Multiteka

Programming Guidelines

Version 1.0

Istorija revizija

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Verzija** | **Opis** | **Autor** |
| 30.3.2015. | 1.0 |  | Aleksandra Vujadinovic |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Sadržaj

[1. Uvod 4](#_Toc415590368)

[1.1 Svrha 4](#_Toc415590369)

[1.2 Područje 4](#_Toc415590370)

[1.3 Definicije, skrećenice i akronimi 4](#_Toc415590371)

[1.4 Reference 4](#_Toc415590372)

[1.5 Pregled 4](#_Toc415590373)

[2. Organizacija koda i stil 4](#_Toc415590374)

[3. Komentari 5](#_Toc415590375)

[4. Imenovanje 5](#_Toc415590376)

[5. Deklaracija 5](#_Toc415590377)

[6. Izrazi i naredbe 6](#_Toc415590378)

[7. Memorija 6](#_Toc415590379)

[8. Rukovanje greškama i izuzecima 6](#_Toc415590380)

[9. Prenosivost 6](#_Toc415590381)

[10. Ponovna upotreba 7](#_Toc415590382)

[11. Problemi pri kompajliranju 7](#_Toc415590383)

Programiranje – početne smjernice

# Uvod

Dokument „Programiranje - početne smjernice “ opisuju standarde i konvencije će biti korištene prilikom programiranja informacionog sistema Multiteke. Postoji mnogo razloga zbog kojih je bitno držati se ovih smjernica tokom programiranja, a jedan od njih je da se veći dio životnog ciklusa programa odnosi na održavanje. Na osnovu ovog dokumenta moguće je uskladiti stil programiranja koja će se svi članovi tima držati i na taj način znatno olakšati čitljivost za sve članove.

## Svrha

Svrha dokumenta je da opiše kolekciju standarda, konvencija i smjernica koji će se koristiti pri pisanju java koda i kreiranju programa.

Smjernice u programiranju takođe imaju za cilj da naglase pravila kojih će se programer držati u procesu pogramiranja. Ovim se omogucava lakse čitanje i razimjevanje koda u buducnosti, a samim tim je olakšano ažuriranje i održavanje programa.

## Područje

Dokument je direktno vezan za sistem Miltiteke jer predstavlja smjernice koje se slijede u procesu njegovog programiranja.

## Definicije, skrećenice i akronimi

Klasa - skup objekata koji imaju zajedničku strukturu i ponašanje

Varijabla – promjenljiva

Exception- izuzetak

Overload – preopterećenje

Properties – svojstva

## Reference

## Pregled

U okviru dokumenta „Programiranje - početne smjernice “ dalje će biti govora o pisanju komentara, imenovanju klasa, promjenjivih, metoda, interfejsa, načina deklaracije i drugih stavki koje su direktno vežu za način programiranja ovog sistema.

# Organizacija koda i stil

Kompletan program porebno je uraditi u Javi. Kreirati posebne pakete za smještanje slika, uključenih biblioteka i drugih referecni u kodu. Organizovati u pakete i foldere srodne klase i interfejse. U svakoj klaci držati se odredjenog gnijezna, uvlačenja u pravljenja podstruktura koje trebaju biti lako vidljive i uočljive. Potrebno je određeno iskustvo u korišćenju softvera da bi mogli razviti nešto poput sopstvenog “stila” kodiranja, tako da smjernice služe samo kao grubi oblik koraka ka tom procesu. Ono sto je najbitnije jeste sama urednost, jasna čitljivost i vidljivost onoga što se želi prikazati. Kao jedinicu uvlačenja koristiti 4 razmaka ili jedan tab što je i približno jednako. Kod treba svoju urednost da ogleda u pisanju komentara

# Komentari

Komentari implementacije se odnose na komantarisanje samog koda. Oni treba da sadrže osnovna uputstva kako bi sam autor ili neki drugi saradnik mogao da ima jasan uvid u logiku koda. Ukoliko dolazi do težeg pisanja komentara, te ukoliko se on ogleda u dužim opisima, treba ramisliti da li postoji neko kraće i efikasnije rješenje u kodu

Koriste se slijedeće vrste komentara:

* Komentari počinju sa „//“ i završavaju se do kraja linije. Ovakav nacin komentarisanja se koristi za jednoliniske komentare i kratka upustva ili objašnjenja linija koda ispod komentara
* Komentari počinju sa „/\*“ i završavaju sa „ \*/“ . Ovakav način komentarisanja je specifiačan za komentare koji se prostiru u jednom ili više linija i služe za detaljnije opise, npr funkcija i njenih parametara

Glavna namjena komentara je da podpomognu čitanje i razumjevanje koda programa

# Imenovanje

Za imenovanje klasa koriste se riječi engleskog jezika koje imaju smislia i opisuju klasu na pravi način. Obavezno počinju velikim slovom. Svaka naredna riječ u imenu počinje velikim slovom bez razdvajanja i dodatnih znakova.

Za imenovanje promjenjivih se koriste riječi koje opisuju promjenjivu i daju joj smisao i funkciju koja se lako može zapamtiti. Obavezno počinju malim slovom.Za pisanje imena koristi se engleski alfabet tako da nisu dozvoljena slova sa kvakicama. Pored slova dozvoljeni su i brojevi ( ali ne na početnoj poziciji) kao i specijalni karakteri „ – “ i „\_“ . Za razdvajanje više riječi u imenu koriste se specijalni karakteri ili konvencija da sledeća riječ počinje velikim slovom bez razmaka.

Za imenovanje metoda ( funkcija ) koriste se riječi koje smisleno opisuju metodu na način koji je razumljiv programeru ali i drugim čitaocima koda.. Ime metode se pise prvim malim slovom .Za pisanje imena koristi se engleski alfabet tako da nisu dozvoljena slova sa kvakicama. Pored slova dozvoljeni su i brojevi ( ali ne na početnoj poziciji) . Za razdvajanje više riječi u imenu koriste se konvencija da sledeća riječ počinje velikim slovom bez razmaka. Pisanje naziva argumenata u metodama je isto kao i pisanje promjenjivih u kodu.

Za imenovanje konstantnih veličina koriste se riječi koje opisuju konstantu i daju joj smisao i osobinu koja se lako može zapamtiti. Imena konstanti se uvjek pisu sa svim velikim slovima. Uvjek se deklarišu na početku klase u kojoj se koriste. Za pisanje imena koristi se engleski alfabet tako da nisu dozvoljena slova sa kvakicama.Ukoliko se naziv konstante nsastoji od više rijeci sva slova se pišu velika a razmak se pravi pomoći sprecijalnog karaktera „\_“ ( donje crte)

Za imenovanje interfejsa važe ista pravila kao i za imenovanje klasa.

# Deklaracija

Lokalne promjenjive se deklarišu na vrhu bloka koda, odnosno na početku posle vitičaste zagrade.

U jednoj liniji se nalazi deklaracija jedne lokalne promjenjive, a ne više promjenjivih.

Samo je brojače dozvoljeno dodatno deklarisati u drugim linijama koda (indeksa za for petlju, jer se kod Jave može deklarisati unutar for izraza )

Deklarisanje klase vrši se slijedećim redoslijedom:

* public member funkcije
* public polja
* protected member funkcije
* protected polja
* private member funkcije
* private polja

# Izrazi i naredbe

Svaka linija treba da sadrži najviše jedan izraz. Ukoliko je izraz komplikovan , radi lakseg razumjevanja moze se razdvojiti na više linija. Iskazi se unose na način kako ih je najlakše čitati. Najpoželjnije je da se razdvajaju uslovi, vitišaste zagrada (svaka da ide u novi red) i ostali elementi. Potrebno je i na odredjeni način vršiti uvlačenje dijela koda radi lakseg razimjevanja šta se nalazi unutar nekih finkcija.

# Memorija

"Upravljanje memorijom" izvještava o načinu na koji je došlo do alokacije i dealokacije tokom intervala korišćenja. Svi objekti koji su kreirani trebaju biti i upotrebljeni.U suprotnom, oni koji se ne upotrebljavaju se brišu iz programskog kodaPotrebno 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 i njihove maksimalnu dizine , a ne pretjerivati.

Koristiti „garbage collector” u Javi.

# Rukovanje greškama i izuzecima

Izuzetci se koriste isključivo za rukovanje logičkim i programskim greškama, konfiguracionim greškama, oštećenim podacima i iscrpljenim resursima.Izuzeci predstavljaju pojavu koja se može predvidjeti i obraditi na odgovarajući način, dok greške dovode do nasilnog prekidanja programa. 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. Try blok sadrži naredbe koje potencijalno mogu da sadrže greške. Ako dođe do greške, generiše se odgovarajuči izuzetak – exception, nakon čega se programski tok preusmjerava na odgovarajući catch blok koji sadrži kod za obradu generisanog izuzetka.Catch blokovi se pišu odmah iza try bloka. Može biti više catch blokova koji hvataju razne vrste izuzetaka.Catch handler je procedura koja može biti preopterećena – overloaded, tj. može biti više catch procedura koje imaju različite argumente.Različiti tipovi argumenata catch procedure odgovaraju različitim tipovima izuzetaka koji imaju zajednička svojstva – properties, npr. Message koje daje tekstualni opis greške.Ako nijedan od catch blokova ne sadrži odgovarajući tip argumenta za proizvedeni izuzetak u try bloku, onda se izlazi iz date procedure u spoljašnju proceduru koja je pozvala proceduru u kojoj je nastala greška, tj. koja je generisala izuzetakNa osnovu toga je smanjena mogućnost totalnog prestanka rada prograama. Potrebno je pružiti mogučnoct korisniku , da ukoliko je greška u unosu podataka, može da se ispravi. Izvještaje o nastalim greškama prikazati na ekran korisniku, sa mogućnostima dodatnog opisa greške.

# Prenosivost

Program radi na svim platformama na kojima je podržana Java. Microsoft Windows, MAC OS, Linux itd. Inače, program se ne kompajlira direktno u mašinski kod vec u takozvani bytecode koji se moze izvršavati na bilo kojoj Java virtualnoj mašini što Javi omogućava potpunu nezavisnost od platforme.

# Ponovna upotreba

Svaki paket, biblioteka Java klase treba biti potvrđena i korištenja kao 100% čista Java. Na ovaj način se osigurava rad na svim platformama. Ponovno korištenje koda dovodi do velike uštede vremena jer se postojeći moduli ne kodiraju iznova, već se obavlja prilagođavanje postojećih modula novim modelima i potrebama

# Problemi pri kompajliranju

Opisuje zavisnosti između modula, prevodilačkih direktiva, kao i svih problema koji se mogu javiti pri samoj upotrebi i izvršavanju zamišljenih dijelova projekta koji je planiran da se obradi. Svi problemi i greške koje se jave pogledati u listi grešaka i vidjeti koji tip greške se odnosi na koji dio u programskom kodu te isti i popraviti.