static lazy varの挙動を追ってみた

平野朋也

lazyのおさらい

```
lazy var value: Int = 0
```

valueはアクセスされるまで生成されない

try!Swift2018でのテクニック

```
lazy var value: Int = { preconditionFailure() }()
```

初期化必須なプロパティをlazy varで宣言することで、代入前にアクセスすると落とすことができる。
またNeverで返すことで!を使わずに宣言できる。

static変数のlazy var

tutorial001.swift

```
class Class {
  static lazy var value: Int = { preconditionFailure() }()
}

Class.value = 1
```

```
$ swiftc tutorial001.swift
tutorial001.swift:2:10: error: 'lazy' must not be used on an already-lazy global
  static lazy var value: Int = { preconditionFailure() }()
         ^~~~~
```

"'lazy' must not be used on an already-lazy global"

「global変数は既にlazyだから、lazyいらないよ😄」

tutorial002.swift

```
class Class {
  static var value: Int = { preconditionFailure() }()
}

Class.value = 1
```

lazyを外してコンパイルも成功!

$./tutorial002

```
Fatal error: file tutorial002.swift, line 2
```

先にpreconditionFailureが呼ばれている…😈

調査

xxxxx.swift

```
class Class {
  static var xxxxx: Int = { preconditionFailure() }()
}
```

変数をわかりやすくxxxxにしてSIL[1]を出力してみる

[1] Swift Intermediate Language Swiftの中間言語

🦊 @noppefoxwolf, 2018

Swift Compiler

swiftc -emit-silgen xxxxx.swift > xxxxx.sil

```
// static Class.xxxxx.setter
sil hidden [transparent] @_T05xxxxx5ClassCAASivsZ : $@convention(method) (Int, @thick Class.Type) -> () {
// %0                                             // users: %8, %2
// %1                                             // user: %3
bb0(%0 : $Int, %1 : $@thick Class.Type):
  debug_value %0 : $Int, let, name "value", argno 1 // id: %2
  debug_value %1 : $@thick Class.Type, let, name "self", argno 2 // id: %3
  // function_ref Class.xxxxx.unsafeMutableAddressor
  %4 = function_ref @_T05xxxxx5ClassCAASivau : $@convention(thin) () -> Builtin.RawPointer // user: %5
  %5 = apply %4() : $@convention(thin) () -> Builtin.RawPointer // user: %6
  %6 = pointer_to_address %5 : $Builtin.RawPointer to [strict] $*Int // user: %7
  %7 = begin_access [modify] [dynamic] %6 : $*Int // users: %9, %8
  assign %0 to %7 : $*Int                        // id: %8
  end_access %7 : $*Int                          // id: %9
  %10 = tuple ()                                 // user: %11
  return %10 : $()                               // id: %11
} // end sil function '_T05xxxxx5ClassCAASivsZ'
```

%4 ~ %5 初期化

%6 ~ %7 代入

```
%4 = function_ref @_T05xxxxx5ClassCAASivau : $@convention(thin) () -> Builtin.RawPointer // user: %5
%5 = apply %4() : $@convention(thin) () -> Builtin.RawPointer // user: %6
```

function_ref sil-function-name : sil-type SIL関数への参照を作成

_T05xxxxx5ClassCAASivau（辿っていくとpreconditionFailure）の参照作成

applyで実行

代入前にpreconditionFailureが呼ばれているのは確実

https://bugs.swift.org/browse/SR-1178

```
'lazy' must not be used on an already-lazy global
```

'lazy' -> 遅延格納プロパティのlazyとは違う意味