Projektovanje softvera

Version 1.0

Istorija revizija

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Verzija** | **Opis** | **Autor** |
| 17/11/2016 | 1.0 | Početna verzija dokumenta | Mileta Stanišić |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Sadržaj

[1. Uvod 4](#_Toc468383797)

[1.1 Cilj 4](#_Toc468383798)

[2. Organizaicja i stil koda 4](#_Toc468383799)

[3. Komentari 5](#_Toc468383800)

[4. Imenovanje 5](#_Toc468383801)

[5. Deklaracija 5](#_Toc468383802)

[6. Upravljanje memorijom 5](#_Toc468383803)

[7. Upravljanje greškama i izuzecima 6](#_Toc468383804)

[8. Ponovna iskoristiviost 6](#_Toc468383805)

# Uvod

Ovaj dokument definše upute u programiranju koje programeri trebaju da slijede. Poštovanje ovih uputa olakšava održavanje koda, ponovno iskorištavanje dijelova koda, kao i nadogradnju sistema. Takođe poštovanjem nekih smjernica olakšava programerima da međusobno kontrolišu jedni druge i bolje rade u timu.

## Cilj

Svrha je u tome da definše smjernice za programiranje. Pod smjernicama programiranja definisane su:

* Organizacija i stil koda
* Komentari
* Imenovanje
* Deklaracija
* Upravljanje memorijom
* Upravljanje greškama i izuzecima
* Ponovna iskoristivost

# Organizaicja i stil koda

Za organizaciju koda koristiće se sledeće Java konvencije:

1. Source Files

Materijal u svim fajlovima će biti organizovan na sledeći način:

* Package statement
* Import statements
* A comment explaining the purpose of this file
* A public class
* Other classes, if appropriate

Komentar koji opisuje svrhu ovog fajla treba biti u skladu sa Javadoc konvencijom.

1. Classes

Komentar prethodi svakoj klasi i opisuje svrhu klase. Prvo se navode public elementi a zatim private elementi. Kod public i private elemenata koristimo sledeći redosled navodjenja:

* Constructors
* Instance Methods
* Static Methods
* Instance Fields
* Static Fields
* Inner classes

1. Methods

Svaki metod (osim main) počinje komentarom u Javadoc formatu. Poželjno je da metode imaju najviše 30 linija koda, pri tome opis metode, komentari, prazne linije i linije koje sadrže samo zagrade se ne računaju.

1. Variables and Constants

Sve varijable se ne definišu na početku bloka. Varijable definišemo na mjestima gdje se one prvi put koriste. Ne definišemo dvije varijable u jednom redu. Ne koristimo magic numbers. Pod magičnim brojem podrazumjeva se bilo koja numerička konstanta ugrađena u kod koja ne posjeduje definiciju. Svaki broj osim -1, 0, 1, i 2 se smatra magičnim.

# Komentari

Prilikom programiranja i implementacije dizajna programeri će koristiti dvije vrste komentara, dokumentacione i implementacione komentare. Na osnovu doc komentara će se pomoću javadoc komentara generisati kompletna dokumentacija projekta. Implementacioni komentari će se koristiti da opišu kako pojedini dijelovi koda rade. Za doc komentare koristi se sintaksa:

/\*\*

documentation

\*/

Za implementacione komentare se koristi sintaksa:

/\* text \*/ ili // text

# Imenovanje

Koriste se sledeće konvencije prilikom implementacije:

1. Paketi – imena paketa se uvijek pišu malim ASCII slovima i trebaju biti na vrhu domenskih imena.
2. Klase – imena klasa moraju biti imenice, pri čemu se svaka riječ piše početnim velikim slovom. Imena klasa treba da budu kratka i deskriptivna.
3. Interfejsi – imena interfejsa trebaju da se pišu sa početnim velikim slovom kao što je to slučaj i sa imenima klasa.
4. Varijable – imena varijabli se trebaju pisati sa početnim malim slovom, a zatim se svaka nova riječ u imenu piše početnim velikim slovom.
5. Konstante – imena konstanti treba da se pišu velikim ASCII slovima i za razmak između dvije riječi treba da se koristi donja crtica ("\_").
6. Metode – imena metoda traba da budu glagoli, tako da je prvo slovo metode malo a zatim prvo slovo svake nove riječi je veliko slovo.

# Deklaracija

Komponente koje treba da sadrži deklaracija klasa su:

* Modifiers such as public, private etc.
* The class name, with the initial letter capitalized by convention
* The name of the class’s parent (superclass), preceded by the keyword implements. A class can implement more than one interface.
* The class body, surrounded by braces {}.

# Upravljanje memorijom

Za programiranje koristimo programski jezik Java koji ima Garbage Collection tako da će se zaostali objekti uvijek pobrisati. Takođe kod današnjih računara memorija nije problem i sve se manje vodi računa o efikasnom korišćenju memorije. Softver je namjenjen korisnicima čije mašine nemaju problem sa memorijom.

# Upravljanje greškama i izuzecima

Sistem će imati podršku za rukovanje izuzecima. Za pojedine komponente sistema mogu se koristiti različite strategije opsluživanja izuzetaka. Sistem će biti robustan i otporan na greške. Za programiranje se koristi programski jezik Java tako da će se koristiti mehanizmi podržani od strane ovog programskog jezika kao što su try-catch blokovi i slično.

# Ponovna iskoristiviost

Prilikom modelovanja sistema i kreiranja class dijagrama korišćeni su šabloni. Šabloni omogućavaju da sistem ima jako svojstvo polimorfizma što omogućava njegovo jednostavno proširivanje i ponovno korišćenje dijelova koda.