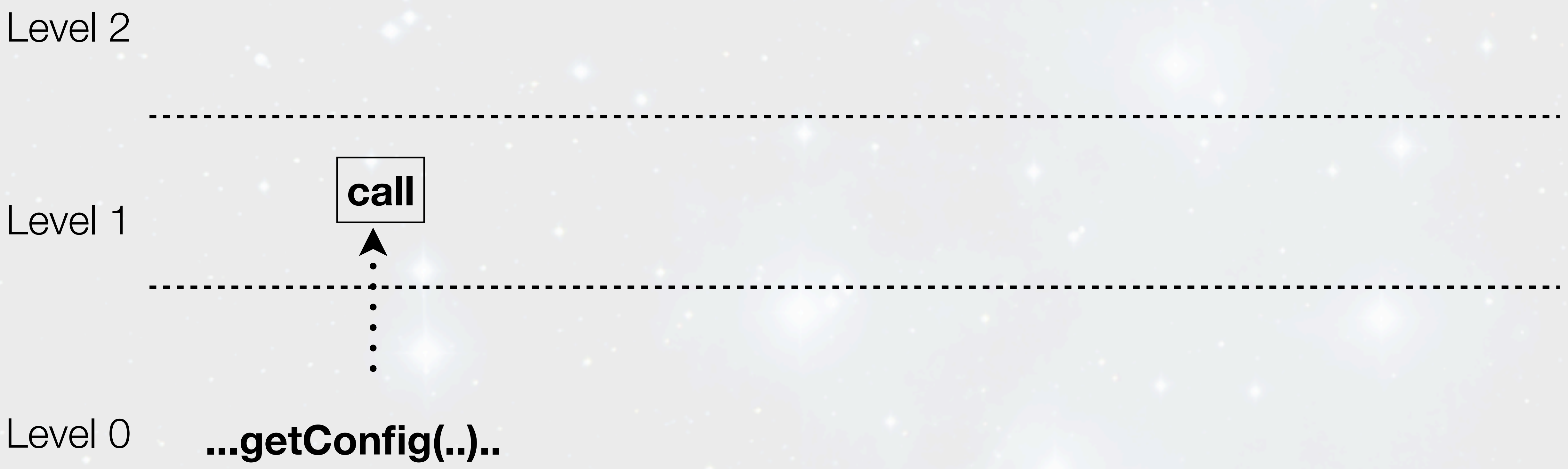
Execution Levels in a Nutshell

Tanter (AOSD ’10)



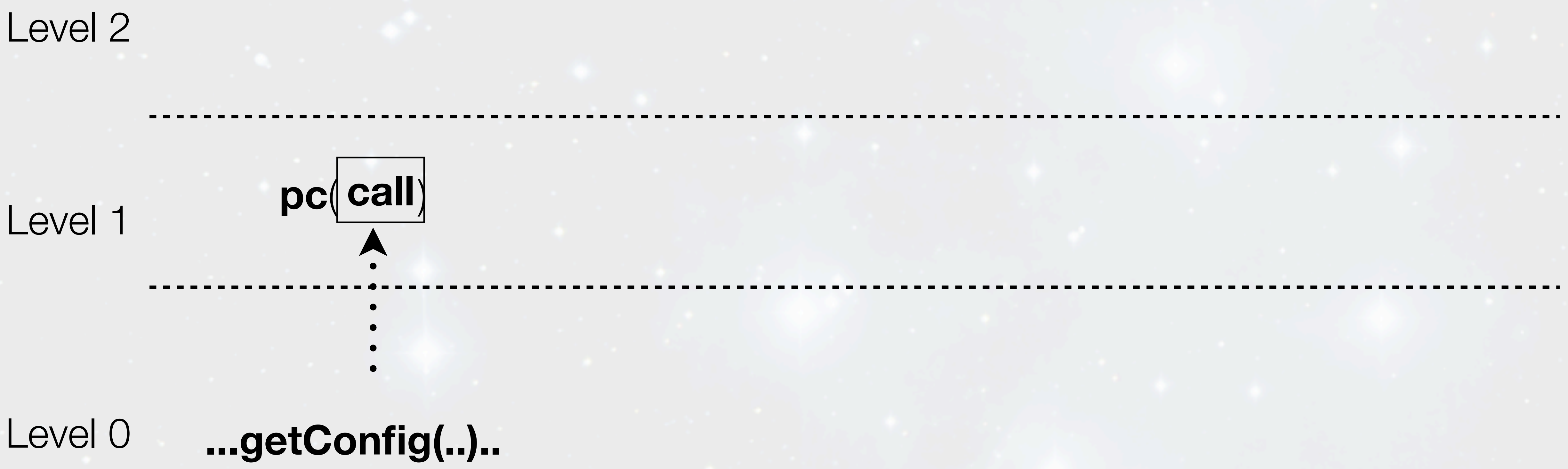
Execution Levels in a Nutshell

Tanter (AOSD '10)



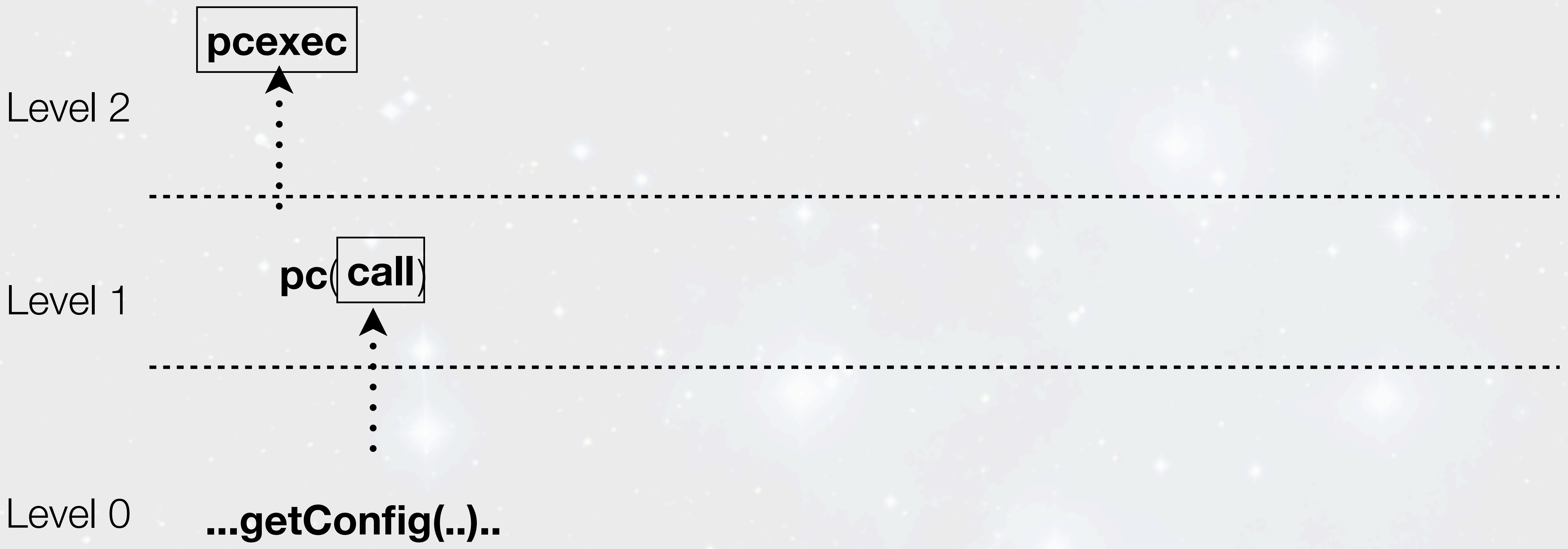
Execution Levels in a Nutshell

Tanter (AOSD ’10)



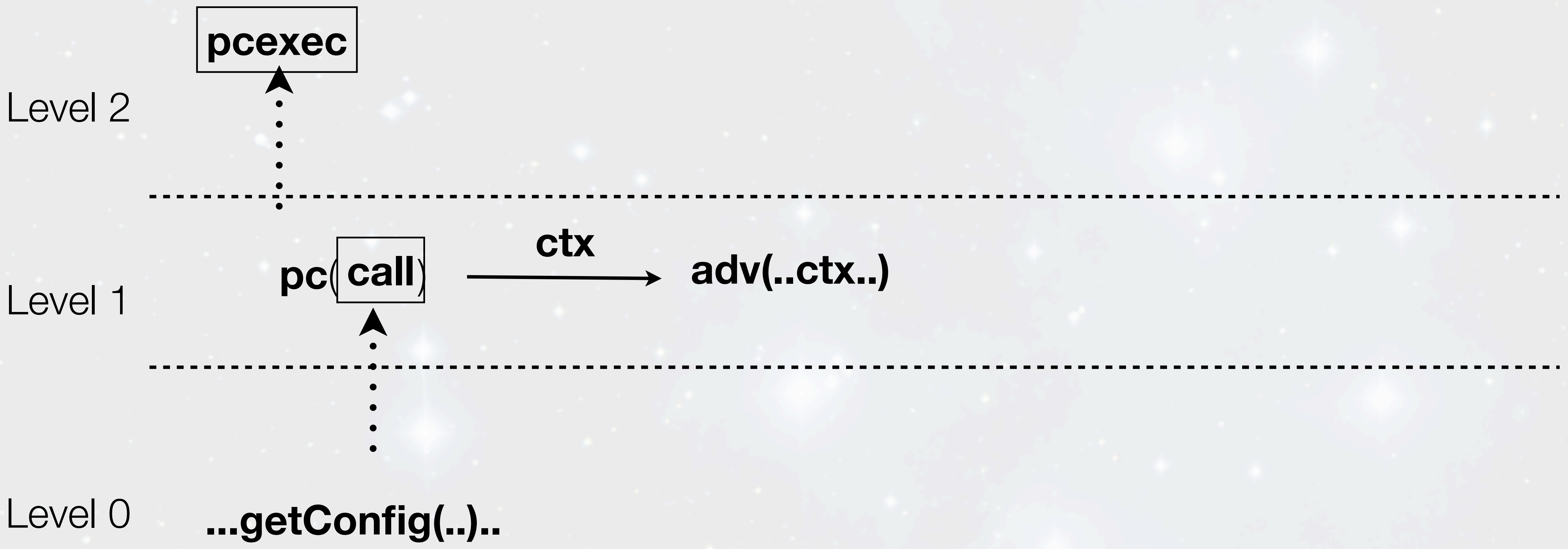
Execution Levels in a Nutshell

Tanter (AOSD '10)



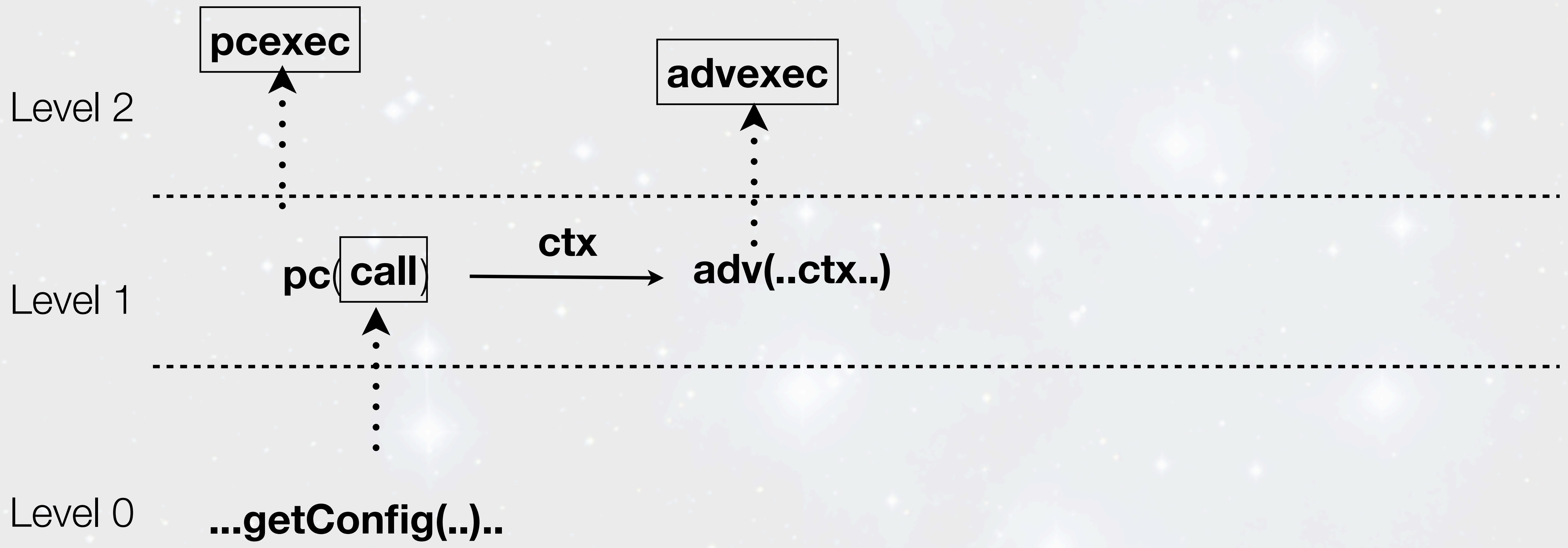
Execution Levels in a Nutshell

Tanter (AOSD '10)



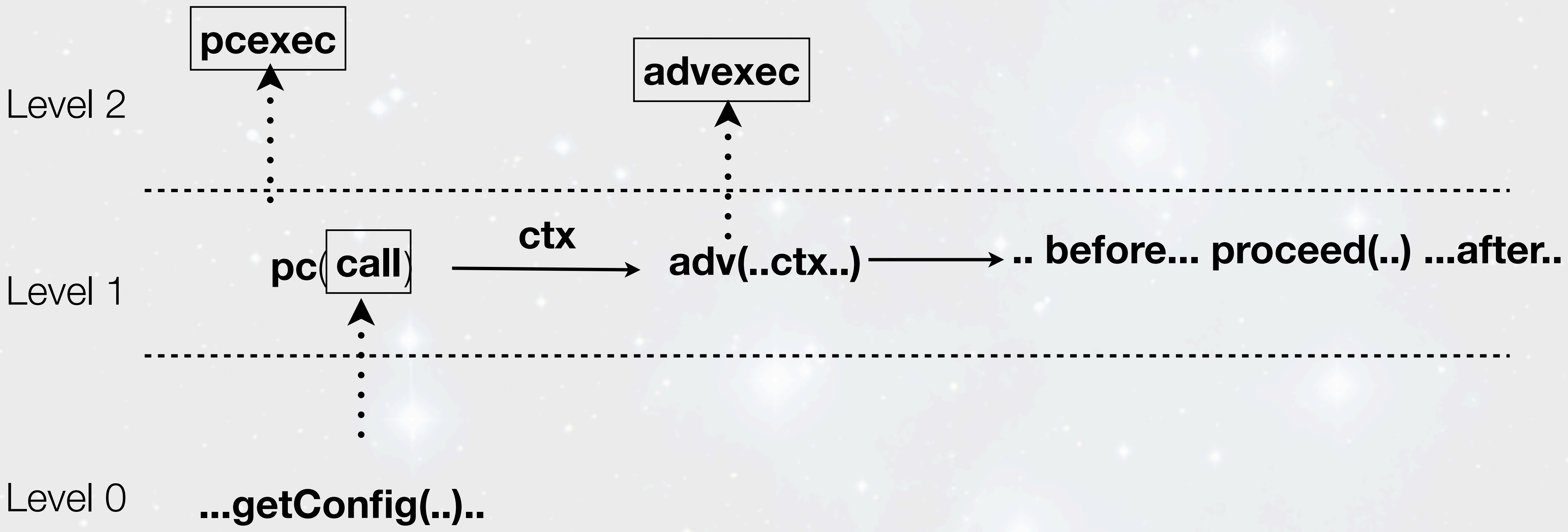
Execution Levels in a Nutshell

Tanter (AOSD ’10)



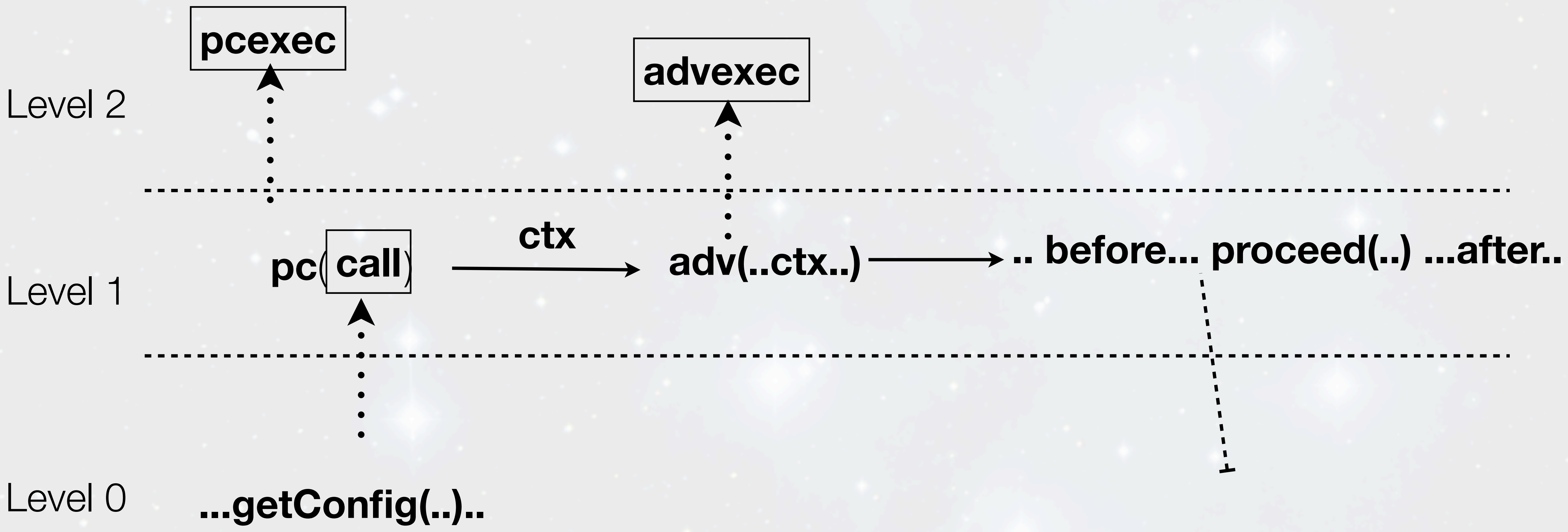
Execution Levels in a Nutshell

Tanter (AOSD '10)



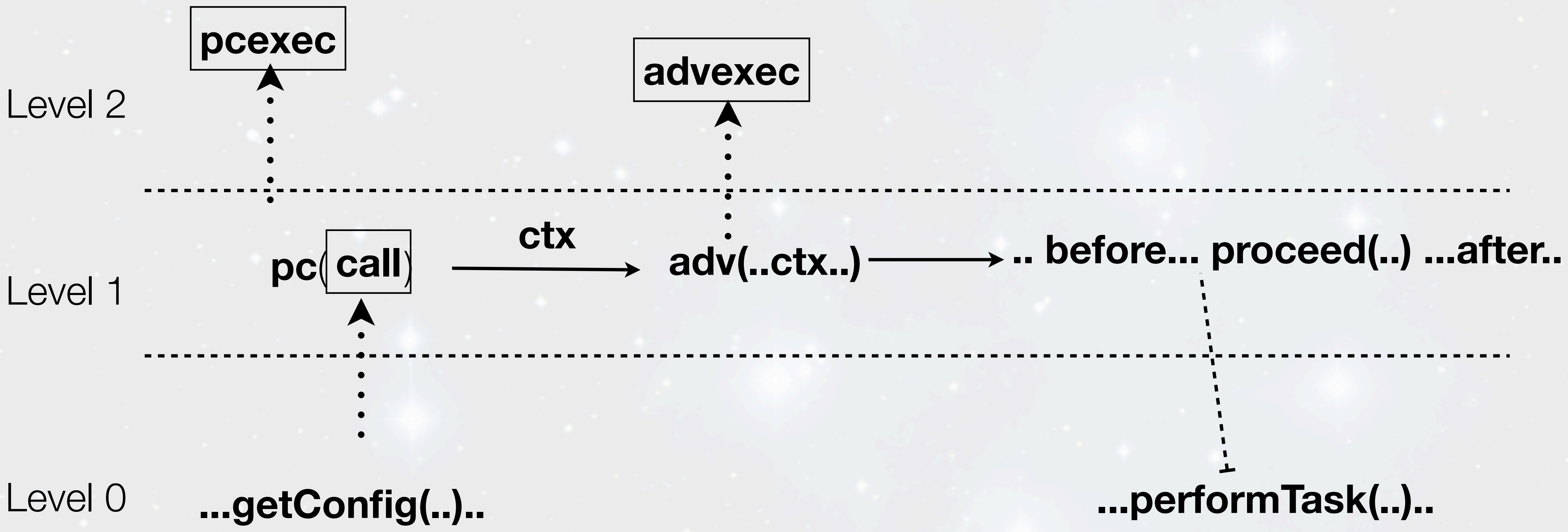
Execution Levels in a Nutshell

Tanter (AOSD '10)



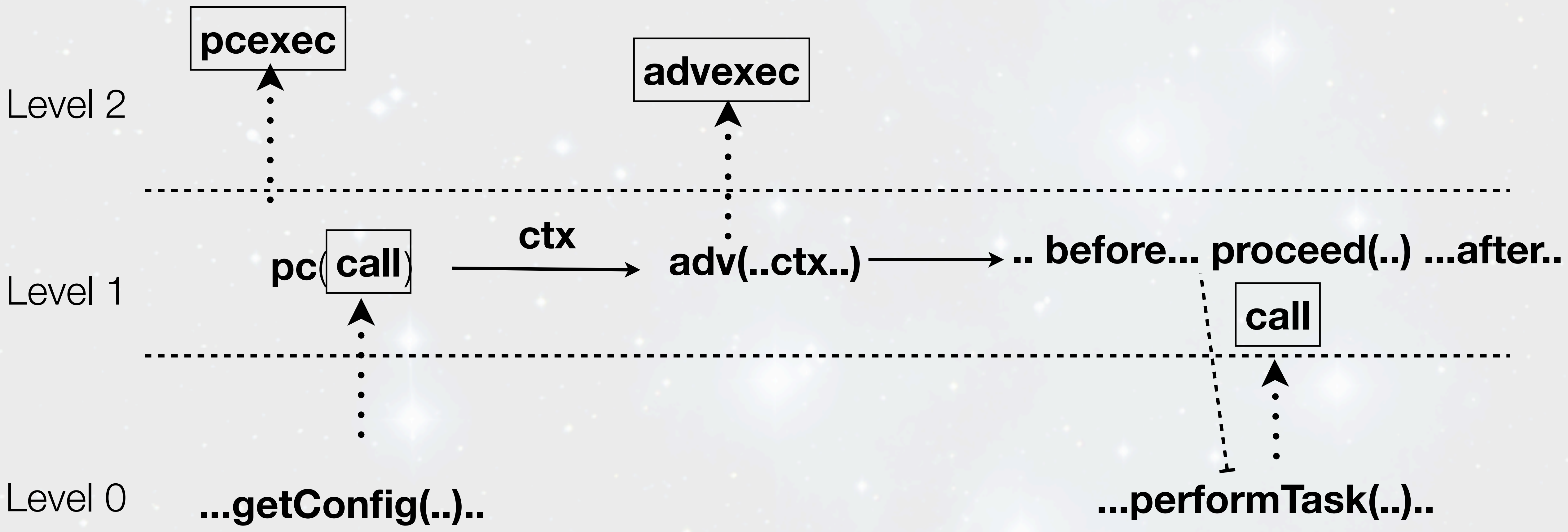
Execution Levels in a Nutshell

Tanter (AOSD '10)



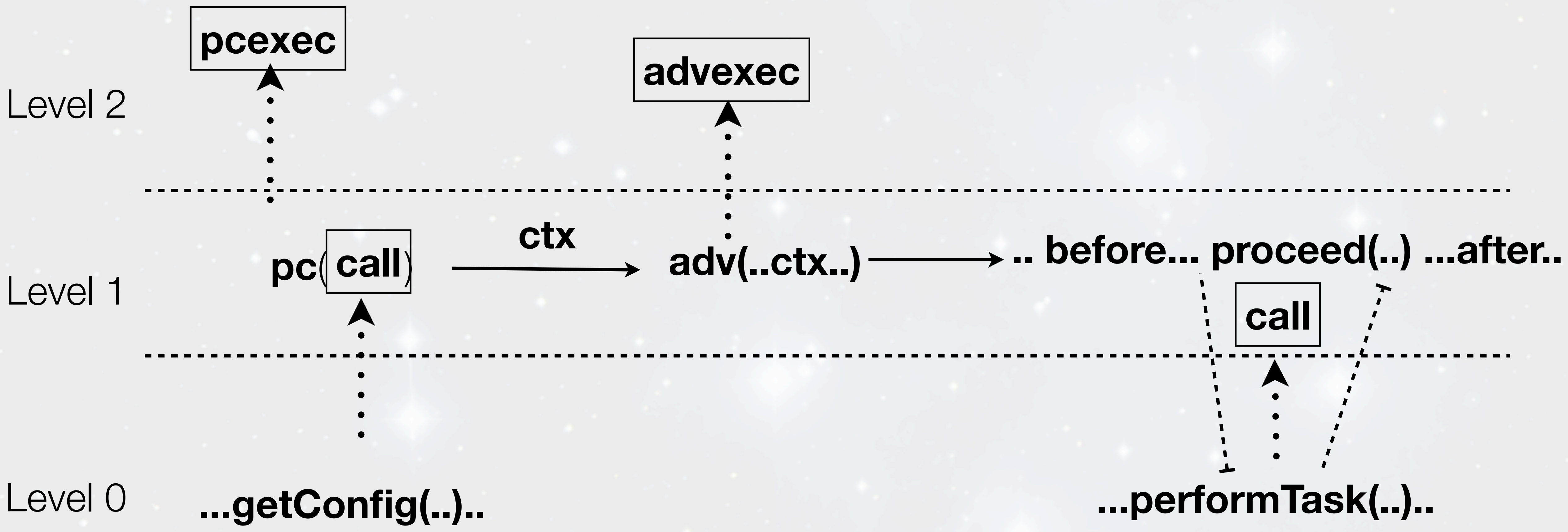
Execution Levels in a Nutshell

Tanter (AOSD '10)



Execution Levels in a Nutshell

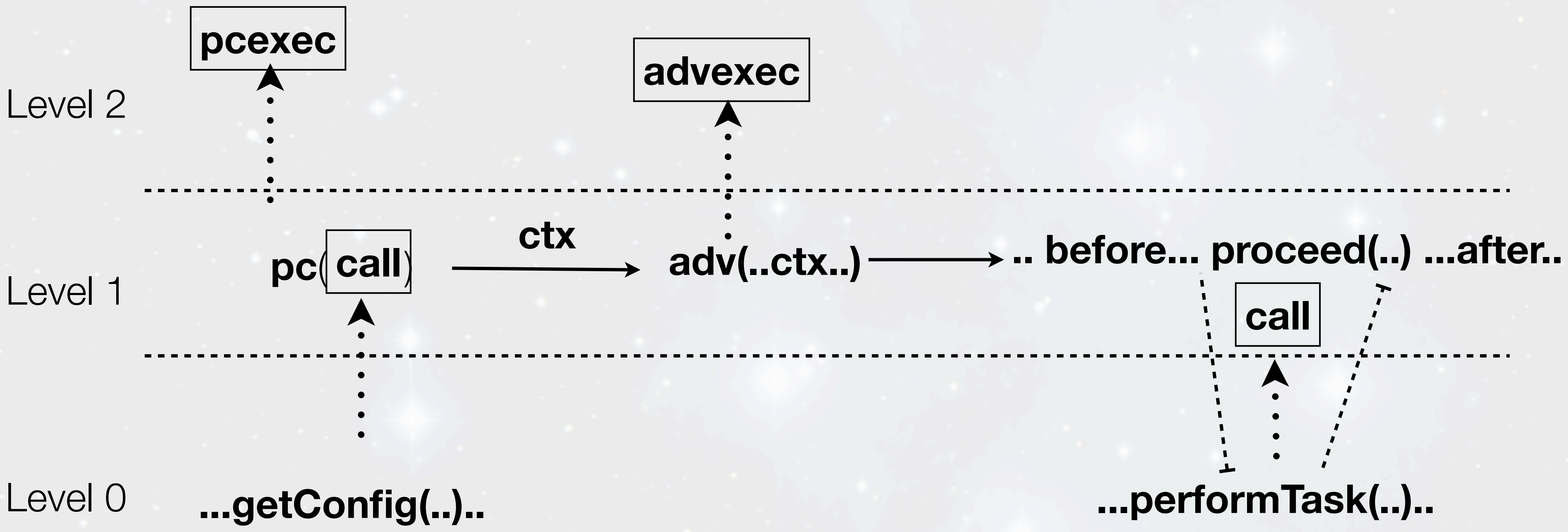
Tanter (AOSD '10)



Execution Levels in a Nutshell

Tanter (AOSD '10)

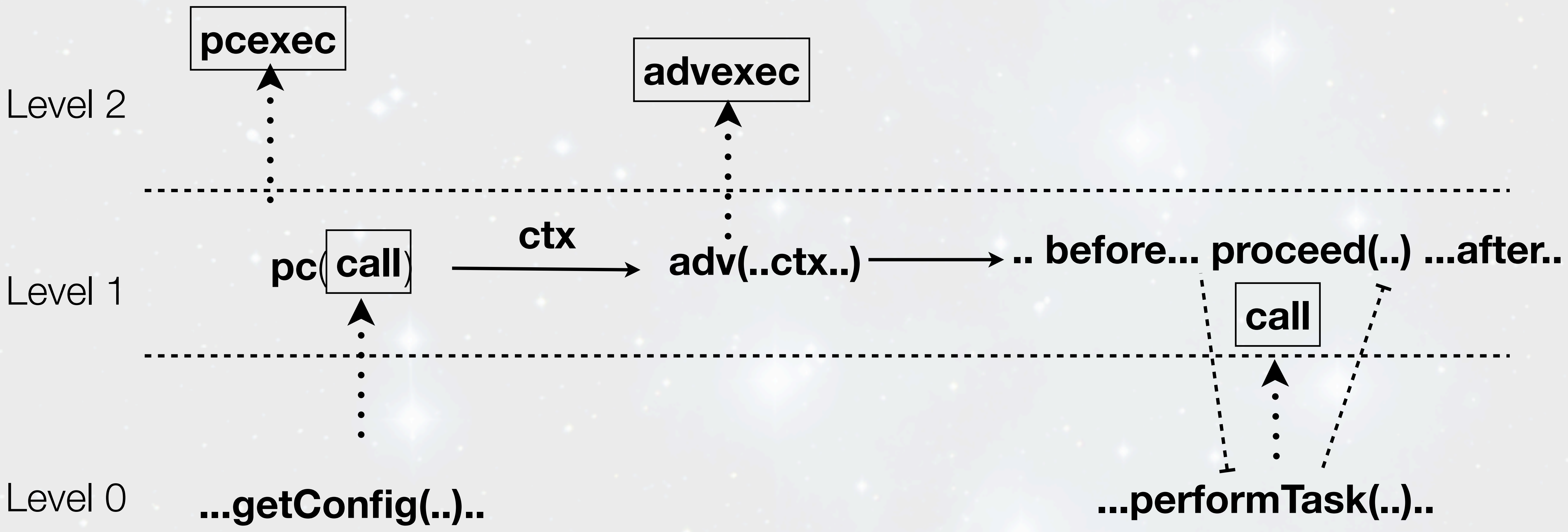
We can have as many levels as needed



Execution Levels in a Nutshell

Tanter (AOSD '10)

We can have as many levels as needed
e.g. composition of dynamic analysis aspects (Tanter *et.al.*,GPCE '10)



Execution Levels in a Nutshell

Execution Levels in a Nutshell

up and **down** operators to explicitly shift level

Execution Levels in a Nutshell

up and **down** operators to explicitly shift level

a **level-capturing function** always evaluate at their **definition-time level**

New Exceptions Semantics

New Exceptions Semantics

In essence:

New Exceptions Semantics

In essence:

bind exceptions and handlers
at the level they originated at

New Exceptions Semantics

In essence:

bind exceptions and handlers
at the level they originated at

an exception is caught by a suitable handler
only if they are both bound at the same level

New Exceptions Semantics

In essence:

bind exceptions and handlers
at the level they originated at

an exception is caught by a suitable handler
only if they are both bound at the same level

safe defaults

New Exceptions Semantics

In essence:

bind exceptions and handlers
at the level they originated at

an exception is caught by a suitable handler
only if they are both bound at the same level

safe defaults

flexibility using level-shifting operators

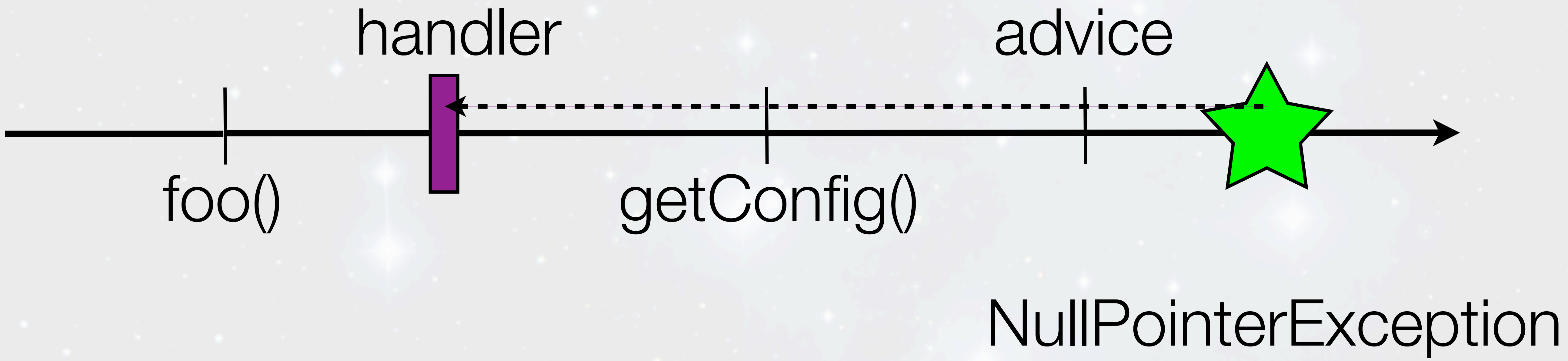
New Exceptions Semantics

New Exceptions Semantics

Recall our first example

New Exceptions Semantics

Recall our first example



New Exceptions Semantics

New Exceptions Semantics

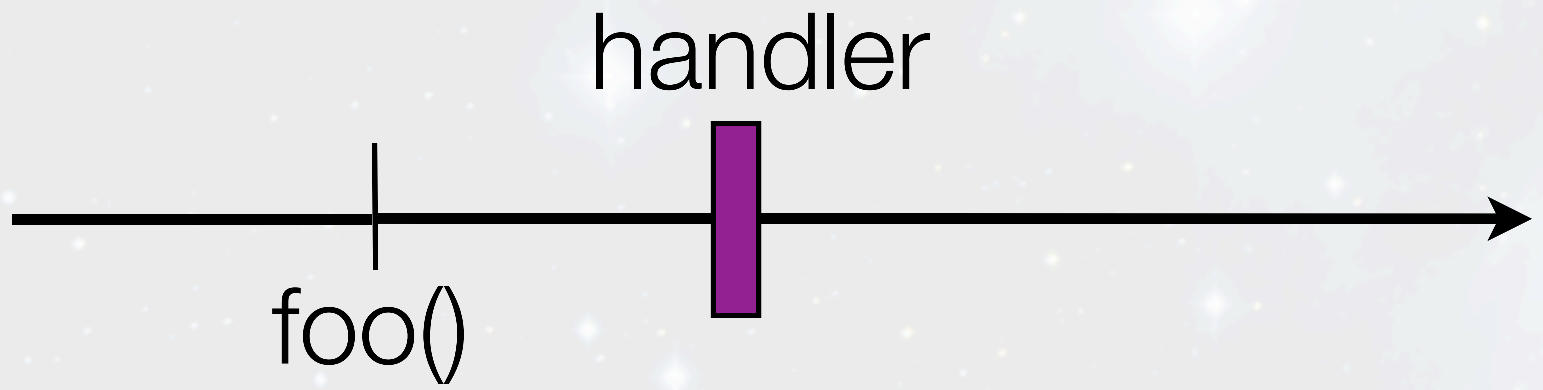
With the new semantics

New Exceptions Semantics

With the new semantics

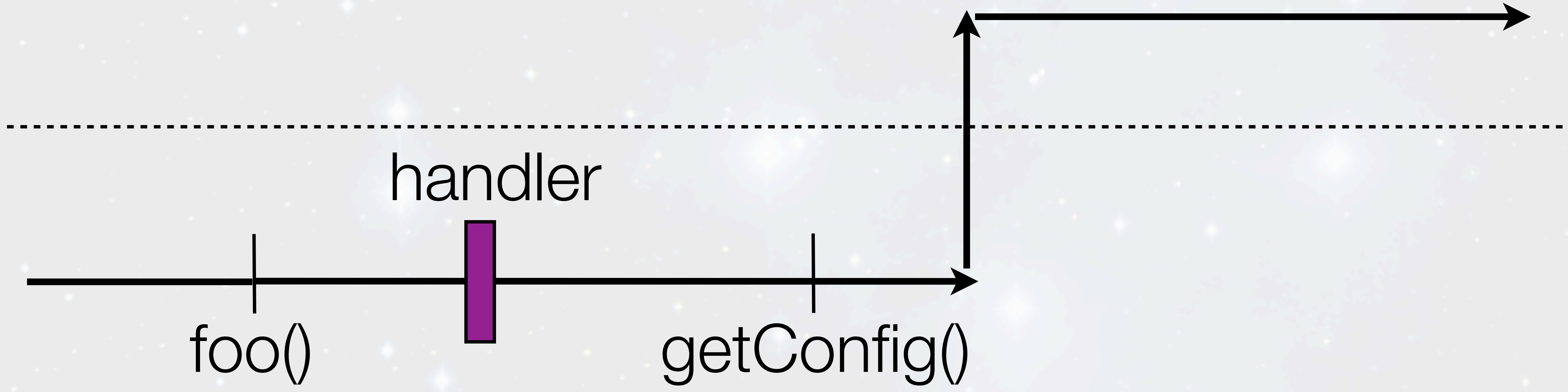
New Exceptions Semantics

With the new semantics



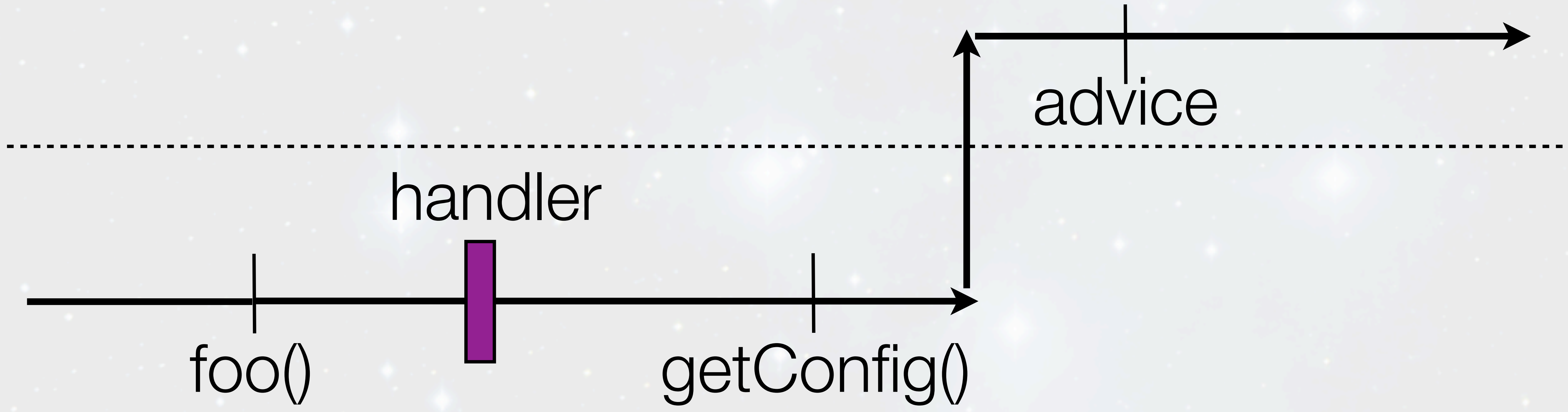
New Exceptions Semantics

With the new semantics



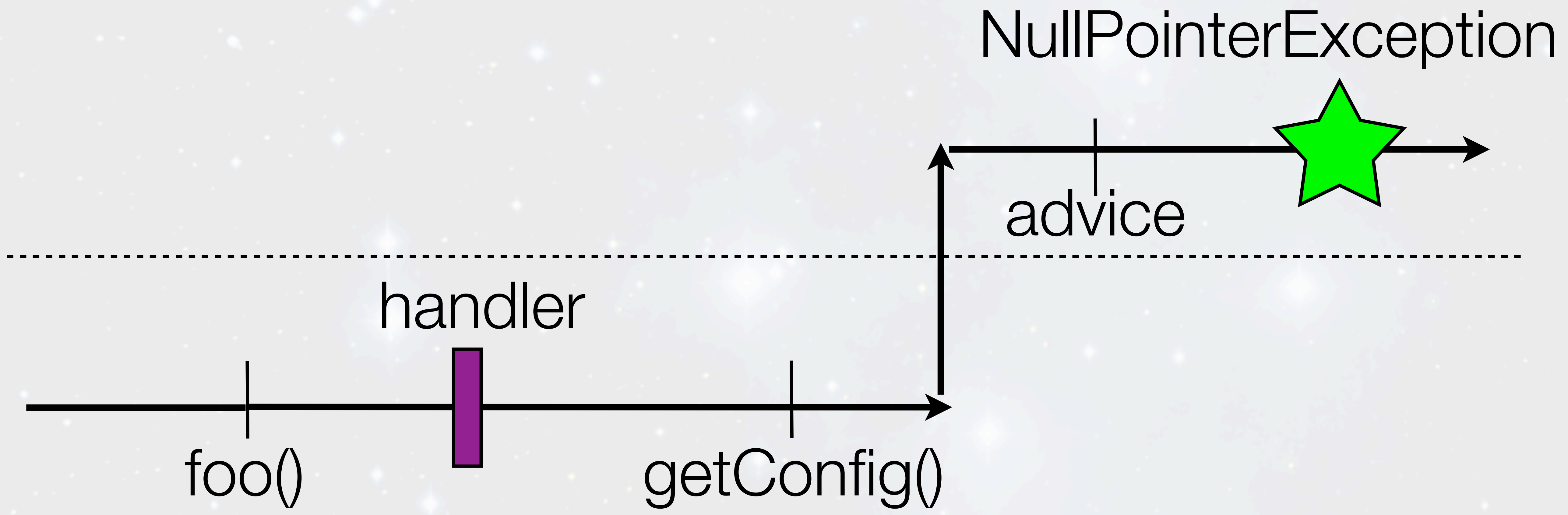
New Exceptions Semantics

With the new semantics



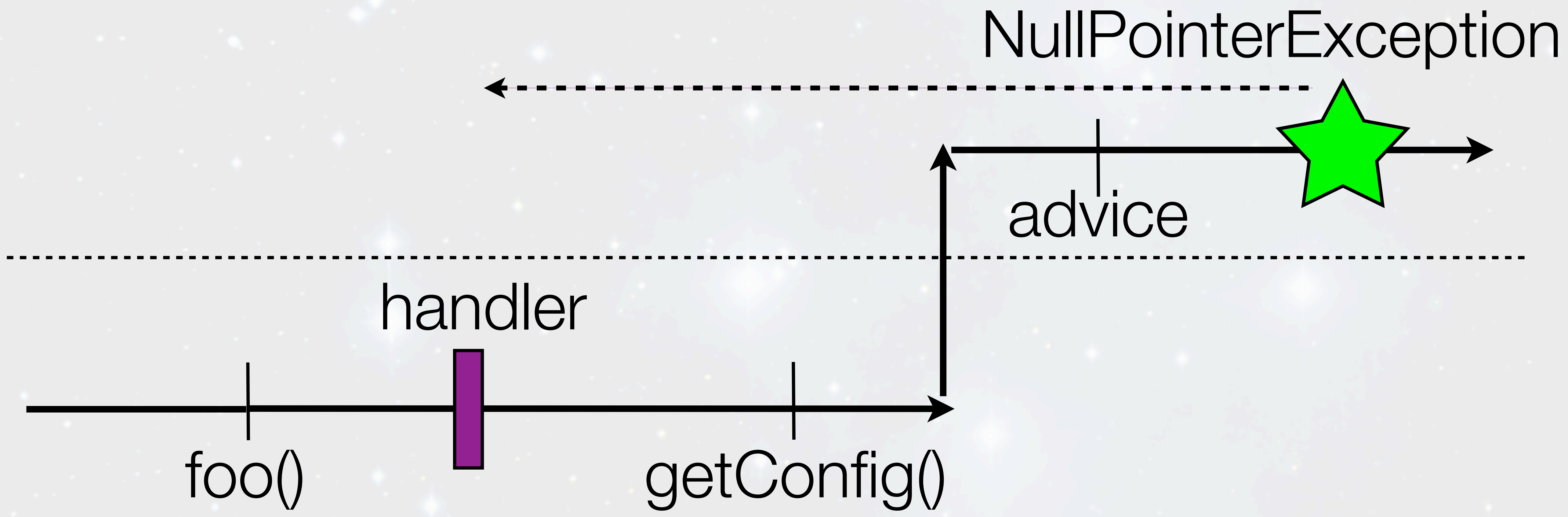
New Exceptions Semantics

With the new semantics



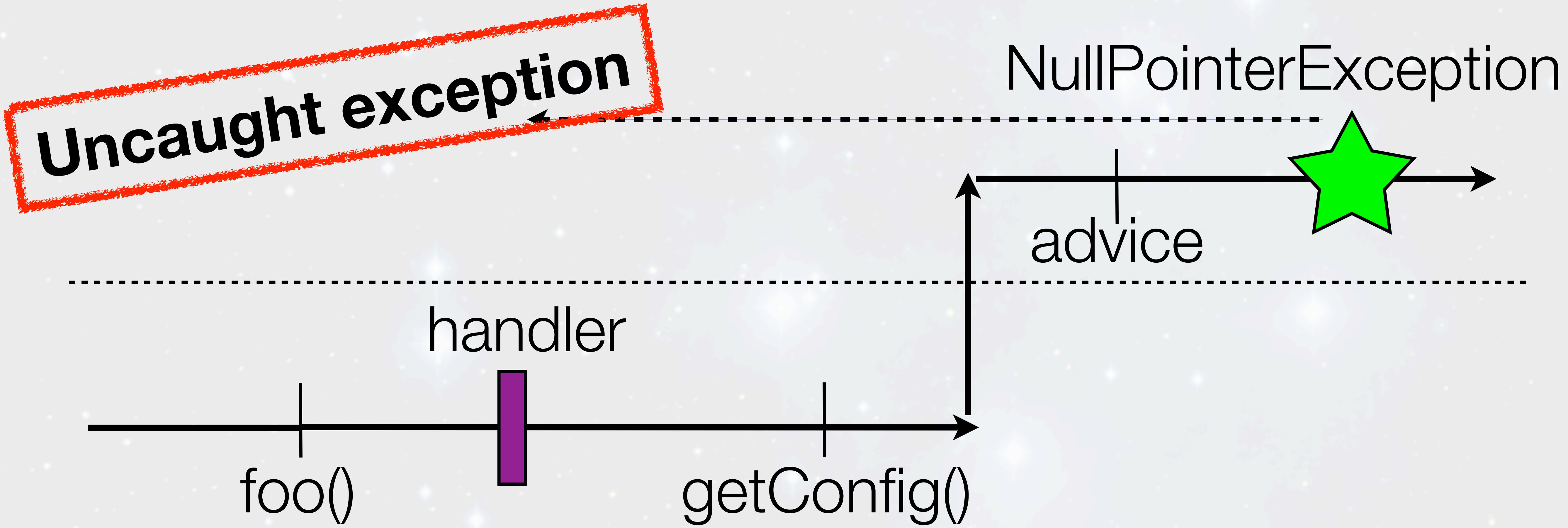
New Exceptions Semantics

With the new semantics



New Exceptions Semantics

With the new semantics



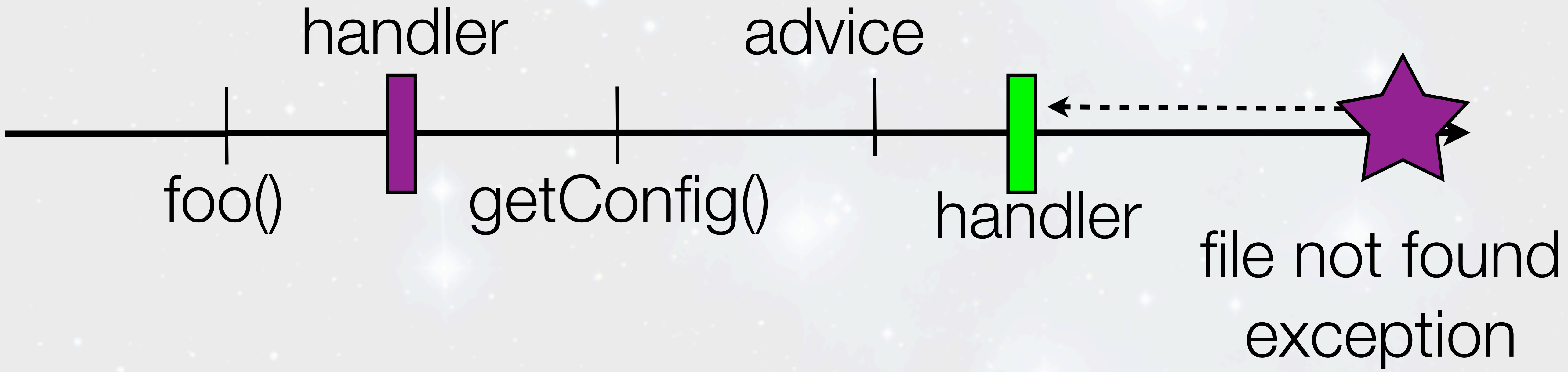
New Exceptions Semantics

New Exceptions Semantics

And so the second example

New Exceptions Semantics

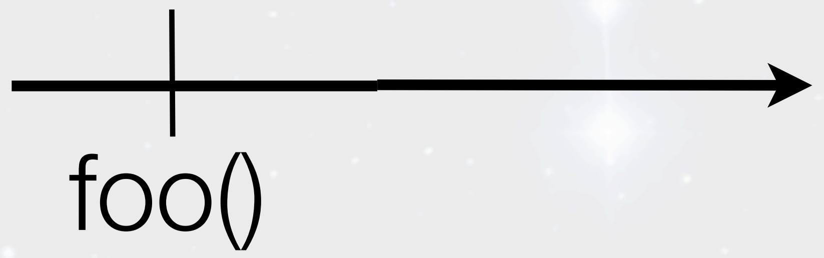
And so the second example



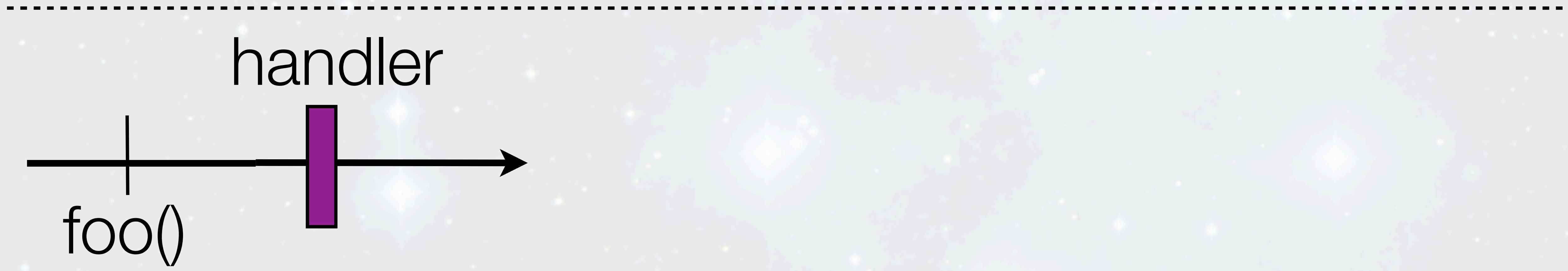
New Exceptions Semantics

New Exceptions Semantics

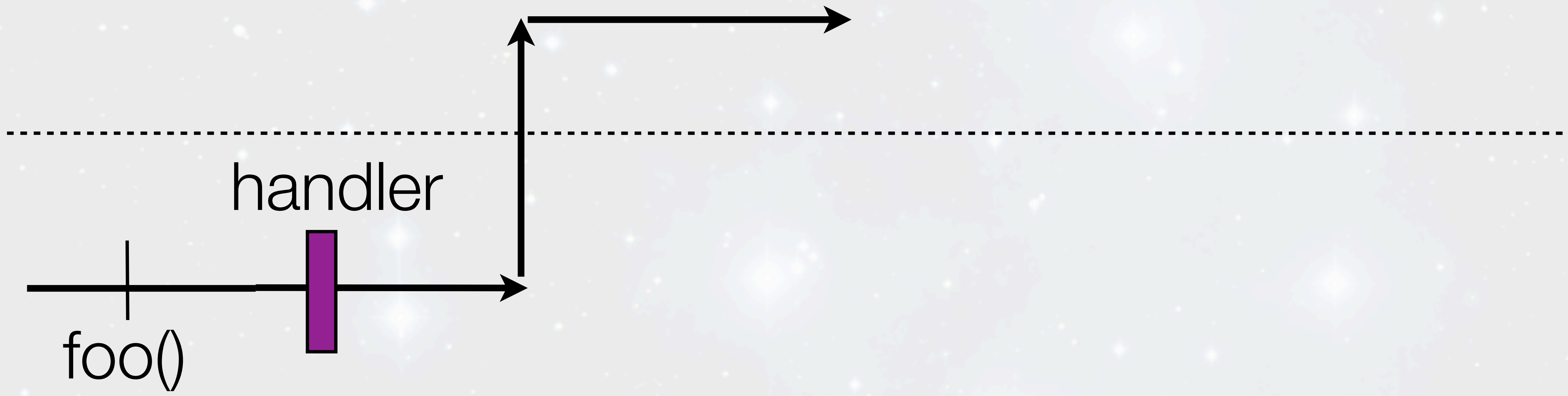
New Exceptions Semantics



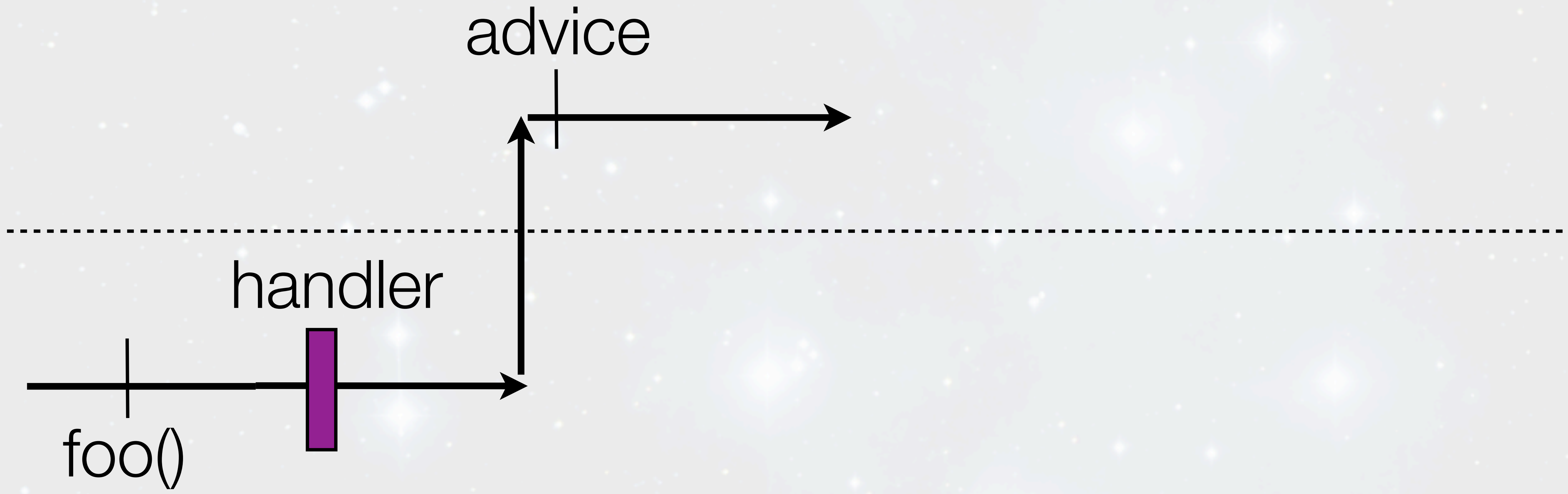
New Exceptions Semantics



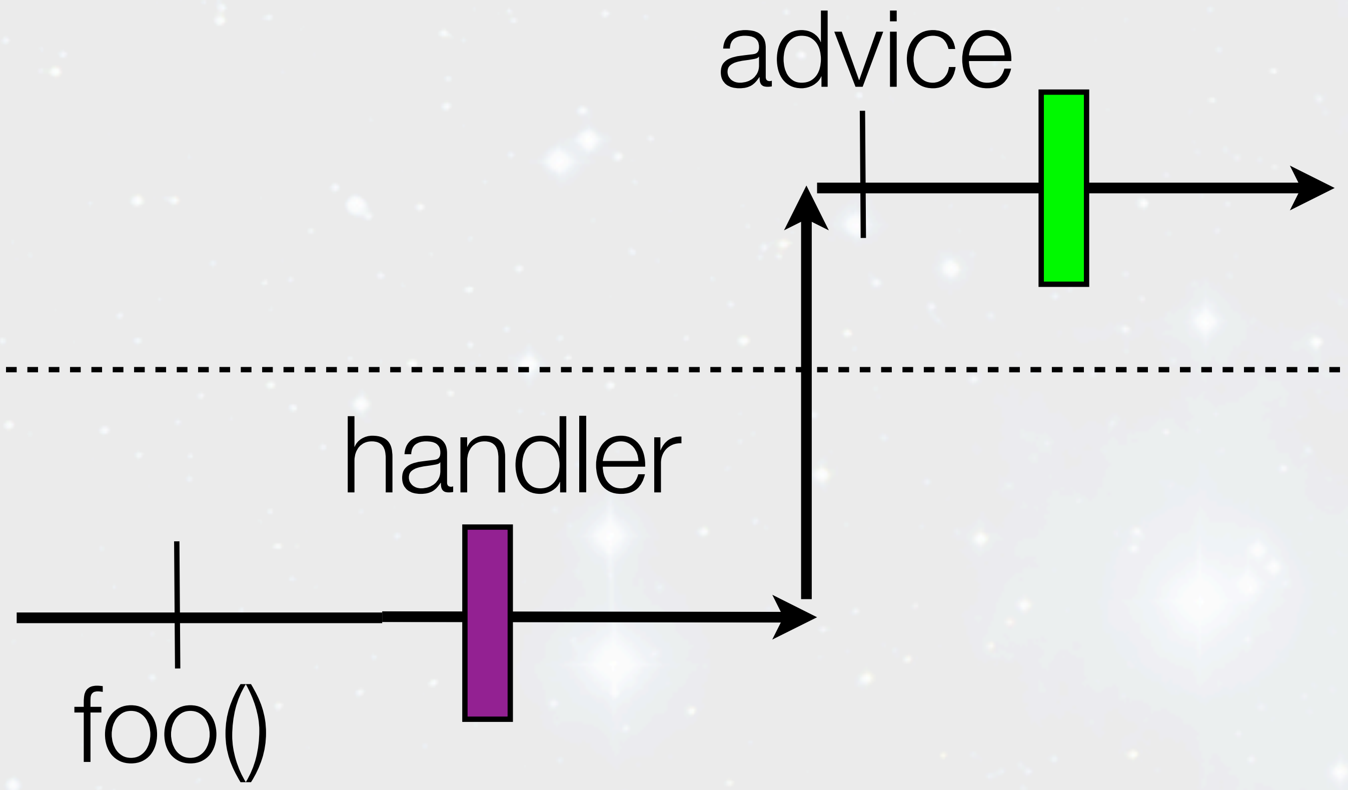
New Exceptions Semantics



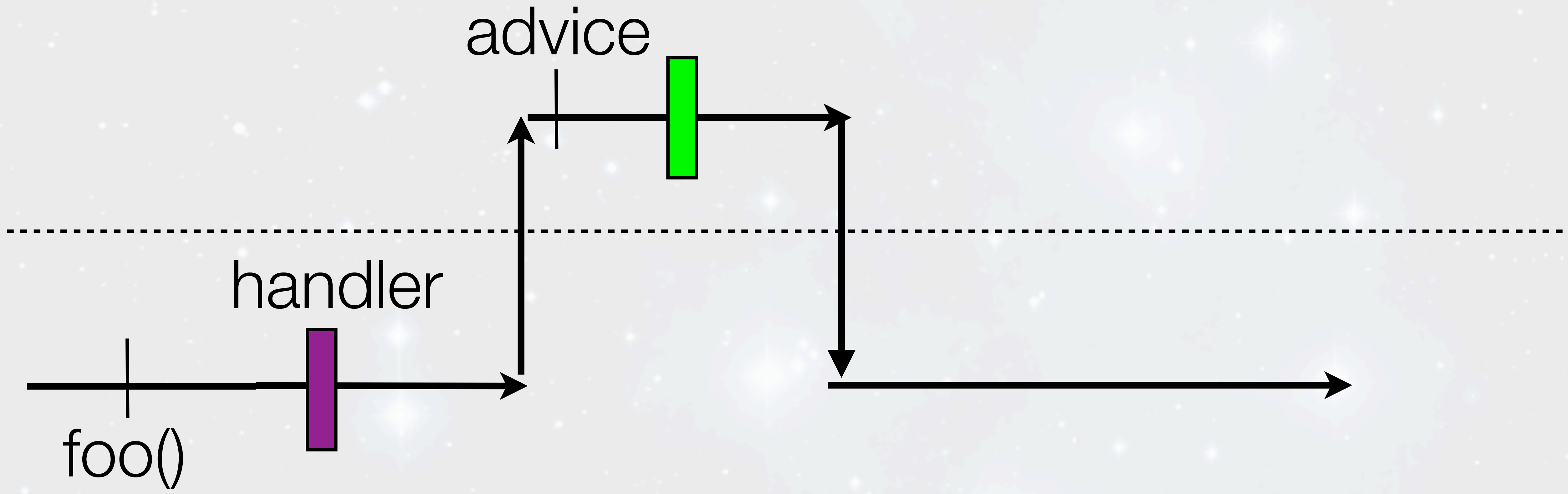
New Exceptions Semantics



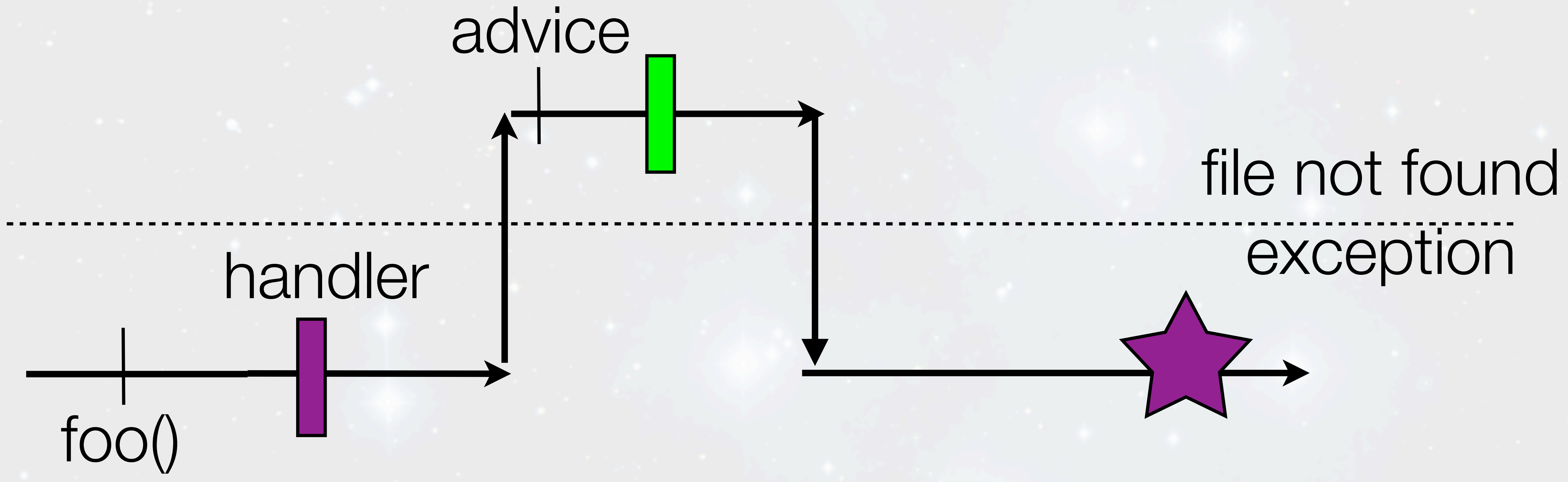
New Exceptions Semantics



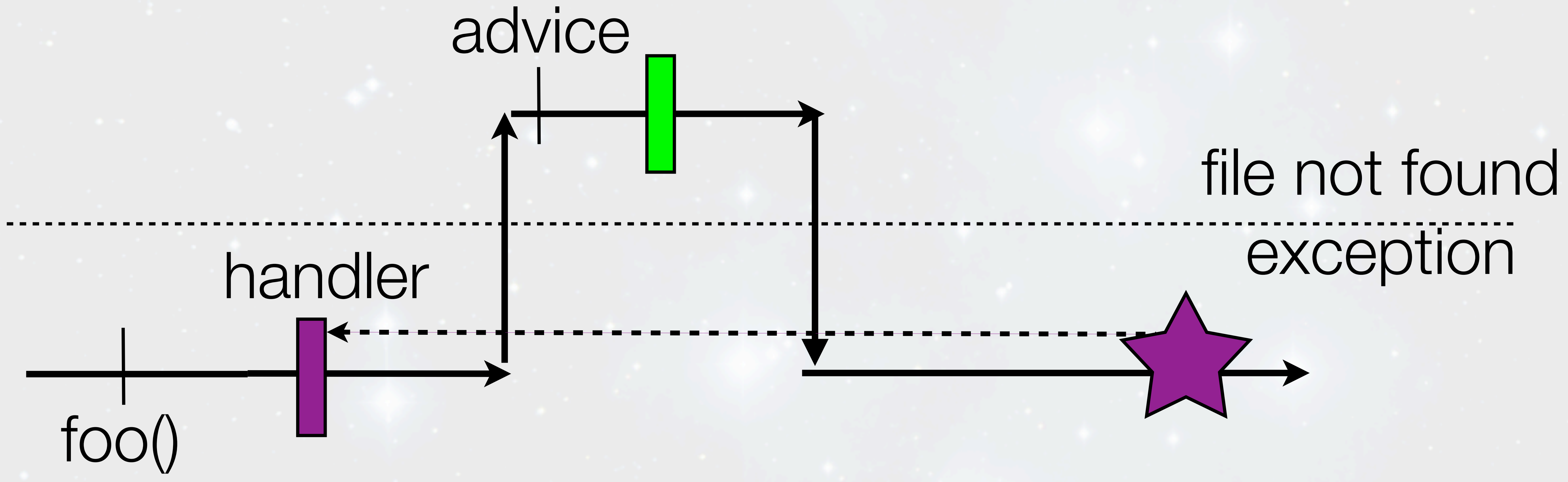
New Exceptions Semantics



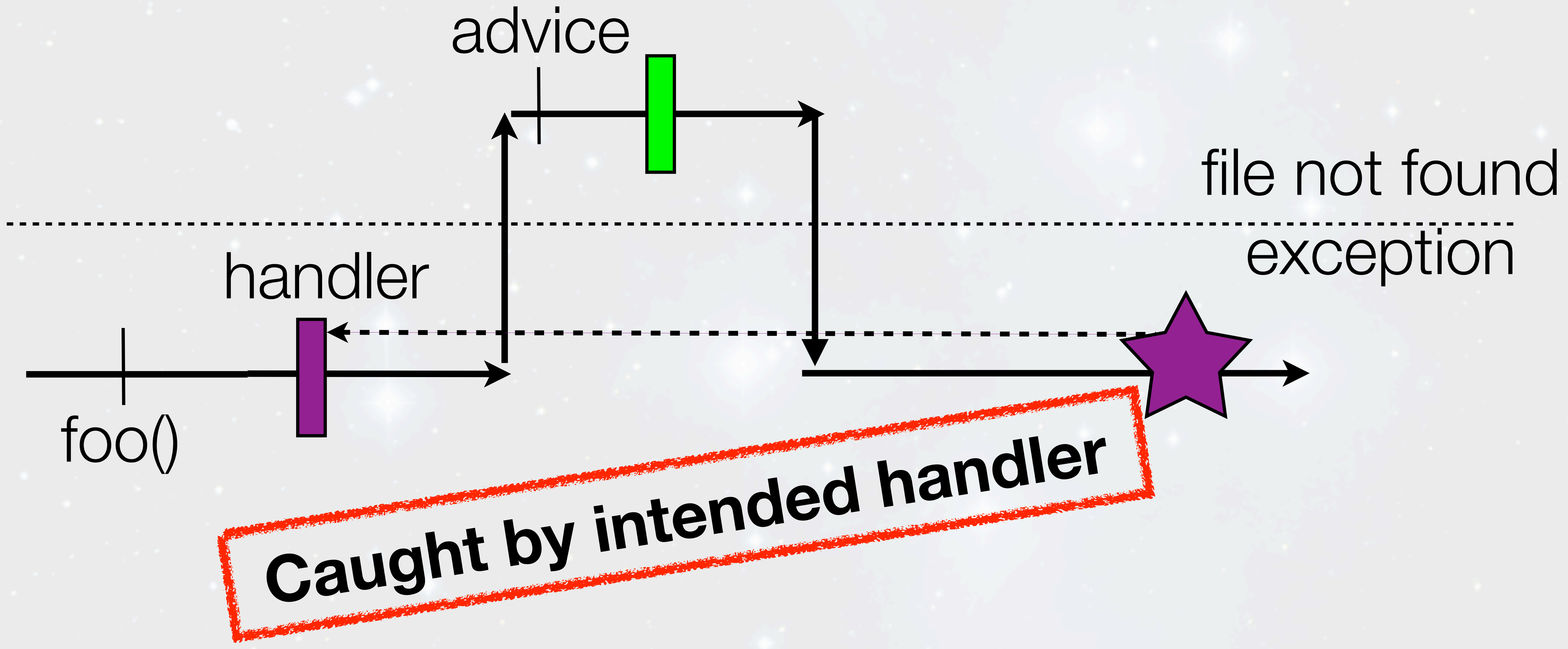
New Exceptions Semantics



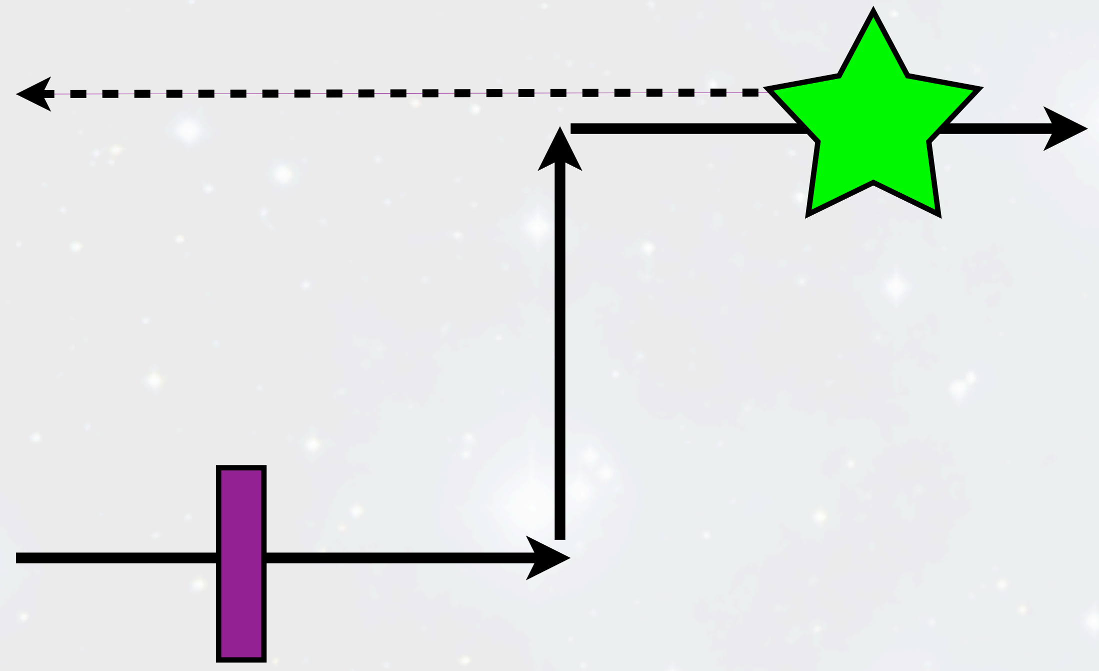
New Exceptions Semantics



New Exceptions Semantics

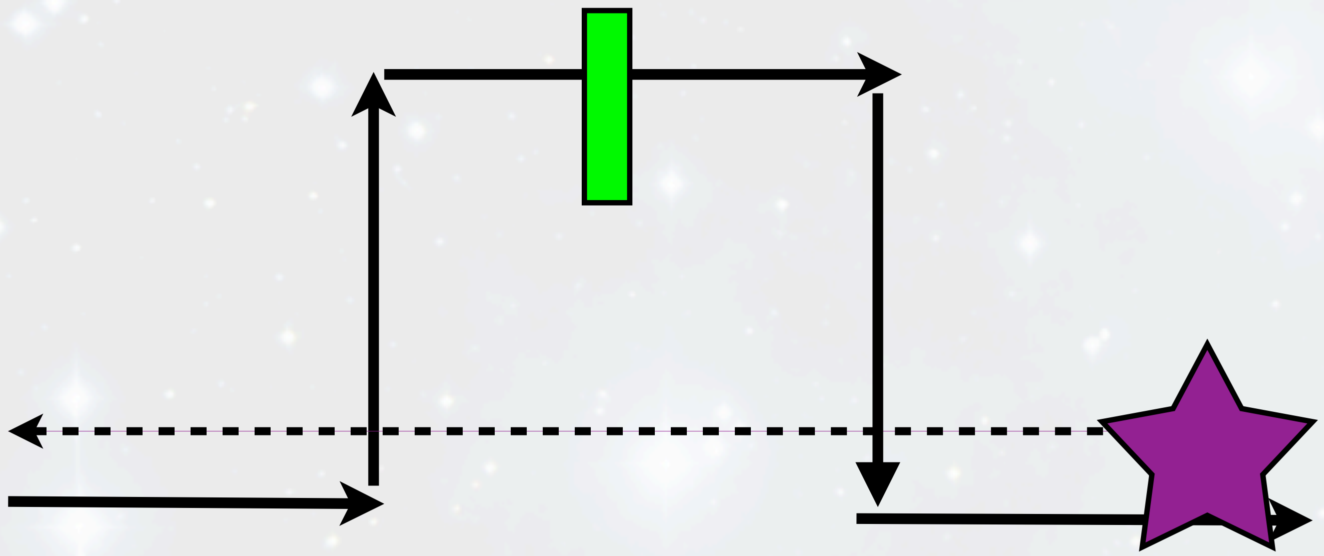


Interaction Patterns



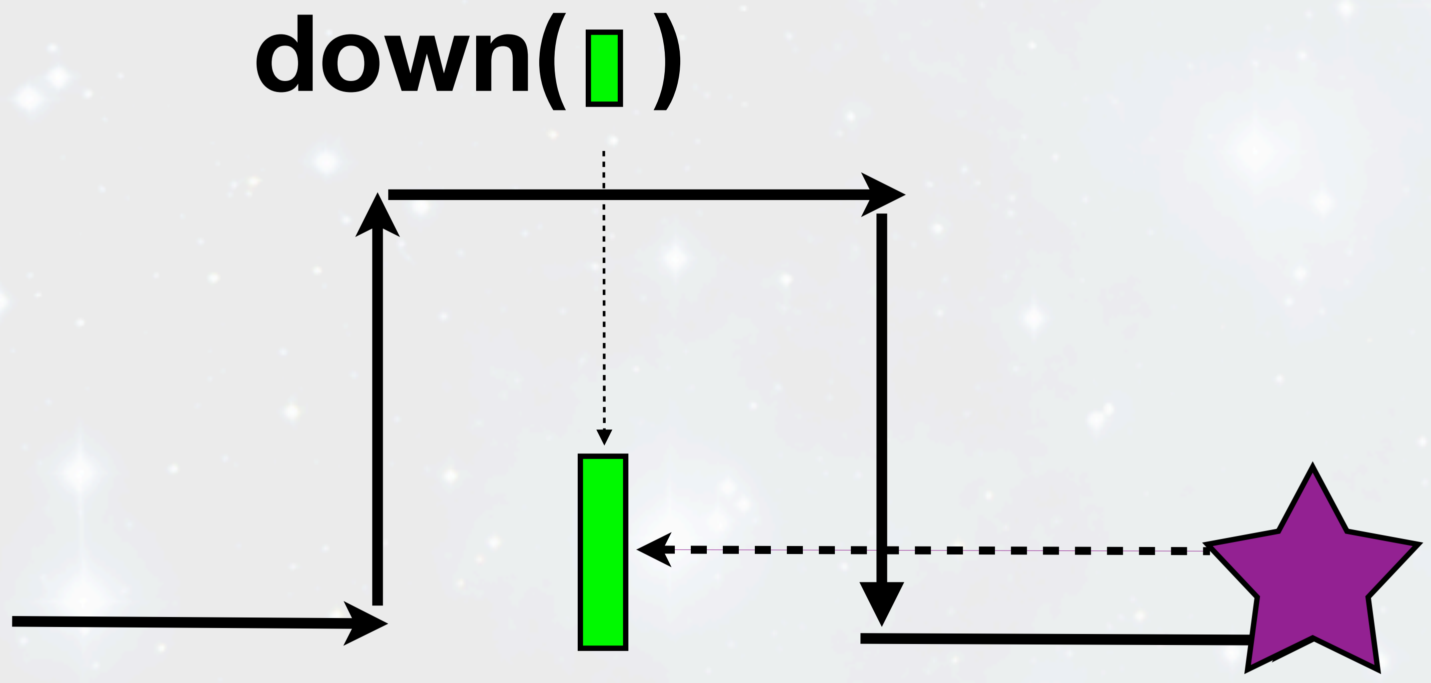
No interference of aspect exception with base handler

Interaction Patterns



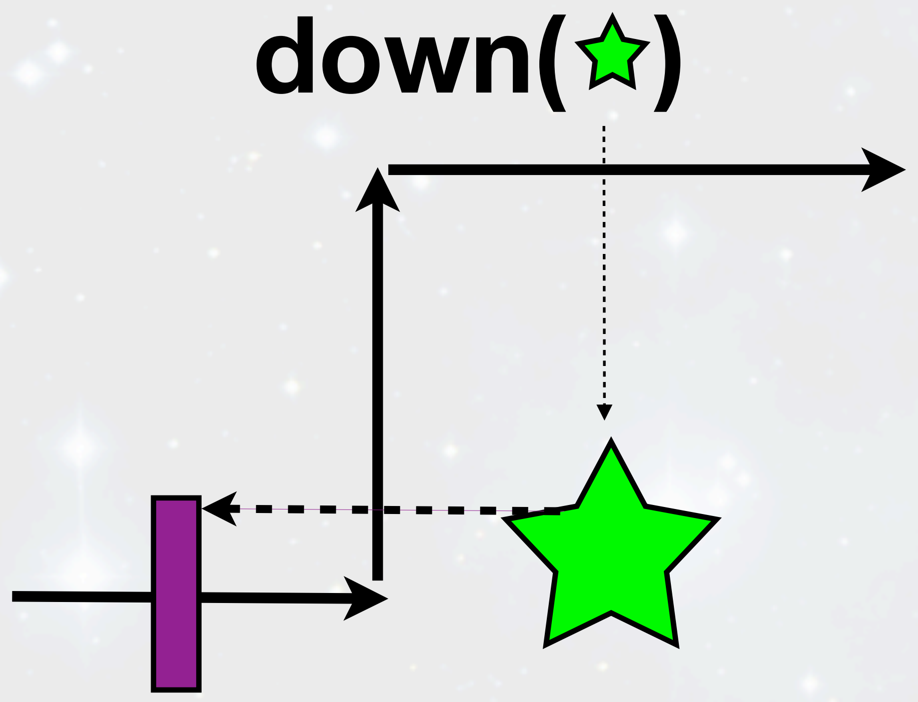
No interference of base exception with aspect handler

Interaction Patterns



down operator to catch base exception

Interaction Patterns



down operator to throw aspect exception at base level

Contributions

Contributions

Exception Conflation Problem

Contributions

Exception Conflation Problem

Core calculus of execution levels + exceptions (see paper!)

Contributions

Exception Conflation Problem

Core calculus of execution levels + exceptions (see paper!)

Implementation using PLT Redex

Contributions

Exception Conflation Problem

Core calculus of execution levels + exceptions (see paper!)

Implementation using PLT Redex

http://pleiad.cl/research/scope

Contributions

Exception Conflation Problem

Core calculus of execution levels + exceptions (see paper!)

Implementation using PLT Redex

http://pleiad.cl/research/scope

Still left to do:

Pleiad

Contributions

Exception Conflation Problem

Core calculus of execution levels + exceptions (see paper!)

Implementation using PLT Redex

http://pleiad.cl/research/scope

Still left to do:

AspectScheme prototype

Contributions

Exception Conflation Problem

Core calculus of execution levels + exceptions (see paper!)

Implementation using PLT Redex

http://pleiad.cl/research/scope

Still left to do:

AspectScheme prototype

Case studies: design-by-contract framework

Pleiad

Contributions

Exception Conflation Problem

Core calculus of execution levels + exceptions (see paper!)

Implementation using PLT Redex

http://pleiad.cl/research/scope

Still left to do:

AspectScheme prototype

Case studies: design-by-contract framework

Prove properties using the calculus

Contributions

Exception Conflation Problem

Core calculus of execution levels + exceptions (see paper!)

Implementation using PLT Redex

http://pleiad.cl/research/scope

Still left to do:

AspectScheme prototype

Case studies: design-by-contract framework

Prove properties using the calculus

Dealing with *finally* blocks