Česká zemědělská univerzita v Praze

Technická fakulta

Možnosti tvorby aplikací na systémech Mac OS

bakalářská práce

Vedoucí bakalářské práce: doc. Ing. Vojtěch Merunka, Ph.D.

Vypracoval: Radek Jonáš

PRAHA 2011

Česká zemědělská univerzita v Praze Technická fakulta

Katedra informačního inženýrství Akademický rok 2009/2010

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Radek Jonáš

obor Informační a řídící technika v agropotravinářském komplexu

Vedoucí katedry Vám ve smyslu Studijního a zkušebního řádu ČZU v Praze čl. 16 určuje tuto bakalářskou práci.

Název práce: **Možnosti tvorby aplikací na systémech MacOS**

Osnova bakalářské práce:

1. Úvod
2. Cíl práce a metodika
3. Literární rešerše
4. Historie Apple Inc.
5. Mac OS X
6. Závěr
7. Seznam použitých zdrojů
8. Přílohy

Rozsah hlavní textové části: 30 - 40 stran

Doporučené zdroje:

KAHNEY, Leander. Jak myslí Steve Jobs. Praha : Computer Press, 2009. 280 s. ISBN 978-80-251-2361-4.

WOZNIAK, Steve; SMITH, Gina. IWoz : Muž, který postavil první osobní počítač. Vyd. 1. Praha : Pragma, 2007. 360 s. ISBN 978-80-7349-064-5.

ČADA, Ondřej. Cocoa : úvod do programování počítačů Apple. Vyd. 1. Praha : Grada, 2009. 200 s. ISBN 978-80-247-2778-3.

SIMON, William; YOUNG, Jeffrey. IKona Steve Jobs : Znovuzrození Apple - nejúžasnější návrat v historii byznysu. Vyd. 1. Praha : Eugenika, 2008. 360 s. ISBN 978-80-8100-077-5.

KOCHAN, Stephen G. Objective-C 2.0. Praha : Computer Press, 2010. 552 s. ISBN 978-80-251-2654-7.

POGUE, David. Mac OS X Snow Leopard : Kompletní Průvodce. Praha : Computer Press, 2010. 952 s. ISBN 978-80-251-2793-3.

POGUE, David. IPhone 3GS : Průvodce s tipy a triky. Praha : Computer Press, 2010. 368 s. ISBN 978-80-251-2852-7.

DAVIDSON, James Duncan. Learning Cocoa with Objective-C. Second Edition. Cupertino : O`Reilly Media, 2002. 384 s. Dostupné z WWW: <http://oreilly.com/catalog/9780596003012>.

Vedoucí bakalářské práce: **doc. Ing. Vojtěch Merunka, Ph.D.**

Termín zadání diplomové práce: listopad 2009

Termín odevzdání bakalářské práce: duben 2011

Vedoucí katedry Děkan

V Praze dne: 30. 11. 2009

Čestné prohlášení

Prohlašuji, že jsem bakalářskou práci na téma: „Možnosti tvorby aplikací na systémech Mac OS“ vypracoval samostatně pod vedením doc. Ing. Vojtěcha Merunky, Ph.D. a využil jsem pouze citovaných pramenů, které jsou uvedeny v seznamu použité literatury. Informace, u kterých není uvedena citace, vycházejí ze zkušeností autora práce nebo z informací odborníka pracujícího v tomto oboru.

V Praze dne: …….

Radek Jonáš

Poděkování

Děkuji vedoucímu bakalářské práce doc. Ing. Vojtěchu Merunkovi, Ph.D. za cenné rady a odbornou pomoc při zpracování této bakalářské. Dále bych chtěl poděkovat panu Ondřeji Čadovi za poskytnutí cenných informací v programování, které získal mnohaletou praxí v daném oboru a za laskvaé propůjčení jeho ukázkového programovacího kódu.

V Praze dne: ……

Radek Jonáš

Možnosti tvorby aplikací na systémech Mac OS

bakalářská práce

Abstrakt:

Obsahem této práce je seznámení s historií počítačové firmy Apple Inc., jejích dosavadních verzí operačního systému a možností tvorby aplikací pro nejnovější verze systému Mac OS. V neposlední řadě také seznámení se základy programovacího jazyku Objective-C a hlavně s nejvyspělejším programovacím rozhraním, kterým je právě prostředí Cocoa.

Klíčová slova: Apple, Mac, Mac OS, Objective-C, Cocoa

Abstract:

The content of this work is to introduce the history of Apple Computer Inc.., its currentoperating system version and the possibilities for creating applications for the latestversion of Mac OS. Last but not least, familiar with basic programming languageObjective-C, and especially with the most advanced programming environment, which isjust the Cocoa environment.

Key words: Apple, Mac, Mac OS, Objective-C, Cocoa

Obsah

1 Úvod

Firma Apple Inc. v dnešní době patří k vysoce uznávaným společnostím v odvětví informačních technologií. Je to právě firma Apple, kdo udává trend v počítačovém průmyslu. Zásadní inovace v technologiích a dech beroucí design přitahuje na svou stranu stále více spokojených uživatelů, mezi které patřím již 4 rokem i já. Toto téma bakalářské práce jsem si vybral, nejen protože jsem spokojený uživatel Mac OS X, ale také proto, že vidím velký potencionál v programování pro tuto platformu. Po předvedení prvního telefonu iPhone Apple rozjel nový převratný způsob prodeje aplikací. Jeho online obchod App Store je v dnešní době velice oblíbený, obsahuje přes 300 000 aplikací pro telefony iPhone, přehrávače iPod Touch a tablet iPad. Aplikace jsou vždy testovány a pouze kvalitní se dostanou do rukou uživatelů. Obchodní podmínky pro vývojáře jsou velice přívětivé, tudíž jejich počet rapidně v poslední době vzrostl. Nedávno Apple představil jeho mladší sourozence Mac App Store pro svůj operační systém Mac OS X. Mě osobně programování pro tuto platformu velice láká. Proto bych se mu chtěl v této bakalářské práci věnovat. Částí práce se zmíním o historii společnosti a následně zrekapituluji všechny dosavadní verze jejího operačního systému Mac OS.

Informuji o možnostech, které existují pro tvorbu aplikací pro Mac OS. Jejich výhody a případné nedostatky. Dále bych se chtěl věnovat vyspělému objektově-orientovanému programovacímu jazyku Objective-C, se kterým bych se chtěl seznámit a následně popsat jeho základní části a vlastní funkce tohoto jazyku. Seznamování se syntaxí budou doprovázet ukázky programového kódu. Po seznámením s tímto programovacím jazykem bych se chtěl na závěr práce přesunout do praktické časti. Pokusím se s popisem postupu o vytvoření jednoduché aplikace ve vývojovém prostředí Xcode. Seznámím se tak s jeho prostředím, s používáním některých jeho klíčových součástí součástí. Zopakuji si přitom i obecný návrh aplikace v obejktově-orientovaném jazyce. Od návrhu datového modelu, přes grafické uživatelské rozhraní, až po řízení vnitřní logiky programu. Věřím, že tato práce mě zavede do tajů vývoje aplikací pro paltformu Mac a možná napomůže mému velkému snu, kterým je, jednou se stát vývojářem pro operační systém, který již dlouhou dobu spokojeně používám a nedám na něj dopustit.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem práce je seznámit se s historií společnosti Apple, se systémy Mac OS a srovnání nejběžnějších možností tvorby aplikací pro ně. Zároveň se naučit základům objektově-orientovaného programovacího jazyku Objective-C a nakonec se pokusit vytvořit jednoduchou aplikaci pomocí vývojového prostředí Xcode, od zmíněné společnosti.

2.2 Metodika

Téma jsem si vybral z důvodu jeho zajímavosti a mé dlouhodobé pozitivní zkušenosti se systémem Mac OS X. Informace jsem čerpal z odborných publikací o této problematice a o historii společnosti Apple. Nejdříve jsem postupoval teoreticky, se snahou se seznámit s možnostmi tvorby aplikací pro systémy Mac OS a zároveň pochopit a následně popsat základy programvacího jazyku Objective-C. Následně jsem se věnoval praktické stránce práce; a to tvorbě jednoduché aplikace v prostředí Xcode a programovacím rozhraním Cocoa, které je v dnešní době základem většíny aplikací systémů Apple.

Podle zásad standardnímho postupu při vývoji aplikace v objektově-orientovaném jazyku jsem v ukázkové praktické části nejprve navrhl datový model aplikace, pomocí ER modelu. Ve kterém jsem definoval entity podle zadání a jejich nezbytné atributy, u kterých jsem si jednotlivě nadefinoval jejich datové typy. Dalším krokem byla tvorba relací mezi zmíněnými relacemi a určení typu relace (1:1, 1:N). Výsledkem bylo grafické znázornění tohoto datového modelu s entitami a propojeními mezi nimi se šipkami podle typu relace. Dále jsem přešel k vytvoření grafického rozhraní aplikace, díky velice propracovanému prostředí Xcode a výbornému nástroji Interface Builder to bylo poměrně snadné. Tento výborný nástroj totiž dokáže zpracovávat dvě důležité vrsty programu soubežně. Jsou jimi, jak vrstva Vzhled (grafické uživatelské rozhraní), tak i důležitá vrstva Řízení, která propojuje vrstvu Model (datový model) s vrstvou Vzhled. Poměrně snadno se tak vytvořilo grafické rozhraní a zároveň i vnitřní logika programu; co které tlačítko ovládá a jak. Po úpravě vzhledu a vytvoření aplikace jsem se zabýval doladěním a vylepšením některých neduhů aplikace, které se mi zdály jako zbytečné.

3 Historie Apple Inc.

3.1 Počátek

11. srpna roku 1950 se v San Jose v Kalifornii narodil velice nadaný chlapec jménem Steve Wozniak. Už od malička se vášnivě věnoval elektronice. Sám dokázal vytvářet kalkulačky a již v 6. třídě získal licenci HAM Radio (sestavování amaterských rádií, pozn. autora). Dnes je znám jako muž, který sestavil první osobní počítač. V březnu roku 1975 se konalo první setkání neobyčejné skupiny nadšenců zvané Homebrewský počítačový klub (Homebrew Computer Club, dále jen Homebrew). Byl to kolektiv lidí, které uchvátila technologie a její možnosti. A právě díky tomuto společenskému klubu se jednoho dne setkal s neméně nadaným, o 5 let mladším, Stevem Jobsem.

Steve Jobs, narozen 24 února 1955 v Los Altos, byl vychován svými adoptivními rodiči a již od raného dětství byl neuvěřitelně komunikativní a hlavně vždy dokázal prosadit svůj názor u ostatních. Stejně jako Wozniaka, i Jobse fascinovala počítačová technologie a viděl v ní budoucnost. A právě spojení těchto dvou rozdílně nadaných osob se stejným zájmem o počítače dopomohlo později k revoluci v samotném počítačovém odvětví. [1]

Homebrew Computer Club byl plný nadaných lidí, kteří se snažili zpřístupnit počítač normálním lidem. V té době zabíraly počítače svou rozlohou celé místnosti a jejich cena byla neuvěřitelně vysoká. To měl změnit Altair. Funkční počítač, který se dal z dostupných součástek složit, měl však malou chybu. K jeho používání bylo potřeba dokoupit velké množství dalšího vybavení, aby byl tento počítač se 30 čipy použitelný. Jeden člen klubu svým vlastním výtvorem vše změnil. [1;2]

3.2 Apple I

Steve Wozniak už tou dobou měl zkušenosti s vytvářením počítačů (o jeho prvním počítači Cream Soda již 5 let předtím napsal dokonce místní plátek). Uvědomoval si, že jeho 5 let starý počítač Cream Soda je v podstatě totožný jako drahý Altair, jediný rozdíl byl v tom, že Altair měl CPU v jednom čipu, kdežto CPU v Cream Sodě sídlilo ve více čipech. Steve chtěl poukázat na svůj talent a hlavně chtěl ukázat všem členům Homebrew, že si mohou sestavit vlastní levný počítač a ne se jen stále slepě zajímat o předražený Altair. Díky jeho zálibě se mu podařilo načtrtnout vylepšený Cream Soda počítač. Obsahoval také 30 čipů jako Altair, ale pro jeho použití již nebylo třeba dokupovat žádné další vybavení. Levný počítač, který opravdu umožňuje programovat, přesně to byl jeho návrh, který později vešel ve známost jako Apple I. Na dalším setkání pak rozdal schémata zapojení jeho výtvoru. Tím zapadl do společnosti a uvedl se na scénu.

Za nedlouho se stal “parťákem” se Stevem Jobsem, kterého tento projekt velice zaujal. Napadlo ho, že by mohli stávající návrh ještě vylepšit.

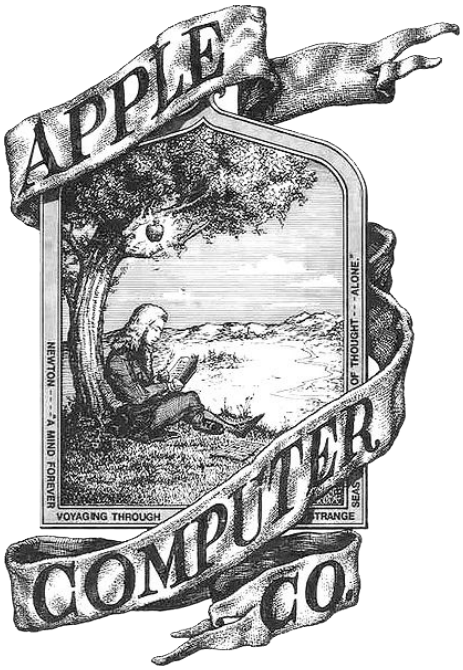
Díky svému komunikativnímu daru dokázal sehnat všechny potřebné čipy od velkých společností. Jeho hlavní vizí bylo, nechat si od počítačové společnosti vyrábět předtištěné desky, které by pak mohli prodávat nadšencům a ti by si podle schémat sami připájeli všechny své čipy sami. Nejen, že by to pro ně bylo finančně přínosné, ale i zákazníci by byli spokojeni. Steve Wozniak si nebyl moc jistý tím, že by se jim peníze z toho mohly vrátit. Steve Jobs na to odvětil: “Dobrá, i kdybychom na tom prodělali peníze, budeme mít společnost. Aspoň jednou v životě budeme mít společnost.”

Tento výrok přesvědčil Steva Wozniaka a ten souhlasil s návrhem. [1;2]

3.3 Apple Computer

Po sehnání potřebných financí, kvůli kterým museli Stevové prodat některé své milované věci, už zbývalo jen vymyslet název. Steve Jobs navrhl název - Apple Computer. Vrátil se totiž právě z návštěvy v Oregonu, z místa zvaného „jablečný sad“. A protože nepřišli na lepší, více technicky znějící název, ujal se Apple Computer jako konečný název jejich nové společnosti založené 1. dubna 1976. Steve Jobs oslovil svého bývalého spolužáka Ronalda Wayna z Atari, který by dokázal navrhnout základní návrh desek pro hromadnou výrobu. Společně mu nabídli 10 procent ve firmě a tím se stal jejich partnerem. Byl to také právě on, kdo vyryl Newtona pod jabloní, který se pak nacházel v počítačovém manuálu (viz obr. 3.1). Později se stalo hlavním logem nakousnuté jablko v barvách duhy (viz. obr. 3.2). V tomto designu vydrželo až do konce 90. let.

Obr. 3.1: Newton Logo



Zdroj: [5]

Obr. 3.2: Duhové logo

Zdroj: [5]

Od založení společnosti už se všechno rozjelo ve velkém. Firma Hewlett-Packard, ve které Steve Wozniak pracoval, slíbila, že si nebude dělat žádné nároky na jeho Apple I a dala mu volnou ruku v jeho užívání. Následovala velká zakázka od prodejce počítačů, smlouva s dodavatelem čipů a desek, to vše přišlo poměrně rychle za sebou. Ronald Wayne se staral o papírování, Steve Jobs sháněl nové objednávky na jejich počítač Apple I a Steve Wozniak pracoval na verzi BASICu pro Apple I (viz. obr. 3.3). [1;3]

Obr. 3.3: Wozniak, Jobs, Apple I

Zdroj: [1]

3.4 Macintosh

Po velkém úspěchu Apple I následoval Apple II, byl to velký krok vpřed. Od začátku podporoval barvy, obsahoval polovinu čipů co Apple I, byl podstatně rychlejší, podporoval grafiku ve vysokém rozlišení, zvuk a schopnost připojit herní ovladače. Prodej stále stoupal, zájem o Apple II byl veliký. Společnost rychle rostla, počet zaměstnanců s ní. Apple III byl však propadák. Byl navržen hlavně marketingovým oddělením. Nebyl zdaleka tak spolehlivý, nepodobal se ani trochu Apple II. Měl vážné hardwarové problémy. Skoro každý kus se vracel na reklamaci. Byl to počítač navržený bez Steva Wozniaka. Trvalo celý rok, než Apple vyrobil takový kus Apple III, který by se neporouchal hned při prvním naběhnutí v obchodě. Tou dobou stále vládl počítačovému světu Apple II, roku 1981 však firma IBM poprvé přišla s odpovědí na Apple II. Díky tomu, že hodně velkých firem již bylo zákazníky IBM, nový osobní počítač IBM PC se tak prodával velice dobře. Když se poprvé objevil, Apple byl trochu arogantní a nechal vytisknout celostránkovou reklamu ve *Wall Street Jurnal* se slovy: „Vítej IBM. Vážně.“ V roce 1983 pak IBM PC předstihl Apple II, do té doby nejprodávanější počítač na světě, tou dobou byl Steve Wozniak na své dlouhodobé dovolené, ke které se rozhodl po letecké havárii v roce 1981. Koncem roku 1983 se rozhodl vrátit zpět do Apple,

ale pouze jako vývojář Apple II, malé divize, která byla založena po zrušení Apple III a neúspěchu uživatelsky přívětivého počítače, ale příliš drahého Apple Lisa.

Pracovali na upravovaných a vylepšených verzích vycházejících z Apple II. Tou dobou už byli ve firmě 2 souběžné divize. Tou druhou byla divize zabývající se novým převratným výrobkem s názvem Macintosh. Jméno dostal podle oblíbené odrůdy jablek McIntosh. Tento počítač byl od začátku navrhnut Jobsem jako rival pro PC. Jeho vývoj byl stále větší prioritou a divize Apple II se začala cítit utlačovaná. Roku 1984 za masové reklamy vešel do prodeje a začal se prodávat neuvěřitelně dobře. [1;2;3]

3.5 Úpadek a Vzestup

Steve Wozniak miloval malé firmy, velké firmy ho desily. To byl po utlačování Apple II divize další z důvodů, proč nakonec roku 1987 definitivně skončil v Apple Computer. Zato Steve jobs opustil Aplle nedobrovolně již roku 1985. John Sculley, který byl ve vedení firmy se nestotožňoval s vizí Jobse a jeho návrhy. Jobs byl tak veřejně mediálně propuštěn. Byl dlouho dobu na dně. Být vyhozen z firmy, kterou sám založil, bylo pro něj to nejhorší, co se mu mohlo stát. Jobs poté založil technologicky vyspělou, ale komerčně nepříliš úspěšnou firmu NeXT Computer. Následně koupil menší animační studio od George Lucase, které pak přejmenoval na Pixar Animation Studios, nikdy však nepřestal milovat Apple, svou první firmu. Mezi tím Apple upadal. Ať se snažil, jak chtěl, pod vedením Sculleyho prodělával a ač se další verze Macintoshe poměrně dobře prodávali v audio a video průmyslu, Apple nedokázal nakonec konkurovat klonům Macintoshe.[1;4]

Apple se proto rozhodl roku 1997 požádat Steva Jobse o pomoc a návrat do firmy. Steve souhlasil, Apple odkoupil jeho firmu NeXT Computer, jeho operační systém pak posloužil jako základ pro operační systém MacOS X. Následovalo nečekané oznámení o uzavření dohody s Microsoftem, které bylo nejdříve odsuzováno, nakonec se však ukázalo jako krok správným směrem. Koncem roku se Steve jobs stal CEO firmy. Logo firmy bylo roku 1998 změněno na monochromatické jablko (viz. obr. 3.4) [3;4]

Obr. 3.4: Monochromatické logo

Zdroj: http://edibleapple.com/wp-content/uploads/2009/04/silver-apple-logo.png

Dalším krokem Jobse bylo uvedení Apple Store, online obchodu, který následně pomohl společnosti v rostoucím prodeji. Velkým zlomem byl pak počítač typu vše v jednom s názvem iMac. Tento počítač integroval přímo do CRT monitoru celé tělo počítače. Měl velký úspěch a dopomohl spolu s dalšími uvedenými produkty k návratu Apple na vrchol pomyslného žebříčku. [1;3;4]

3.6 Apple a 21. století

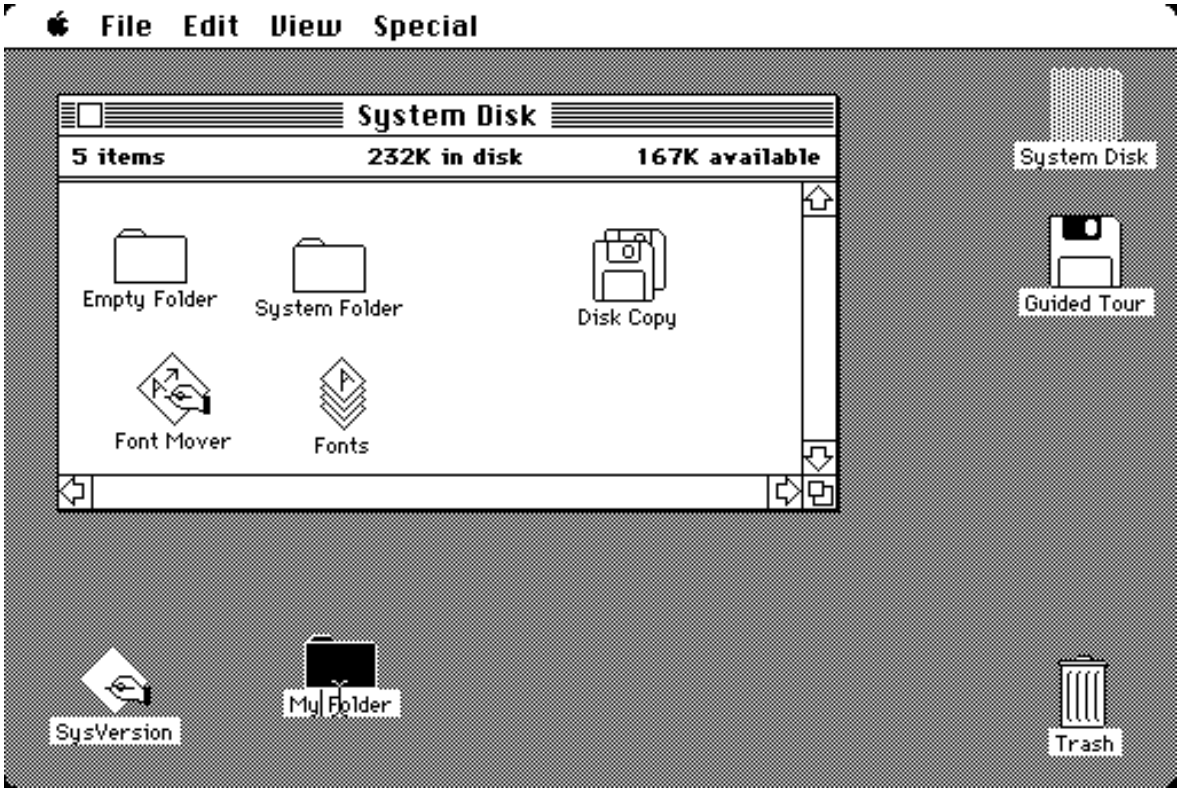
V roce 2001 Apple představil převratný mediální přehrávač iTunes a revoluční přenosný přehrávač iPod. Vznikly první kamenné obchody Apple. V roce 2003 byl otevřen první zahraniční obchod v Japonsku. Stevu Jobsovi byla roku 2005 sdělena diagnóza rakoviny slinivky břišní. Rok 2005 byl pro Apple přelomový v tom, že uzavřela kapitolu počítačů s procesory PowerPC a přešla definitivně na procesory Intel. V roce 2007 byl zkrácen název firmy Apple Computer Inc. na Apple Inc. Steve Jobs v jeho Keynote prezentaci zdůvodnil, že již Apple není pouze počítačovou firmou, a na konci pak představil další revoluční výrobek; tím byl Apple iPhone (viz. obr. 3.5) a jeho operační systém iPhone OS (dnes znám jako iOS). Nedílnou součástí iPhone se stal App Store. Online obchod s aplikacemi, který v dnešní době nabízí přes 300 000 aplikací. Zatím posledním zásadním výrobkem je iPad, tablet od firmy Apple, který rozdělil svět na 2 části. První tento tablet miluje, druhá jím pohrdá. V dnešní době se prodává již 4. Verze mobilního telefonu iPhone, v prodeji je také druhá verze přenosného tabletu iPad, ultra tenké unibody notebooky Macbook Air, nebo například velmi výkonné stolní hliníkové ploché počítače vše v jednom iMac. V dnešní době patří firma Apple k největším prodejcům hudby a filmů. Zároveň je uznávána jako silně inovátorská společnost, co udává směr ve výpočetní technice a designu. [3;4]

4 Mac OS

4.1 System

Když v roce 1984 přišel na trh Macintosh, byl opatřen jedním z prvních operačních systémů s grafickým uživatelským prostředím (tzv. GUI) (viz. obr. 4.1). Byl tedy lehko rozpoznatelný od ostatních konkurenčních systémů. Jeho hlavní částí byl Finder, aplikace, která se používala pro snadnou správu souborů. Tento systém byl pojmenovaný snadně - System. S postupem času a příchodem novějších verzí Macintoshů se objevovaly i nové verze Systemu.[3;6]

Obr. 4.1: System



Zdroj: http://en.wikipedia.org/wiki/File:Apple_Macintosh_Desktop.png

Od roku 1985 až do roku 1987 postupně přinesly verze 2, 3 a 4 vylepšení v podobě souborového systému HFS (Hiearchical File System), vetší podporu diskových uložišť a disketových mechanik a podporu laserové tiskárny LaserWriter. Verze 5 již přišla s tzv. MultiFinder, s rozšířením, které dovolovalo mít spuštěno více programů současně. V roce 1988 se objevil System 6, narozdíl od předchůdců měli jednotlivé hlavní softwarové části systému stejné číslování jako systém, aby nedocházelo k nesrovnalostem jako v minulosti. Hlavní výhodou byla stabilita systému a podpora nových periférií a standartů. V roce 1991 pak přišel systém verze 7. Jednalo se o velký upgrade dosavadního systému. Celkové doladění a oprava uživatelského rozhraní, nové aplikace, zvýšení

stability a podpora virtuální paměti. Přímo integrovaný byl i kooperativní multitasking. Jako první také System 7 začal používat zvláštní skrytou složku pro Trash (koš na smazané soubory), do té doby se vždy vyprázdnil s vypnutím počítače nebo MultiFinderu. Teď už smazané soubory vydržely vypnutí a zapnutí počítače a ke smazání bylo zapotřebí vybrat položku “Emty Trash” z nabídky. Pozdější verze Systemu 7 pak přinesly podporu PowerPC Maců a další menší vylepšení. [2;3;6]

4.2 Mac OS

Mac OS 8, přesně tak se jmenoval první systém Applu čistě s názvem Mac OS. Bylo to nedlouho poté, co se do firmy v roce 1997 vrátil Steve Jobs. Mac OS 8 měl vylepšený multitasking, GUI bylo přepracováno do nového šedého kabátu pojmenovaného “Platinum” a přineslo i nový panel nástrojů (viz obr. 4.2). Jeho následné verze přinesly podporu nové verze souborového systému s názvem HFS+, podporu USB a Firewire pro některé stroje a verze 8.5 byla první verze Mac OS, která běžela výhradně na strojích osazených PowerPC procesory.

Obr. 4.2: Mac OS 8

Zdroj: http://en.wikipedia.org/wiki/File:MacOS81_screenshot.png

V roce 1999 byl vypuštěn nástupce dosavadního systému s názvem Mac OS 9. Jednalo se obecně spíše o evoluci a celkově stabilnější verzi předchozího verze systému. Přinesl podporu pro AirPort bezdrátové připojení. Obsahoval první implementaci rozhraní více uživatelů. Lepší vyhledávání, vylepšenou správu paměti. Obsahoval jako první centralizovaný software update s názvem Apple Software Update. [2;6]

4.3 Mac OS X

Systém, který byl narozdíl od svých předchůdců, založený na Darwin-Unixovém jádře. Byl postavený na technologii firmy NeXT vyvíjené od druhé poloviny 80. let až do roku 1997, kdy Apple firmu koupil. První verze Mac OS 10.0 „Cheetah“ (viz. obr. 4.3)přišla na na trh v roce 2001 a jednalo se o úplně nový kód naprosto oddělený od Mac OS 9 a ostatních předchůdců. Nový vzhled, nový aplikační 2D dok na spodní časti obrazovky, naprosto nová správa paměti. Obsahoval jak vývojové prostředí Cocoa (vývojové prostředí firmy NeXT pro systém OS X), tak i prostředí Carbon (vývojové prostředí Applu, ve kterém se vytvářely aplikace pro předchozí systémy Applu). Byl uveden na trh narychlo, proto jeho stabilita nebyla ze začátku moc valná a dlouhou dobu trpěl i dalšími nešvary.

Obr. 4.3: Mac OS X 10.0 Cheetah

Zdroj: http://en.wikipedia.org/wiki/File:MacOSX10-0screenshot.png

Jeho nástupce 10.1 “Puma” přišel s navýšením výkonu systému a byl nabízen jako bezplatná aktualizace uživatelům verze Cheetah. Avšak i přes jeho zrychlení, pro hodně uživatelů Apple to stále nebyl dostačující důvod pro přechod na Mac OS X. Až verze 10.2 “Jaguar” v roce 2002 byl konečně masově přijat Apple uživateli. Přinesl další optimalizaci výkonu, byl konečně stabilní, umožnoval kompatibilitu s ostatními Unix systémy. Byla to také první verze, u které bylo použito kódové označení v marketingu a reklamě. V roce 2003 byl uveden Mac OS X Panther (v10.3),

který přišel se skoro 150 novinkami. Jednou z nich bylo Safari, které vystřídalo doposud primární prohlížeč Internet Explorer od firmy Microsoft, který byl součástí dohody z roku 1997. Dalšími novinkami bylo například šifrování dat na pevném disku, jmenovitě pak domovské složky uživatele a Xcode pro vývoj aplikací pod Mac OS X. Roku 2005 se dostal na trh Mac OS X Tiger (v10.4). Nejúspěšnější systém Applu od jeho počátku, jak ho sama společnost nazvala. Bežel celých 30 měsíců, než ho vystřídal další následník. V prvních 6 týdnech se ho prodalo přes 2 milióny kusů, čož odpovídalo v té době cca 16,7 procentům všech uživatelů Mac OS X. Přinesl nové rychlé fultextové vyhledávání v systému Spotlight, VoiceOver pro postižené, nebo například Dashboard s miniaplikacemi. Jeho následovníkem se stal roku 2007 Mac OS X Leopard (v10.5). Nově předělaný 3D dok, částečně průhledný menu bar, Finder s náhledovým zobrazením Cower Flow, automatickou zálohu pomocí aplikace Time Machine, možnost psaní 64-bit aplikací, rodičovskou ochranu Parental Controls, možnost použití více virtuálních ploch díky Spaces, nebo například souhrný soubor pomůcek pro umožnění práce s počítačem postiženým osobám Universal Access. Leopard je poslední verzí systému Mac OS X, která běží na PowerPC strojích. Aktuální verze Mac OS X Snow Leopard (v10.6) byla uvedena jako celková optimalizace Leopardu s menším počtem novinek, ale zato s perfektní stabilitou a rychlosti uzivatelského rozhraní. Je to hybridní systém podporující jak 32-bit aplikace, tak i 64-bitové. Stejně tak může jadro naběhnout v obou módech. Rozšiřuje mulitouch ovládání a všechny aplikace jsou již psané v Cocoa vývojářském prostředí (viz. obr. 4.4).

Obr. 4.4: Mac OS X 10.6 Snow Leopard

Zdroj: http://farm4.static.flickr.com/3141/2654807040_8365c15d5a_o.jpg

Letos v létě se chystá nová a zásadní verze Mac OS a to verze Lion. Bude spojovat výhody sytému Mac OS X a výhody systému iOS, který z Mac OS X vyšel. Také slogan „Back To Mac“ na to upozorňuje. Podle prvních ukázek a vývojářských verzí půjde o velký krok dopředu, hlavně co se multitouch ovládání a přívětivosti k uživateli týče.[2;3;6]

4.4 iOS

Původně iPhone OS je operační systém Applu pro iPhone, iPod Touch a iPad. Systém byl inspirován Mac OS X, ale od základu psaný iPhone na míru. V první verzi v roce 2007 byl hodně omezený, postupem času se ale vypracoval na verzi 4.3, která je již v dnešní době plnohodnotným operačním systémem pro mobilní zařízení. Jeho rozhraní a smysl ovládání inspirovalo Apple při vývoji nadcházející verze Mac OS X Lion. V dnešní době existuje obrovské množství aplikací pro tento systém, internetový obchod App Store je velice úspěšný. Vývojové prostředí Xcode v posledních verzích v sobě implementuje iOS SDK, tzv. Software Development Kit pro vytváření aplikací pro iOS. Toto odvětví je v dnešní době velice silné a mnoho vývojářů se plně věnuje vývoji aplikací pro iOS, protože díky promyšlenému App Store systému je pro vývojáře poměrně výhodné vytvářet takové aplikace.[3;5]

5 Možnosti tvorby aplikací pro Mac OS

5.1 Carbon

Procedurální rozhraní firmy Apple pro programování aplikací pod Mac OS. Používal programovací jazyk C pro přístup k Macintosh systémovým službám. Dovoluje velice dobrou kompatibilitu běhu programů na dnes již zastaralých systémech Mac OS 8 a 9. Carabon obsahuje obsáhlou sadu funkcí pro správu souborů, paměti, data, uživatelské rozhraní a další systémové služby. Pokrývá více rámců (frameworks). Carbon framework pro práci s UI, CoreServices framework s rozhraními pro práci se strukturou dat na nižších vrstvách a ApplicationServices framework zapadající svou funkcí mezi dva přechozí. Carbon se masově používal až do nedávna. Jeho limitace se projevila až v roce 2007, kdy Mac OS X Leopard přišel s prvím přechodem na 64-bit aplikace. Apple nepodporoval kompaktibilita grafického rozhraní Macintosh a programovacího jazyku C v 64-bitovém prostředí. Vyžadoval místo toho jazyk Objective-C s programovacím rozhraním Cocoa. Mnoho aplikací ještě donedávna běželo na Carbonu. Potřeba přepsat velkou část kódu zvětšovala prodlevu ve vydání nových verzí velkých programů již pod záštitou prostředí Cocoa. Až Snow Leopard přišel s celkovým přechodem na prostředí Cocoa a jazyk Objective-C. [7;8]

5.2 Cocoa

Nativní objektově-orientované rozhraní pro programování aplikací pro Mac OS X firmy Apple. Vzniklo jako vývojové prostředí operačního systému NeXTStep v polovině osmdesátých let minulého století – tedy před více než 20 lety. To je velkou výhodou prostředí Cocoa: díky tomu totiž jsou jeho dětské chyby již dávno vychytány. Operační systém NeXTStep se nikdy masově nerozšířil – příčinou zřejmě byla nepříliš dobrá marketingová politika firmy NeXT. Dobyl si však pevné místo na trhu jako daleko nejlepší prostředí pro tvorbu klíčových aplikací. Pravě tady vznikl World Wide Web. Když Apple po koupi firmy NeXT v roce 1998 představil Mac OS X, jeho rozhraní pro programování aplikací (API) bylo znovupokřtěno na Cocoa (kakao) a spolu s Carbonem se stalo vývojářskou součástí nového systému. Cocoa je stále nejvyspělejší programovací rozhraní na trhu. Dokáže pracovat i s jinými programovacími jazyky, od Javy přes Ruby až po AppleScript, ale Objective-C je jazykem základním, nejflexibilnějším, a také daleko nejefektivnějším. Cocoa je rozdělena mezi dva programovací rámce. The AppKit framework poskytuje objekty a služby vyšší vrstvy pro psaní aplikací, obsahuje také Aqua UL elementy. The Foundation framework poskytuje objekty a služby užitečné pro všechny programy, jakými je třeba kolekce datových typů, Unicode string podpora a další. Rozdělení na tyto dva rámce tedy pak dovoluje použít Foundation části, aniž by muselo přinést celé grafické uživatelské rozhraní. Například příkazový řádek napsaný v Objective-C může jednoduše používat Foundation. [8;9;10;14]

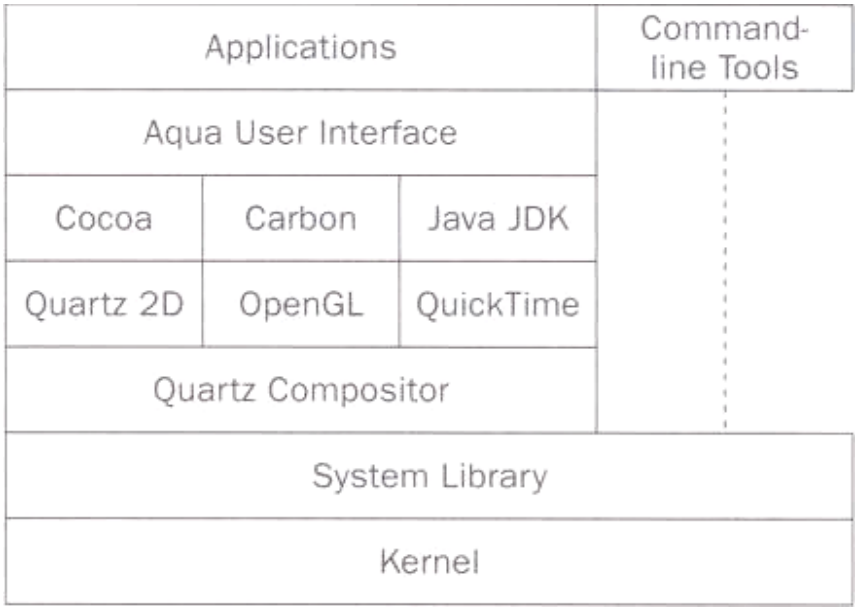
5.3 Java JDK

Mac OS X má v sobě zabudovanou podporu pro Java aplikace. Java je objektově-orientovaný programovací jazyk vytvořený firmou Sun Microsystems pro vyvíjení celistvých programů rozšiřitelných na širokou škálu počítačů a zařízení, v tom je také její největší síla. V tom Java nemá soupeře. Na druhou stranu pro psaní specifické aplikace pro Mac OS X, aplikační rámce Javy mají vážné nevýhody. Java se, kvůli své rozšiřitelnosti na různé stroje, musí zaměřovat na běžně používané techologie a koncepty. Je tudíž složité dosáhnout přístupu k unikatním vlastnostem Mac OS X, jako například výkon CoreGraphics, protože tato vlastnost není dostupná na všech Java systémech a proto v jejích rámcích chybí. To je důvod, proč se Java pro vývoj specifických aplikací pro Mac OS X moc nepoužívá. [8]

5.4 Shrnutí

Popsali jsme si stručně asi nejhlavnější řešení pro tvorbu aplikací pro Mac OS. Carbon se v dnešní době 64-bitových aplikací přestává používat, možná spíše pro tvorbu programů pro předchozí verze sytému. Java JDK nedokáže využívat některé jedinečné vlastnosti Mac OS X, díky kterým právě převyšuje konkurenci, proto se ve větší míře pro vývoj aplikací výhradně pro Mac OS X nepoužívá. Nejlepším řešením je tak bezesporu rozhraní Cocoa s programovacím jazykem Ruby, nebo nejlépe s jeho výchozím jazykem Objective-C, s jehož základy se v následující kapitole seznámíme blíže. Na obrázku jsou znázorněny jednotlivé vrstvy od jádra Kernel, přes systémové služby a aplikační rámce, až po grafické rozhraní (viz. obr. 5.1).

Obr. 5.1: Vrstvy programu

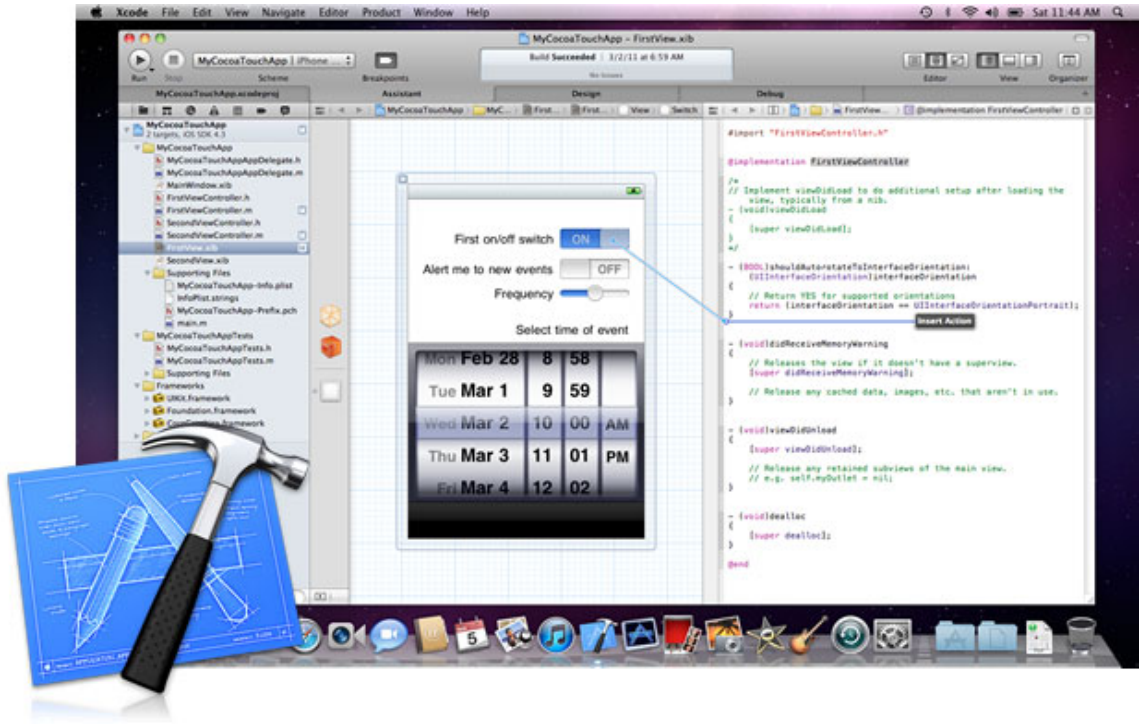


Zdroj: [8]

5.5 Xcode IDE

Integrované vývojové prostředí od společnosti Apple pro vývoj software pro Mac OS X a iOS. Xcode sada také zahrnuje většinu vývojářské dokumentace Apple, a Interface Builder, aplikaci pro konstrukci grafického uživatelského rozhraní. Podporuje moho rozdílních jazykových kódů. Například C, C++, Objective-C, Java, AppleScript, Ruby, Python, a další. Umožňuje vývojářům bezproblematicou práci v uživatelsky přívětivém prostředí. Jeho silnou součástí je Interface Builder. Jeho poslední verze se jmenuje Xcode 4, je to první verze Xcode, která je zpoplatněna. Používá se jak pro tvorbu aplikací pro Mac OS X, tak i pro tvorbu aplikací pro iOS. Vzhled programu je vidět níže (viz. obr. 5.2).[11]

Obr. 5.2: Xcode 4



Zdroj: http://en.wikipedia.org/wiki/File:Xcode4.png

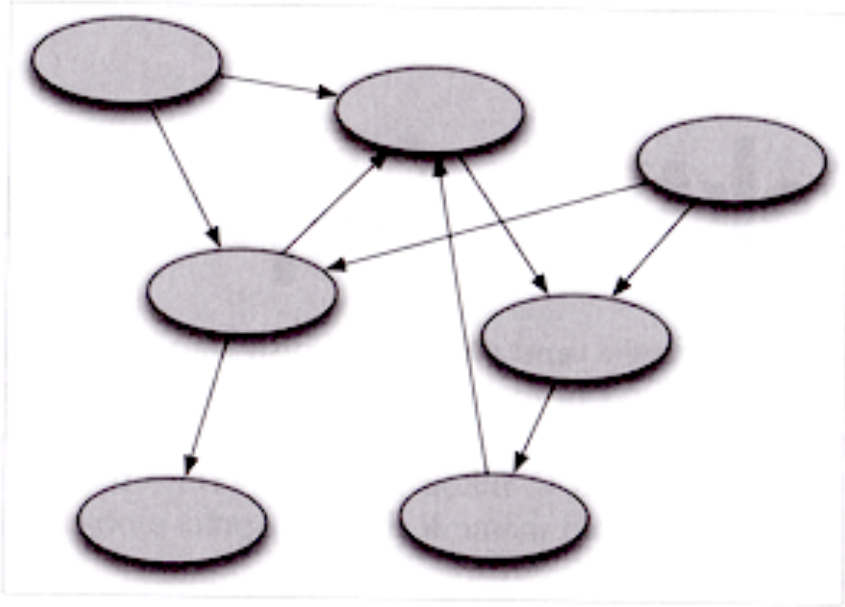
6 Objective-C

Objektová nadstavba progragramovacího jazyku C. Vyvinuli ho Brad Cox a Tom Love roku 1986 jako programovací jazyk pro operační systém NeXTStep firmy NeXT. Jedná se o částečné spojení jazyku Smalltalk a jazyku C.[10]

6.1 Objekty a Zprávy

Objektové programování je takový přístup, který se dívá na program jako na volnou síť vzájemně komunikujících objektů (v podstatě tak mapuje reálný svět, v němž je to podobné) (viz. obr. 6.1) [10]

Obr. 6.1: Objektová síť



Zdroj: [10]

Objekty [10;12;13]

Prvky modelované reality (jak data, tak související funkčnost) jsou v programu seskupeny do entit (objektů). Tyto objekty si pamatují svůj stav a navenek poskytují *operace* (přístupné jako metody pro volání). Jedním ze základních pravidel objektových systémů je zapouzdření (encapsulation):

- *Zapouzdření* – zaručuje, že nelze přímo přistupovat k „vnitřnostem“ jednotlivých objektů. Každý objekt navenek zpřístupňuje *rozhraní*, pomocí kterého (a nijak jinak) se s objektem pracuje. Z hlediska kódu, který s objektem pracuje, musí objekt být "černá skříňka", cosi, co leží kdesi v paměti na určité adrese, a do čeho není vidět. Máme k dispozici jen určitý standardní protokol, jehož prostřednictvím můžeme po objektu (po kterémkoli objektu) požadovat služby; objekt sám se musí "rozhodnout" jakým způsobem požadavky vyplní.

Jazyk C je jazykem poměrně nízké úrovně, a pro "cokoli v paměti" vždy používal adresy – ukazatele; Objective-C tuto tradici dodržuje, i v něm jsou objekty representovány ukazateli. Protože do "vnitřností" objektů nám kvůli zapouzdření nic není, použijeme pro ně nejobecnější void* (viz. ukázka kódu):

```
void *obj1,*obj2;
...
if (obj1==NULL) printf("Objekt 1 neexistuje");
if (obj1==obj2)
printf("proměnné obj1 i obj2 representují tentýž objekt")
```

Ve normálních hlavičkových souborech Objective-C definováno pro "typ objekt" nové jméno ‘id’; podobně namísto NULL je k dispozici konstanta nil s naprosto stejným významem. Překladači je to naprosto jedno, je to jen pro nás lidi, pro lepší srozumitelnost a čitelnost kódu. Lze tedy předchozí ukázku napsat jako druhým způsobem, aniž bysme se museli obávat, že překladač to vyhodnotí jinak (viz. ukázka kódu):

```
id obj1,obj2;
...
if (obj1==nil) printf("Objekt 1 neexistuje");
 if (obj1==obj2)
printf("proměnné obj1 i obj2 representují tentýž objekt")
```

Nad objekty je definována jediná operace: objektu můžeme zaslat tzv. zprávu. [10]

Zprávy [10]

Zpráva je jakýsi "balíček", obsahující jméno zprávy a její případné parametry. Základní vlastností každého objektu je schopnost přijímat a zpracovávat zprávy: objekt tento "balíček" rozbalí a podle jména zprávy (a jejích případných parametrů) se rozhodne, co se zprávou provede. Nejběžněji provede nějakou akci odpovídající zprávě, může však stejně dobře zprávu třeba uložit pro pozdější zpracování, nebo předat jinému objektu, nebo prostě odmítnout (to vede pak k běhové chybě). Ve zdrojovém kódu se pro zaslání zprávy používá hranatých závorek a sestavení [<příjemce> <zpráva>]; příjemcem je libovolný objekt (tj. libovolný výraz, jehož výsledek bude typu id). Syntaxe pro zprávu je zhruba převzatá ze Smalltalku: jménem zprávy může být jakýkoli identifikátor, obsahující libovolné množství dvojteček, kde dvojtečky representují parametry.

Dvojtečky mohou ve zprávě stát kdekoli, a parametry se píší přímo za ně (takže jméno zprávy je "roztrhané" a parametry jsou uvnitř něj), to napomáhá tomu, že i velmi složité zprávy jsou snadno čitelné (viz. ukázka kódu):

```
id obj;
[obj zpravaBezParametru];
[obj zpravaSJednimParametrem:1];
[obj zpravaSParamteremX:1 aParametremY:2];
// reálná zpráva by vypadala například následovně:
[obj drawCircleWithCentreX:10 Y:10 radius:12 title:"Terč"];
// nebo může zpráva vracet hodnotu:
int suma=[obj intValue]+23;
// vrací-li hodnotu typu id (objekt), můžeme mu hned zaslat další zprávu:
[[obj dejMiObjekt] zpravaProVracenyObjekt];
```

Při zasílání zpráv, narozdíl od jazyku C a jemu podobným, to, jaká operace bude na základě přijetí zprávy provedena, rozhodne až přijímající objekt ve chvíli, kdy zprávu dostal. Protože se vazba mezi požadavkem toho, kdo zprávu odesílá, a reakcí toho, kdo ji přijímá, naváže co nejpozději to je možné (až v okamžiku odesílání zprávy za běhu programu), nazývá se tento systém pozdní vazba (late binding).

Výhodou pozdní vazby je její flexibilita, například triviální funkce pro výpočet průměru (viz. ukázka kódu):

```
double average(id *o) { // pole objektů, končí hodnotou nil
  double cnt=0,sum=0;
  while (*o) {
    sum+=[o doubleValue];
    cnt++; o++;
  }
  return sum/cnt;
}
```

Kdybysme něco takového napsali v prostém C, pro každý typ hodnot by bylo třeba nové implementace. Funkce, která počítá průměr 'celých čísel' by neuměla spočíst průměr 'čísel

desetinných', o ostatních variantách ani nemluvě. V dynamickém objektovém prostředí je ale funkce naprosto flexibilní. Je naprosto jedno, jaké objekty dostane, jeden může representovat třeba celé číslo, druhý číslo v pohyblivé řádové čárce, a třetí například matici (doubleValue spočte a vrátí její determinant). Čtvrtým objektem může být zase něco úplně jiného — třeba textové pole v uživatelském rozhraní, jež vrací svůj obsah interpretovaný jako číslo... Funkce bude pořád korektně pracovat! Kdybychom funkci předali objekt, který zprávě doubleValue nerozumí? Program by byl ukončen, a do konsole by se vypsala příčina chyby ("objekt třídy X dostal zprávu doubleValue, již nedokáže zpracovat").

Této flexibilnosti, univerzálnosti a jednoduché přenositelnosti kódu z jednoho projektu do druhého nelze dosáhnout v programovacích jazycích, jež zprávy nepoužívají. Fakt, že libovolnému objektu můžeme poslat libovolnou zprávu a on si ji korektně zpracuje po svém se nazývá polymorfismus, a je považován za zcela základní rys objektových systémů.

- *Polymorfismus* – odkazovaný objekt se chová podle toho, jaké třídy je instancí. Pokud několik objektů poskytuje stejné rozhraní, pracuje se s nimi stejným způsobem, ale jejich konkrétní chování se liší podle implementace. Představme si například vypínač. Princip každý zná. Mohou vypadat stejně, každý ale ovládá jiné zařízení, mohou se lišit vnitřním uspořádáním a způsobem zapojení. Na venek jsou totožné, ale uvnitř se značně liší. Nebo například, když pošleme zprávu “vypočti obsah” objektu čtverec a obdélník, výsledek bude hodnota obsahu, ale každý objekt k výpočtu použije jiný vzorec a popřípadě postup. [10;13]

6.2 Třídy a dědičnost

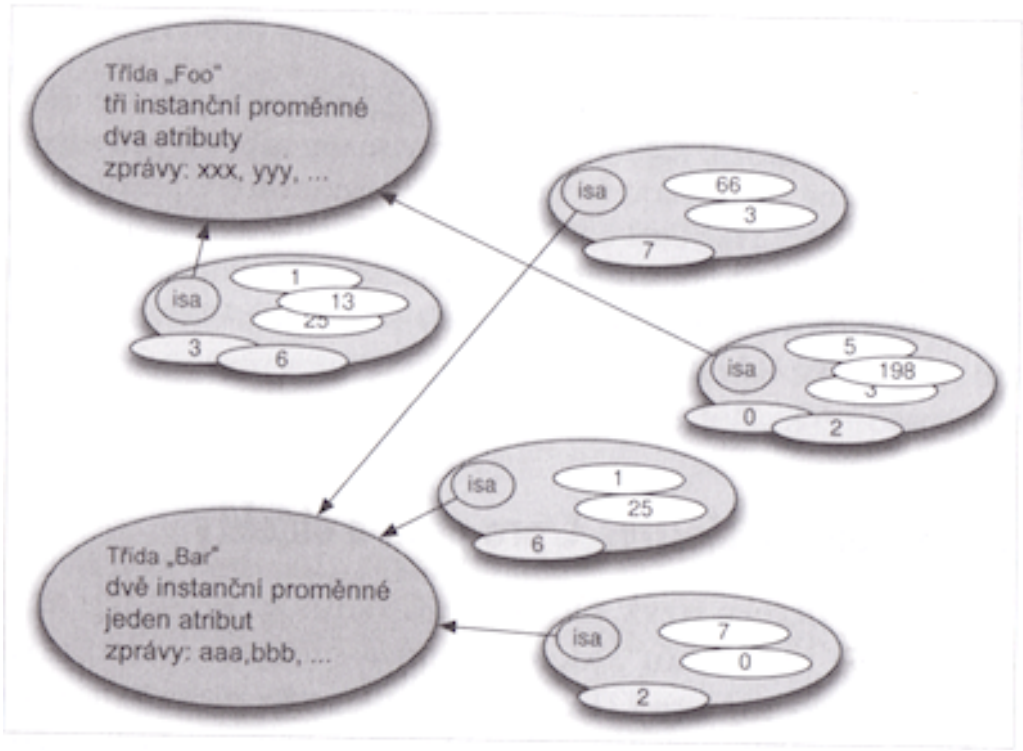
Pro pohodlné programování jsou potřeba ještě tzv. třídy (classes), representující objekty stejného nebo podobného druhu, a dědičnost (inheritance), sloužící pro pohodlnou tvorbu nových tříd.

Třídy [10;12:13]

V praxi se obvykle setkáváme s množstvím objektů stejného druhu. Například v databázovém systému knihovny je množství "oddělení" a ještě více "knih". Každý objekt "kniha" se přitom velice podobá všem ostatním objektům "kniha" v tom smyslu, že vždy stejným způsobem reaguje na stejné zprávy — pouze vrací jiné konkrétní hodnoty. Když jakémukoliv objektu "kniha" pošleme zprávu autor, vždy dostaneme informaci, kdo knihu napsal, nikdy nedostaneme například informaci o počtu stránek. Kdyby měl proto programátor systému určovat způsob reakce třeba na zprávu autor pro každý objekt "kniha" zvlášť, bylo by to velice nepraktické a neefektivní. Místo

toho programátor sestaví třídu "kniha", a v jejím rámci určí obecnou reakci na kteroukoli zprávu, již objekty "kniha" mají zpracovávat. Každý konkrétní objekt pak ví, které třídě patří a dostane-li objekt nějakou zprávu, vyhledá si mechanismus zpracování zprávy ve své třídě. Všechny objekty téhož typu jen obsahují odkaz na tuto třídu, zhruba tak, jak ukazuje obr. 6.2.

Obr. 6.2: Třídy a instance



Zdroj: [10]

Nesmíme zapomenout, že třídy jsou také objekty. Pro komunikaci s třídami můžeme použít naprosto standardní mechanismus zpráv. Pro lepší čitelnost se ale pro třídy používá místo typu id typ Class a namísto hodnoty nil hodnota Nil. Je to však pouze pro nás lidi, aby pro nás byly zdrojové kódy lépe čitelné. Překladači je opět naprosto jedno, můžeme klidně použít void a Null. Naše "normální objekty" nejsou ničím jiným, než instancemi tříd. Jinými slovy, v Objective-C pracujeme s objekty, jimiž mohou být buďto třídy, nebo instance. Třídy jsou zcela standardními objekty až na jednu výjimku: samy již nemají žádnou "třídu tříd", metatřídu.

Dědičnost [10;]

Objekty jsou organizovány stromovým způsobem, kdy objekty nějakého druhu mohou *dědit* z jiného druhu objektů, čímž přebírají jejich schopnosti, ke kterým pouze přidávají svoje vlastní rozšíření. Každá třída může dědit od jiné třídy (v některých programovacích jazycích i z několika jiných tříd). Máme-li například třídu Člověk (jméno, příjmení, atd.), vytvoříme třídu Opravář jako

jejího potomka. Třída Opravář tímto dědí všechny atributy i metody třídy Člověk (není třeba je v kódu znovu psát a dají se upravovat na jediném místě), navíc bude mít metodu opravit.

6.3 Rozhraní, implementace a metody

Popis třídy má dvě jasně oddělené části: první je rozhraní, jež obsahuje informace o tom že třída vůbec existuje, jaké má jméno, koho je dědicem, a jak se s jejími instancemi i s ní samotnou pracuje, a tou druhou je implementace, ta určuje jak objekty budou zpracovávat zprávy. Ve zdrojových kódech Objective-C pro rozhraní a implementaci tříd slouží direktivy @interface, @implementation a @end. [10]

Rozhraní [10]

Nejjednodušší rozhraní prostě jen určí jméno nově vytvářené třídy. Při využívání dědičnosti, zapíšeme za jméno nové třídy dvojtečku, a za ni jméno již existující třídy, od které chceme novou děděním odvodit (nazveme jí nadtřída) (viz. kód):

```
@interface MyClass:NSObject @end
```

Pokud překladač zpracuje tento řádek, pak ví o existence třídy MyClass a je schopen jí i jejím instancím zasílat libovolné zprávy. (Samozřejmě, aby třída doopravdy existovala, musíme ji – třeba v úplně jiném modulu – skutečně vytvořit pomocí direktivy @implementation).

Ve většině případů je třeba, aby každá instance třídy obsahovala nějaké vlastní proměnné (často se jim říká anglicky properties), jež nějak definují její obsah: objekt "kniha" by asi měl proměnné "autor", "název", "počet stran", "vydavatel" a podobně. Všechna objektová prostředí proto umožňují v rámci třídy takové proměnné definovat. Je celkem zřejmé, že se obsah těchto proměnných stane součástí toho "něčeho v paměti", co representuje objekt. Pouze v tomto smyslu se objekt v Objective-C trochu podobá klasické struktuře v C jazyku. Ve zdrojovém textu můžeme takové proměnné definovat ve složených závorkách hned za jménem třídy a nadtřídy, syntaxe přesně odpovídá "vnitřku" standardní céčkové deklarace struct (viz. kód):

```
@interface MyClass2:NSObject {
  // každý objekt třídy MyClass2 bude mít vlastní...
  int i,j; // ... 2 proměnné typu int...
  double d; // ... 1 typu double...
  id o1,o2,o3; // ...a 3 odkazy na objekty (id je ukazatel).
}
@end
```

Zpráva intValue vrací číslo typu int, zpráva doubleValue vrací číslo typu double; tři argumenty zprávy drawCircleWithCentreX:Y:radius:title: byly v jednom předchozím příkladě typu int, a čtvrtý char*, jak to má překladač poznat? Snadno: poslední standardní součástí rozhraní je totiž deklarace zpráv a jejich typů. Syntaxe je jednoduchá — před každou zprávu se napíše znak '-', argumenty se označí libovolnými identifikátory, a před ně i před celou zprávu napíšeme patřičné typy v závorkách (viz. kód):

```
@interface MyClass3:NSObject { ... }
-(int)intValue;
-(double)doubleValue;
-(void)drawCircleWithCentreX:(int)x Y:(int)y radius:(int)r
title:(char*)tt;
@end
```

Jedná jen o informaci pro překladač a o nic jiného! Za běhu pak díky pozdní vazbě může libovolný objekt dostat libovolnou zprávu, zcela bez ohledu na to, jestli je zapsaná v jeho rozhraní nebo ne. Stejně snadno je také možné používat i zprávy, jež nejsou zapsané v žádném rozhraní: překladač pak považuje jejich návratové hodnoty i jejich případné argumenty za objekty (tedy typy id) – což znamená, že to stejně dobře může být cokoli, co se dá na id bez ztráty převést (typicky int či obecný ukazatel na cokoli).

Objective-C je v zásadě beztypový jazyk: všechny objekty jsou téhož typu (tedy id/Class); to, že mohou být instancemi různých tříd či třídami samotnými, se projeví až za běhu, a to jen a jenom tím, kterým zprávám rozumějí a kterým ne.

Implementace [10]

Implementace není nic jiného, než naprogramování několika tzv. metod. Metoda je v zásadě zcela standardní funkce jazyka C. Namísto hlavičky funkce však použijeme hlavičku, jež přesně odpovídá deklaraci zprávy v rozhraní (není zakončena středníkem). Překladač pak udělá dvě věci: (a) přeloží kód metody, (b) umístí do třídy informaci, že dostane-li kterákoli její instance zprávu, jejíž identifikátor odpovídá hlavičce metody, bude vyvolána právě tato metoda.

Na rozdíl od deklarací v rozhraní tedy metody v implementaci skutečně popisují chování objektu: dostane-li instance zprávu, jíž neodpovídá žádná z jejích metod, odmítne ji, a dojde k běhové chybě (viz. kód):

```
@implementation MyClass3
-(int)intValue {
  return 1;
}
-(double)doubleValue {
  return 1.0;
}
-(char)charValue {
  return 'a';
}
@end
```

To, že v implementaci je metod více, je naprosto běžné: odpovídající zprávy z toho či onoho důvodu nejsou součástí rozhraní, ale instance třídy MyClass3 je přesto dokáží zpracovat.

Metody tříd [10]

Třída sama je objektem, a sama tedy dokáže přijímat a zpracovávat zprávy. Proto můžeme v rozhraní kromě deklarace zpráv, určených pro instance této třídy, deklarovat i zprávy určené pro třídu samotnou. Podobně v implementaci můžeme definovat "třídní" metody – tedy takové, jež budou vyvolány v případě, že třída sama dostane zprávu, odpovídající hlavičce metody. V obou případech je deklarace i definice takřka stejná jako minule; jen znak '-' na začátku je nahrazen znakem '+' (viz. ukázka kódu):

```
@interface MyClass4:NSObject
+(char*)myName; // pro samotnou třídu
-(char*)myName; // pro její instance
@end
@implementation MyClass4
+(char*)myName {
  return "Třída MyClass4";
}
-(char*)myName {
  return "Instance MyClass4";
}
@end
```

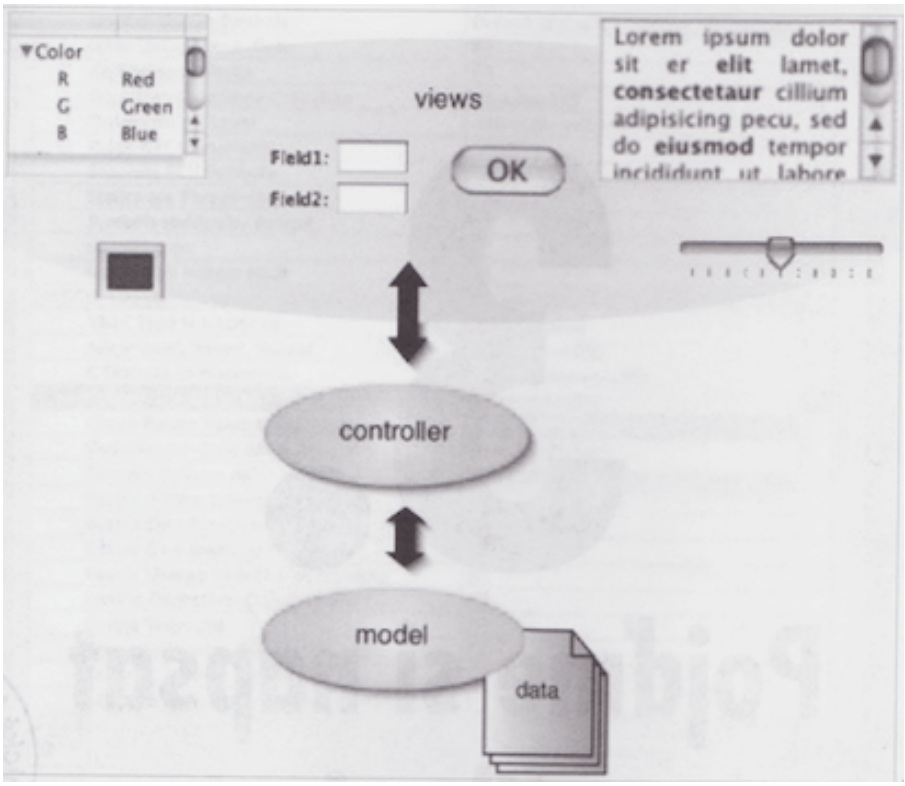
Poslední informace, která chybí k dokončení seznámení se základy programování v Objective-C, je jak se dostaneme k "objektu třída" z programu. Pokud se jméno třídy použije v hranatých závorkách jako příjemce zprávy, representuje právě požadovaný "objekt třída". A to by bylo k již opravdu vše k seznámení s Objective-C.

7 Aplikace v Cocoa

Každou aplikaci lze z pohledu objektového designu rozdělit na tři do značné míry nezávislé vrstvy, které spolu však spolupracují (viz. obr. 8.1).

- Model – reprezentace dat, jež aplikace zpracovává
- Vzhled – uživatelské rozhraní aplikace
- Řízení – tzv. vnitřní logika aplikace, která předešlé dvě váže dohromady [10]

Obr. 8.1: Struktura model-zvhled-kontroler



Zdroj: [10]

K vytvoření aplikace použijeme integrované vývojové prostředí Mac OS X, kterým je Xcode verze 3.2.

7.1 Datový model

Aplikace se bude jmenovat Projekt_BC. Bude mít v sobě obsah účtů a pro každý účet bude mít seznam transakcí, jež byly na něm provedeny. To je v tomto případě také popis struktury dat, s nimiž bude aplikace pracovat (tzv. datový model). Tento model se stane součástí aplikace a třídy knihovny CoreData jej načtou po spuštění. Modelování v CoreData je založeno na entitách, relacích a atributech. Jinak se také těmto modelům říká “ER modely” podle zkratky anglického “entity-relationship”.

- Entita – typ datového objektu (všechny účty)
- Atributy – například jméno, částka transakce
- Relace – vzájemné vztahy mezi entitami [10]

Návrh datového modelu pro náš případ [10]

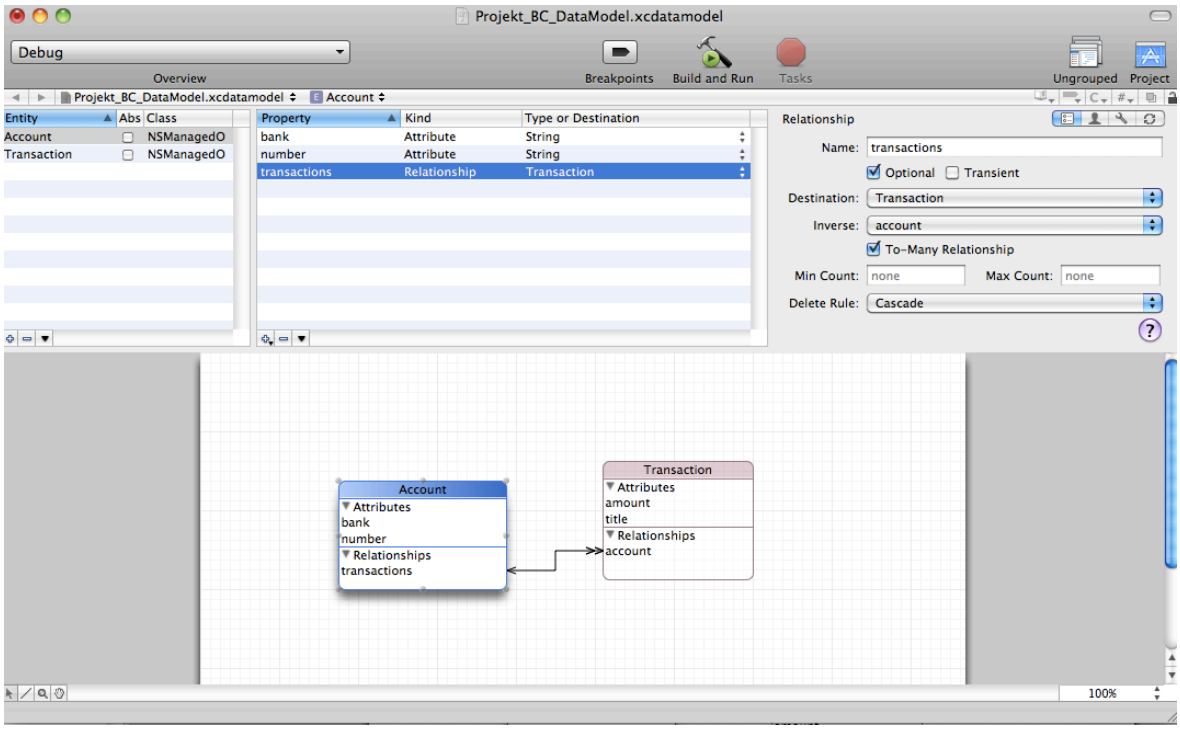
V tomto případě máme dvě entity, *Account* a *Transaction*. Account bude mít atributy bank a *number* (číslo účtu). Budou to textové řetězce. Transaction pak bude mít atribut *title* (popis transakce, textový řetězec) a jeden atribut *amount* (částka transakce, desetinné číslo).

Pak je třeba mezi entitami definovat relaci jménem *account*, vedenou z entity *Transaction* do entity *Account* (jde o relaci 1:1). Druhá relace bude mít jméno *trasactions*, povede z entity *Account* do entity *Transaction* (relace typu 1:N, protože pro daný účet vrátí seznam všech relací, jež mu patří). Relace budou inversní, tuto informaci také uložíme do modelu. Nemůže se tak pak stát, aby účet A obsahoval transakci B v relaci transactions, a přitom transakce B neměla právě účet A ve své relaci account.

Sestavení modelu v Xcode [10]

Vytvoříme nový projekt v Xcode se vzorem “Core Data Application”. Model najdeme v projektu pod názvem “Simple_Accountant_DataModel.xcdatamodel” ve skupině “Models”. Otevřem si ho v samostatném okně. Zde přidáme entity a jim dané atributy (a jejich datové typy). Pro relace je nejlepší přepnout editor na do režimu práce tím, že aktivujeme čáru. Myší pak přetáhneme dvě čáry, jednu z entity Account do entity Transaction, druhou pak opačným směrem. Dvě nově vzniklé relace si pojmenujeme jako v návrhu, tedy transactions a account. Postupě je pak nastavíme v jejich nastaveních (odkuď kam, typ relace) a nakonec u relace transactions zvolíme v “Delete Rule” variantu “Cascade”. Tímto je model kompletně hotov (viz. obr. 8.2), zbývá už jen vytvořit vrstvu vzhled a řízení aplikace.

Obr. 8.2: Datový model



7.2 Interface Builder

Vrstvy vzhledu a řízení aplikace v prostředí Cocoa se sestavují najednou. Vzhled jako strukturů objektů GUI, řízení pak jako patřičné zkonfigurované řídicí objekty a vazby.. To vše v jedné aplikaci – Interface Builderu. Je to editor souborů ve formátu „xib“. Ten v sobe uchovává všechny informace, které Interface Builder postupně vytváří a konfiguruje. Aplikace se pak jen v načte z disku, a má celé GUI a vrstvu řízení hotové. [10]

Nejprve Xcode vygenerovaný „xib“ upravíme na míru naší aplikace. V „MainMenu“ zrušíme „File“ smažeme vše kromě položek „Close“ a „Save“. Smažeme i nabídky „Format“ a „View“. V inspektoru hlavního okna aplikace upravíme titulek okna a zakážeme zavírání okna.Díky tomu, že Interface Builder rozumí datovým modelům, dokáže na základě jejich obsahu sestavit základní grafické uživvatelské rozhraní automaticky podle zvolené entity a připojit k němu odpovídající řídicí objekt. [10]

Přetahneme tak z datového modelu entitu Account do hlavního okna aplikace. Vybereme z nabídky, že chceme sestavit “Master/Detail view“. A vybereme jen potřebné editační objekty. V dalším kroku pak ponecháme označené pouze řádky *bank* a *number*, pak zakončíme tlačítkem Finnish. Vytvoří se tabulka (seznam účtů), byhledávací pole, textová pole, tlačítka pro přidání, zrušení účtu a další. S těmito prvky si pohrajeme aby nám vzhed nejvíce vyhovoval.

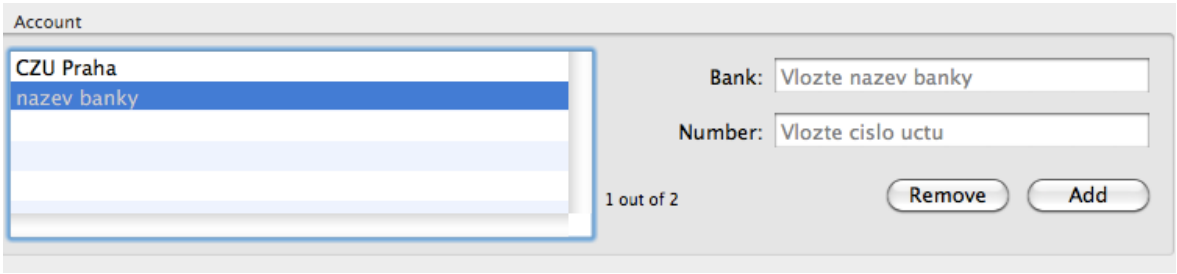
Přejdeme na entitu Transaction, kterou vhodíme také do hlavního okna z datového modelu. Postupujeme stejně, jen tady vybereme i editační objekt "Search Filed". Upravímé opět trochu rozhraní. Je třeba zajistit, aby v tabulce transakcí byly videt jen ty transakce, jež patří vybranému účtu. Toto je třeba řešit na úrovni řídicích objektů (nikoliv tak objektů vzhledu). Upravíme tak "Transaction Array Controller". Otevřeme vazbu contentSet a propojíme ji s řídícím objektem entity Account, s právě vybraným prvkem entity Account (můžeme využít příkazu selection), poté se seznamem transakcí vybraného účtu (do "Model Key" zapíšeme „transaction"). Nyní sestavíme a spustíme aplikaci pomocí "Bulid & Go" v Xcode. Máme tak hotovou funkční aplikaci. Nastává další fáze, kterou je doladění vylepšení. [10]

7.3 Doladění

Prázdná textová pole [10]

Prázdná textová pole po přidání nového účtu nebo transakce nahradíme například zástupným textem. Použijem k tomu v interface Builderu inspektor v režimu Bindings. Napíšeme vždy zástupná slova do políčka „Null Placeholder" (viz. obr. 8.3).

Obr. 8.3: Zástupná slova



Součty transakcí [10]

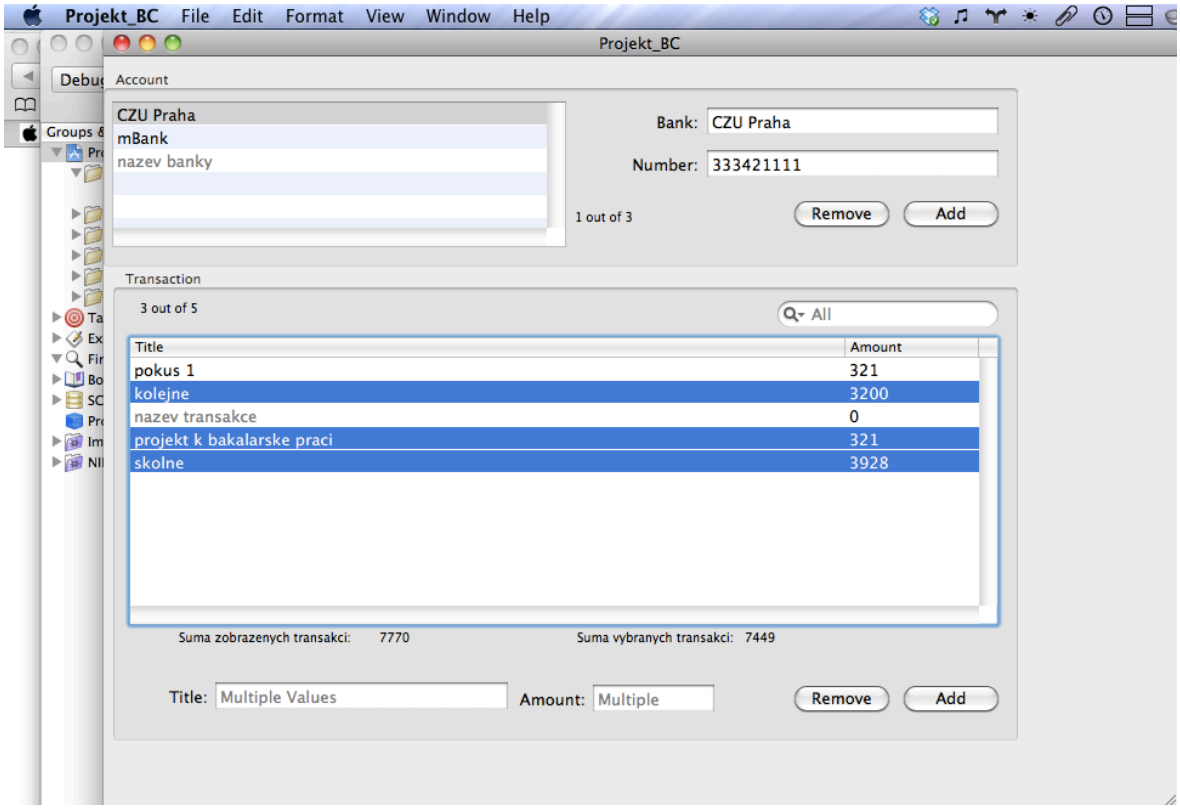
Větším nedostatkem je to, že nedokáže sčítat částky u zobrazených transakcí. Ta je ale potřebná. Do hlavního okna aplikace vhodně doplníme čtyři textová pole (2 budou nadpisy, 2 výsledné součty). Zapneme "Bindings" a vzby vytvoříme takto:

- v "Bind to:" zvolíme „Transaction Array Controller";
- v „Control Key" vybereme součet všech atributů arrangedObjects a zaškrtne mselection;
- v "Model Key Path" si vyžádáme součet pomocí "@sum.amount";

A to je vše, ještě můžeme doplnit vhodné hodnoty do prázdných políček pomocí „Null Palceholder“ a „No Selection Placeholder“.

Poté už jen v panelu Library vyhledáme skupinu „Formatters“, nalezneme „Number Formatter“ a myší jej přetáhneme na obě textová pole. Přepneme inspektor do Attributes a nastavíme požadované formáty. A jsme na konci, aplikace je hotová a po doladění nám i sčítá hodnoty transakcí (viz obr. 8.4).

Obr. 8.4: Hotová aplikace



8 Závěr

V bakalářské práci jsem se zabýval možnostmi tvorby aplikací na operačních systémech firmy Apple. Firma Apple má za sebou nespočet zásadních technologií, které alespoň částečně ovlivnily celou naši společnost. Proto bylo nezbytné se alespoň malou částí zmínit o její historii, která je provázána nejen životy tvůrců společnosti Apple, ale třeba také prvním osobním počítačem na světě. Dále jsou v kapitole o historii uvedeny zlomy, které vedly jak k úspěchům, tak i k existencionálnímu úpadku firmy. K úpadku došlo v polovině devadesátých let za nepřítomnosti jejího zakladatele Steva Jobse. Po jeho návratu zpět dokázal Steve Jobs navrátit firmu zpět na pomyslný vrchol, na kterém zůstává dodnes díky rozvíjejícím inovacím v technologickém odvětví.

Následně jsem se ve své práci věnoval popisu všech dosavadních verzí systému Mac OS. Od prvního systému počítače Macintosh v roce 1984 pojmenovaným System až po aktuální verzi vyspělého systému nové generace Mac OS X Snow Leopard. U každého ze zmíněných systémů jsem se snažil uvést jejich hlavní rysy a novinky oproti předchůdcům, jako například podpory nových technologií a postupné zdokonalování uživatelského rozhraní. Nový operační systém iOS pro mobilní telefon iPhone a tablet iPad, ktarému byl operační sytém Mac OS X vzorem a základním kamenem, pak celou kapitolu zakončuje.

Dále je rozebírána problematika tvorby aplikací na systémech Mac OS. Kde jsou srovnány možnosti vývoje aplikací v rozhraních Carbon, Cocoa a Java JDK. Po prostudování všech informací o těchto programovacích rozhraních následném srovnání všech tří mi vyšlo, že Java JDK je pro programování aplikací určených výhradně pro Mac OS X nepříliš vhodná právě proto, že nedokáže z důvodu zaměření na více platforem využívat některé jedinečné vlastnosti Mac OS X, kterými se tento systém liší od konkurence. Naopak se hodí pro tvorbu aplikací pro více platforem, jejich migrace je díky podstatě tohoto rozhraní velice jednoduchá. Corbon, který přišel s prvním počítačem Macintosh v roce 1984 byl až do dnešní doby majoritní řešení pro tvorbu aplikací na Mac OS, bohužel podpora moderních technoligií systému Apple není dále v Carbonu podporována a tvorba 64-bitových aplikací není možná. To má za následek, že se Carbon již skoro nepoužívá a dokonce i ty největší softwarové firmy svůj kód přepracovali do Cocoa. Rozhraní Cocoa je součástí Mac OS X od jeho vzniku a bylo navrženo pro maximální využití sytémových knihoven a všech systémových nástrojů, a tudíž je s objektově-orientovaným programovacím jazykem Objective-C nejlepší, nejefektivnější a hlavně nejlépe integrované řešení pro tvorbu moderních aplikací pro tento sytém. Využívá se i pro tvorbu aplikací pro systém iOS.

Důležitou částí této práce je kapitola o objektově-orientovaném jazyku Objective-C, který vznikl částečným spojením programovaacích jazyků Smalltalk a prostého C. S Objective-C jsem se sám důkladně seznámil, abych mohl pak přiblížit základní části tohoto programovacího jazyku, jako jsou: objekty, práce s nimi, jejich zapouzdření, dále pak zprávy, důležitost a výhody dědění, používání tříd a jejich rozhraní. Samozřejmostí jsou i názorné ukázky programovacích kódů, které dopomáhají lepší představě o fungování tohoto programovacího jazyku a samotnémé syntaxi jazyku. Jedná se spíše však jen o základy tohoto jazyku, z důvodu rozsáhlasti Objective-C jsem nemohl popsat jazyk do podrobna.

Na závěr jsem vytvořil jednoduchou aplikaci ve vývojovém prostředí Xcode firmy Apple a jeho programovacím rozhraním Cocoa s popisem postupu při jeho tvorbě. Na této jednoduché aplikaci jsem si vyzkoušel, jak je integrované prostředí Xcode velice programátorsky přívětivé. Díky propracovaným frameworkům a celkovému propojení všech potřebných služeb, je vývoj v tomto prostředí velice příjemný. Samotné prostředí s pomocí programovacího rozhraní dokáže programátorovi pomoci s částmi kódu, které by jinak musel vždy sám znovu programovat. Jedinečná součást Xcode, kterou je bezesporu Interface builder, pak nabízí tvořit vrstvu grafického uživatelského rozhraní a zároveň vrstvu řízení souběžně a jednodušše. Také je názorný obecný postup při tvorbě programu v objektově-orientovaném programovacím jazyku, od počátku, až po doladění.

Díky této práci jsem se seznámil s jazykem Objective-C, který mě překvapil svojí vyspělostí a propracovaností. Případná navazující diplomová práce se tak přímo nabízí. Můj zájem o programování v Cocoa touto prací vysoce vzrostl, a za to jsem velice vděčný.

9 Citovaná literatura

1. **Wozniak, Steve; Smith, Gina.** *iWoz: Muž, který postavil první osobní počítač.* [překl.] Jana Novotná. 1. Praha : Pragma, 2007. str. 360. 978-80-7349-064-5.

2. **Tracy, Ed.** History of computer design: Apple I. *Landsnail.com.* [Online] 1997. [Citace: 2. April 2011.] http://www.landsnail.com/apple/local/design/apple1.html.

3. **The Apple Museum** [online]. 1998, 2009 [cit. 2011-04-02]. The history of Apple Computer, Inc. Dostupné z WWW: <http://www.theapplemuseum.com/index.php?id=1>.

4. **Simon** , **William**; **Young, Jeffrey**. IKona Steve Jobs : Znovuzrození Apple - nejúžasnější návrat v historii byznysu . 1.v. Praha : Eugenika, 2008. 360 s. ISBN 978-80-8100-077-5.

5. Wikipedia.org [online]. 1998, 2011 [cit. 2011-04-02]. History Of Apple Inc. Dostupné z WWW: <http://en.wikipedia.org/wiki/History_of_Apple_Inc.>

6. Wikipedia.org [online]. 1998, 2011 [cit. 2011-04-02]. History Of Mac OS. Dostupné z WWW: <http://en.wikipedia.org/wiki/History_of_Mac_OS>.

7. **Apple inc.** Developer.apple.com [online]. 2008 [cit. 2011-04-02]. Carbon. Dostupné z WWW: <http://developer.apple.com/carbon/>.

8. **Trent, Michael; McCormack, Drew.** *Beginnig Mac OS X Snow Leopard Programming.* Indianapolis : Wiley Publishing, 2010. 978-0-470-57752-

9. **Apple Inc.** Developer.apple.com [online]. 2011 [cit. 2011-04-02]. Cocoa. Dostupné z WWW: <http://developer.apple.com/technologies/mac/cocoa.html>.

10. **Čada, Ondřej.** *Cocoa - úvod do programování počítačů Apple.* Praha : Grada Publishing a.s., 2009. 978-80-247-2778-3.

11.Wikipedia.org [online].2011 [cit. 2011-04-02]. Xcode. Dostupné z WWW: <http://en.wikipedia.org/wiki/Xcode>.

12. **Merunka, Vojtěch.** *Objektové modelování.* Praha : Alfa Nakladatelství s.r.o., 2008. 978-80-87197-04-2.

13. **Keogh, Jim; Giannini, Mario.** *OOP bez předchozích znalostí - Průvodce pro samouky.* [překl.] Veronika Matějů. Brno : Computer Press a.s., 2006. 80-251-0973-9.

14. **Marick, Brian.** *Programming Cocoa with Ruby.* North Carolina : The Pragmatic Bookshelf, 2009. 1-934356-19-0.

10 Přehled obrázků

11 Seznam zkratek

Zkratky: SDK, JDK, IDE, HFS, CPU