Groupe 4 – ‘Matuidi Pharo’
OpenDev
Exceptions

KUNSANGABO Berfy

DRINCQBIER Gautier

JANKOWSKI Gaël

Année Universitaire : **2017-2018**

Introduction

Lever une exception

ZeroDivide new signal

ZeroDivide signal : "class-side convenience method does the same as above"

Capturer une exception

Ablock on: exceptionClass do: handlerAction

[1+2] on: ZeroDivide do: [:exception | 33]
→3
[1/0] on: ZeroDivide do: [:exception | 33]
→33
[1+2. 1+ 'kjhjkhjk'] on: ZeroDivide do: [:exception | 33]
→raise another Error

Passer à travers une exception

```
anyBlock ensure: ensuredBlock "ensuredBlock will run even if anyBlock fails"

| writer |
writer := GIFReadWriter on: (FileStream newFileNamed: 'Pharo.gif').
[ writer nextPutImage: (Form fromUser) ]
ensure: [ writer close ]

[ 1 ] ensure: [ 0 ]
 →1"not 0"
```

Passer à travers une exception

```
[^ 10] ifCurtailed: [Transcript show: 'We see this'] Transcript show: 'But not this'

[^ 10] ifCurtailed: [Transcript show: 'This is displayed'; cr]

[10] ifCurtailed: [Transcript show: 'This is not displayed'; cr]

[1 / 0] ifCurtailed: [Transcript show: 'This is displayed after selecting Abandon in the debugger'; cr]
```

Privilégier exceptions plutôt que message d’erreur

Avec les messages d’erreurs :

```
source := 'log.txt'.
destination := 'log-backup.txt'.
success := 1.
"define two constants, our error codes"
failure := 0.
fromStream := FileStream oldFileNamed: (FileSystem workingDirectory / source).
fromStream ifNil: [
UIManager default inform: 'Copy failed--could not open', source.
^ failure
"terminate this block with error code"].
toStream := FileStream newFileNamed: (FileSystem workingDirectory / destination).
toStream ifNil: [ fromStream close.
UIManager default inform: 'Copy failed--could not open', destination.
^ failure ].
contents := fromStream contents.
contents ifNil: [fromStream close. toStream close.
UIManager default inform: 'Copy failed--source file has no contents'.
^ failure ].
result := toStream nextPutAll: contents.
result ifFalse: [fromStream close. toStream close.
UIManager default inform: 'Copy failed--could not write to ', destination.^ failure ].
fromStream close.
toStream close.
^ success.
```

Privilégier exceptions plutôt que message d’erreur

Avec les exceptions :

```
| source destination fromStream toStream |
source := 'log.txt'.
destination := 'log-backup.txt'.
[ fromStream := FileStream oldFileNamed: (FileSystem workingDirectory /
source).
toStream := FileStream newFileNamed: (FileSystem workingDirectory /
destination).
toStream nextPutAll: fromStream contents ]
on: FileStreamException
do: [ :ex | UIManager default inform: 'Copy failed--', ex description ].
fromStream ifNotNil: [fromStream close].
toStream ifNotNil: [toStream close].
```

Hiérarchie des classes d'exceptions

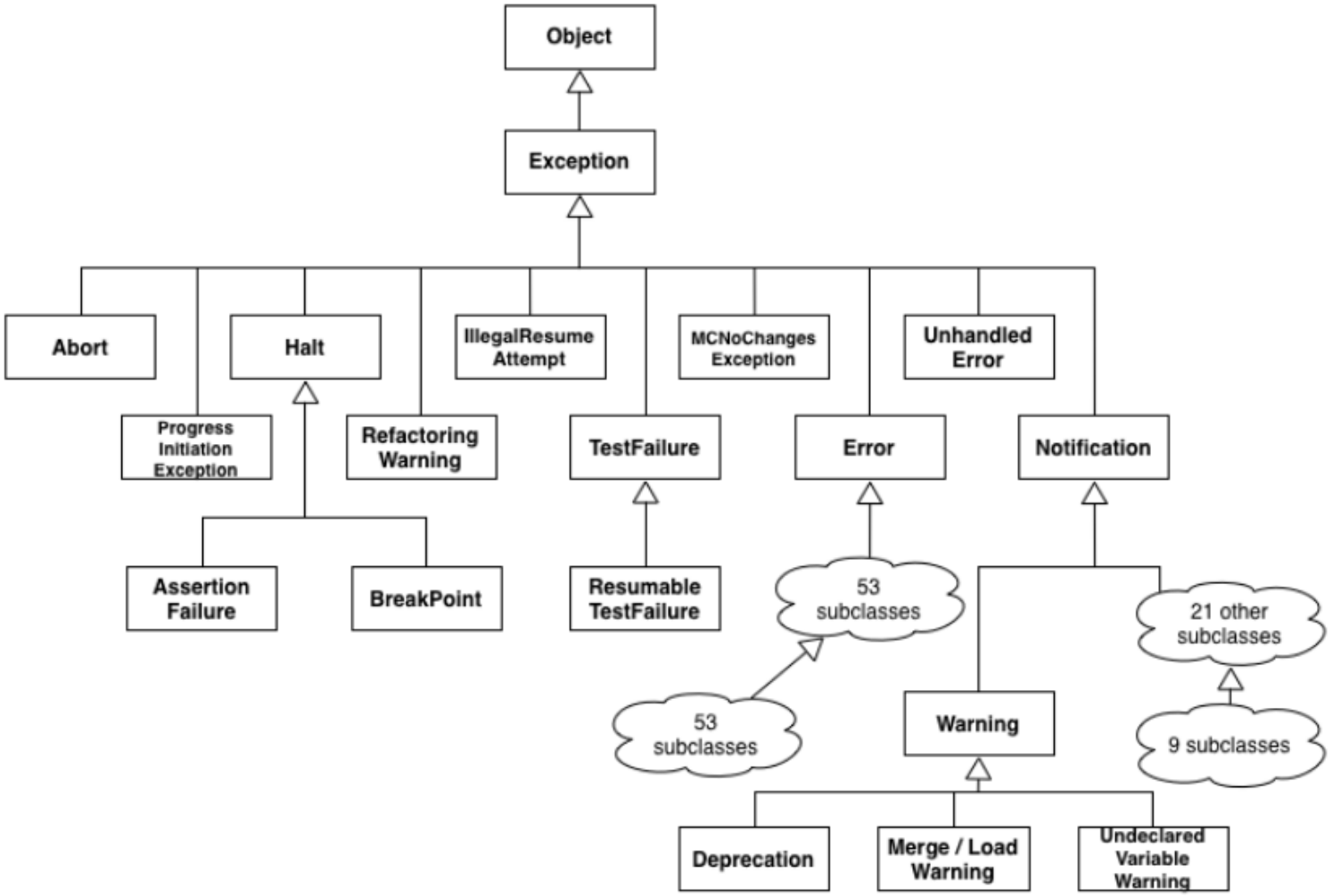

Figure 13.4: A part of Pharo exception hierarchy.

Hierarchie de classes d’exceptions

[...] on: Error do: [...] or
[...] on: FileStreamException do: [...] or
[...] on: FileDoesNotExistException do: [...]

result := [Warning signal . 1/0]
on: Warning, ZeroDivide
do: [:ex | ex resume: 1].
result→1

Différentes manières de gérer une exception

```
answer := [ |result|
result := 6*7.
Error signal.
result "This part is never evaluated"]
on: Error
do: [ :ex | 3 + 4 ].
Answer →7

result := [Error signal]
on: Error
do: [ :ex | ex return: 3 + 4 ].
result→7
```

Différentes manières de gérer une exception

```
[Error signal] on: Error do: [:ex | ex retry]
"will loop endlessly"

result := [ theMeaningOfLife*7 ]
"error--theMeaningOfLife is nil"
on: Error
do: [:ex | theMeaningOfLife := 6. ex retry ].
result→42

x := 0.
result := [ x/x ]
"fails for x=0"
on: Error
do: [:ex | x := x + 1. ex retryUsing: [1/((x-1)*(x-2))] "fails for x=1 and x=2"].
result→(1/2) "succeeds when x=3"]
```

Différentes manières de gérer une exception

```
result := [ | log |
log := OrderedCollection new.
log addLast: 1.
log addLast: MyResumableTestError signal.
log addLast: 2.
log addLast: MyResumableTestError signal.
log addLast: 3.
log ]
on: MyResumableTestError
do: [ :ex |  ex resume: 0 ].
Result → an OrderedCollection(1 0 2 0 3)
```

Différentes manières de gérer une exception

```
Object»performAll: selectorCollection
selectorCollection do: [:each |
[self perform: each]
on: MessageNotUnderstood
do: [:ex | (ex receiver == self and: [ex message selector == each])
ifTrue: [ex return]
ifFalse: [ex pass]]]
"pass internal errors on"

[:ex | (ex receiver == self and: [ex message selector == each])
ifTrue: [InvalidAction signal]
ifFalse: [ex pass]]

[:ex |  (ex receiver == self and: [ex message selector == each])
ifTrue: [ex resignalAs: InvalidAction]
ifFalse: [ex pass]]
```

Différentes manières de gérer une exception

```
Object»performAll: selectorCollection
selectorCollection do: [:each |
[self perform: each]
on: MessageNotUnderstood
do: [:ex | ex return]] "also ignores internal errors"

Object»performAll: selectorCollection
selectorCollection do: [:each |
[self perform: each]
on: MessageNotUnderstood
do: [:ex | (ex receiver == self and: [ex message selector == each])
ifTrue: [ex return]
ifFalse: [ex pass]]] "pass internal errors on"
```

Tester les exceptions

```
testNameOfMonth

self assert: (Date nameOfMonth: 1) equals: #January.

self
shouldnt: [ Date nameOfMonth: 2 ]
raise: SubscriptOutOfBounds.

self
should: [ Date nameOfMonth: 13 ]
raise: SubscriptOutOfBounds.
```

Pas à pas avec les exceptions

```
Exception new isResumable    ⟶    true
Error new isResumable    ⟶    false
Notification new isResumable    ⟶    true
Halt new isResumable    ⟶    true
MessageNotUnderstood new isResumable    ⟶    true
```

Conclusion