|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ  **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ** |  |

Име и презиме Немања Шобо

**НАСЛОВ РАДА**

ДИПЛОМСКИ РАД

- Основне академске студије -

Нови Сад, година

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ ⚫ **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**  21000 НОВИ САД, Трг Доситеја Обрадовића 6 | | | |
| **КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА** | | | |
| Редни број, **РБР**: | | |  | |
| Идентификациони број, **ИБР**: | | |  | |
| Тип документације, **ТД**: | | | Монографска публикација | |
| Тип записа, **ТЗ**: | | | Текстуални штампани документ/ ЦД | |
| Врста рада, **ВР**: | | | Дипломски рад | |
| Аутор, **АУ**: | | | име презиме | |
| Ментор, **МН**: | | | др име презиме, звање | |
| Наслов рада, **НР**: | | | наслов | |
| Језик публикације, **ЈП**: | | | Српски (писмо – ћирилица или латиница) | |
| Језик извода, **ЈИ**: | | | Српски/енглески | |
| Земља публиковања, **ЗП**: | | | Србија | |
| Уже географско подручје, **УГП**: | | | Војводина | |
| Година, **ГО**: | | | година | |
| Издавач, **ИЗ**: | | | Ауторски репринт | |
| Место и адреса, **МА**: | | | Факултет техничких наука (ФТН), Д. Обрадовића 6, 21000 Нови Сад | |
| Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога) | | | ?/?/?/?/?/?/? | |
| Научна област, **НО**: | | | Електротехничко и рачунарско инжењерство | |
| Научна дисциплина, **НД**: | | | Примењене рачунарске науке и информатика | |
| Предметна одредница/Кључне речи, **ПО**: | | | кључне речи | |
| **УДК** | | |  | |
| Чува се, **ЧУ**: | | | Библиотека ФТН, Д. Обрадовића 6, 21000 Нови Сад | |
| Важна напомена, **ВН**: | | |  | |
| Извод, **ИЗ**: | | | сажетак | |
| Датум прихватања теме, **ДП**: | | |  | |
| Датум одбране, **ДО**: | | |  | |
| Чланови комисије, **КО**: | | Председник: | др име презиме, звање |
|  | | Члан: | др име презиме, звање | Потпис ментора |
|  | | Члан, ментор: | др име презиме, звање |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | UNIVERSITY OF NOVI SAD ⚫ **FACULTY OF TECHNICAL SCIENCES**  21000 NOVI SAD, Trg Dositeja Obradovića 6 | | | |
| **KEY WORDS DOCUMENTATION** | | | |
| Accession number, **ANO**: | | |  | |
| Identification number, **INO**: | | |  | |
| Document type, **DT**: | | | Monographic publication | |
| Type of record, **TR**: | | | Textual material, printed/CD | |
| Contents code, **CC**: | | | Bachelor thesis | |
| Author, **AU**: | | | ime prezime | |
| Mentor, **MN**: | | | dr ime prezime, zvanje na engleskom | |
| Title, **TI**: | | | naslov na engleskom | |
| Language of text, **LT**: | | | Serbian | |
| Language of abstract, **LA**: | | | Serbian/English | |
| Country of publication, **CP**: | | | Serbia | |
| Locality of publication, **LP**: | | | Vojvodina | |
| Publication year, **PY**: | | | godina | |
| Publisher, **PB**: | | | Author’s reprint | |
| Publication place, **PP**: | | | Faculty of Technical Sciences, D. Obradovića 6, 21000 Novi Sad | |
| Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes) | | | ?/?/?/?/?/?/? | |
| Scientific field, **SF**: | | | Electrical and computer engineering | |
| Scientific discipline, **SD**: | | | Applied computer science and informatics | |
| Subject/Key words, **S**/**KW**: | | | ključne reči na engleskom | |
| **UC** | | |  | |
| Holding data, **HD**: | | | Library of Faculty of Technical Sciences, D. Obradovića 6, 21000 Novi Sad | |
| Note, **N**: | | |  | |
| Abstract, **AB**: | | | sažetak na engleskom | |
| Accepted by the Scientific Board on, **ASB**: | | |  | |
| Defended on, **DE**: | | |  | |
| Defended Board, **DB**: | | President: | dr ime prezime, zvanje na engleskom |
|  | | Member: | dr ime prezime, zvanje na engleskom | Menthor's sign |
|  | | Member, Mentor: | dr ime prezime, zvanje na engleskom |  |

|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ ⚫ **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**  21000 НОВИ САД, Трг Доситеја Обрадовића 6 | Датум: |
|  |
| **ЗАДАТАК ЗА ИЗРАДУ ДИПЛОМСКОГ (BACHELOR) РАДА** | Лист/Листова: |
| 1/1 |

*(Податке уноси предметни наставник - ментор)*

| Врста студија: | Основне академске студије  Основне струковне студије |
| --- | --- |
| Студијски програм: |  |
| Руководилац студијског програма: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент: |  | Број индекса: |  |
| Област: |  | | |
| Ментор: |  | | |
| НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА  ИЗДАЈЕ СЕ ЗАДАТАК ЗА ДИПЛОМСКИ (Bachelor) РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:   * проблем – тема рада; * начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна; * литература | | | |

**НАСЛОВ ДИПЛОМСКОГ (BACHELOR) РАДА:**

|  |
| --- |
|  |

**ТЕКСТ ЗАДАТКА:**

|  |
| --- |
|  |

|  |  |
| --- | --- |
| Руководилац студијског програма: | Ментор рада: |
|  |  |

|  |
| --- |
| Примерак за:  - Студента;  - Ментора |

Образац **Q2.НА.15-04** - Издање 1

**SPISAK KORIŠĆENIH SKRAĆENICA**

Skraćenica Puni naziv

(spisak sortirati po skraćenicama)

**SADRŽAJ**

**1. UVOD**

**2. OPIS REŠAVANOG PROBLEMA**

**2.1**

**2.2**

**2.3**

**3. OPIS KORIŠĆENIH TEHNOLOGIJA I ALATA**

**4. OPIS REŠENJA PROBLEMA**

# **5. Zaključak**

**6. LITERATURA**

**PRILOG 1**

**PODACI O KANDIDATU**

(nazivi poglavlja u sadržaju su okvirni; UVOD i ZAKLJUČAK pisati na kraju, nakon završetka preostalih poglavlja)

**1. UVOD**

Navesti kratak pregled rada.

Navesti kratak prikaz ostalih poglavlja rada.

Orijentaciona dužina uvoda je oko jedne stranice teksta.

Navesti detaljan opis problema rešavanog u radu. Pozvati se na relevantnu literaturu i prodiskutovati već postojeća rešenja razmatranog problema.

Ispod svake tabele (slike ili listinga) navesti broj i naziv, kao za Tabelu 2.1.

|  |  |  |
| --- | --- | --- |
|  | kolona 1 | kolona 2 |
| red 1 | element 1,1 | element 1,2 |

Tabela 2.1 Primer tabele

U tekstu obavezno prodiskutovati svaku tabelu (sliku ili listing).

Korišćena literatura se citira navođenjem rednog broja reference između srednjih zagrada, na primer [1]. U tekstu obavezno citirati svaku referencu iz literature.

Samo napraviti pregled raspoloživih tehnologija i alata, navesti razloge izbora tehnologije i alata, kao i njihova bitna svojstva i prednosti**.**

**2. OPIS REŠAVANOG PROBLEMA**

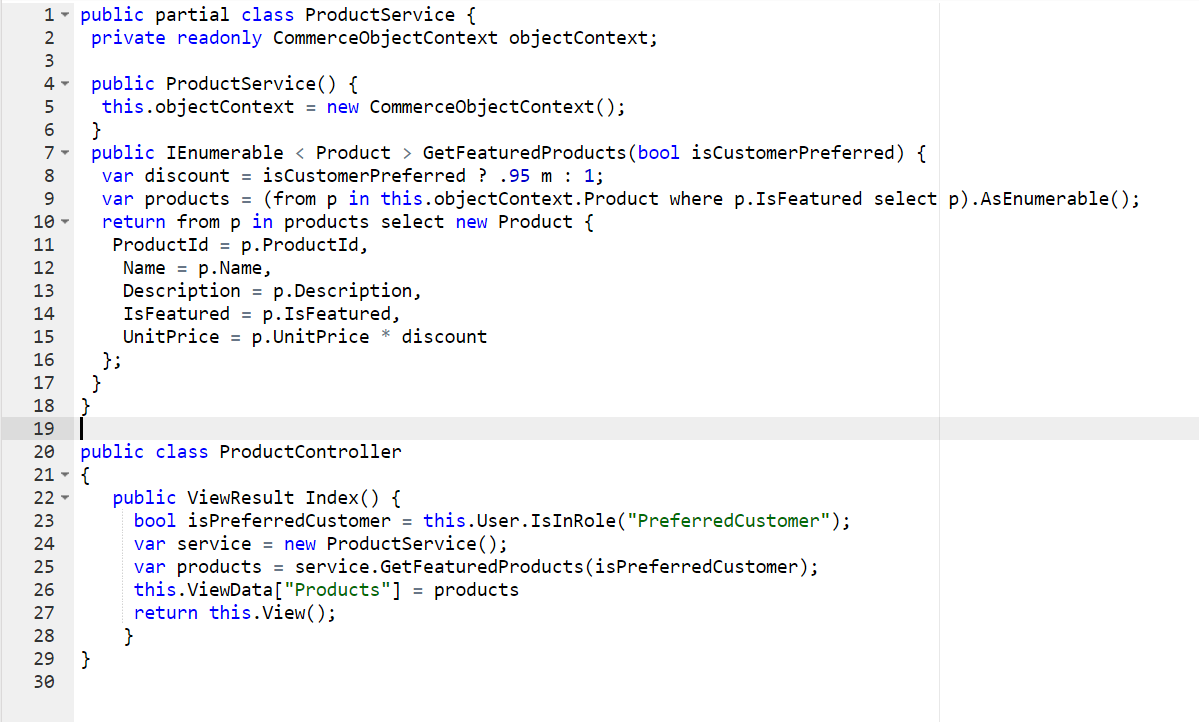
2.1. Motivacija zakorištenje dependency injection-a

Živimo u vremenu kada posao od ljudi zahteva čestu promenu i konstantan razvoj i proširenje postojećih softverskih rešenja. Veoma često klijenti posle prvog puštanja nekog softverskog proizvoda u produkciju dolaze do različitih zaključaka i promene mišljenja kako bi unapredili svoje rešenje kao i ispoštovali zahteve korisnika koji usled korištenja aplikacije očekuju promene koje će im korištenje iste učiniti lakšim i pristupačnijim. Takođe se dešava da klijenti menjaju mišljenje o servisima koji vrše određeni deo posla na različitim slojevima aplikacije kako bi optimizovali njen rad ali i zamenili zastrele servise novim koji funkcionišu mnogo bolje i njihovo održavanje je lakše i isplativije. Obzirom na gore navedeno na nama je da sve to učinimo što lakšim kako bi smo rešenju omogućili konstantan napredak i održivost na visokom nivou.

2.2. Čvrsto spregnuti kod

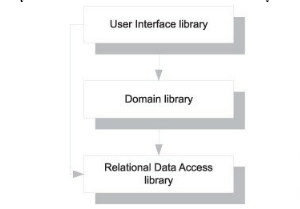
Projektovanjem i kreiranjem softvera koji je interno čvrsto spregnut činimo održivost i proširivost velikih projekata skoro nemogućim i samim tim onemogućujemo njegov dalji razvoj i unapređenje. Čvrsta spregnutost koda ili Tight Coupling je uobičajeno loša praksa pisanja a njene posledice su slaba fleksibilnost, gotovo nemoguća ponovna iskorsitivost, dodavanje određenih izmena je veoma problematično i testiranje aplikacije je veoma otežano. Tightly Coupled komponente su one komponente koje mnogo znaju jedna o drugoj, odnosno o samoj implementaciji i toliko su spregnute da izmena jedne nužno znači i izmenu druge.

2.3 Primer čvrsto spregnutog koda



Slika 2.1 Primer čvrsto spregnutog koda

Na slici 2.1 vidimo primer čvrsto spregnutih slojeva aplikacije. Pretpostavimo da ProductService klasa pripada sloju aplikacije koji je zadužen za upravljanje poslovnom logikom aplikacije, ComerceObjectContect pripada sloju koji je zadužen za komunikaciju sa bazom podataka, a ProductController sloju koji je zadužena za slanje podataka u onom obliku u kom treba da budu prikazani korsniku aplikacije. Čvrsta spregnutost se najviše ogleda u dirketnom instanciranju komponenti iz drugih slojeva( u ovom slučaju library-a) aplikacije i samim tim potpuno onemogućujemo izlociju ovih modula tako da mogu da rade zasebno bez oslanjanja jedan na drugog. Glavna prednost višeslojnih aplikacija je u tome što nam omogućava totalno razdvajanje poslova za koji je svaki sloj zadužen, kao što je na primer dobavljanje podataka iz baze, biznis logika, prikaz podataka itd. Povezivanje slojeva na ovaj način spregnuli smo aplikaciju tako da naše komponente ne mogu da egzistiraju jedna bez druge i samim tim narušili koncept slabo spregnutih rešenja.



Slika 2.2. Graf zavisosti između slojeva aplikacije

Slika 2.2. nam prikazihe graf zavisnosti između slojeva (Library) aplikacije oslikavajući kod sa slike 2.1. odnosno prikaz zavisnosti slojeva kojima klase prikazane u njoj pripadaju. User Interface library sloj u kom se nalazi ProductController klasa čiji je zadatak da podatke koje dobija iz servisa šalje na komponente koje se bave prikazom, referencira Domain library odnosno servisni sloj u kome se nalazi ProductService klasa a zbog svoje visoke spregnutosti sa njom mora da referencira i RelationalDataAccess library u kojoj se nalazi kontekstna klasa odnosno CommerceObjectContext. Graf nam ukazuje na prolongiranje Product klase iz sloja za upravljanje bazom podataka u sve slojeve koji sarađuju s njim. Samim tim izbacivanje ovog sloja u slučaju da želimo na primer da zamenimo servis koji komunicira sa sql bazom podataka za neki drugi koji je takođe zadužen za komunikaciju sa bazom nam je otežano i zahteva izmene u svim slojevima aplikacije, što u velikim sistemima može da prestavlja jako veliki problem kao i ogroman utrošak vremena i sredstava. Cilj dependency injectiona-a je omogućavanje slabe spregnutosti između slojeva tako da rade u potpunoj izolaciji jedna od druge oslanjajuci se samo na komunikaciju putem interfejsa gde je konkretna implementacija servisa nebitna i u svakom momentu zamenjiva.

2.4 Dependecy Injection

Jedna od mnogih definicija : Dependency Injectionje set softverskih dizajn principa i paterna koji nam omogućavaju da pisemo slabo spegnut programski kod(loosely coupled).

Prednosti slabo spregnutog koda :

1. Servisi su lako zamenjivi drugim servisima. Vrlo značajno kad je u pitanju standardan softver prilikom izmene zahteva i načina implementacije.
2. Kod je proširiv kao i ponovo iskoristiv u situacijama koje nisu bile planirane niti predviđene da bi do njih moglo doći.
3. Paralelni razvoj rešenja odnosno aplikacije što je veoma bitno kod velikih i kompleksnih aplikacija. Kod malih aplikacija ova prednost ne dolazi do izražaja.
4. Veoma održiv kod, klase kojima je jasno definisan njihova odgovornost su lake za održavanje.
5. Takođe u slučaju korištenja unite testova pisanje slabo spregnutnog koda je tehnika koja testiranje čini lakšim.

Omogućava nam da različite odgovornosti nekih klasa razdvojimo u posebne odnosno enkapsulirane jedinice koje su zadužene da obavljaju određeni deo posla. Takođe oslobađa klase od kreiranja instanci objekata od kojih zavisi i sa kojima je u saradnji odnosno na čiji interfejs se oslanjaju. Dependecy Injection nam omogućava da komponujemo aplikaciju presretajući zavisnosti unutar nje i kontrolisanje životnog ciklusa tih zavisnosti.

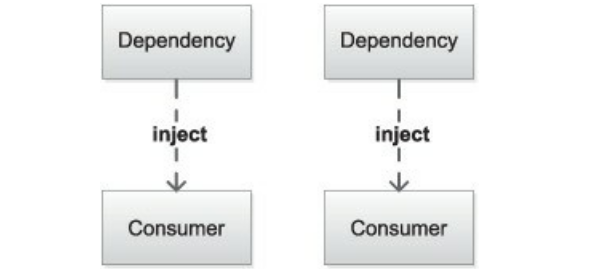
Kompozicija objekata(OBJECT COMPOSITION), presretanje(INTERCEPTION) i upravljane životnim ciklusom(LIFTIME MANAGAMENT) predstavljaju tri dimenzije DI.

2.4.1 Kompozicija objekata(Object composition)

Da bi smo osetili prednosti proširivosti, paralelnog razvoja aplikacije kasnije i lake odživosti moramo prvo biti u mogućnosti da izvršimo kompoziciju klasa u aplikaciju. Jedan od osnovnih razloga zbog koga bi smo uopšte koristili dependency injection je upravo ta što nam omogućava kompoziciju objekata. Ona nam ustvari ukazuje na to da neki odvojeni moduli mogu biti sastavljeni u jednu aplikaciju. Kompozicija predstavlja najvažniji aspekt dependency injectiona jer bez fleksibilne kompozicije nema ni presretanja a simim tim ni potreble za upravljanjem životnim ciklusom samih objekata odnosno njihovih instanci.

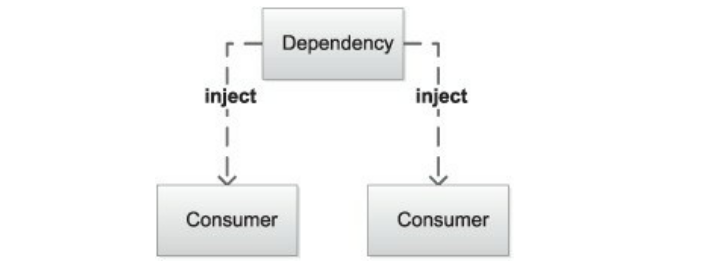
2.4.2 Upravljanje životnim ciklusom (Liftime managament)

Klase koje nemaju kontrolu nad svojim zavisnostima, omogućavaju nam da sami pravimo izbor u odabiru implementacije određene apstrakcije. Takođe nam daju kontrolu nad upravljanjem životnim ciklusom instanci, odnosno odabiru kada će neke instance biti biti kreirane ili uništene. Takođe upravljanje životnim ciklusom se bavi i brojem kreiranih istanci nekog objekta.



Slika 2.3 Primer više instanci istog zavisnotg objekta prem različitim korisnicima

Slika 2.3 prestavlja grafički prikaz prisupu kreiranja onoliko instanci jednog zavisnog objekta koliko postoji servisa koji ga koriste.



Slika 2.4 Jedna instanca opslužuje sve korisnike

Slika 2.4 prikazuje pristup kreiranju jedne insance zavisnog objekta koji će opsluživati sve servise koji ga korsite. Odabir jednog od dva pristupa kreiranja zavisi od konteksta i upotrebe samog zavisnog objekta.

2.4.3 Presretanje(Interception)

Kada klasa nema kontrolu nad kreiranjima instanci zavisnih objekata, daje nam mogućnost same izmene objekta pre nego što joj ga prosledimo. Ovo nam je omogućeno upravo zbog toga jer se korisnici nekih zavisnih objekata uvek oslanjaju na njihove interfejse odnosno njihove apstrakcije, a nikad na kontkretne implementacije, pa nam samim tim to ostavlja prostor da objekte menjamo i uređujemo pre nego što budu prosleđeni ili ti injektovani u servis koji zahteva njihovu instancu.

**3. OPIS KORIŠĆENIH TEHNOLOGIJA I ALATA**

**3.1 C# programski jezik**

Cilj ovog programskog jezika je da obezbedi jednostavan, bezbedan i moderan objektno orjentisani programski jezik visokih performansi za .NET development. Veoma lak za korištenje ali u stanju da omogući implementaciju modernih koncepata programiranja. Sadrži podršku za struktuirano, komponentno bazirano, objektno orjentisano programiranje koje se očekuje od modernog jezika koji je nastao na osnovu C++ i Jave. Kao i svaki objekto orjentisni jezik osnovna namena mu je da obezbedi podršku za definisanje i posao sa klasama. Odnosno za definisanje novih tipova koji će nam omogućiti bolje modelovanje problema koji pokušavamo da rešimo. Podžava nas u kreiranju novih klasa, njihovih metoda i atributa, kao i implementacije enkapsulacije, nasleđivanja i polimorfizma kao tri osnovna principa objekto orjentisanog programiranja.

3.2. .NET razvojno okruženje

# **4. Opis rešenja problema**

# **5. Zaključak**

Osvrt na rešenje problema, uz ukazivanje na njegove dobre i eventualno loše strane, kao i na moguće pravce daljeg istraživanja. Osvrt na rešenje pisati u prošlom vremenu.

Orijentaciona dužina zaključka je oko jedne stranice teksta.

**6. LITERATURA**

[1] ***Cleaveland, J.C., Building Application Generators, 1988, Software, IEEE, Jul 1988, Vol 5, Is 4***

[2] ***M. E. Lesk and E. Schmidt, Lex - A Lexical Analyzer Generator, 2005, www.compilertools.net***

[3] ***Wikipedia, Regular expression, 2006, Wikipedia***

[4] ***ISO/IEC, EBNF - ISO 14977 standard, 1996, ISO/IEC***

[5] ***Free Software Foundation, Inc., Bison manual, 2005, http://www.gnu.org/software/bison/bison.html***

[6] ***A.V. Aho, R. Sethi, and J.D. Ullman, Compiler: Principles, Techniques and Tools, 1986, Addison-Wesley***

[7] [***www.w3.org***](http://www.w3.org), spisak ključnih reči prezentacije kako bi bila dostupna i u slučaju promene URL

**PRILOG 1**

**PODACI O KANDIDATU**

Kandidat ime i prezime je rođen(a) ?. godine u mestu. Završio(la) je srednju vrsta školu naziv u mestu ?. godine. Fakultet tehničih nauka u Novom Sadu je upisao(la) ?. godine. Ispunio(la) je sve obaveze i položio(la) je sve ispite predviđene studijskim programom.