# 1 Ohjelmoinin tyyliopas JouTo-projektissa

Tässä dokumentissa käsitellään JouTo-projektin lähdekoodeissa käytettävää ohjelmointityyliä, sisennyksiä, lohkorakenteiden merkitsemistä, kommentointia sekä lähdekoodissa käytettävää nimeämistyyliä.

Tyylien pohjana on käytetty yleisesti hyväksyttyä ja käytettyä Sun Microsystemsin Java Coding Style Guide -ohjetta. Kyseinen ohje löytyy osoitteesta http://www.sun.com/forte/whitepapers/java-style.pdf.

## 1.1 Lohkorakenteiden merkintä

Lohkon aloittavan aaltosulun sijainti on heti lohkon nimen perään samalla rivillä. Lopettava aaltosulku sijaitsee omalla rivillä sisennettynä samalle tasolle kuin lohkon aloittavan rivin ensimmäinen kirjain.

Esimerkki:

```
public class Program {
   ... koodia ...
}
```

Sama tyyli kuin edellä koskee myös esimerkiksi for()- ja while()-silmukkoja. Esimerkki:

```
public int someFunction(int i) {
   for (int j = i; j <= i*i; ++j) {
    System.out.println("j:n arvo: " + j);
   }
   return j;
}
```

## 1.2 Nimeämistyylit

Tässä luvussa selvitetään eri elementtien nimeämiskäytännöt, joita käytetään JouTo-projektissa.

### 1.2.1 Muuttujat

Muuttujat nimetään seuraavalla tavalla:

- Ensimmäinen kirjain kirjoitetaan pienellä, sitä seuraavien sanojen ensimmäinen kirjain isolla ja muut kirjaimet pienellä.

- Kaikki muuttujat nimetään käyttäen englannin kieltä tai yleisesti käytössä olevia muuttujannimiä (kuten x,y,i tai j)
- Vakiot kirjoitetaan kokonaan isoilla kirjaimilla
- Kaikki välit korvataan alaviivoilla (_)

Esimerkki:

```
int exampleVariable;
int CONSTANT_VALUE;
```

Kaikkiin luokkamuuttujiin liitetään tieto niiden näkyvyydestä. Näitä ovat mm.

- public
- private
- static

Lohkomuuttujiin tätä käytäntöä ei tarvitse soveltaa.
Esimerkki:

```
public class ExampleClass {

   public int example1;
   private boolean testValues;

}
```

### 1.2.2 Metodit

Funktiot nimetään seuraavalla tavalla:

- Ensimmäiset sanan kirjain pienellä, seuraavien sanojen alkukirjaimet isolla.
- Käytetään kielenä englantia.
- Metodien tyyppi on määrättävä.
- Metodien eteen on liitettävä tieto niiden näkyvyydestä.

Esimerkki:

```
public int exampleMethod(int variable) {
    . . .
}
```

### 1.2.3 Luokat

Luokat nimetään seuraavalla tavalla:

- Jokainen sana aloitetaan isolla kirjaimella.
- Käytetään kielenä englantia.
- Luokan eteen on liitettävä tieto sen näkyvyydestä.

Esimerkki:

```
public class TestingClass {
    . . .
}
```

## 1.3 Kommentointi

Kaikki itse tuotettu lähdekoodi kommentoidaan niin, että siitä saadaan JavaDoc:illä valmis API-kuvaus. Rivikommentteihin käytettään //-merkkejä. Esimerkki rivikommenteista

```
public int testMethod(int x) {

    int x;   // Tässä on rivikommentti

}
```

### 1.3.1 Yleistä JavaDoc:ista

JavaDoc-dokumentti alkaa merkeillä /** ja loppuu merkkeihin *. Jokaisen kommenttirivin aloittaa *-merkki.
Tarkempaa tietoa funktiosta, luokasta tai muuttujasta annetaan @parametri -tagilla sekä siihen liittyvällä määritelmällä. Seuraavassa taulukossa käytettävät tagit ja niiden selitykset.

| | |
|---|---|
| @author | Tämä tagi ilmoittaa koodit authorit (=kirjoittaja) |
| @param | Tällä tagilla ilmoitettaan parametrien tarkoitus |
| @return | Tällä tagilla ilmoitetaan palautusarvo ja sen tarkoitus |
| @exception | Tällä tagilla ilmoitetaan mahdolliset keskeytykset |
| @see | Viite liittyvään luokaan tai pakettiin |

Nämä tagit laitetaan aina JavaDoc-kommentin sisään kuten muukin kommentti. Jokainen tagi tulee omalle rivilleen. Tagin perään tuleva teksti erotetaan tabulaattorilla tagista eikä siihen lisätä heittomerkkejä tai vastaavia.

Esimerkki:

```
/**
* This is just an example of JavaDoc comment
* General description of program should be put here
* Also HTML-tags are possible, but do not use Heading or Font-elements.
*    <ul>
* <li>List element 1</li>
* <li>List element 2</li>
* </ul>
*
* @author   Harri Tuuloskoski
* @author   Lari Laine
* @see      Java.Lang
*/
```

Esimerkissä Harri ja Lari ovat molemmat kirjoittajia, joista Harri kronologisesti vanhempi kirjoittaja.
Jossain tageissa tarvitaan parametrejä. Ne laitetaan seuraavasti

```
/**
* Example of JavaDoc comment
*
* @exception   IOException    This function throws an exception
* @param       x              Parameter x is used as initial value
*
*/
```

### 1.3.2 Luokat

Jokaisessa luokassa on oltava ainakin seuraavat kommentit:

```
/**
```

```
  * Complete description Class, its'
  *   - purpose
  *   - things that must be taken into consideration when using the class
  *
  * @author      // all the authors of this class
  * @see         // if needed
  */
public class TestiLuokka {
```

### 1.3.3 Metodit

Metodit kommentoidaan seuraavalla tavalla

```
  /**
  * Description of method and its' operation
  *
  * @param      // all the parameters with descriptions
  * @return     // return value with description and purpose
  * @exception  // description of exception and purpose
  * @see        // Reference to related classes if needed
  */
  public int testMethod() {
```

### 1.3.4 Muuttujat

Muuttujia ei kommentoida erikseen JavaDoc-kommenteilla. Seuraavat kommenttivaatimukset ovat kuitenkin voimassa

- Jokainen muuttuja on laitettava omalle rivilleen esittelylohkossa
- jokaiseen muuttujaan on liitettävä kuvaus sen käyttötarkoituksesta sekä luokassa että metodissa. Tämä liitetään käyttämällä //-kommenttimerkkiä. Nämä kommentit ovat lähdekoodin lukijoita varten, eivätkä ne näy JavaDoc-ohjelman tulosteissa.

```
Esimerkki:
  public int testMethod(int parameter) {

     int x;                        // Used to store temporary variables
     boolean truthImplicator;      // Represents of someting is correct
```

### 1.3.5 Huomioitavaa

Jokaiset toteuttajan tulee omalta osaltaan lähdekoodia kirjoittaessaan huomioida seuraavat asiat.

- Jokainen monimutkaisempi lauserakenne, ehtolause tai algoritmi tulee kommentoida niin että myös toinen ryhmäläinen pystyy saamaan käsityksen kyseisen osuuden toimintalogiikasta mahdollisimman helposti.

## 1.4 Sisennys

Ohjelman lähdekoodi sisennetään käyttäen tabulaattoria sisennysmerkkinä. Näin jokainen pystyy asettamaan sisennyksen haluamallaan tavalla käyttämässään ohjelmassa.
Lohkorakenteen sisällä olevat koodirivit sisennetään yhdellä tabulaattorilla.
Esimerkki:

```
public class Test {
   String x;

   public String returnString(int index) {

      return this.x;
   }
}
```

Mikäli ehtorakenne, metodin kuvaus tai vastaava sulkeiden sisällä oleva rakenne on niin pitkä että se menee toiselle riville, niin se sisennetään käyttäen tabulaattoria ja välilyöntiä niin että seuraava rivi alkaa edellisen rivin ehtolauseen tms. tasalta. Tabulaattorilla siis tasataan edellisen tason alkuun, ja välilyönneillä parametrin tai ehtolaiseen tasalle.
Esimerkki:

```
 if ((value1 == a) || ((value1 == b) &&
     (value2 == c))) {
```

`tai`

```
 method(parameter1, parameter2,
        parameter3, parameter4);
```