**

queri

Smjernice za programiranje

Verzija 1.2

Istorija revizija

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Verzija** | **Opis** | **Autor** |
| 31.12.2022. | 1.0 | Napisana inicijalna verzija dokumenta | Mladen Todorović |
| 01.01.2023. | 1.1 | Formatirane reference, dodan primjer za vitičaste zagrade | Mladen Todorović |
| 02.01.2023. | 1.2 | Formatiran izgled dokumenta. | Mladen Todorović |

Sadržaj

1. Uvod 5

1.1 Svrha 5

1.2 Područje primjene 5

1.3 Definicije, akronimi i skraćenice 5

1.4 Reference 5

1.5 Pregled 5

2. Organizacija i stil koda 5

2.1 Indentacija (uvlačenje) 5

2.2 Dužina linije 5

2.3 Razbijanje redova 5

3. Prazne linije i razmaci 6

3.1 Prazne linije 6

3.2 Razmaci 6

4. Komentari 6

4.1 Komentari dokumentacije 6

4.2 Komentari implementacije 7

4.3 Posebne oznake 7

5. Imenovanje 7

6. Deklaracije 7

7. Inicijalizacije 8

8. Izrazi i izjave 8

8.1 Izrazi 8

8.2 Izjave 9

8.2.1 Return izjave 9

8.2.2 If/else izjave 9

8.2.3 For izjave 9

8.2.4 While izjave 9

8.2.5 Do-while izjava 10

8.2.6 Switch izjava 10

8.2.7 Try-catch izjava 10

8.2.8 Pristup promjenljivim klasa/instanci 10

8.2.9 Referenciranje promjenljivih i metoda statičkih klasa 10

9. Upravljanje memorijom 10

10. Upravljanje greškama i izuzecima 10

11. Prenosivost 11

12. Ponovno korištenje 11

13. Problemi pri kompajliranju 11

14. Aneks: sažetak smjernica 11

Smjernice za programiranje

# Uvod

Dokument *Smjernice za programiranje* sadrži detaljno opisane standarde, konvencije i smjernice za pisanje koda u Java programskom jeziku, i namijenjen je za programere koji rade na razvoju softverskog alata *queri*.

## Svrha

Svrha dokumenta je da izloži konvencije programiranja kojima će se voditi programeri angažovani na izradi softverskog alata *queri*. Ovim se izbjegavaju sukobi stilova i omogućava se pisanje koda lakog za razumijevanje, održavanje i poboljšanje, te se produktivnost tima programera značajno povećava.

## Područje primjene

Dokument utiče na način kodiranja prototipa i finalnih verzija softverskog alata *queri.* Samim tim ograničava realizaciju na okruženje Eclipse, u programskom jeziku Java uz dodatak Swing seta alata za GUI.

## Definicije, akronimi i skraćenice

**JVM** - Java Virtual Machine predstavlja virtuelnu mašinu koja može izvršavati Java kompajlirani byte kod, tzv. Javin bytecode

Sve potrebne definicije, akronimi i skraćenice, u domenu problema koji *queri* rješava, sadržane su u dokumentu *Rječnik*, koji je dio projektne dokumentacije.

## Reference

1. [Code Conventions for the Java Programming Language](https://www.oracle.com/java/technologies/javase/codeconventions-introduction.html)
2. [Google Java Style Guide](https://google.github.io/styleguide/javaguide.html)
3. [Stackify blog](https://stackify.com/best-practices-exceptions-java)

## Pregled

U nastavku dokumenta *Smjernice za programiranje* opisana je organizacija i stil koda, kao i način pisanja komentara, imenovanja, deklaracija, izraza i izjava, upravljanja memorijom, upravljanja greškama i izuzecima, opis prenosivosti, ponovnog korištenja i potencijalnih problema tokom kompajliranja.

# Organizacija i stil koda

## Indentacija (uvlačenje)

Kao jedinicu uvlačenja treba koristiti 4 razmaka. Eclipse ima ugrađen formater koji se brine o pravilnom uvlačenju pojedinih sekcija koda.

## Dužina linije

Dužina linije treba biti manja od 80 karaktera, uz napomenu da bi komentari dokumentacije trebali biti kraći, do 70 karaktera po liniji.

## Razbijanje redova

Kada red ne može ispoštovati gornje pravilo, pisati u novi, vodeći se sljedećim pravilima:

* Razbiti poslije zareza.
* Razbiti prije operatora.
* Novu liniju poravnati sa početkom izraza u prošloj liniji.
* Poželjno je razbijati na višem sintaksičkom nivou (nakon zatvaranja zagrade, “.” separatora itd) .

Ako je kod još uvijek nejasan, onda ga uvući za 8 razmaka.

# Prazne linije i razmaci

## Prazne linije

Prazne linije poboljšavaju čitljivost koda izdvajanjem dijelova koda koji su logički povezani.

Dvije prazne linije koristiti:

* između sekcija source file-a
* između definicija klasa i interfejsa

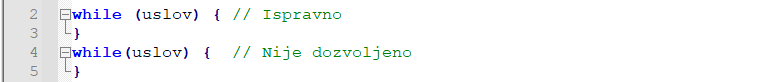
Jednu praznu liniju koristiti:

* između lokalne varijable u metodi i njene prve izjave
* prije blokovskog i jednolinijskog komentara
* između logičkih cjelina unutar metode
* između uzastopnih inicijalizatora (grupa) klase: polja, konstruktori, metode, ugnježdene klase, statički inicijalizatori.

## Razmaci

Razmake koristiti:

* između rezervisane riječi i zagrade “(“



* sa obje strane bilo kog binarnog ili ternarnog operatora, izuzev dva slučaja:
  + kod operatora dvije dvotačke “::” za referenciranje statičke metode, metode instance ili konstruktora



* + kod korištenja tačke “.” za “dohvatanje” polja/metoda klase/instance



* poslije zareza “,”, dvotačke “:”, tačke sa zarezom “;” i zatvarajuće zagrade “)”
* sa obje strane dvostruke kose crte “//”, koja započinje komentar na kraju reda. Ovdje je dozvoljeno i više razmaka, da bi se poboljšala čitljivost koda
* između tipa i naziva podatka u deklaraciji

Razmake ne koristiti između naziva metode i otvorene zagrade “(”, za listu parametara. Takođe, unarne operatore, inkrement i dekrement, ne treba odvajati razmakom od operanda.

# Komentari

Pisanje komentara uz kod predstavlja dobru praksu. Iako se nastoji da sam kod bude pregledan i lako razumljiv, određeni dijelovi koda su pogodni za dodatno pojašnjenje. Ta pojašnjenja mogu biti “zlata vrijedna”, kada različiti ljudi vrše određene izmjene u kodu, ali i prilikom održavanja cijelog softvera poslije nekog dužeg vremena. Kvalitetni komentari će značajno poboljšati kod, ali učestalost komentara može da odražava loš kvalitet samog koda. Kada se osjeti potreba za pisanjem komentara, poželjno je razmisliti o izmjenama u kodu kako bi postao jasniji.

## Komentari dokumentacije

Netrivijalne metode, klase, interfejse je poželjno opisati komentarima dokumentacije. Opisati jasno ulaz i izlaz (gdje je to primjenljivo), srpskim jezikom. Ovaj tip komentara ima za cilj da opiše specifikaciju koda iz perspektive bez implementacije. Okruženje Eclipse omogućava generisanje JavaDoc (Java dokumentacije). Počinju sa “/\*\*” i završavaju se sa “\*/”.

## Komentari implementacije

Komentare implementacije koristiti da bi svrha koda bila jasna na prvi pogled, ili da se pruži informacija koja nije prisutna ili lako razumljiva u samom kodu. Komentare je potrebno na odgovarajući način odvojiti od samog koda. Ne koristiti kosu crtu unutar komentara implementacije. U Eclipse okruženju razlikujemo:

* jednolinijske komentare: počinju sa “//” i idu do kraja linije koda,
* višelinijske komentare počinju sa “/\*” i završavaju se sa “\*/”.

## Posebne oznake

Koristiti oznaku “XXX” u komentaru za dio koda koji nije adekvatan, ali radi.

Koristiti oznaku “POPRAVITI” u komentaru za dio koda na kojem “puca” program.

# Imenovanje

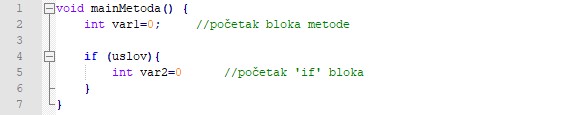
| **Tip identifikatora** | **Pravila imenovanja** | **Primjeri** |
| --- | --- | --- |
| Paketi | Prefiks unikatnog imena paketa se uvijek piše svim malim ASCII slovima. Koriste se obrnuti domeni. U slucaju da postoje ključne riječi sadržane u imenu domene, ili ime počinje brojem, tada se ispred ubacuje znak “ ”. | com.sun.eng  com.apple.quicktime.v2  edu.cmu.cs.bovik.cheese |
| Klase | Klasna imena trebaju biti imenice, sa prvim slovom svake riječi velikim. Koristiti čitave riječi, a ne skraćenice (osim ako su skraćenice poznatije kao takve npr. URL ili HTML). Svaka sljedeća riječ počinje velikim slovom. | class Raster; class ImageSprite; |
| Interfejsi | Imena interfejsa treba pisati kao i imena klasa. | interface RasterDelegate; interface Storing; |
| Metode | Metode moraju biti glagoli, sa malim početnim slovom prve riječi, a sa velikim početnim slovom ostalih riječi. | run(); runFast(); getBackground(); |
| Promjenljive | Promjenljive pisati sa istim pravilima kao i metode.  Imena trebaju biti kratka ali značajna, tj. ono što promjenljiva predstavlja treba biti jasno na prvi pogled. Imena dužine jednog karaktera izbjegavati, osim kod privremenih promjenljivih (npr. brojača). | int i;  char c;  float myWidth; |
| Konstante | Imena promjenljivih deklarisanih kao klasne konstante pišu se sa svim velikim slovima i riječima odvojenim donjom crtom (“\_”). | static final int MIN\_WIDTH = 4;  static final int MAX\_WIDTH = 999;  static final int GET\_THE\_CPU = 1; |

# Deklaracije

Pisati jednu deklaraciju po liniji radi lakšeg pisanja komentara.

Snimak ekrana 2022-05-05 214523

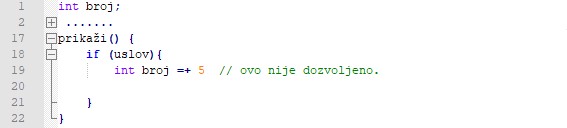
Deklaracija se piše na početku bloka koda.



Jedini izuzetak ovog pravila je deklaracija brojača u for petlji.

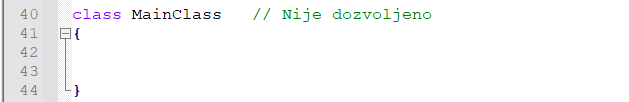
Snimak ekrana 2022-05-05 215822

Zabranjene su lokalne deklaracije koje skrivaju globalne.



Prilikom deklarisanja klasa, interfejsa i metoda otvorena vitičasta zagrada “{“ se nalazi u istoj liniji odmah poslije deklaracije. Zatvorena vitičasta zagrada “}” stoji sama na kraju klase, osim ako je tijelo prazno, onda se piše odmah poslije otvorene vitičaste zagrade “{“.





# Inicijalizacije

Inicijalizovati lokalne promjenljive odmah nakon deklaracije. Jedini razlog, da se promjenljiva ne inicijalizuje tamo gdje je deklarisana, je kada početna vrijednost zavisi od nekog izračunavanja koje se prvo dogodi.

# Izrazi i izjave

Izrazi izvršavaju operacije nad podacima i premiještaju podatke. Izjave predstavljaju osnovnu jedinicu izvršenja.

## Izrazi

U izrazima se mora voditi računa da ne bude dvosmiselnih dodjela, tj. da ne dolazi do eksplicitne konverzije podataka. Ukoliko se radi o složenim izrazima potrebno je korisiti zagrade radi lakšeg razumijevanja koda.



## Izjave

Svaka linija može imati najviše jednu izjavu.

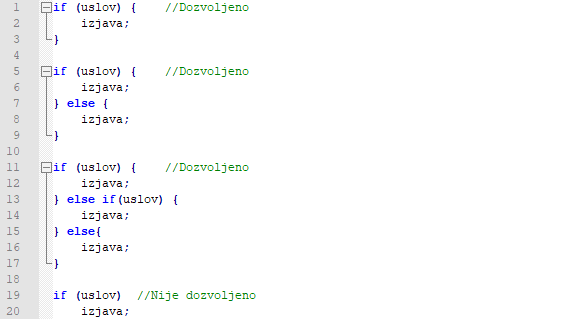
Snimak ekrana 2022-05-06 115246

### Return izjave

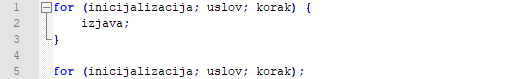
Return izjave ne bi trebale da se pišu sa zagradama, osim ako se radi o složenim izrazima.

Snimak ekrana 2022-05-06 121042

### If/else izjave



### For izjave



Operator zarez u inicijalizaciji ili ažuriranju vrijednosti promjenljivih koristiti sa najviše tri promjenljive. Ako se javlja potreba za više promjenljivih, usložnjavanje koda izbjeći izmještanjem inicijalizacije prije for petlje, i ažuriranja na kraj petlje.

### While izjave

While izjava ima sljedeću formu:

Snimak ekrana 2022-05-06 151450

Prazna while izjava:



### Do-while izjava

Do-while izjava ima sljedeću formu:



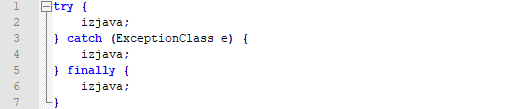
### Switch izjava

Switch izjava ima sljedeću formu:



### Try-catch izjava

Try-catch izjava ima sljedeću formu:

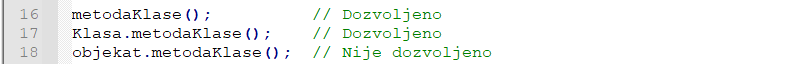


### Pristup promjenljivim klasa/instanci

Promjenljive klasa/instanci ne smiju biti javne (public) bez jakog razloga. Često, promjenljive instanci se mogu dobiti kao bočni efekti u pozivima metoda, pa se ne moraju eksplicitno postavljati ili dohvatiti. Primjer javne promjenljive instance je u slučaju gdje je klasa esencijalno struktura podataka, bez ponašanja.

### Referenciranje promjenljivih i metoda statičkih klasa

Nije dozvoljeno korištenje objekta za pristup promjenljivim ili metodama statičkih klasa. Koristiti ime klase.



# Upravljanje memorijom

Koristi se sistem za automatsko upravljanje memorijom koji se naziva *garbage collector*, koji radi u pozadini i briše neiskorištene objekte.

# Upravljanje greškama i izuzecima

* Prilikom kreiranja novog tipa izuzetka potrebno ga je dokumentovati.
* Potrebno je čuvati svaku grešku i izuzetak u log file.
* Prvo se hvataju specifični, pa uopšteni izuzeci.
* Prilikom bacanja izuzetaka potrebno je opisati razlog nastajanja greške korisniku.
* Ukoliko se desi izuzetak potrebno je omogućiti nesmetan rad korisnika.

# Prenosivost

Softverski alat *queri* može da radi na svakom računarskom sistemu koji posjeduje JVM. Izvršni kod se ne mora mijenjati da bi zadovoljio posebne potrebe računarskog sistema.

# Ponovno korištenje

* Pridržavati se principa modularnosti.
* Moduli moraju biti slabo spregnuti.
* Klase i metode trebaju da obavljaju jednu funkciju.
* Koristiti enkapsulaciju.
* Praviti testove za klase i učiniti da su klase jednostavne za testiranje.
* Ne ponavljati poslovni kod, jer je moguće da će biti mijenjan i nekompatibilan s ostatkom koda.

# Problemi pri kompajliranju

Potrebno je posjedovati sve potrebno za kompajliranje Java koda što je opisano u dijelu prenosivost. Koristiti alate za upravljanje zavisnostima između modula (Maven je preporučen). Obrisati neiskorištene zavisnosti i komentarisati importe zavisnosti. Koristiti poruke u kodu u slučaju nastanka grešaka pri kompaljiranju.

# Aneks: sažetak smjernica

* Organizacija i stil koda – opisan izgled i formatiranje koda radi lakše čitljivosti.
* Komentari – opisano kada i kako komentarisati i dokumentovati kod.
* Imenovanje – data tabela s pravilima imenovanja elemenata kojima se programer služi.
* Deklaracije – opisan način deklarisanja promjenljivih.
* Inicijalizacije – opisan način inicijalizovanja promjenljivih.
* Izrazi i izjave – opisan način pisanja petlji, matematičkih i logičkih izraza programskog koda.
* Upravljanje memorijom – opisan rad garbage collector-a.
* Upravljanje greškama i izuzecima – opisano rješavanje mogućih grešaka pomoću try-catch izraza.
* Prenosivost – opisani uslovi i mogućnosti prenosa softverskog alata *queri* na druge platforme.
* Ponovno korištenje – opisani principi dobre prakse koji omogućuju da kod bude ponovo iskoristiv u budućim projektima ili interno.
* Problemi pri kompajliranju – opisane modulne zavisnosti u cilju izbjegavanja problema pri kompajliranju.