

Swift进阶第二节课：值类型与引用类型

延迟存储属性

类型属性

单例的正确写法

结构体的初始化

结构体是值类型

延迟存储属性

```
1 class LGTeacher{
2     lazy var age: Int = 10
3 }
```

- 用 lazy 修饰的存储属性
- 延迟存储属性必须有一个默认的初始值
- 延迟存储在第一次访问的时候才被赋值
- 延迟存储属性并不能保证线程安全
- 延迟存储属性对实例对象大小的影响

类型属性

```
1 class LGTeacher{
2     static var age: Int = 10
3 }
```

- 使用关键字 static 修饰
- 类型属性必须有一个默认的初始值
- 类型属性只会被初始化一次

单例的正确写法

```

9 #import "LGThread.h"
10
11 @implementation LGThread
12
13 + (instancetype)sharedInstance {
14     static LGThread *sharedInstance = nil;
15     static dispatch_once_t onceToken;
16
17     dispatch_once(&onceToken, ^{
18         sharedInstance = [[LGThread alloc] init];
19     });
20     return sharedInstance;
21 }
22
23 @end
24 ,

```

```

1 class LGTeacher{
2     static let sharedInstance: LGTeacher = LGTeacher()
3     private init(){}
4 }

```

- 使用 static let 创建声明一个实例对象
- 给当前 init 添加访问控制权限 private(什么意思)

结构体的初始化

- 结构体不需要自定义初始化方法，对比下面两段代码

```

1 //1
2 struct LGTeacher{
3     var age: Int
4     var name: String
5 }
6
7 //2
8 class LGStudent{

```

```

9     var age: Int
10    var name: String
11 }

```

其中 2 在代码编译过程中会报错: "Class 'LGStudent' has no initializers"

这是因为编译器在结构体中自动帮我们合成了初始化方法, 也就意味着我们可以这样调用:

```

var t = LGTeacher(age)

```

M LGTeacher (age: Int, name: String)

这里我们通过 `SIL` 来查看:

```

struct LGTeacher {
    @_hasStorage var age: Int { get set }
    @_hasStorage var name: String { get set }
    init(age: Int, name: String)
}

```

- 如果我们的属性有默认初始值, 系统会提供不同的默认初始化方法

```

// default argument 0 of LGTeacher.init(age:name:)
sil hidden @default argument 0 of main.LGTeacher.init(age: Swift.Int, name: Swift.String) -> main.LGTeacher : @$convention(thin) () -> Int {
bb0:
    %0 = integer_literal $Builtin.Int64, 18 // user: %1
    %1 = struct $Int (%0 : $Builtin.Int64) // user: %2
    return %1 : $Int // id: %2
} // end sil function 'default argument 0 of main.LGTeacher.init(age: Swift.Int, name: Swift.String) -> main.LGTeacher'

// LGTeacher.init(age:name:)
sil hidden @main.LGTeacher.init(age: Swift.Int, name: Swift.String) -> main.LGTeacher : @$convention(method) (Int, @owned String, @thin LGTeacher.Type) -> @owned LGTeacher {
// %0 // user: %3
// %1 // user: %3
bb0(%0 : $Int, %1 : $String, %2 : @$thin LGTeacher.Type):
    %3 = struct $LGTeacher (%0 : $Int, %1 : $String) // user: %4
    return %3 : $LGTeacher // id: %4
} // end sil function 'main.LGTeacher.init(age: Swift.Int, name: Swift.String) -> main.LGTeacher'

```

- 如果我们自定义初始化方法, 系统就不会帮我们生成初始化方法

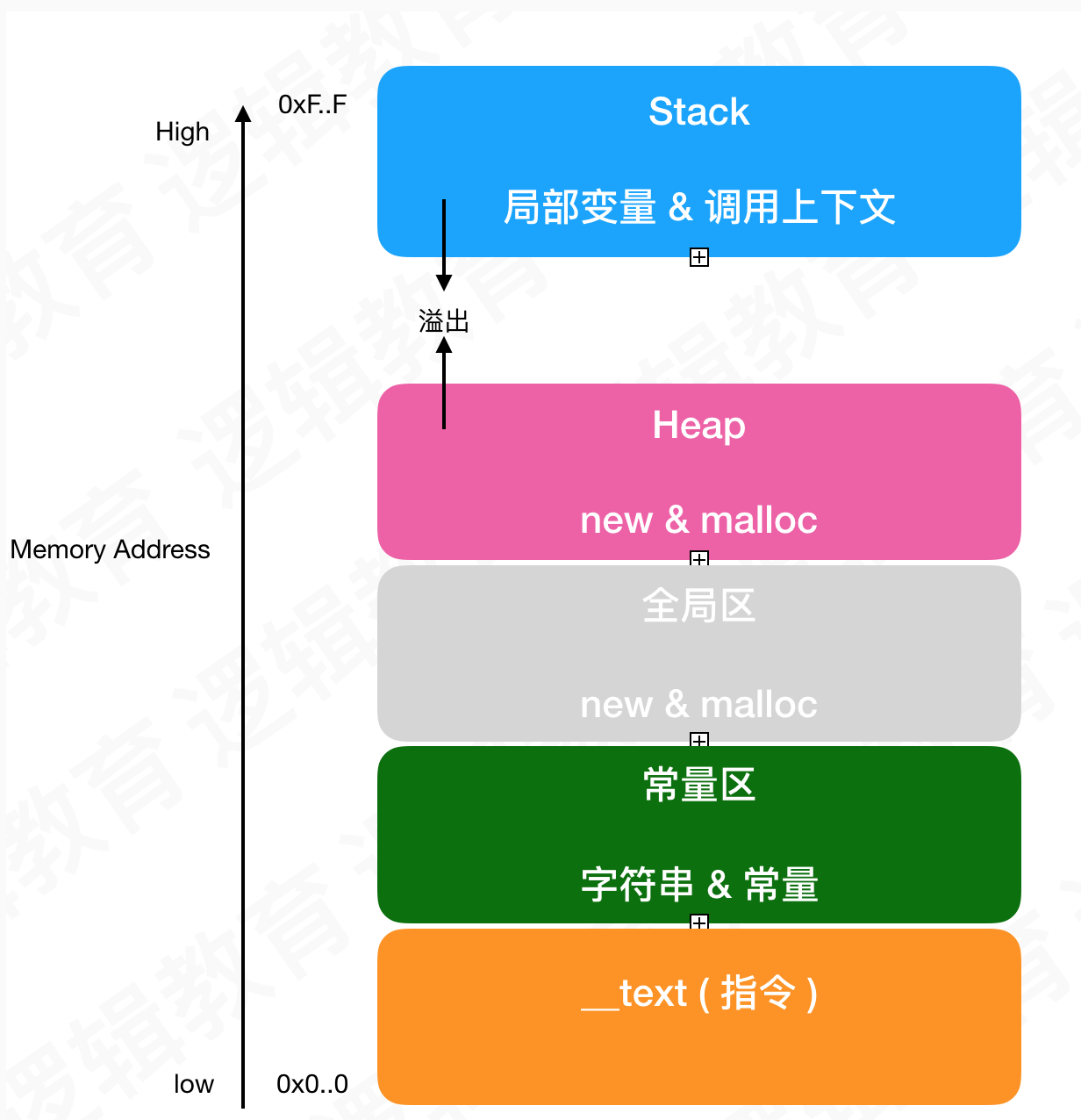
结构体是值类型

- 什么是值类型

我们首先来看一个例子:

```
1 func test(){  
2     var age = 18  
3  
4     var age2 = age  
5  
6     age = 30  
7  
8     age2 = 45  
9  
10    print("age=\(age),age2=\(age2)")  
11 }
```

- 内存分区模型：



- 通过 LLDB 查看结构体的内存模型

其实值类型和引用类型就像什么，一个是在线表格，一个是本地的excel,当我和你共享编辑一个在线表格的时候其实就是一个引用类型，而当我们通过QQ传给你一个excel的时候，这个时候就相当于一个值类型，你修改的内容我是不知道的。

- mutating

值类型本身创建之后是不允许修改的，如果要修改，需要使用 mutating 关键字

- 结构体中的方法调度是静态调度（编译，链接完成之后当前的函数地址就已经确定存放在了代码段）

Executable (X86_64)

Mach64 Header

► Load Commands

▼ Section64 (__TEXT,__text)

Assembly

► Section64 (__TEXT,__stubs)

► Section64 (__TEXT,__stub_helper)

► Section64 (__TEXT,__cstring)

Section64 (__TEXT,__const)

Section64 (__TEXT,__swift5_typerref)

Section64 (__TEXT,__swift5_reflstr)

Section64 (__TEXT,__swift5_fieldmd)

Section64 (__TEXT,__swift5_types)

Section64 (__TEXT,__unwind_info)

► Section64 (__TEXT,__eh_frame)

▼ Section64 (__DATA_CONST,__got)

Non-Lazy Symbol Pointers

Section64 (__DATA_CONST,__const)

Section64 (__DATA_CONST,__objc_imageinfo)

▼ Section64 (__DATA,__la_symbol_ptr)

Lazy Symbol Pointers

Section64 (__DATA,__data)

► Dynamic Loader Info

► Function Starts

▼ Symbol Table

Symbols

Data in Code Entries

▼ Dynamic Symbol Table

Indirect Symbols

String Table

Code Signature

- Symbol Table：存储符号位于字符串表的位置

Executable (X86_64)
Mach64 Header
Load Commands
Section64 (__TEXT,__text)
Assembly
Section64 (__TEXT,__stubs)
Section64 (__TEXT,__stub_helper)
Section64 (__TEXT,__cstring)
Section64 (__TEXT,__const)
Section64 (__TEXT,__swift5_typeref)
Section64 (__TEXT,__swift5_reflist)
Section64 (__TEXT,__swift5_fieldmd)
Section64 (__TEXT,__swift5_types)
Section64 (__TEXT,__unwind_info)
Section64 (__TEXT,__eh_frame)
Section64 (__DATA,__const, __got)
Non-Lazy Symbol Pointers
Section64 (__DATA,__const, __const)
Section64 (__DATA,__const, __objc_imageinfo)
Section64 (__DATA,__la_symbol_ptr)
Lazy Symbol Pointers
Section64 (__DATA,__data)
Dynamic Loader Info
Function Starts
Symbol Table
Symbols
Data in Code Entries
Dynamic Symbol Table
String Table
Code Signature

Offset	Data	Description	Value
#0			
00003268	000002CC	String Table Index	__\$s11LGSwiftTest9LGTeacherV3age5iSgvM.resume.0
0000326C	0E	Type	N_SECT
0000326D	01	Section Index	1 (__TEXT,__text)
0000326E	0000	Description	
00003270	0000000100000A90	Value	4294970000 (\$+208)
#1			
00003278	000002FB	String Table Index	__\$s5Sprint_9separator10terminatoryypd_S2stfFA0_
0000327C	1E	Type	N_SECT
0000327D	01	Section Index	1 (__TEXT,__text)
0000327E	0000	Description	
00003280	0000000100000B60	Value	4294970208 (\$+416)
#2			
00003288	0000032B	String Table Index	__\$s5Sprint_9separator10terminatoryypd_S2stfFA1_
0000328C	1E	Type	N_SECT
0000328D	01	Section Index	1 (__TEXT,__text)
0000328E	0000	Description	
00003290	0000000100000B80	Value	4294970240 (\$+448)
#3			
00003298	0000035B	String Table Index	__swift_memcpy9_8
0000329C	1E	Type	N_SECT

- Dynamic Symbol Table : 动态库函数 位于符号表的偏移信息

Executable (X86_64)
Mach64 Header
Load Commands
Section64 (__TEXT,__text)
Assembly
Section64 (__TEXT,__stubs)
Section64 (__TEXT,__stub_helper)
Section64 (__TEXT,__cstring)
Section64 (__TEXT,__const)
Section64 (__TEXT,__swift5_typeref)
Section64 (__TEXT,__swift5_reflist)
Section64 (__TEXT,__swift5_fieldmd)
Section64 (__TEXT,__swift5_types)
Section64 (__TEXT,__unwind_info)
Section64 (__TEXT,__eh_frame)
Section64 (__DATA,__const, __got)
Non-Lazy Symbol Pointers
Section64 (__DATA,__const, __const)
Section64 (__DATA,__const, __objc_imageinfo)
Section64 (__DATA,__la_symbol_ptr)
Lazy Symbol Pointers
Section64 (__DATA,__data)
Dynamic Loader Info
Function Starts
Symbol Table
Symbols
Data in Code Entries
Dynamic Symbol Table
Indirect Symbols
String Table
Code Signature

Offset	Data	Description	Value
000039C8	0000006D	Symbol	__\$S521_builtinStringLiteral17utf8CodeUnitCount7isASCIIS8Bp_BwB11_tcfC
		Indirect Address	0x100000E2 (\$+0)
000039CC	0000006F	Symbol	__\$s27_allocateUninitializedArraySayx6_BptBwLF
		Indirect Address	0x100000E8 (\$+6)
000039D0	00000070	Symbol	__\$s5Sprint_9separator10terminatoryypd_S2stf
		Indirect Address	0x100000E0 (\$+12)
000039D4	00000072	Symbol	__swift_beginAccess
		Indirect Address	0x100000E4 (\$+18)
000039D8	00000073	Symbol	__swift_bridgeObjectRelease
		Indirect Address	0x100000E1A (\$+24)
000039DC	00000074	Symbol	__swift_endAccess
		Indirect Address	0x100000E20 (\$+30)
000039E0	0000006E	Symbol	__\$S521_builtinStringLiteral17utf8CodeUnitCount7isASCIIS8Bp_BwB11_tcfC
		Indirect Address	0x100001000 (\$+0)
000039E4	00000071	Symbol	__\$sypN
		Indirect Address	0x100001008 (\$+8)
000039E8	00000075	Symbol	dyld_stub_binder
		Indirect Address	0x100001010 (\$+16)
000039EC	0000006D	Symbol	__\$S521_builtinStringLiteral17utf8CodeUnitCount7isASCIIS8Bp_BwB11_tcfC
		Indirect Address	0x100001000 (\$+0)
000039F0	0000006F	Symbol	__\$s27_allocateUninitializedArraySayx6_BptBwLF
		Indirect Address	0x100001008 (\$+8)
000039F4	00000070	Symbol	__\$s5Sprint_9separator10terminatoryypd_S2stf
		Indirect Address	0x100001010 (\$+16)
000039F8	00000072	Symbol	__swift_beginAccess
		Indirect Address	0x100001018 (\$+24)
000039FC	00000073	Symbol	__swift_bridgeObjectRelease
		Indirect Address	0x100001020 (\$+32)
00003A00	00000074	Symbol	__swift_endAccess
		Indirect Address	0x100001028 (\$+40)

- 类中的方法调度是一般函数表调度，同时和函数声明的位置与关键字有关。

- extesion声明方法
- final关键字
- static关键字
- @objc关键字
- dynamic关键字