Grupa1

SIMBL

Smjernice za programiranje

Verzija 1.0.1

Istorija revizija

Datum	Verzija	Opis	Autor
22.05.2024	1.0.1	Uradjen čitav dokument	Dušan Marilović

Sadrzaj

Smjernice za programiranje

1. Uvod

Ovaj dokument opisuje smjernice za programiranje, odnosno pisanje Java koda, i namijenjen je za programere koji rade na razvoju grafičkog BPMN editora SIMBL.

1.1 Svrha

Svrha dokumenta je prikaz smijernica za programere, tj konvencije programiranja kojima će se voditi programeri koji rade na projektu. Ovim se izbjegavaju sukobi stilova i omogućava se jasan i čitljiv kod.

1.2 Područje primjene

Dokument Smjernice direktno utiče na kodiranje i realizaciju softverskog alata SIMBL u editor Eclipse, programskom jeziku JAVA (paket Swing).

1.3 Definicije, akronimi i skraćenice

Sve definicije, akronimi i skraćenice, korištene u dokumentu, su opisane u dokumentu Rječnik, koji je dio projektne dokumentacije.

1.4 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)

1.5 Pregled

U nastavku dokumenta Smjernice za programiranje prikazan je način organizacije i stila kodiranja. Sva pravila referencijaru se na JAVA standarde kodiranja.

2. Organizacija i stil koda

2.1 Indentacija (uvlačenje)

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

2.2 Dužina linije

Zbog citljivosti trebalo bi izbjegavati redove duze od 80 karaktera. Za komentare treba da vazi slično pravilo, oni bi trebali biti ograničeni na 70 karaktera po liniji.

2.3 Razbijanje redova

Kada nije moguce ispostavati gore navedeno pravilo, razbijanje se vrši po sledećim pravilima:

- Razbijanje nakon zareza
- Razbijanje prije operatora
- Razbijanje nakon operatora,
- Novu liniju poravnati sa početkom izraza u prošloj liniji
- Ukoliko prethodna pravila generišu vizuelno loše formatiran kod, koristiti uvlačenje od 8 razmaka.

3. 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.

3.1 Komentari dokumentacije

Netrivijalne metode, klase, interfejse opisati komentarima dokumentacije. Opisati jasno ulaz izlaz (gdje je to primjenljivo), srpskim jezikom.

3.2 Komentari implementacije

Komentare implementacije koristiti da bi svrha koda bila jasna, ili da pruži dodatne informacije koje nisu prisutne ili lako razumljive. Potrebno je naravno, komentare 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 "*/".

4. Imenovanje

Tip identifikatora	Pravila imenovanja	Primjeri
Paketi	Prefiks unikatnog imena paketa se uvijek piše svim malim 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.example.mypackage javax.swing com.sun.eng
Klase	Imena klasa predstavljaju imenice. Na pocetku je prvo slovo veliko, kao i kod svake sledeće unutrašnje riječi. Imena bi trebala da budu jedostavna, tako da je pozeljno koristiti čitave riječi.	class Raster; class ImageSprite;
Interfejsi	Imena interfejsa treba pisati kao i imena klasa.	interface RasterDelegate; interface Storing;

Metode	Metode treba da budu glagoli, prva riječ malim slovima, a ostale riječi sa prvim velikim slovom.	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	`int i;` `float myWidth;`

	jednog karaktera izbjegavati, osim kod privremenih promjenljivih (npr. brojača).	

Konstante	Imena promjenljivih deklarisanih kao klasne konstante pišu se sa svim velikim slovima i riječima odvojenim donjom crtom (“_”).	`int MIN_WIDTH = 4;`
		`int MAX_WIDTH = 999;`
		`int GET_THE_CPU = 1;`

5. Deklaracije

Jedna deklaracija po liniji radi lakšeg pisanja komentara.

```
1  float cijena; //Cijena nekog proizvoda
2  int brojGodinaStaza; //Broj godina radnog staza nekog radnika
```

Deklaracija se piše na početku bloka koda.

```
1  void MainMethod() {
2        int i = 0;       //Početak bloka metode
3        int j = 0;
4
5        if(i == 0){
6            j = 1;       //Početak if bloka
7        }
8  }
9
```

Zabranjeno je koristiti deklaraciju u više linija, osim u slučaju da su podaci usko povezani, tj da su podaci međusobno zavisni.

```
1  int brojDijagrama, velicina; //Nije dozvoljeno
2  int niz[], duzina; // Dozvoljeno
3  int matrica[][], brojKolona, brojRedova;  //Dozvoljeno
```

Jedini izuzetak ovog pravila je za for petlju.

```
1  for(int i = 0; i < duzina; i++){
2       suma += 1;
3  }
```

Zabranjeno je lokalne deklaracije sakrivaju globalne.

```
int broj = 0;

prikazi(){
    if(uslov){
        int broj += 5; // Ovo nije dozvoljeno
    }
}
```

Deklaracija klasa i interfejsa se započinje otvorenom vitičastom zagradom '{' odmah posle njene deklaracije. Zatvorena vitičasta '}' zagrada stoji sama u liniji na kraju klase, osim u slučaju kada u tijelu klase nema ništa. Onda ona stoji odmah posle '{'.

```
class MainClass{



}

class Student {}
```

6. Izrazi i izjave

U izrazima se mora voditi računa da bude nedvosmislena dodjela, tj da ne dolazi do eksplicitnog pretvaranja podatka.

6.1 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.

```
x + y / z // Nije dozvoljeno
(x + y) / z //Dozvoljeno
x + (y / z) //Dozvoljeno
```

6.2 Izjave

Svaka linija može imati najviše jednu izjavu.

```
i++; // Dozvoljeno
j++; k--; // Nije dozvoljeno
```

6.2.1 Return izjave

Return izjave ne bi trebale da koriste zagradu osim u slučaju da se radi o nekom kompleksnom izrazu.

```
return;
return myDisk.size();
return (opcija ? x : y);
```

6.2.2 *If/Else izjave*

```
if(uslov){ //Dozvoljeno
    izjava;
}

if(uslov){ //Dozvoljeno
    izjava;
}
else{
    izjava;
}

if(uslov){ //Dozvoljeno
    izjava;
}
else if(uslov){
    izjava;
}
else{
    izjava;
}

if(uslov) //Nije dozvoljeno
    izjava;
```

6.2.3 *For izjave*

```
for(Inicijalizacija; Uslov; Korak){
    izjava;
}

for(Inicijalizacija; Uslov; Korak)
```

Kada imamo više od tri izjave u inicijalizaciji ili u koraku, onda je potrebno inicijalizovati varijable prije for petlje ili na kraju petlje.

6.2.4 *While izjave*

While izjava ima sljedeću formu:

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

6.2.5 *Do-while izjava*

Do-while izjava ima sledeći oblik:

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

6.2.6 *Switch izjava*

Switch izjava ima sledeću formu:

```
switch (opcija){
    case 1:
        izjava;
        break;
    case 2:
        izjava;
        break;
    default:
        izjava;
        break;
}
```

6.2.7 *Try-catch izjava*

Try-catch izjava ima sledeću formu

```
try{
    izjava;
}
catch(ExceptionClass e){
    izjava;
}
finally{
    izjava;
}
```

7. Upravljanje memorijom

Programski jezik JAVA automatski upravlja memorijom. Koristi sistem za automatsko upravljanje memorijom koji se naziva garbage collector. Od programera se ne zahtjeva da primjeni logiku upravljanja memorijom u aplikaciji.

8. Upravljanje greškama I izuzecima

Greškama se treba rukovati tako da korisnik dobije obavještenje da ne radi funkcionalnost na adekvatan način, ali da se ne ometa funckionalnost programa. Sa tim u vidu, mozemo da izdvojimo:

- Try i catch blok se završava sa finally blokom.
- 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.

9. Prenosivost

Da bi se ostvarila mogućnost upotrebe grafičkog BPMN editora SIMBL na drugim računarima nije potrebno instalirati nikakve kompajlere niti dodatke. Kod napisan u JAVA programskom jeziku se može izvršiti na bilo kojoj Java Virtual Machine (JVM) što Javi pruža nezavisnost od platforme na kojoj se izvršava i dozvoljava programerima da “write once, run anywhere”.

10. 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.

11. Dodatak: Rezime smjernica

- Uvod predstavlja document i prikazuje ukratko njegovu svrhu, reference i područje.
- Organizacija i stil koda prikazuju pravila organizacije i prikaz stila po kome će kod biti radjen.
- Komentari – predstavljeni komentari koji će se koristiti u kodu prilikom implementacije, i u dokumentaciji.
- Imenovanja – način po kome će se imenovati elementi koje korisnik deklariše u kodu.
- Deklaracija predstavlja pravilan način deklarisanja varijabli, klasa i interfejsa u kodu.
- Izrazi i izjave detaljno prikazuju kako će se prikazivati izrazi i izjave u cilju nedvosmislenog i čitljivog koda.
- Upravljanje memorijom programeru definiše kako će voditi računa o alokaciji i delokaciji memorije.
- Rukovanje greškama i izuzecima – prikazuje nekolicinu pravila kojih se programer mora pridržavati prilikom rukovanja greškama i izuzecima.
- Prenosivost – pravila koja definišu kako se kod može prenositi na druge uređaje.
- Ponovno korištenje – pravila koja drže kod spremnim i olakšanim za ponovno korištenje.
- Problemi pri kompajliranju – prikazuje kako postupati ako dođe do njih.