

Dokumentácia k štýlu programovania

(Textový editor obohatený o grafické prvky)

Tímový projekt

Obsah

Úvod.....	3
Slovník pojmov (terminológia)	3
#1 Vytváranie názvov.....	4
#2 Odsadenia	4
#3 Písanie zátvoriek	5
#4 Písanie komentárov	5
#5 Písanie metód	6

Úvod

Tento dokument definuje štýl písania kódu, ktoré budeme dodržiavať v rámci firemnej kultúry tímového projektu. Definované sú pravidla, ktoré musí každý člen tímu dodržiavať pre sprehľadnenie zdrojového kódu a tým zamedzeniu možných nedorozumení a konfliktov, ktoré môžu vzniknúť z nejednotného štýlu programovania.

Slovník pojmov (terminológia)

PascalCase – notácia pri ktorej sa začiatocne slová názvov vždy začínajú veľkým písmenom (napr. „TimovyProjekt“)

camelCase – je notácia pri ktorej sa vždy prvé slovo píše malým písmenom a každé nasledujúce slovo začína veľkým písmenom (napr. „urobitSkusku“)

Mad'arska notácia - je typ notácie pri ktorej sa používa prefix pred prvé slovo, ktoré niečo zvyčajne indikuje (napr. „btnOK“ - btn pre tlačidlo a OK je názov tlačidla)

#1 Vytváranie názvov

Použitie spravených názvov je kľúčové k sprehľadneniu kódu, názvy treba zvoliť zmysluplne aby vystihovali podstatu riešenia.

Názvy budú písane po slovensky a bez diakritiky

Triedy

1. používať notáciu PascalCase
2. názvy by mali byť podstatnými menami

Metódy

1. používať notáciu camelCase
2. názvy by mali byť slovesného tvaru

Premene

1. názvy sú písane malými písmenami
2. voliť zmysluplne názvy nie nič nehovoriace skratky ako (napr. „v“)
3. v prípade dlhších názvom používať pre oddelenie slov podtrhovník (narp. „nazov_nazov“)
4. ak je možné tak premenu v tom istom riadku aj inicializovať

Konštanty

1. písane sú veľkými písmenami

Ovládacie prvky (GUI)

1. používať Maďarsku notáciu
2. používať prefix, ktorý vystihuje aký je to typ ovládacieho prvku (napr. „btnOK“ - btn pre tlačidlo a OK je názov tlačidla)

#2 Odsadenia

Pre sprehľadnenie štruktúru blokov v zdrojovom kóde treba dodržiavať nasledujúce pravidla.

1. Každý vnorený riadok musí byť odsadený tabulátorom o jednu pozíciu do ľavá

Správne:

```
nazovFunkcie()
{
    if()
    {
        nejakypríkaz;
    }
}
```

#3 Písanie zátvoriek

1. Zložene zátvorky píšeme vždy na novom riadku nepoužívať štýl K&R

Správne:

```
if()  
{  
    nejakyprikaz;  
}
```

Nesprávne:

```
if() {  
    nejakyprikaz;  
}
```

2. Písanie okrúhlych zátvoriek vždy za kľúčovým alebo nejakým príkazom slovom bez použitia medzery

Správne:

```
if (a == 2)
```

Nesprávne:

```
if      (a == 2)
```

#4 Písanie komentárov

1. Komentáre písať čo možno stručne nevytvárať v žiadnom prípade laterálne diela
2. Komentovať treba každú metódu, podmienku, cyklus, triedu ale aj premenu
3. Komentáre písať bez diakritiky a prvé slovo začína veľkým písmenom

Komentovanie súborov

1. Každý súbor alebo ak je v jednom súbore trieda by mal obsahovať v hlavičke základne informácie, ktoré by nás mali informovať čo daný súbor obsahuje a k čomu je určený

Formát:

```
/*  
Nazov suboru  
-----  
popis suboru načo slúži prípadne ake triedy obsahuje  
  
[datum zmeny - meno autora ktory zmenu vykonal]  
*/
```

Komentovanie vetvení

1. Vetvenia vždy komentovať za podmienkou v tom istom riadku využívať 2x stlačenie tabulátora pre odsadenia

Správne:

```
if (podmienka)           // Komentár
{
}
else if (podmienka)      //Komentár
{
}
else                     // Komentár
{
}
```

Komentovanie metód

1. Používať jednoriadkové komentáre
2. Komentovať načo funkcia slúži aké príma a odosiela parametre

napr.

```
vypisTextu() //Metóda vypisuje text na obrazovku o aktuálnom dátume
{
    Nejakyprikaz;
}
```

Používanie skratiek v komentároch

1. TODO: bude značiť niečo čo je potrebné v budúcnosti implementovať

napr.

```
//TODO: doplniť funkcionality
```

2. BugID: bude signalizovať že na tomto mieste je známa chyba a ak je zaznačená nástroji pre manažment zmien tak aj jej id

napr.

```
//Bug12: chyba nesprávneho výpisu hodnôt
```

#5 Písanie metód

Odporúčania:

1. Snažiť sa programovať krátke a jasné metódy. Ak je problém rozložený na viacero menších problémov a každý z nich je riešený na samostatnom mieste, je jednoduchšie pochopiť celý problém
2. Odstránenie duplicity kódu, čo zvyšuje kvalitu kódu a prispieva k lepšej udržiateľnosti
3. Vyhýbať sa tzv. „mŕtvemu kódu“ čo je taký kód, ktorý je napr. v komentároch, už sa nepoužíva len tam ostal ako história po predošlých verziách