

Programování zařízení Apple

IZA, Martin Hrubý, FIT VUT, 2018

Úvod do předmětu

Úvod do Objective-C

Cílová zařízení

- Primárně zařízení pod iOS — iPhone, iPod Touch, iPad.
- Následně deriváty — Apple TV, Apple Watch.
- Přehledově macOS — OS X, iMac.

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.

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.*
- Proto náš cíl: pochopit principy, naučit se modelovat vnitřní stav programu.

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,
 - proměnlivost prostředí (přepínání aplikací, účtů, přerušení telef. hovorem, ...) — aplikace nemá "chvilku klidu",
 - multi-platformnost (víc zařízení jednoho uživatele).

V čem je problém?

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

Elegance programování

- Programování je umělecké řemeslo. Vliv zkušeností.
- Maximální znovupoužitelnost a flexibilita.
- Rozsah použitelných zařízení.
- Rozsah morální životnosti. Dlouhověkost SW :)
- Koncepce knihoven Foundation.

Typická aplikace — uložená data

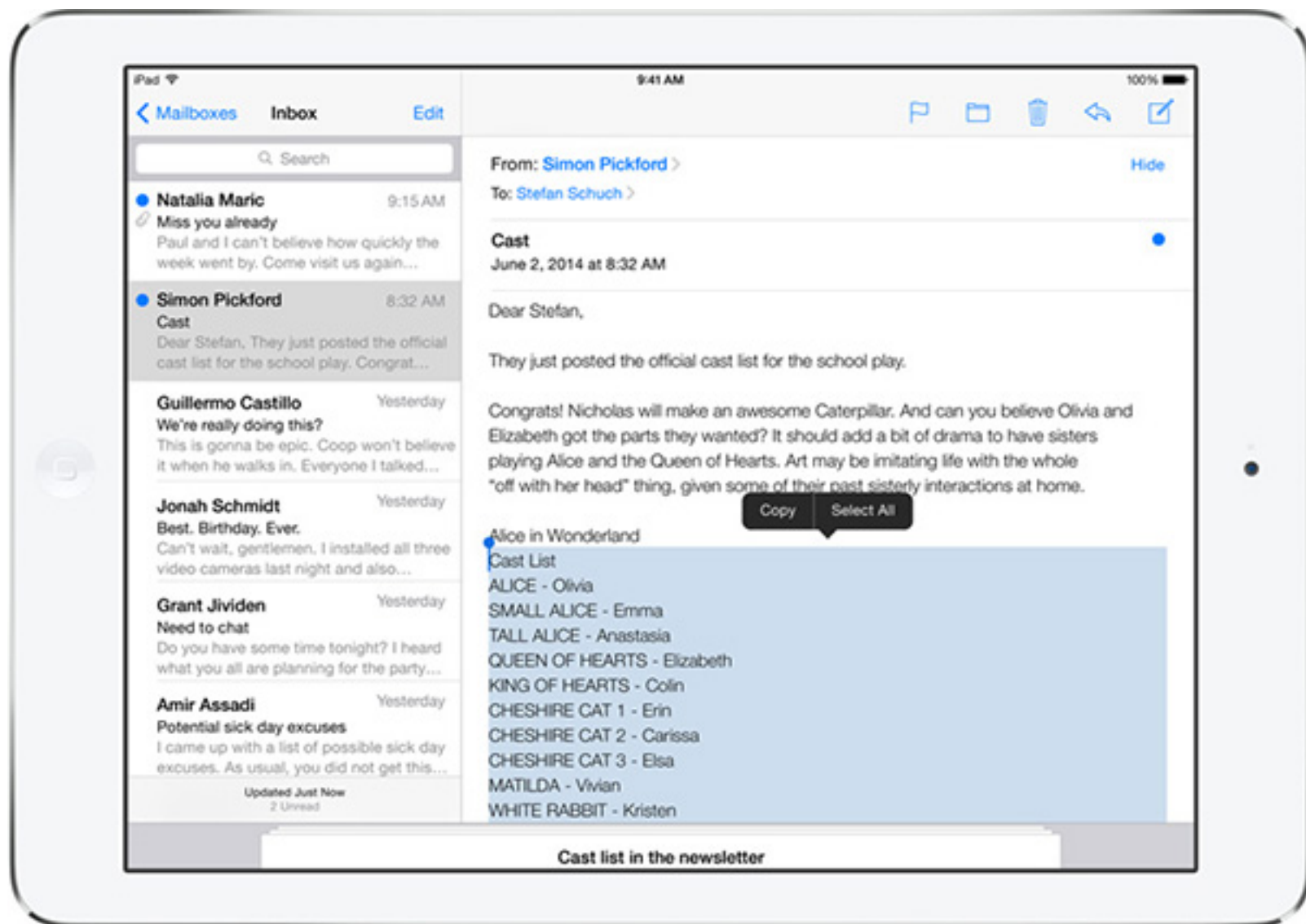
- Uživatelská konfigurace (stav) — lokální/synchronizovaná.
- Datové schema lokální a exportované.
 - Dokumenty. Synchronizace. Verzování.
 - Lokální DB — CoreData.
 - Externí DB — iCloud (CloudKit).
- Rozhraní na data pro UI, tzv. data source.

Typická aplikace — řízení

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

Typická aplikace — UI

- Existují zvyklosti. User Experience.
- Schvalování aplikací do AppStore.
- Návrh stylu UI — iPhone/iPad verze.
- Estetika aplikace, logika ovládání aplikace, koncept naprogramování UI.



iPad

9:41 AM

100%

< Mailboxes Inbox Edit

🚩 📁 🗑️ ↩️ ✍️

🔍 Search

● **Natalia Maric** 9:15 AM
Miss you already
Paul and I can't believe how quickly the week went by. Come visit us again...

● **Simon Pickford** 8:32 AM
Cast
Dear Stefan, They just posted the official cast list for the school play. Congrat...

Guillermo Castillo Yesterday
We're really doing this?
This is gonna be epic. Coop won't believe it when he walks in. Everyone I talked...

Jonah Schmidt Yesterday
Best. Birthday. Ever.
Can't wait, gentlemen. I installed all three video cameras last night and also...

Grant Jividen Yesterday
Need to chat
Do you have some time tonight? I heard what you all are planning for the party...

Amir Assadi Yesterday
Potential sick day excuses
I came up with a list of possible sick day excuses. As usual, you did not get this...

Updated Just Now
2 Unread

From: [Simon Pickford](#) >[Hide](#)To: [Stefan Schuch](#) >

Cast
June 2, 2014 at 8:32 AM

Dear Stefan,

They just posted the official cast list for the school play.

Congrats! Nicholas will make an awesome Caterpillar. And can you believe Olivia and Elizabeth got the parts they wanted? It should add a bit of drama to have sisters playing Alice and the Queen of Hearts. Art may be imitating life with the whole "off with her head" thing, given some of their past sisterly interactions at home.

Copy

Select All

Alice in Wonderland
Cast List
ALICE - Olivia
SMALL ALICE - Emma
TALL ALICE - Anastasia
QUEEN OF HEARTS - Elizabeth
KING OF HEARTS - Colin
CHESHIRE CAT 1 - Erin
CHESHIRE CAT 2 - Carissa
CHESHIRE CAT 3 - Elsa
MATILDA - Vivian
WHITE RABBIT - Kristen

Cast list in the newsletter



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.

Organizace předmětu

- Přednášky. Pozn.: doba přednášky. IJA.
- Semináře.
- Projekt.
- Zkouška.
- Technické prostředky: Mac + XCode.

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. Paměť. Garbage collecting.
- Objective-C.
- Swift. Kolekce. Foundation.
- Model-View-Controller (MVC).
- Konstrukce aplikace v iOS.

Osnova přednášek — core

- Kódování dat (Document, iCloud).
- CoreData. OO databáze v aplikaci.
- Paralelismus v aplikacích (GCD, NSOperation).
- CloudKit.

Osnova přednášek

- Životní cyklus aplikace.
- SpriteKit.
- Programování pro MacOS.
- Programování pro WatchOS, tvOS.
- Přednáška hosta.

Democvika, semináře.

- Několik democvičení s praktickými ukázkami programů.
- Ideálně formou hromadných konzultací, diskuzí.
- 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.
 - 1985 — Jobs vyhozen z Apple.
 - 1997 — vrací se.
- 80 / 90 léta. Éra vzniku osobních počítačů.
- Walter Isaacson: Steve Jobs.

Vliv Smalltalku

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

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.



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, Windows NT, ...
- IBM OS/2.
- UNIXový svět.
- Procesory Motorola, Intel (286, 386, 486, ...).
 - Apple: začínal na Motorole, PowerPC, později Intel. Dnes ARM.

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. Homepod.
- Maximální znovupoužitelnost 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.



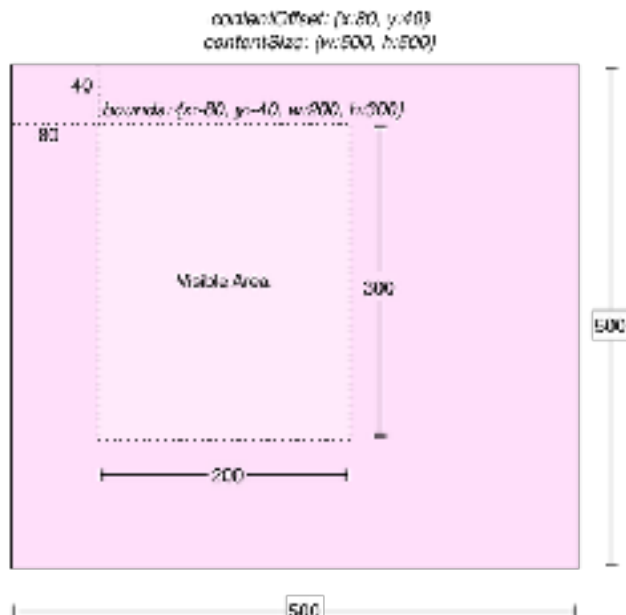
Počátky multi-touch

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



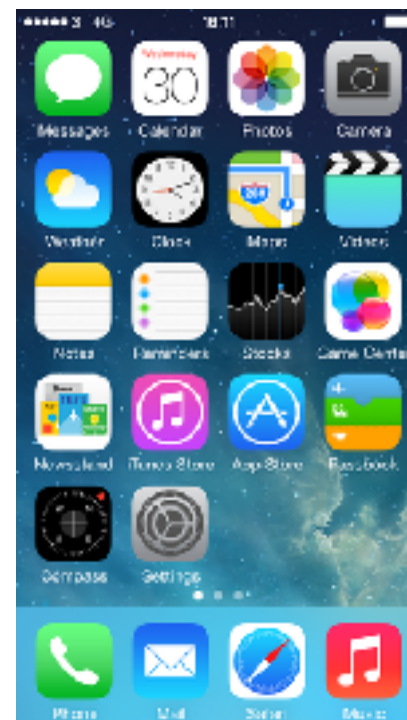
Vývoj v počátcích

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



Projekt vývoje iPhone

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



Jazyky pro programování

- Objective-C — převzat Appllem a dále rozvíjen.
- Swift (4.0).
- 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.
- *Kategorie* — rozšiřitelnost rozhraní třídy.
- Garbage collection.
- *Properties* — atributy s programovatelnými setter / getter. Základ KVC / KVO.
- Bloky.
- Zpráva pro NULL objekt.

Swift

- Představen na WWDC 2014.
- Syntaktická revize Obj-C a pár konceptů navíc.
- Koncept hodnota versus reference (a NULL).
 - Deklarace "var" a "let"
 - Wrapped value — String?, Int?
 - `mujString?.count`

Koncepty Swiftu

- Poněkud překošatělé možnosti jazyka.
- struct / class — předávání hodnoty nebo ref.
- enum — výčet hodnot (strukturovaných).
- N-tice (nepojmenované struktury).
- Licence pro volné využití i mimo Apple zařízení.

Objective-C

Koncept jazyka

- Objective-C je směska C a třídní nadstavby.
- Čistá OO jazyka Smalltalk: vše je objektem, hlavní natřída.
- Obj-C kombinuje primitivní DT a objekty (NSObject).
 - čísla — int, long, ...
 - ukazatele.

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) c { number = c; }
```

```
@end
```

Práce s objektem, zprávy

- `TRIDA *obj = [[TRIDA alloc] init];`
 - Třídní metoda `alloc`, alokace paměti.
 - Inicializace — Obj-C / Swift nemluví o konstrukci. Inicializace je v podstatě (téměř) dobrovolná.
 - 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!

```
@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 {
    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.

KVC

- Metody valueForKey / setValue(forKey): jsou generovány automaticky, tj. reagují na deklarované properties.
 - [b valueForKey: @"textb"]
 - [b setValue: @"ahoj" forKey: @"textb"]
 - ... včetně hierarchie dědičnosti
- Jak do KVC zapadají primitivní DT?
- Jaké může být použití KVC v programech?

KVC, NSNumber

- int, BOOL, long apod. nelze provozovat v KVC (typ id).
- Foundation zavádí abstrakci NSNumber.
- Všechna aplikačně používaná čísla jsou instance NSNumber.
 - @(1), @(3.14), @(YES)
- Nutnost i pro kolekce (NSArray apod).

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.

Vlastnictví objektu

- Objekt A vlastní objekt B.
- Objekt B nemůže žít déle než A.
- Vlastnictví objektu: během destrukce A se dealokuje i B.
- Problém: referenci B chceme v programu předávat (ukládat do kolekcí, předávat při posílání zpráv, vázat na jiné objekty).
- Rozdíl ukazatel versus reference?

Paměť aplikace

- V programu rozlišujeme data organizovaná v:
 - Datovém segmentu aplikace, statická, globální.
 - Dynamicky přidělovaná z haldy (heap, alloc, new, ...) — low-memory warning, uvolňování paměti.
 - Dynamicky alokovaná na zásobníku.
- Vztah vznikající: $\text{TRIDA } *b = [\text{TRIDA alloc}];$
 - Data reference "b" — paměťové místo.
 - Objekt třídy TRIDA — typicky z heapu.

Proměnná "TRIDA *b"

- Je opět v nějakém kontextu vyšší struktury:
 - Je instanční proměnnou objektu (nebo třídní proměnnou třídy).
 - Je lokální proměnnou procedury (tj. na zásobníku).
 - Je globální proměnnou programu.
 - Je prvkem kolekce libovolného typu.

Vlastnictví objektu

- Objekt A vlastní objekt B. S koncem A se dealokuje i B.
 - konstruktor a destruktork
 - Potřebujeme B referencovat i z objektu C.
- Strong / weak reference.
 - strong — provádí dealokaci B, weak — referenci "pouze čte".
- Pořádek ve vlastnictví — největší problém programování.

Reference counting

- Objekt B bude mít *int counter* čítač, kolikrát je referencován.
- Referencování — inkrementace counter.
- DReferencování — dekrement counter.
- Po dosažení `counter==0`, dealokace.
 - Implementace — okamžitá, odložená (pool).
- Pomohli jsme si?

V C++, Obj-C, Swift, Java ...

```
-(void) metoda
{
    TRIDA *b = [[TRIDA alloc] init];
    // konfigurace "b"
    ...
    //
    return;
}
```

Problémy ref-countingu.

- Manuální incr / decr čítače — zodpovědnost programátora (retain / release, autoreleasepool).
 - při volání, noveJmeno — inkrementuje ref-count?
- Paralelizmus — výlučnost přístupu na counter.
- Časová režie — incr / decr.

```
-(void) setJmeno: (NSString *)noveJmeno
{
    [noveJmeno retain];
    [_jmeno release];
    //
    _jmeno = noveJmeno;
}
```

Automatic RefCounting (ARC)

- Automatizuje retain / release a pool.
- Inkrementace:
 - Přiřazení reference,
 - předání ref jako argumentu volání metody (režie!).
- Dekrement:
 - Nulování reference.

C++

```
typedef std::shared_ptr<Trida> Trida_ptr;  
typedef Trida *Trida_wptr;  
//  
Trida_ptr _ptr = std::make_shared<Trida>(...);  
  
void metoda(const Trida_ptr &aTrida)  
{  
    //  
    Trida_wptr _wp = &*aTrida;  
}
```

Property v ARC

- `@property (strong) NSString *jmeno;`
- `@property (readonly) Document *doc;`

Property, lazy vyhodnocení

```
@synthesize doc = _doc;

-(Document *) doc {
    //
    if (_doc == NULL) {
        _doc = [[Document alloc] init];
        [_doc readAndInit];
    }
    //
    return _doc;
}

//
self.doc;
//
_doc = NULL;
```

Synchronní / asynchronní

- Co znamená synchronní volání?
- Vlákna v programu.
- Řízení běhu programu.
- Všechny časově náročné operace se provádí asynchronně.

```
-(void) akceNaTlacitko
{
    // aktivuj sluzbu jadra
    // napr. nacteni souboru
}

-(void) souborBylNacten
{
    // aktualizuj UI
}
```

Kolekce — kontainery *objektů*

- Základní dělení: mutable, immutable.
 - NSMutableArray, NSArray.
- Typy kolekcí:
 - Vektor — Array,
 - Množina — Set,
 - Slovník — Dictionary.
- Obj-C kolekce jsou heterogenní.

Závěr

- Příště bude Swift.