

Programski jezik Objective-C

Seminarski rad u okviru kursa
Metodologija stručnog i naučnog rada
Matematički fakultet

Dalma Beara, Denis Aličić, Mateja Marjanović, Anja Miletić
kontakt email prvog, a.denis96@gmail.com, trećeg, anya.miletic@gmail.com

6. april 2019

Sažetak

U ovom tekstu je ukratko prikazana osnovna forma seminarskog rada. Obratite pažnju da je pored ove .pdf datoteke, u prilogu i odgovarajuća .tex datoteka, kao i .bib datoteka korišćena za generisanje literature. Na prvoj strani seminarskog rada su naslov, apstrakt i sadržaj, i to sve mora da stane na prvu stranu! Kako bi Vaš seminarski zadovoljio standarde i očekivanja, koristite uputstva i materijale sa predavanja na temu pisanja seminarskih radova. Ovo je samo šablon koji se odnosi na fizički izgled seminarskog rada (šablon koji *morate* da ispoštujete!) kao i par tehničkih pomoćnih uputstava. Pročitajte tekst pažljivo jer on sadrži i važne informacije vezane za zahteve obima i karakteristika seminarskog rada.

Sadržaj

1 Uvod	3
2 Osnovna uputstva	3
3 Slike i tabele	3
4 Nastanak i istorijski razvoj, mesto u razvojnem stablu, uticaji drugih programskih jezika	3
4.1 Nastanak i istorijski razvoj	3
4.2 Uticaj drugih programskih jezika	4
5 Osnovna namena programskog jezika, svrha i mogućnosti	4
6 Osnovne osobine ovog programskog jezika, podržane paradigme i koncepti	5
6.1 Osnovni koncepti	5
6.1.1 Klasteri klasa	5
6.1.2 Introspekcija	5
6.1.3 Alokacija objekata	5
6.1.4 MVC	5
7 Najpoznatija okruženja (framework) za korišćenje ovog jezika i njihove karakteristike	6

8 Instalacija i uputstvo za pokretanje na Linux/Windows operativnim sistemima	6
8.1 Instalacija na Linux operativnim sistemima	6
8.1.1 Instalacija neophodnih paketa	6
8.1.2 Kompajliranje	6
8.2 Instalacija na Windows operativnim sistemima	7
8.2.1 Kompajliranje i izvršavanje	7
9 Primer jednostavnog koda i njegovo objašnjenje	7
10 Zaključak	8
Literatura	8

1 Uvod

Kada budete predavali seminarski rad, imenujete datoteke tako da sadrže redni broj teme, temu seminarskog rada, kao i prezimena članova grupe. Precizna uputstva na temu imenovnja će biti data na formi za predaju seminarskog rada. Predaja seminarskih radova biće isključivo preko veb forme, a NE slanjem mejla. Link na formu će biti dat u okviru obaveštenja na strani kursa. Vodite računa da prilikom predavanja seminarskog rada predate samo one fajlove koji su neophodni za ponovno generisanje pdf datoteke. To znači da pomoćne fajlove, kao što su .log, .out, .blg, .toc, .aux i slično, **ne treba predavati**.

2 Osnovna uputstva

Vaš seminarski rad mora da sadrži najmanje jednu **sliku**, najmanje jednu **tabelu** i najmanje **sedam referenci** u spisku literature. Najmanje jedna slika treba da bude originalna i da predstavlja neke podatke koje ste Vi osmislili da treba da prezentujete u svom radu. Isto važi i za najmanje jednu tabelu. Od referenci, neophodno je imati bar jednu **knjigu**, bar jedan **naučni članak** iz odgovarajućeg časopisa i bar jednu adekvatnu **veb adresu**.

Dužina seminarskog rada treba da bude od 10 do 12 strana.

3 Slike i tabele

Slike i tabele treba da budu u svom okruženju, sa odgovarajućim naslovima, obeležene labelom da koje omogućava referenciranje.

Na svaku sliku neophodno je referisati se negde u tekstu. Na primer, na slici ?? prikazane su pande.

Primer 3.1 *I tabele treba da budu u svom okruženju, i na njih je neophodno referisati se u tekstu. Na primer, u tabeli 1 su prikazana različita poravnanja u tabelama.*

Tabela 1: Različita poravnanja u okviru iste tabele ne treba koristiti jer su nepregledna.


centralno poravnanje	levo poravnanje	desno poravnanje
a	b	c
d	e	f

4 Nastanak i istorijski razvoj, mesto u razvojnom stablu, uticaji drugih programskih jezika

4.1 Nastanak i istorijski razvoj

Iako se pojam jezika Objective-C primarno vezuje za proizvode kompanije Epl (engl. Apple) – MAC OS X, iPhone itd, on je zapravo nastao mnogo pre njih. Njegove temelje postavili su Bred Koks i Tom Lav 1981.

godine, težeći da nađu način za povećavanje produktivnosti programera. Te godine pojavio se novi, revolucionarni jezik Smalltalk [?], koji je tada unapredio koncept objektno-orijentisanog programiranja. U osnovi tog jezika bilo je posmatranje programa kao skupova objekata koji su mogli da komuniciraju jedni sa drugima dinamički pozivajući metode. To je omogućilo da se stanje programa menja pod uticajem korisnika. Koks je u ovoj ideji video mogućnost da se vrtoglavo ubrza pisanje programa, pošto su se u njemu mogle praviti biblioteke objekata i posle koristiti u drugim programima bez izmene. Međutim, Smalltalk je bio veoma spor i zahtevao je da se svi programi pišu i pokreću u posebnom okruženju. Tada se javila potreba za spajanjem objektno-orijentisanih ideja Smalltalk-a i brzog jezika C. Koks je 1983. objavio naučni rad u kom je predstavio objektno-orijentisani prekompilator (eng. *Object-Oriented Precompiler*). Da bi ovu svoju ideju izbacili na tržište, njih dvojica su osnovali kompaniju Stepstone (eng. *Stepstone*), izmenili kompilator za OOPC i preimenovali jezik u Objective-C. Iako je glavna ideja ovog projekta bila pisanje i prodavanje pomenutih biblioteka koje su mogle ponovo da se koriste, korisnici su imali dosta zamerki, te je jezik sve više poprimao karakteristike objektno-orijentisanih jezika kakve ih danas znamo, kao što su sakupljač otpada (eng. *garbage collector*) i interpreter, te je kompanija uskoro propala. Stiv Džobs je 1988. kupio licencu, a nekoliko godina kasnije i sva prava za Objective-C za potrebe svoje kompanije NeXT Computer koja se 1996. pripojila Apple-u. Sa tom tranzicijom, sam jezik je pretrpeo razne promene, a najvažnije su uvođenje dveju novih „komponenti”: kategorija (eng. *categories*), danas poznatih kao ekstenzije (eng. *extensions*) i protokola (eng. *protocols*). Kategorije su služile da programerima omoguće da sami dodaju funkcije u objekte iz već postojećih biblioteka, a protokoli su omogućavali objektima da komuniciraju jedni s drugima. Oni su danas podržani u jeziku Java i poznati su kao interfejsi (eng. *interfaces*).



razvojno_stablo.pdf

Slika 1: Razvojno stablo

4.2 Uticaj drugih programskih jezika

Kao što je već navedeno, Objective-C je nastao kao kombinacija koncepata na kojima počiva Smalltalk utočenih u sintaksu jezika C. Isprva je čak funkcionisao tako što je kod prevođen na C i onda izvršavan. S druge strane, on je uticao na jezik Swift koji je razvio Epl. Swift [?] se često naziva i „Objective-C bez C-a” kao i na Javu [1](#).

5 Osnovna namena programskog jezika, svrha i mogućnosti

Krajem 80-ih godina prošlog veka, popularnost Objective-C jezika je rasla sa razvojem NeXT sistema. Najviše se koristi za razvoj softvera za Apple iOS operativni sistem. Uticaj Objective-C jezika je vidljiv u Java programskom jeziku. Objective-C definiše mali ali moćan skup ekstenzija ANSI C programskog jezika koje omogućavaju objektno-orijentisano

programiranje. Cocoa, Apple-ov API za macOS je napisan uz pomoć Objective-C jezika, takodje veliki deo aplikacija je napisan koristeći taj jezik.

6 Osnovne osobine ovog programskog jezika, podržane paradigme i koncepti

Objective-C je nadskup C programskog jezika koji pruža objektno-orijentisane sposobnosti i dinamičko izvršno okruženje (eng. *runtime*). Objective-C nasledjuje sintaksu, primitivne tipove i naredbe za tok izvršavanja od jezika C, a dodaje sintaksu za definisanje klasa i metoda. Takodje omogućava dinamičko tipiziranje i povezivanje.

6.1 Osnovni koncepti

6.1.1 Klasteri klasa

Klasteri klasa (eng. *class clusters*) su programski šablon koji se često koristi u pisanju programa na Objective-C jeziku. Klasteri klasa grupišu određeni broj privatnih podklasa unutar javne apstraktne nadklase. Ovakvo grupisanje pojednostavljuje javno vidljivu arhitekturu objektno-orijentisanih biblioteka i radnih okvira, bez smanjenja njihove funkcionalnosti. Apstraktna nadklasa mora da deklarise metode za kreiranje instanci svojih privatnih podklasa.

6.1.2 Introspekcija

Introspekcija je moćno svojstvo objektno-orijentisanih jezika i okruženja. Ono predstavlja mogućnost objekta da predstavi detalje o sopstvenoj implementaciji u toku izvršavanja programa. Ti detalji mogu biti njegovo mesto u drvetu nasledjivanja, da li implementira određeni protokol i da li odgovara na neku poruku. Primer:

```
1000 while ( id anObject = [objectEnumerator nextObject] ) {  
      if ( [self class] == [anObject superclass] ) {  
1002         // do something appropriate ...  
      }  
1004 }
```

6.1.3 Alokacija objekata

Prilikom alokacije objekata, dodeljuje se zahtevana memorija iz regiona virtualne memorije dostupne programu. Da bi se izračunalo koliko memorije treba alocirati, uzimaju se u obzir instancirane promenljive tog objekta - uključujući njihove tipove i redosled - kao što je navedeno u klasi objekta.

6.1.4 MVC

MVC (eng. *Model-View-Controller*) je programski šablon (eng. *design pattern*) visokog nivoa koji predstavlja globalnu arhitekturu aplikacije i razvrstava objekte prema ulozi koju imaju u toj aplikaciji. Objektno-orijentisano programiranje ima više koristi od ovog modela; povećava

se modularnost korišćenih objekata, njihovi interfejsi su bolje definisani. Sami programi se bolje adaptiraju promeni zahteva. MVC smatra da postoje tri vrste objekata: model, pogled i kontroler objekti. MVC šablon definiše uloge koje ovi objekti imaju u aplikaciji i način na koji komuniciraju. Pri dizajniranju aplikacije, bitan korak je biranje - ili pravljenje - klasa za objekte koji pripadaju jednoj od ovih uloga.

7 Najpoznatija okruženja (framework) za korišćenje ovog jezika i njihove karakteristike

8 Instalacija i uputstvo za pokretanje na Linux/Windows operativnim sistemima

8.1 Instalacija na Linux operativnim sistemima

8.1.1 Instalacija neophodnih paketa

Da bismo mogli da kompajliramo programe napisane u Objective-C, potrebno je prvo da instaliramo par paketa. Za kompajliranje Objective-C programa koristi se GCC kompajler[?], uz još par dodataka. Prvo što treba da uradimo je da instaliramo GCC podršku za Objective-C.

```
1000 $ sudo apt-get install gobjc
```

Zatim treba da instaliramo radno okruženje (eng. *framework*) na kom se moderni Objective-C zasniva i bez kog ne bi imalo puno smisla raditi (iako je moguće). To okruženje se zove GNUstep [?] i instalira se komandama:

```
1000 $ sudo apt-get install gnustep
$ sudo apt-get install gnustep-devel
```

8.1.2 Kompajliranje

Svaki put kada pokrenemo novu sesiju, moramo da pokrenemo skript GNUstep.sh, da bismo mogli da kompajliramo Objective-C programe.

```
1000 $ chmod +x /usr/share/GNUstep/Makefiles/GNUstep.sh
$ /usr/share/GNUstep/Makefiles/GNUstep.sh
```

Da ne bismo svaki put morali da izvršavamo taj skript, možemo da uradimo sledeće:

```
1000 $ echo /usr/share/GNUstep/Makefiles/GNUstep.sh >> ~/.bashrc
```

I na taj način, svaki put kad se pokrene nova sesija, biće izvršen GNUstep.sh i nećemo morati da brinemo.

Kao što smo već naveli, Objective-C koristi GNU-ov C kompajler, uz dodatnu podršku. Sem standardnih argumenata, GCC-u su potrebni i dodatni argumenti (eng. *flags*), kao što su `-lobjc`, `-lgnustep-base`, itd.

Dakle ako želimo da prevedemo `hello.m` program, moramo da izvršimo sledeće:

```
1000 $ gcc hello.m -o hello `gnustep-config --objc-flags` -  
      lgnustep-base -lobjc
```

Izvršavanje programa se radi uobičajeno:

```
1000 $ ./hello
```

8.2 Instalacija na Windows operativnim sistemima

8.2.1 Kompajliranje i izvršavanje

Kada smo instalirali MinGW i GNUstep, možemo da kompajliramo i izvršavamo Objective-C programe u shell okruženju, koje pokrećemo tako što otvorimo Shell program koji se nalazi u GNUstep. Inicijalno se nalazimo u C:

GNUstep

home

<username> direktorijumu. Program ćemo kompajlirati sledećom naredbom:

```
1000 $ gcc `gnustep-config --objc-flags` -L /GNUstep/System/  
      Library/Libraries hello.m -o hello -lgnustep-base -  
      lobjc
```

A izvršavanje sa:

```
1000 ./hello.exe
```

9 Primer jednostavnog koda i njegovo objašnjenje

Zatim, navedimo primer zastarelog (od verzije MAC 10.5), ali zanimljivog koda koji omogućava „imitiranje” (eng. *posing*).

```
1000 #import <Foundation/Foundation.h>  
1002 @interface MyString : NSString  
1004 @end  
1006 @implementation MyString  
1008 - (NSString *)stringByReplacingOccurrencesOfString:(  
      NSString *)target
```

```

1010 withString:(NSString *)replacement {
    NSLog(@"The Target string is %@",target);
    NSLog(@"The Replacement string is %@",replacement);
1012 }
1014 @end
1016 int main() {
    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc]
        init];
1018 [MyString poseAsClass:[NSString class]];
    NSString *string = @"Test";
1020 [string stringByReplacingOccurrencesOfString:@"a"
        withString:@"c"];
1022 [pool drain];
        return 0;
1024 }

```

10 Zaključak

Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak.
Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak.
Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak.
Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak.