Grupa1

LightsOn
Smjernice za programiranje

Verzija 1.0

Istorija revizija

Datum	Version	Opis	Autor
10.12.2024	1.0	Izrada dokumenta Smjernice za programiranje.	Dušan Marilović

Sadržaj

Smjernice za programiranje

1. Uvod

Dokument "Smjernice za programiranje" opisuje skup standarda, konvencija i smjernica za pisanje čitljivog Java koda. Ove smjernice se zasnivaju na čvrstim, dokazanim principima softverskog inženjerstva koji dovode do koda koji je lako razumjeti, održavati i poboljšati.

1.1 Svrha

Svrha dokumenta "Smjernice za programiranje" je da uspostavi skup pravila prema kojima je potrebno kodirati softver *LightsOn*. Budući da na projektu radi više osoba, cilj ovog dokumenta je da definiše jasna pravila kodiranja kako bi se izbegli sukobi različitih stilova.

1.2 Područje

Dokument "Smjernice za programiranje" direktno utiče na način kodiranja *LightsOn* softvera. Projekat se realizuje u integrisanom razvojnom okruženju Eclipse, koristeći programski jezik Java, uz dodatak Swing seta alata za GUI.

1.3 Pojmovi, akronimi i skraćenice

Svi korišteni, nepoznati pojmovi, akronimi i skraćenice su opisani u dokumentu Rječnik.

1.4 Reference

[1] Code Conventions for the Java TM Programming Language (https://www.oracle.com/java/technologies/javase/codeconventions-contents.html)
[2] Google Java Style Guide *(https://google.github.io/styleguide/javaguide.html)*
[3] Code Complete 2nd Edition, Steve McConnell, Microsoft Press 2004
[4] Programski jezik JAVA sa rešenim zadacima, Laslo Kraus, AKADEMSKA MISAO, Beograd 2013

1.5 Pregled

U nastavku dokumenta, opisana je organizacija i stil koda, kao i način pisanja komentara, imenovanja, deklaracija, izraza i izjava. Takođe, dokument pokriva upravljanje memorijom, upravljanje greškama i izuzecima, prenosivost, ponovno korišćenje, i potencijalne probleme pri kompajliranju.

2. Organizacija i stil koda

U ovom dijelu opisane su tehnike kodiranja koje čine kod čitljiviji.

2.1 Pragraf ili uvlačenje koda

Kao jedinicu uvlačenja koristite se 4 razmaka. Može se koristiti i jedan tab. Eclipse ima ugradjen formater koji se brine o pravilnom uvlačenju pojedinih sekcija koda.

2.2 Razbijanje redova

Ukoliko izraz ne može stati u jedan red, treba ga razbiti prema sljedećim pravilima:

- Razbijanje nakon zareza,
- Razbijanje nakon operatora,
- Poravnanje nove linije sa početkom izraza na istom nivou kao prethodna linija,
- Ukoliko prethodna pravila generišu vizuelno loše formatiran kod, koristiti uvlačenje od 8 razmaka.

2.3 Upotreba razmaka i praznih linija

- Jedan razmak između ključnih riječi (if, while, for, switch, ...) i lijeve zagrade.
- Jedan razmak i sa lijeve i sa desne strane binarnih operatora.
- Jedan razmak poslije zareza f(a, b).
- Praznom linijom odvojiti cjeline koda.

2.4 Zagrade

Male zagrade se koriste u izrazima sa mnogo operatora da bi se izbjegli problemi prioriteta.

2.5 Pisanje kratkih pojedinačnih linija

- Izbjegavati linije duze od 80 karaktera. Izuzetak ovog pravila je pri upotrebi podataka koji se ne smiju razbijati (npr. URL).
- Nije dozvoljeno kodiranje više izraza u jednoj liniji.

3. Komentari

Tip komentara	Pravila korištenja	Primjer
Dokumentacija	Dokumentacijski komentari se koriste prije deklaracija interfejsa, klasa i funkcija radi njihovog dokumentovanja. Nakon dokumentacijskih komentara generiše se *javadoc.*	/** * Primjer klasa koja ne radi nista, a sadrzi jednu metodu i jedan atribut. * * @author D.M * @author A.M * @version 2.0 13.11.2024. */ **public class MojaKlasa**
C tip	Ovu vrstu komentara koristiti ukoliko postoji deo koda koji više nije prihvatljiv, ali koji treba da bude saučuva.	/* Ovaj kod je zakomentarisan ...(Komentarisani kod) */
Jednolinijski	Koristiti jednolinijske komentare u unutrašnjosti funkcija radi dokumentovanja korištene logike i cjelina u kodu. Ne koristiti komentare na dijelu koda koji je već precizno jasan.	// -----Sortiranje----- // dodati …..

4. Imenovanje

Tip identifikatora	Pravila imenovanja	Primjeri
Paketi	Imena paketa se pišu malim slovima a uzastopne riječi su jednostavno povezane sa tačkom. Nije dozvoljeno povezivati riječi sa donjom crtom ili kombinacijom malih i velikih slova. U slučaju da postoje	com.podaci.citanje int_.primjer com.primjer._123ime

	ključne riječi sadržane u imenu domene ili ako riječ počinje sa brojem, tada se ispred ključne riječi ubacuje znak _.	
Klase	Imena klasa su imenice, miješano veliko i malo slovo, sa prvim velikim slovom svake unutrašnje riječi. Imena klasa treba da budu jednostavna. Koristiti čitave riječi - a izbjegavati akronime, ukoliko akronim nije široko poznat (HTML, URL...).	class Registar; class MojaKlasa:
Interfejsi	Imena interfejsa se pišu na isti način i klase.	interface Racunanje; interface Smjestanje;

Metode	Metode treba da budu glagoli, prva riječ malim slovima, a ostale riječi sa prvim velikim slovom.	pokreni();
		sortirajNiz();
		dohvatiPozafinu();

Varijable	Prva riječ kod imena varijabli piše se malim slovom, a unutrašnje riječi sa velikim prvim slovom. Imena varijabli treba da budu kratka ali sa značenjem. Izbor imena varijable treba biti mnemonik koji ukazuje na njeno značenje. Varijable koje se sastoje od jednog karaktera treba izbjegavati, osim ako su u pitanju privremene varijable. Standardna imena za privremene varijable su i, j, k, m i n za int varijable; c, d i e za char varijable.	int i; float brzinaKretanja;
Konstante	Imena konstanti pišu se velikim slovima sa riječima odvojenim donjom crtom ("_").	int MIN_DUZINA = 4; int MAX_DUZINA = 999;

5. Deklaracije

5.1 Broj deklaracija po liniji

Koristi se jedna deklaracija po liniji.

```
1  if  (a == b && c == d)
2  if  ((a == b) && (c == d )
3
```

Izbjegavati deklaraciju varijabli i metoda na istoj liniji.

```
long x, y;
int x, niz[];
```

5.2 Pozicija

Stavljati deklaracije samo na početku blokova koda, odnosno na početku unutar vitičastih zagrada “{“i”}“. Izbjegavati deklarisanje varijabli „na zahtjev“, odnosno onda kada su neposredno potrebne.

```
public void metoda()
{
    int x;

    if (uslov)
    {
        int y:
        ...
    }
}
```

Izuzetak je kod indeksa za for petlju.

```
for (int i = 0; i < duzina; i++)
{
    ....
}
```

5.3 Inicijalizacija

Pravilo je da se lokalne varijable inicijalizuju gdje su deklarisane. Treba izbjegavati u slučaju ako inicijalna vrijednost zavisi od nekih prethodnih izračunavanja koje uzimaju vrijeme i prostor. Tako, na primjer, treba izbjegavati instanciranje objekata do onoga trenutka kada su zaista potrebni.

5.4 Deklaracija klasa i interfejsa

- Nema razmaka između imena metode i lijeve zagrade “(”
- Lijeva vitičasta zagrada “{”se nalazi na kraju iste linije kao izjava deklaracije
- Desna vitičasta zagrada “}”se nalazi sama u liniji, osim ako je blok prazan onda ta zagrada treba da se pojavi odmah nakon “{”

```
class Klasa {
    int x;
    int y;

    primjer (int i, int j) {
        x = i;
        y = j;
    }

    int metoda () {}

    ...
}
```

6. Izrazi i izjave

6.1 Izrazi

Izrazi se pišu nedvosmisleno i precizno. Ukoliko se radi o složenim izrazima potrebno je koristiti

zagrade radi razjašnjenja koda.

```
x  + y / 100 //Nejasno
(x + y) / 100  //Jasno
x + (y / 100)   //Jasno
```

6.2 Jednostavne izjave

Svaka linija treba da sadrži jednu izjavu.

```
argv++; //Pravilno
argc++; //Pravilno
argv++;   argc++;  //Izbjegavati
```

6.3 Složene izjave

- Ograđene izjave trebaju biti uvučene za jedan nivo više od složenih izjava
- Lijeva vitičasta zagrada treba biti na kraju reda koji započinje složeni izraz; desna vitičasta zagrada treba započeti liniju i biti uvučena do početka složeog izraza.
- Zagrade se koriste oko svih izraza, čak i pojedinačnih izraza, kada su deo kontrolne strukture, kao što je if-else ili for naredba.

6.4 Return izjave

Return izjava sa vrijednosti ne bi trebala da koristi zagrade osim ako na neki način povratnu vrijednost čine očiglednijom.

6.5 If, if-else, if-else-if else izjave

If, if-else, if-else-if else izjave treba da imaju sledeću formu.

```
if (uslov)
{
    izjave;
}


if (uslov)
{
    izjave;
}
else
{
    izjave;
}


if (uslov)
{
    izjave;
}
else if (uslov)
{
    izjave;
}
else
{
    izjave;
}
```

6.6 For petlje

For petlje treba da imaju sledeću formu:

```
for (inicjalizacija; uslov; korak)
{
    izjave;
}
```

Kada se koristi operator zarez u inicijalizaciji ili koraku u for izjavi, izbjegavati komplikovanje korišćenjem više od 3 varijable. Ako je potrebno, deklarisati izjave prije for petlje (za inisjalizaciju) ili na kraju petlje (za korak).

6.7 While petlje

While petlje treba da imaju slijedeću formu:

```
while (uslov)
{
    izjave;
}
```

6.8 Do-while petlje

Do while petlje treba da imaju slijedeću formu:

```
do
{
    izjave;
}
while (uslov);
```

6.9 Switch izjave

Switch izjave treba da imaju slijedeću formu:

```
switch  (uslov)
{
    case ABC:
        izjave;
        break;
    case DEF:
        izjave;
        break;
    default:
        izjave;
        break;
}
```

6.10 Try-catch izjave

Try-catch izjave treba da imaju slijedeću formu:

```
try
{
    izjave;
}
catch (ExceptionClass e)
{
    izjave;
}
```

7. Upravljanje memorijom

Java posjeduje automatski menadžment memorijom, *Garbage collector* koji radi u pozadini i briše neiskorištene objekte i čisti memoriju. Stoga, nije potrebno razmišljati o uništavanju objekata i kako utiču na memoriju.

8. Upravljanje greškama i izuzecima

- Koristiti izuzetke za upravljanje sa logičkim i programerskim greškama, konfiguracijskim

greškama, oštećenim podacima i iscrpljivanjem resursa.
- Ne koristiti izuzetke za česte i predvidive događaje.
- Omogućiti nesmetan rad programa
- Ako postoje catch blokovi sa nekoliko tipova izuzetaka, potrebno je blokove postaviti tako da se prvo hvata izuzetak najniže podklase pa redom prema navišoj podklasi.
- Na kraju try-catch uvjek staviti blok finally.
- Ukoliko se pravi poseban izuzetak potrebno ga je dokumentovati.

9. Prenosivost

Java programi su prenosivi i mogu da rade na bilo kom računarskom sistemu koji posjeduje Java interpreter. Izvorni kod se može kompajlirati na bilo kom računaru koji ima Java kompajler. Izvršni kod se ne mora mjenjati da bi zadovoljio posebne potrebe računarsko sistema.

10. Ponovno korištenje

Svaka biblioteka ili paket Java klase koji se kupuje ili koristi iz spoljnog izvora mora biti sertifikovana kao 100% čista Java. Neke od smjernica da se se kod učini pogdnijim za ponovno korištene su:
- Ne dopustiti da se kod ponavlja.
- Napraviti da klase i metode rade jednu stvar.
- Napraviti jedinstvene testove za klase i učiniti da su klase lake za testiranje
- Ukloniti poslovnu logiku ili “main” kod iz svakog frejmvork koda
- Što više koristiti apstrakciju, u smislu interfejsa i apstraktnih klasa
- Napraviti kod koji će se lakše proširiti u budućnosti
- Ne pisati kod koji nije potreban

11. Problemi pri kompajliranju

- Potrebno je otkloniti sve sintaksne greške da bi program mogao da se kompajlira.
- Potrebno je posjedovati sve potrebno za kompajliranje Java koda što je opisano u dijelu prenosivost.
- Koristiti poruke u kodu u slučaju nastanka greški pri kompaliranju.