Multiteka

Design Guidelines

Verzija 1.0

Istorija revizija

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Veszija** | **Opis** | **Autor** |
| 30.3.2015. | 1.0 |  | Aleksandra Vujadinović |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Sadržaj

[1. Uvod 4](#_Toc415591359)

[1.1 Svrha 4](#_Toc415591360)

[1.2 Područje 4](#_Toc415591361)

[1.3 Opseg 4](#_Toc415591362)

[1.4 Definicije, skrećenice i akronimi 4](#_Toc415591363)

[1.5 Reference 4](#_Toc415591364)

[2. Generalne smjernice za dizajn i implementaciju 4](#_Toc415591365)

[3. Smjernice za dizajn baze 6](#_Toc415591366)

[4. UML Stereopitovi 6](#_Toc415591367)

Dizajn- početne smjernice

# Uvod

Dokument „Dizajn- početne smjernice“ sliži za definisanje pravila, standarda i ostalih zakona u okviru tima, na osnovu kojih će se omogućiti lakša komunikacija, sporazimjevanje i riješavanje zadataka. Na osnovu dokumenta omogučeno je lakse razumjevanje i uniformno pisanje koda kao i ostale dokumentacije.

## Svrha

Svrha ovog dokumenta je da opise standarde dizajna koji će se primjenjivati u nastavku, zatim konvencije i idiome koji se koriste u opštem dizajniranju sistema, tako i u ovom sistemu.

## Područje

Dokument je direktno vezan za sistem Miltiteke jer predstavlja smjernice koje se slijede u procesu njegovog nastajanja, kreiranja i održavanja.

## Opseg

Opseg ovog dokumenta odnosi se na standarde pisanja koda koji će se usvojiti i primjenjivati u kreiranju informacionog sistema, kao i na standarde pisanja dokumentacije i opšta pravila kodiranja.

## Definicije, skrećenice i akronimi

GUI - grafičko korisničko okruženje (Graphic User Interface)

UML- Unified Modeling Language

## Reference

# Generalne smjernice za dizajn i implementaciju

* Mapiranje od projektovanja do implementacije

Prekirati pakete u programskom jeziku i na smislen i logičan način rasporediti klase unutar njih.

Davati smislena ime paletma i klasama, po potrebi krirati i stablo radi lakšeg uopčavanja povezanosti

* Dokumentovanje operacije

U pisanju koda, za potrebe pisanja i korištenja operacija korističe se standardan način opisivanja operacija. Operacije će se sastojati od imena, argumenata i po potrebi dodatnog opisa i specifične implementacije.

Nije neophodno da svi formalni argumenti budu posebno dokumentovani i detaljno opisani, posebno kada su u pitanju dijelovi koda koji se rijetko koriste.

Po potrebi, ili na zahtjev korisnika sistema i tipovi argumenata će biti dodatno dokumentovani.

* Dokumentovanje poruka

Nazivi promjenjivih bi trebali ukratko da opisuju svrhu promjenjive. Ako je potreban dodatan opis svrhe promjenjive isti je potrebno napisati u vidu komentara u kodu. Ovi komentari služe za lakše razumjevanje rada metode i ne ulaze u JavaDoc dokumentaciju.

* Otkrivanje, opsluživanje i izvještaji o greškama

Svaka kritična naredba, kao i kod za koji postoji mogućnost da se ne izvrši pravilno da zablokira i izbaci gršku mora da se stavi u try/catch blok.

Na osnovu toga je smanjena mogućnost totalnog prestanka rada prograama.

Pružiti mogučnoct korisniku , da ukoliko je greška u unosu podataka, može da se ispravi.

Izvještaje o greškama prikazati na ekran korisniku, sa mogućnostima dodatnog opisa greške.

* Upravljanje memorijom

Potrebno ja da se, ukoliko je to moguće, izvršava optimizacija koda i zauzima sto manje memorije za promjenjive koje ne trebaju u čitavom kodu.

Kao tipove promjenjivih uzimati realne vrijednosti ( maksimalnu dizinu) i ne pretjerivati.

Koristiti „garbage collector” u Javi.

* Distribucija softvera

Primjena softvera na različitim mjestima i kod različitih vrsta klijenata.

Korištenju softvera će predhoditi potrebna obuka , prikazivanje načina za distribuciju i korištenje.

* Posebna upotreba jezika i njegove karakteristike

Za davanje imena paketma, klasama, interfesjima kao i promjenjivim, konstantama, funkcijama, metodama. Parametrima koristi se engleski alfabet.

Radi lakšeg snalaženja i kompajliranja nije dozvoljeno korištenje znakova sa klakicama i nekih sprecijalnih znakova.

U nazivima mogu se još koristiti i brojevi ( ali ne na prvom mjestu) kao i specijalni karakteri „-” i „\_”

* Programska struktura

Deklaracije promjenljivih osim brojača su na početku koda.

Nazivi promjenljivih, klasa i funkcija moraju biti smisleni i logični.

U slučaju da se naziv promjenljive, klase ili funkcije sastoji od dvije ili više riječi, one se pišu spojeno, a nova riječ se označava velikim početnim slovom.

Nazivi promjinljivih počinju malim slovom, kao i nazivi funkcija anazivi klasa velikim početnim slovom.

Svaka naredba ide u novi red.

Svaka vitičasta zagrada zauzima po čitav red.

Razmak između blokova naredbi je po jedan red.

Pisanje dodatnih komentara u novom redu.

* Dokumentacija metoda

Upisuje se ime metode i navode se ulazni i izlazni parametri.

Opisuje se svrha metode i njena osnovna funkcionalnost.

Svaki od ulaznih i izlaznih parametara treba biti ukratko opisan, čemu sluzi i koji tip je potreban za pravilan unos.

# Smjernice za dizajn baze

U ovom pasusu potrebno je definisati pravila i preporuke za kreiranje baze podataka

* Aplikacija je generičkog tipa i nije vezana ni za jednu strukturu baze podataka. Strukturu baze aplikacija učitava iz posebno struktuiranog XML fajla na osnovu koje na adekvatan način prikauje podatke.
* Dizajn baze podataka, sve potrebne klase i atribute predstaviti u konceptualnom, logičkom i fizićkom modelu i na osnovu njega genetisati potreban kod
* Konvencija imenovanja u bazi podataka

Za davanje entitetima, vezama i atributima koristi se engleski alfabet.

Radi lakšeg snalaženja nije dozvoljeno korištenje znakova sa klakicama i nekih sprecijalnih znakova.

U nazivima mogu se još koristiti i brojevi ( ali ne na prvom mjestu) kao i specijalni karakteri „-” i „\_”

Nazivi promjenljivih, klasa i funkcija moraju biti smisleni i logični.

Nazivi klasa ( entiteta) pišu se velikim slovima

Nazivi atributa pisu se malim slovima, u nayivu treba da stoji i oznaka entiteta kojem atbibut pripada radi lakšeg snalađenja i razumjevanja šeme baze

# UML Stereopitovi

Poželjno je iskoristiti što više dizajn šablona koji su obrađeni u toku nastave iz predhodnih predmeta. Šabloni koji bi bili poželjni pri realizaciji su:

* MVC (Model-View-Controller)

Šablon MVC definiše slabo vezivanje između prikaza i logike obrade podataka. Odvojeni prikaz od logike obrade podataka postiže visoku fleksibilnost i može doprinjeti boljem razvoju i arhitekturi aplikacije.Ovaj šablon bi trebalo iskoristiti kao osnovu za realizaciju aplikacije.

* Command

Razdvaja objekat koji aktivira operaciju od objekta koji ima potrebna saznanja da operaciju izvrši. To omogućava fleksibilnost u projektovanju korisničkog interfejsa.Ovaj šablon bi trebalo iskoristiti za realizaciju komandi u aplikaciji.

* Singleton

Obezbjeđuje da klasa ima samo jedan primjerak i daje mu globalnu tačku pristupa.Ovaj šablon bi trebalo iskoristiti za klasu koja uspostavlja konekciju sa bazom pošto je konekcija jedinstvena za instancu aplikacije.

* Composite

Sastavlja objekte u strukture stabla da bi se predstavile hijerarhije dijelova i cjelina. Composite omogućava klijentima ujednačeno postupanje pojedinačnim objektima i kompozicijama objekata.Ovaj šablon bi trebalo iskoristiti za formiranje polja za unos željenog podatka.

* State

Omogućava da objekat promjeni ponašanje kada mu se promjeni unutrašnje stanje. Ova promjena izgleda kao da je objekat promjenio klasu.Ovaj šablon bi trebalo iskoristiti za rješavanje različitih ponašanja u različitim stanjima aplikacije.