

Programování zařízení Apple

IZA, Martin Hrubý, FIT VUT, 2025 / 26

Úvod do předmětu

Úvod do Objective-C

Dosavadní historie IZA

- Počátek — záběr na všechna zařízení
 - Jazyk Objective-C.
 - Jazyk Swift. Orientace na UIKit/ AppKit.
- Počátky SwiftUI. Opatrné experimenty :)
- Nyní:
 - poměrně vyspělý Swift,
 - SwiftUI již lze brát vážně,
 - ... přesto je dobré znát kořeny.

Cílová zařízení

- Primárně zařízení pod *iOS/ipadOS* — iPhone, iPod Touch, iPad.
 - iOS University Developer Program.
- Následně deriváty — Apple TV, Apple Watch.
- Přehled o macOS — OS X, iMac.
 - teoreticky sjednoceno přes SwiftUI.

Cíl předmětu

- Studium velmi významné platformy.
- Inspirace způsobem programování.
 - Jak to dělá někdo jiný...dle libosti srovnejte s např Androidem.
 - Měl by to být *předmět o programování*. Jako platformu pro ukázky si vezmeme Apple zařízení. **Debata o programování.**
- Programování v jazyce Swift (+ Objective C).
- Obecné principy programování *více-vláknových aplikací*. Programy s asynchronním konceptem.

V čem je problém?

- Programování app pro iOS je na první pohled snadné. Copy-paste demo příkladů.
 - Je třeba chápat smysl konstrukcí.
- Složitější programy vyžadují *více-vláknovost* a *asynchronnost*. Tady většina nováčků končí.
 - *Definovat a udržet v programu koncepci*. Návyky. Modely (Petriho sítě).
- Proto náš cíl: pochopit principy, naučit se modelovat *vnitřní stav programu*. SwiftUI — nutnost.

V čem je problém?

- Na iOS aplikace se kladou velké nároky:
 - plynulost ovládání a UI — zvyklosti, úroveň SW,
 - událostní řízení — plus vlákna (UI), uživatel je součástí aplikace,
 - asynchronní volání do jádra, ... SwiftUI/Combine
 - proměnlivost prostředí (přepínání aplikací, účtů, přerušení telef. hovorem, ...) — aplikace nemá "chvilku klidu",
 - multi-platformnost (víc zařízení jednoho uživatele),
 - ekosystém zařízení.

V čem je problém?

- Apple vždy kladl důraz na dokonalé chování svých produktů:
 - externí vývojáři by to mohli kazit...(uživatelský dojem ze zařízení). Co jim dovolíme?
 - uživatelé by si to sami mohli kazit...Co jim dovolíme?
- Koncepce produktů / SW Apple tomu měla vždy bránit.
 - Srovnání s Androidem. Problém "home adresáře".
 - První počítače Apple (I. a II.) byly hackerské DIY stavebnice.

Objective-C -> Swift

- Objective-C:

- triviální koncepce,
- ukecanost kódu, pracnost — ale bastleníčko... 😅

- Swift (poučení z krizového vývoje)

- striktní typová kontrola, optional hodnoty, bigotní init(...)
- kontextové kontroly (throws / try, escaping, override)
- vysoká produktivita práce

Storyboard -> SwiftUI

- Storyboarding (UIKit): MVC

- místy chaotické programování MV-C programu,
- hodně pracné, ukecané — ale tolerující "bastlení".
- geometrie Views byla peklo...

- SwiftUI: MVVM

- striktní architektura programu, "ordnung"
- vysoká produktivita práce,
- abstrakce -> přenositelnost mezi zařízeními Apple.

Apple: 48 let konzistence...

- Swift / SwiftUI stojí na principech předchozího vývoje nástrojů.
- jenom je balí do hezčích kabátků,
- směrem k vyšší produktivitě práce v programování,
- směrem k robustnější koncepci programů.

Milníky pochopení Swiftu

- Pochopení koncepce *struct*.
 - Swift je sice OOP, ale třídy a OOP snad už nikoho nezajímají :)
 - pointer zapouzdřený do reference, ref. zapouzdřená do hodnoty
 - Komunikace mezi objects / structs. SwiftUI.
- Pochopení closures. (closure versus callback)
- Protokoly (koncepty) a extensions.
 - Dříve spíš delegátství (komunikace). Abstrakce v programu.
- Generické programování.

struct / class — immutable / mutable

- Mutable chování má být výjimečné (let / var).
 - Pozn.: paradoxy var v @IBOutlet (později...).
- NS(Mutable)Array versus [Element]
 - Všechny kolekce ve Swiftu jsou **struct** (!!!).
- Předání:
 - reference — teď více stran něco sdílí (je to OK?)
 - hodnota — každý máme svoji vlastní kopii.

Typická aplikace — uložená data

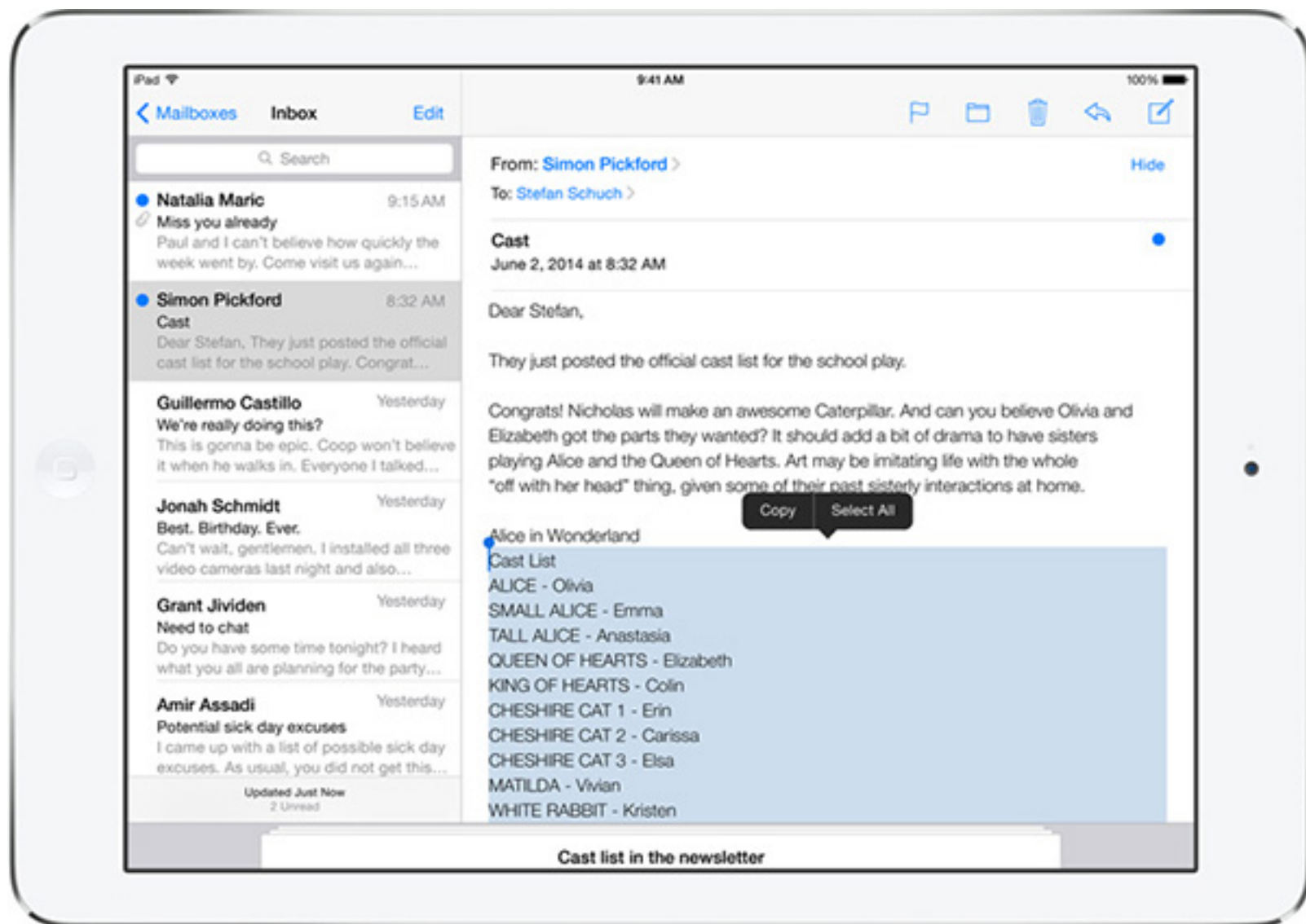
- Uživatelská konfigurace (stav) — lokální/synchronizovaná.
- Datové schema lokální a *exportované*.
 - *Dokumenty*. Synchronizace. Verzování.
 - Lokální DB — CoreData.
 - Externí DB — iCloud (CloudKit).
 - Další — Realm, Firebase, Amazon AWS.
- Rozhraní na data pro UI, tzv. data source.
 - dnes nazýváno Model, ViewModel (SwiftUI).

Typická aplikace — řízení (Cnt)

- Řízení přístupu k datům.
- Řízení komunikace s prostředím. Synchronizace mezi zařízeními uživatele.
- Řízení UI.
- Koncept komunikace uvnitř programu — delegáti, posílání zpráv, KVO.
- Maximální znovupoužitelnost kódu — tzv. controllers.

Typická aplikace — UI

- Existují zvyklosti. Guidelines...
- Typicky Views skládáme z knihovných komponent.
- Schvalování aplikací do AppStore.
- Návrh stylu UI — iPhone/iPad verze.
- Estetika aplikace, logika ovládání aplikace, koncept naprogramování UI.



Jak na to půjdeme?

- Postupně.
- Jazyky a jazykové konstrukce — Objective-C a Swift.
- Konstrukce základní aplikace — UI.
- Data v aplikaci — kódování, DB, Dokumenty.
- Řízení aplikace — vlákna, operace.
- Doplnky: hry, Apple Watch / TV, desktop.

Jak na to půjdeme

- Je třeba chápat *historické souvislosti* — vývoj zařízení Apple, vývoj knihoven, jazyků.
 - Chápat **proč** se styl programování takto vyvinul.
- Je třeba chápat smysl programových konstrukcí.
- Rozbor postupů — rozebrat si špatné a dobré řešení.
- Minimum je zvládnout udělat aplikaci. *Optimum je rozumět ji.*

Motto (S. Jobs)

- Pokud váháš, zda-li přijít v sobotu do práce, v neděli už chodit nemusíš.
- Výrobek musí být dokonale zpracovaný i v částech, na které zákazník přímo nevidí.
 - Kde to končí?

Organizace předmětu

- Přednášky.
- Demo-cvika. Semináře.
- Projekt.
- Zkouška.
- Technické prostředky: Mac + XCode.
 - Virtualizace macOS. Volně šiřitelný Swift. Serverový Swift.

Vývoj aplikací pro iOS

- Vývojové prostředí XCode, Mac.
 - Musí tam být obrázek jablka :)
- Emulátor iOS — virtualizace iOS zařízení.
- Vývojářský účet.
- Podepisování aplikací.
- Návaznost na iCloud — dokumenty, CloudKit.

Osnova přednášek — základy

- Úvod. Objective-C. Historické kořeny.
- Swift I. — Základy, kolekce, class / struct.
- Swift II. — Closures. Protokoly. Extensions.
- Swift III. — Templates.

Osnova přednášek — apps

- MVC: Základy aplikací v MVC (Storyboard).
 - Základy MVC. TableView. Geometrie Views.
- SwiftUI I.: Základy aplikací ve SwiftUI:
 - View (some View), @State, @Binding
 - view modifiers
 - NavigationView, TabView, ...

Osnova přednášek — core SwiftUI

- SwiftUI II.

- Geometrie v aplikacích — VStack/HStack.
- @ViewBuilder — konstrukce View.

- SwiftUI III.

- Combine. @Published. Řízení modelu a procesů.

- SwiftUI IV.

- Prerekvizita: CoreData, Procesy, Kódování dat & Document.
- Pokročilé konstrukce.

Osnova přednášek — DATA & Procesy

- Paralelismus v aplikacích (GCD, Operation).
 - Combine. async / await.
 - MainThread (mutex netřeba). Globální vlákna. Fronty.
- Kódování dat (Document, iCloud, FileManager).
- CoreData. OO databáze v aplikaci.
 - "must" znalost. CoreData v hlavním / vedlejším vlákně.
- CloudKit. — Amazon AWS.

Přednášky hostů

- David Procházka — umění tvorby uživatelského rozhraní aplikace.

Democvika, semináře.

- Několik democvičení s praktickými ukázkami programů.
- Ideálně formou hromadných konzultací, diskuzí.
 - Diskuze nad studentskými programy.
- Začneme 4.-5. týden semestru.

Počátky Apple

- 1976 — Steve Jobs a Steve Wozniak zakládají Apple Computer Inc.
 - Osobní počítače Apple I. (100ks) a II. (miliony ks).
- Různé modely dalších počítačů. Lisa. Macintosh.
 - 1985 — Jobs vyhozen z vedení Apple.
 - 1997 — vrací se.
- 80 / 90 léta. Éra vzniku osobních počítačů.
- Walter Isaacson: Steve Jobs.

První počítače Macintosh, 1984

- Paměťové modely — dvojitý pointer.
- Kooperativní multi-tasking.
- V PC-světě: MS-DOS.
 - 90-léta: Nástup CPU i386
 - SCO Unix na PC (386)
 - Počátky Internetu v ČR.



Vliv Smalltalku

- XEROX.
- Vývoj Smalltalku — koncept GUI, myš (A. Kay, D. Ingalls, A. Goldberg).
 - Licencováno pro počítače Apple.
- Objective C.
 - Foundation. AppKit.

Jobsův "exil"



- NeXT Computer Inc. Dále pak Pixar.
- Počítač NeXT (Cube), NeXTSTEP.
 - Mikrojádro Mach (Carnegie Mellon University).
 - BSD Unix — služby OS.
 - Objektově orientované knihovny Foundation, Cocoa. Prefix *NS*.
 - Objektově orientovaný OS.
 - Objective C.

Jobsův návrat

- Integrace NeXTSTEPu do počítačů Apple.
- OS X — čteme "OS Ten".
- Dnešní OS X (macOS) je pokračovatel NeXTStepu.
 - Mach+BSD základ.
 - Knihovny Cocoa a Foundation — AppKit.
- macOS / iOS jsou UNIXové systémy!

80 / 90 léta

- Apple svět — Apple II, Macintosh apod.
- Microsoft svět — MS-DOS 3.3, 5.0, Windows 3.11 (1992-94), Windows NT, ...
- IBM OS/2.
- UNIXový svět.
- Procesory Motorola, Intel (86, 286, 386, 486, ...).
 - Apple: začínal na Motorole, PowerPC, později Intel.
 - Dnes ARM. Řada procesorů A. Procesor M1/M2.

Linie OS v Apple

- Počítače (stolní): NeXTSTEP -> OS X.
- Mobilní počítače: OS X -> iOS.
- iOS -> watchOS, tvOS.
 - Nositelná zařízení — Apple Watch. Apple AirPods.
 - Domácí spotřební elektronika — Apple TV. HomeKit.
- Plně POSIX kompatibilní.
- Maximální znovu-použitelnost knihoven.

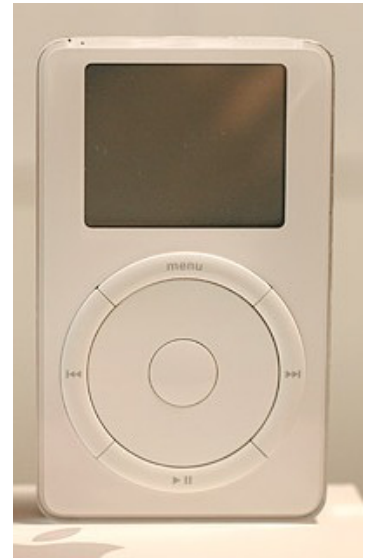
Počátky mobilních počítačů

- PDA — Personal Digital Assistant.
- 1993 — Apple, Newton.
- 1996 — Palm, Palm Pilot.



Hudba, iPod, iTunes

- iPod (2001) a myšlenka oddělené konfigurace zařízení přes iTunes.
 - Ne přímému (datovému) přístupu do zařízení.
 - Žádné datové porty.
 - Žádný home adresář.
- Úspěch iPodu otevřel nové trhy spotřební elektroniky.
 - Otevřel nové trhy: telefony, tablety apod.



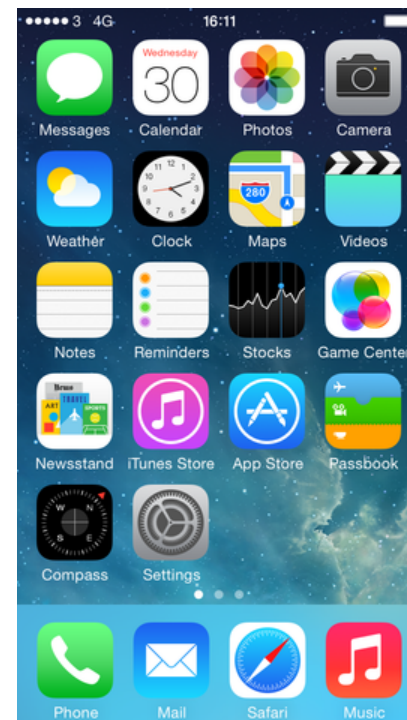
Počátky multi-touch

- iPod, 2001. Projekt Newton.
- Tablet — ovládání tužkou nebo multi-touch.
- Pokusy v Microsoftu. Spolupráce s Motorola (ROKR)



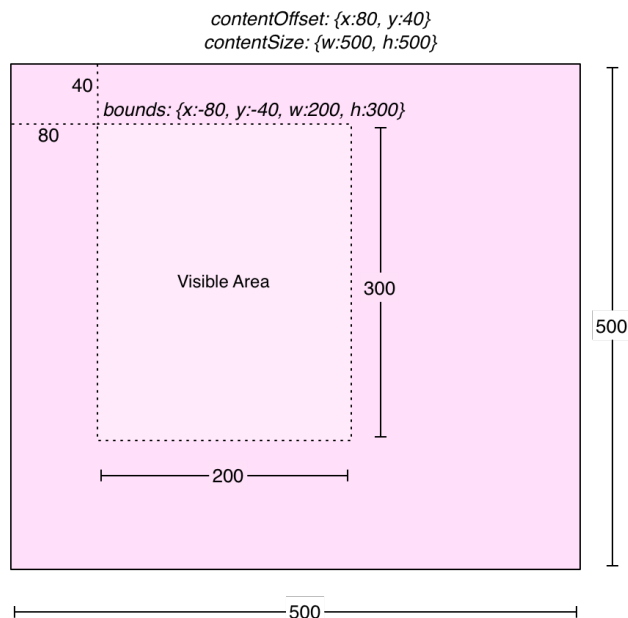
Projekt vývoje iPhone

- 2005 — Projekt extrémní významnosti pro Apple.
- Jaká forma? iPod nebo dotyková obrazovka.
- iPhone 2G — leden 2007. V prodeji červen 2007.



Vývoj v počátcích

- Posuvný obsah obrazovky (UIScrollView).
- Ovládání gesty (rozpoznávače gest).
- Koncept klávesnice.



Jazyky pro programování

- Objective-C — převzat Applem a dále rozvíjen.
- **Swift** (dnes 5.6).
- *Znalost jazyka versus schopnost programovat.*
- Pro pochopení Swiftu a programování AppleDev bude nezbytné poznat Objective-C.
 - Properties. Datové struktury. ARC.
 - Key-Value Coding, Key-Value Observing.
 - Řízení programu. Bloky, delegáti, GCD, vlákna, callback apod.

Objective-C

- C s konceptem Smalltalkového posílání zpráv.
- Původ B. Cox a T. Love, Stepstone.
- Jazyk pro NextSTEP (1988). Později OS X a iOS.
 - 1996 — NeXT přešel pod Apple.
- Knihovna Foundation. Cocoa, Cocoa Touch (UIKit).

Koncepty Objective-C

- *Protokoly* — vícenásobná dědičnost na úrovni rozhraní. Delegáti / dataSource.
- *Extensions* — rozšiřitelnost rozhraní třídy.
- Garbage collection. ARC.
 - Swift: strong-ref, weak, unowned, unsafe(unowned)
- *Properties* — atributy s programovatelnými setter / getter. Základ KVC / KVO.
- Bloky (lambda výrazy, closures).

Swift

- Představen na WWDC 2014.
- Syntaktická revize Obj-C a (pár) konceptů navíc.
- Kompilovaný jazyk (na rozdíl od Java / C#).
- Koncept *hodnota* versus *reference* (a NULL).
 - Deklarace "var" a "let". Optional value — Int?
 - Pointer versus reference?

Koncepty Swiftu

- Prý jazyk vhodný pro začátečníky :D
 - Mimořádně bohaté možnosti abstrakce v programování.
 - Datové typy -> Protokoly.
- struct / class — předávání hodnoty nebo ref.
- enum — výčet hodnot (strukturovaných).
- Generické programování.

Objective-C (ochutnávka)

Koncept jazyka

- Objective-C je směska C a třídní nadstavby.
 - Čistá OO jazyka Smalltalk: vše je objektem, hlavní nadtřída.
 - Instanční / třídní metody. Metaprogramování.
 - *@objc* pro Swift (zvyky Objective-C v knihovnách Swiftu).
- Obj-C kombinuje primitivní DT a objekty (NSObject).
 - čísla — int, long, ..., NSNumber, NSString, ...
 - NSArray — kolekce jsou heterogenní.

Interface třídy

```
@interface TRIDA : NSObject
{
    int number;
    NSString *text;
}

-(id) init;
-(id) initWithNumber: (int) c;

+(TRIDA *) create;

-(int) number;
-(void) setNumber: (int) c;

@end
```

```
@interface NSObject(myExtensions)
-(void) sayHello;
@end

@protocol MyProtocol
-(BOOL) canFly;
@end
```

Implementace rozhraní

```
@implementation TRIDA
```

```
-(id) init {  
    self = [super init];  
    return self;  
}
```

```
-(id) initWithNumber: (int) c {  
    self = [super init];  
    if (self != NULL) {  
        number = c;  
    }  
    return self;  
}
```

```
+(TRIDA *) create  
{  
    return [[TRIDA alloc] init];  
}
```

```
-(int) number { return number; }  
-(void) setNumber: (int) { number = c; }
```

```
@end
```

Práce s objektem, zprávy

- `TRIDA *obj = [[TRIDA alloc] init];`
 - Třídní metoda `alloc`, alokace paměti. Paměť je nulovaná.
 - Inicializace — Obj-C / Swift nemluví o konstrukci. Explicitní inicializace je v podstatě (téměř) dobrovolná. Jak ve Swiftu.
 - Konvence `NSObject`: inicializace reference countingu.
 - `AutoreleasePool`.
- `[obj zprava]; [obj setNumber: 3];`
- Není "virtual". Ve Swiftu naopak — "override".

NULL

- Objective-C všechny proměnné automaticky nuluje.
- TRIDA *obj; // je NULL
- [obj number]; // není seg-fault
 - V Obj-C lze posílat NULL-ref zprávu, vrací se nějaká nulová hodnota (0, NO, NULL).
 - Běžně se s tím v programech počítá. Odpadá to "if (ref) ...".
 - Pozn.: Swift se vyvinul úplně opačným směrem.

setter / getter

- Nelze přistupovat na obj.number, instanční proměnné jsou skryté (Smalltalk).
- Přístupové metody: getter / setter
 - Přístupem na inst. proměnnou voláme metodu objektu!
 - Toho by se dalo nějak využít. KVO — key / value observing.
- Ruční psaní getterů / setterů bylo otravné, takže se zavedly properties (WWDC 2006, Obj-C 2.0).
- Jmenné konvence v Obj-C.

Property

- Property je public instanční proměnná, která automatizuje deklaraci setteru / getteru.
- `@synthesize`, `@dynamic`
- tečková notace
- opět volání metody!
- `@propertyWrapper`

```
@interface TRIDAB
//
@property int numberb;
@property NSString *textb;
@end

@implementation TRIDAB
@synthesize numberb, textb;
@end

//
TRIDAB *b = ...;
//
b.numberb = 3;
[b setNumberb: 3];
int bb = b.numberb;
```

```
@implementation TRIDAB
/* @synthesize numberb = _numberb; */

-(int) numberb {
    // willAccess, didAccess...
    return _numberb;
}

-(void) setNumberb:(int)v {
    [self willChangeValueForKey: @"numberb"];
    _numberb = v;
    [self didChangeValueForKey: @"numberb"];
}
@end
```

Key Value Coding (KVC)

- Základní protokol implementovaný v NSObject.
- Zavádí abstrakci nad properties.
- Každý objekt je slovník (dictionary, map) typu klíč/hodnota:
 - - (id)valueForKey:(NSString *)key;
 - - (void)setValue:(id)value forKey:(NSString *)key;
 - může implementovat libovolné klíče.
- Swift: máme obdobu v tzv. KeyPath

KVC

- Metody valueForKey / setValue:forKey: jsou generovány automaticky, tj. reagují na deklarované properties.
 - [b valueForKey: @"textb"]
 - [b setValue: @"ahoj" forKey: @"textb"]
 - ... včetně hierarchie dědičnosti
- Jaké může být použití KVC v programech?
 - UIKit. CoreData.

Key Value Observing (KVO)

- Vnitřní mechanismus Foundation pro předávání zpráv mezi objekty o změnách hodnot properties.
- Objekt A, jeho property X.
- Observer B — jiný objekt B, který dostane zprávu, když X změní hodnotu.
- Použití konceptu. K čemu je to dobré?
 - SwiftUI abstrakce: `@Binding`, `@ObservedObject`

NSObject ve Swiftu

- Je stále implementován. Protokoly z Foundation.
- Sémantika:
 - KVC — `value(forKey:)`, `set(value: forKey:)`
 - KVO — `addObserver`, `observer(..., didChangeValue: ...)`
 - `respondsToSelector(...)`

Knihovna Foundation (NS...)

- NotificationCenter. UserDefaults.
- Grand Central Dispatch. OperationQueue.
- CoreData. CloudKit.
 - Stále NS kolekce.
- NSCoder — Swift: Codable.
- ...

Závěr

- Objective-C je historický vývojový předek Swiftu.
- V Objective-C je stále napsána spousta kódu aplikací/knihoven.
- Příště bude Swift.