Grupa 2

SculptER
Smjernice za programiranje

Verzija 1.0

Istorija revizija

Date	Version	Description	Author
10.05.2024	0.1	Uvod dokumenta	Risto Berjan
10.05.2024	0.2	Organizacija	Risto Berjan
12.05.2024	0.3	Imenovanje, komentari	Risto Berjan
17.05.2024	0.4	Dodavanje 5. stavke i formatiranje	Aleksej Mutić
17.05.2024	0.5	Dodavanje 6. stavke	Aleksej Mutić
18.05.2024	0.6	Kompletiranje dokumenta(još nije formatiran tekst, moguće manje promjene)	Aleksej Mutić
25.07.2024.	0.7	Postavljanje logoa	Nikola Savić
11.08.2024.	0.8	Zamena logo-a	Nikola Savić
02.09.2024	1.0	Završetak dokumenta	Aleksej Mutić

Sadržaj

Smjernice za programiranje

1. Uvod

Dokument Smjernice za programiranje sadrži u sebi pravila koja se odnose na kodiranje, kao i skup smjernica i standarda za pisanje Java koda. Namijenjen je za programere koji su saradnici na razvoju projekta SculptEr.

1.1 Svrha

Svrha ovog dokumenta je u suštini prikaz smjernica i pravila, koje će poslužiti za kodiranje projekta SculptEr od strane programera.Svrha ovog dokumeta je takođe i prikazivanje stila koda,radi olakšavanja koda drugim saradnicima na projektu. Pravila koda su veome važna, između ostalog jer poboljšavaju čitljivost softvera tj. izvornog koda, pa je samim tim shvatanje koda razumljivije i brže.

1.2 Područje

Dokument Smjernice za programiranje direktno utiče na način kodiranja i realizaciju projekta SculptEr.Realizacija ovog projekta će se vršiti u razvojnom okruženju Eclipse, u programskom jeziku Java.Kao dodatak će se koristiti Swing set alata, za GUI.

1.3 Definicija, akronimi i skraćenice

Sve definicije, akronimi i skraćenice, koji su korišteni u ovom dokumentu, nalaze se u dokumentu Riječnik.

1.4 Reference

[1] Rječnik
[2] Java coding standard

1.5 Pregled

U nastavku dokumenta Smjernice za programiranje prikazan je stil kodiranja, kao i način organizacije. Sva pravila i smjernice se odnose na Java standarde kodiranja. U daljem dijelu ćemo vidjeti pravila komentarisanja, imenovanja, deklaracije izjava i izraza, upravljanje memorijom, greškama i izuzecima, prenosivnost, ponovno korištenje i moguće probleme koji se javljaju pri kompajliranje.

2. Organizacija i stil koda

U ovom djelu su prikazane tehnike kodiranja koje će učiniti kod razumljivijim. To su:

2.1 Dužina redova

Zbog čitljivosti i ljepšeg izgleda, ne bi trebalo koristiti linije duže od 80 karaktera. Komentari, koje koristimo u dokumentaciji, ne bi trebali biti duži od 70 karaktera po liniji.

2.2 Paragraf(Uvlačenje koda)

Kao jedinicu uvlačenja ćemo koristiti 4 razmaka. Možemo koristiti i jedan tab, ali omjer tab i 4 razmaka nije 1 prema 1. Eclipse IDE ima ugrađen formater koji je zadužen za pravilno uvlačenje pojedinih sekcija koda.

2.3 Upotreba razmaka i praznih linija

Neka od pravila za upotrebu razmaka i praznih linija su:

-Prilikom pisanja metode, nije dozvoljena upotreba razmaka izmedju imena metode i lijeve zagrade(npr. F(x)-pravilno, F (x)-nepravilno).
-kod upotrebe binarnih operatora, razmaci su i sa lijeve i sa desne strane operatora.
-Posle zareza korsitimo jedan razmak(npr. F(x, y)).
-Cijeline koda se odvajaju praznom linijom.
-Prilikom korištenja ključnih riječi(if, while…), ključna riječ i lijevi zarez su odvojeni razmakom.

2.4 Razbijanje redova

Ako imamo izraz, koji ne može stati u jedan red, razbijanje se vrši po sledećim pravilima:

-Razbijanje posle zareza
-Razbijanje posle operatora
-Poravnanje nove linije sa početkom izraza na istom nivou kao prethodna linija
-Koristimo uvlačak od 8 razmaka, ako prethodna pravila generišu vizuelno lose formatiran kod.

2.5 Zagrade

Vitičaste zagrade koristimo kod if, else, for, do, while ako im je tijelo prazno, ili unutar tijela imamo samo jednu liniju koda. Zagrade bez ograničenja koristimo kada imamo više operatora u izrazu, da bi se izbjegao problem prioriteta.
Primjer:

```
1   if(x == y || z == g)          //Izbjeći
2   if((x == y || z == g))        //Pravilno
```

2.6 Pisanje kratkih pojedinačnih linija

- Nije dozvoljeno pisanje više izraza u jednoj liniji.
- Odstupanje od ograničenja u vidu maksimalnih 80 karaktera u jednoj liniji postoji kod pisanja URL-a.

2.7 Povratne vrijednosti

Izbjegavati redundantni kod.
Primjer:

```
1   if(VrijednostBool)
2   {
3       return true;
4   }
5   else
6   {
7       return false;                 //Ovakav način pisanja stvara nepotreban višak koda, izbjeći.
8   }
9
10  return VrijednostBool;            //Pravilno.
```

3. Komentari

Prilikom pisanja koda, koristićemo 2 tipa komentara:
1.)Komentari implementacije: Kratki komentari, čija je uloga da pokažu kako je dio koda implementiran, a da nije očigledan. Poboljšavaju razumjevanje koda programerima. Prilikom pisanja komentara, izbjegavaju se višelinijski komentari, osim ako za njima nema krajnje potrebe.

2.)Komentari dokumentacije(JavaDoc):To su komentari koji kratko opisuju klasu, funkcije, informacije o autoru, itd. Editor Eclipse omogućava generisanje koda JavaDoc(Java dokumentacija).Svaki komentar je uključen u dokumentaciju koda.

4. Imenovanje

Imenovanje se odnosi na pravila kodiranja u Javi. Detaljan prikaz u tabeli:

Tip identifikatora	Pravila imenovanja	Primjeri
Paketi	Imena paketa se pišu malim slovima. Ovo se radi da ne bi došlo do konflikta sa imenima klasa i interfejsa.Ako postoje uzastopne riječi, odvojene su tačkom. Riječi ne smiju biti povezane donjom crtom, a nije ni dozvoljeno korištenje kombinacije malih i velikih slova.Ako postoje ključne riječi sadržane u imenu domene ili ime počinje brojem, tada ispred ključne riječi stavljamo znak _.	javax.swing com.example.mypackage com.example._package
Klase	Imena klase su imenice koje se pišu velikim slovom, kao i unutrašnje imenice. Dozvoljena je kombinacija malih i velikih slova. Klase trebaju biti jednostavne, sa jasnim značenjem. Preporučljivo je koristiti gotovu riječ, ali i skraćenice ukoliko se radi o poznatim pojmovima koji su skraćeni(URL,HTTP,).	class Student; class MojaKlasa;
Interfejsi	Ista pravila kao za klase.	interface Animal; interface FirstInterface;

Metode	Metode su glagolske riječi. Prva riječ metode uvijek počinje malim slovom, a ostale riječi, ako ih ima su velika slova.	sort();
		sortirajNiz();
		runFast();

Varijable	Imena varijabli treba da počinju sa malim slovom, dok unutrašnja riječ počinje sa velikim. Imena trebaju da budu jasna sa značenjem, sem ako nisu privremene varijable. i, j, k, m, n-primjeri za int varijable. c, d, e-primjeri za char varijable.	int i; float tezinaIgrača;

Konstante	Konstante treba da se pišu velikim slovima, a ako imaju više riječi, ne koristi se razmak, nego su odvojene sa znakom_.	int MAX=5;
		int MAX_HEIGHT=10;
		int MIN_WIDTH=2;

5. Deklaracije

5.1 Broj deklaracija po liniji

Piše se jedna deklaracija po liniji radi lakšeg pisanja komentara.

```
int ocjena;
float cijena;
```

5.2 Pozicija

Deklaracija se vrši na početku bloka koda, unutar vitičastih zagrada. Izbjegavati deklarisanje varijabli tek kada su neposredno potrebne.

```
public void metoda()
{
    int brojac;                    //početak bloka metode

    if(uslov)
    {
        int suma = 0;              //početak if bloka
        ...
    }
}
```

Jedini izuzetak ovog pravila je kod for petlji.

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

Zabranjene su lokalne deklaracije koje sakrivaju globalne.

```
int suma;
...
racun()
{
    if(uslov)
    {
        ...
        int suma += 10;                    //Ovakva deklaracija nije dozvoljena!!!
        ...
    }

    ...
}
```

5.3 Inicijalizacija

Lokalne varijable inicijalizovati tamo gdje su i deklarisane. Odstupanje od ovog pravila se javlja ukoliko vrijednost varijable zavisi od nekih prethodnih izračunavanja. Inicijalizaciju vršiti onda kada je stvarno potrebno.

5.4 Deklaracija klasa i interfejsa

Deklaracija klasa i interfejsa se počinje otvorenom vitičastom zagradom u istoj liniji sa imenom klase/interfejsa. Zatvorena vitičasta zagrada se nalazi na kraju, sama u jednoj liniji, osim u ako je u pitanju prazno tijelo. Tada se zagrade nalaze jedna do druge bez razmaka.

```
class Student{



}


class Profesor{}
```

6. Izrazi i izjave

Izrazi predstavljaju osnovne jedinice izvršenja koje izvršavaju operacije nad podacima i premještaju podatke.

6.1 Izrazi

Izrazi moraju biti precizni i nedvosmisleni, da ne bi dolazilo do eksplicitne konverzije podataka. Ukoliko se radi o složenim izrazima, radi lakšeg razumijevanja koda koristimo zagrade.

```
x + y / z                //Neispravno
(x + y) / z              //Ispravno
x + (y / z)              //Ispravno
```

6.2 Jednostavne izjave

Svaka linija treba da sadrži najviše jednu izjavu.

```
brojac++; bodovi += 10;              //Neispravno, dvije izjave u jednoj liniji
brojac++;                            //Ispravno
bodovi += 10;                        //Ispravno
```

6.3 Return izjave

Return izjave ne bi trebale da se pišu u zagradama, osim ako je riječ o složenim izrazima.

```
return;
return vrijednost;
return(uspjesnaPrijava? uspjesna: neuspjesna);
```

6.4 If, if-else i if-else if-else izjave

If, if-else, if-else if-else izjave se pišu na sljedeći način:

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

```
if (uslov)                //Provjera uslova
{
    izjave1;
}
else                      //Blok u koji se ulazi ukoliko uslov nije ispunjen
{
    izjave2;
}
```

```
19  if (uslov)                  //Provjera prvog uslova
20  {
21      izjave1;
22  }
23  else if (uslov)             //Ukoliko prvi nije ispunjen, provjera narednog
24  {
25      izjave2;
26  }
27  else                        //Blok u koji se ulazi ukoliko nijedan uslov nije uspunjen
28  {
29      izjave3;
30  }
```

6.5 For izjava

For petlje imaju sljedeći oblik:

```
1  for(inicijalizacija; uslov; korak)
2  {
3      izjave;
4  }
```

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

6.6 While izjava

While petlja ima sljedeći oblik:

```
1  while (uslov)
2  {
3      izjave;
4  }
```

Prazna while izjava:

```
1  while (uslov);
```

6.7 Do-while izjava

Do-while izjava ima sljedeći oblik:

```
1  do {
2      izjave;
3  } while (uslov);
```

6.8 Switch izjava

Switch izjava ima sljedeći oblik:

```
switch (uslov)
{
    case 1:
        izjave1;
        break;
    case 2:
        izjave2;
        break;
    case 3:
        izjave3;
        break;
}
```

6.9 Try-catch izjave

Try-catch izjave imaju sljedeći oblik:

```
try {
    izjave1;
} catch (ExceptionClass e) {
    izjave2;
} finally {
    izjave3;
}
```

7. Upravljanje memorijom

Za upravljanje memorije koristi se Garbage Collector sistem koji u pozadini automatski upravlja memorijom. Garbage collector briše neiskorišćene objekte i objekte koji više nisu u funkciji u programu, čime oslobađa prostor u memoriji.

8. Rukovanje greškama i izuzecima

- Try-catch blok se završava finally blokom
- Svaka greška i izuzetak se trebaju čuvati u log file-u
- Pri kreiranju novog tipa izuzetka potrebno ga je dokumentovati
- Pri nastanku izuzetka potrebno je obezbjediti nesmetan rad korisniku
- Prilikom bacanja izuzetaka potrebno je opisati razlog nastajanja greške korisniku
- Ukoliko postoji više tipova izuzetaka, potrebno je postaviti blokove tako da se izuzeci hvataju redom od najniže ka najvišoj podklasi. Prvo se hvataju specifični, pa uopšteni izuzeci.

9. Prenosivost

Programi napisani u Java programskom jeziku su prenosivi i mogu se kompajlirati na svakom računaru koji ima instaliran JVM. Izvršni kod se ne mora mijenjati da bi zadovoljio posebne potrebe bilo kog računarskog sistema.

10. Ponovno korištenje

Kako bi kod bio pogodan za ponovno korištenje potrebno je pridržavati se sljedećeg:

- Izbjeći ponavljanje koda i pisanje nepotrebnih dijelova koda
- Klase i metode treba da obavljaju jednu funkciju
- Napraviti testove za klase i učiniti klase jednostavnim za testiranje
- Često koristiti apstrakciju

- Napisani kod treba biti lak za proširivanje i održavanje
- Ne ponavljati poslovni kod, jer postoji mogućnost da će biti mijenjan i nekompatibilan s ostatkom koda

11. Problemi pri kompajliranju

Sve što je potrebno posjedovati za kompajliranje Java koda navedeno je u dijelu Prenosivost. Da bi se kod mogao kompajlirati neophodno je otkloniti sve sintaksne greške.

12. Dodatak: Sažetak smjernica

- Organizacija i istil koda – opisan izgled koda i način formatiranja koda radi lakše čitljivosti
- Komentari – opisano kada i kako komentarisati i dokumentovati kod
- Imenovanje – prikazana tabela sa pravilima imenovanja elemenata kojima se programer služi
- Deklaracije – opisan način deklarisanja promjenljivih
- Izrazi i izjave – opisan način pisanja petlji, grananja, matematičkih i logičkih izraza
- Upravljanje memorijom – opisuje rad Java Garbage Collector-a
- Rukovanje greškama i izuzecima – navode se pravila i načini rukovanja greškama i izuzecima
- Prenosivost – dat uvid u mogućnost prenosivosti sistema za različite platforme
- Ponovno korišćenje – navode se prakse kojih se treba pridržavati da bi se omogućilo ponovno korišćenje koda
- Problemi pri kompajliranju – opisano kako postupati ukoliko dođe do problema pri kompajliranju