CODE CONVENTIONS & JavaDoc

Java Code Conventions:

Come scrivere il proprio codice seguendo le convenzioni …

- Sul sito della sun puoi trovare il documento esteso
  - http://java.sun.com/docs/codeconv/
  - Ci sono 24 pagine di convenzioni

1. Introduction - Why Have Code Conventions

- Code conventions are important to programmers for a number of reasons:
  - 80% of the lifetime cost of a piece of software goes to maintenance.
  - Hardly any software is maintained for its whole life by the original author.
  - Code conventions improve the readability of the software, allowing engineers to understand new code more quickly and thoroughly.
- If you ship your source code as a product, you need to make sure it is as well packaged and clean as any other product you create.
- For the conventions to work, every person writing software must conform to the code conventions. Everyone.

2.1 File Organization - Java Source Files

- Each Java source file contains a single public class or interface. When private classes and interfaces are associated with a public class, you can put them in the same source file as the public class. The public class should be the first class or interface in the file.
- Java source files have the following ordering:
  - Beginning comments
  - Package and Import statements
  - Class and interface declarations

2.1.1 Beginning Comments

- All source files should begin with a c-style comment that lists the class name, version information, date, and copyright notice:

```
/*
 * Classname
 *
 * Version information
 *
 * Date
 *
 * Copyright notice
 */
```

2.1.2 Package and Import Statements

- The first non-comment line of most Java source files is a package statement. After that, import statements can follow. For example:

```
package java.awt;

import java.awt.peer.CanvasPeer;
```

- Note: The first component of a unique package name is always written in all-lowercase ASCII letters and should be one of the top-level domain names, currently com, edu, gov, mil, net, org, or one of the English two-letter codes identifying countries as specified in ISO Standard 3166, 1981.

2.1.3 Class and Interface Declarations

- The following lists describes the parts of a class or interface declaration, in the order that they should appear.

- **Class/interface documentation comment** (/**...*/)

- **class or interface statement**

- Class/interface implementation comment (/*...*/), if necessary

- **Class (static) variables**: First the public class variables, then the protected, then package level (no access modifier), and then the private.

- **Instance variables** First public, then protected, then package level (no access modifier), and then private.

- **Constructors**

- **Methods** These methods should be grouped by functionality rather than by scope or accessibility. For example, a private class method can be in between two public instance methods. The goal is to make reading and understanding the code easier.

3 - Indentation

- **Line Length**
- **Wrapping Lines**
- **....**

Suggerimento: usa il formatter automatico del tool in eclipse

4 - Comments

- Java programs can have two kinds of comments: implementation comments and documentation comments. Implementation comments are those found in C++, which are delimited by /*...*/, and //. Documentation comments (known as "doc comments") are Java-only, and are delimited by /**...*/. Doc comments can be extracted to HTML files using the javadoc tool.

- **Implementation Comment Formats**:
  - Block Comments:
    ```
    /*
     * Here is a block comment.
     */
    ```
  - Single-Line Comments:
    ```
    if (condition) {
       /* Handle the condition. */
       ...
    }
    ```

- **Documentation Comments: vedi dopo**

5. Declarations

- **Number Per Line:**
  One declaration per line is recommended since it encourages commenting.
- **Initialization:**
  initialize local variables where they're declared
- **Placement**:
  Put declarations only at the beginning of blocks. (A block is any code surrounded by curly braces "{" and "}".)
- **Class and Interface Declarations**
  - No space between a method name and the parenthesis "(" starting its parameter list
  - Open brace "{" appears at the end of the same line as the declaration statement
  - Closing brace "}" starts a line by itself indented to match its corresponding opening statement, except when it is a null statement the "}" should appear immediately after the "{"

6. Statements

- **Simple Statements**
  Each line should contain at most one statement.
- **Compound Statements…**
- **for Statements**
- **while Statements**
- **…**

- **if, if-else, if else-if else Statements**
  The if-else class of statements should have the following form:
  - ```
    if (condition) {
        statements;
    }
    ```
  - ```
    if (condition) {
        statements;
    } else {
        statements;
    }
    ```
  - ```
    if (condition) {
        statements;
    } else if (condition) {
        statements;
    } else{
        statements;
    }
    ```

Note: if statements always use braces {}.
    ```

7 - White Space

- …

8 - Naming Conventions - package

- **Packages**
- The prefix of a unique package name is always written in all-lowercase ASCII letters and should be one of the top-level domain names, currently com, edu, gov, mil, net, org, or one of the English two-letter codes identifying countries as specified in ISO Standard 3166, 1981. Subsequent components of the package name vary according to an organization's own internal naming conventions. Such conventions might specify that certain directory name components be division, department, project, machine, or login names.
- Esempi:
- com.sun.eng
- com.apple.quicktime.v2
- edu.cmu.cs.bovik.cheese

Naming Conventions - Class

- Class names should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep your class names simple and descriptive. Use whole words-avoid acronyms and abbreviations (unless the abbreviation is much more widely used than the long form, such as URL or HTML).
- Esempi:
- class Raster;
- class ImageSprite;
- Class Esempio4
- Class CalcoloCodiceFiscale

Naming Conventions – Interface - methods

- **Interface**
- Interface names should be capitalized like class names.
  interface RasterDelegate;
  interface Storing;
- **Methods**
- Methods should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.
- Esempi:
- run();
  runFast();
  getBackground();

Naming Conventions - variables

- Variables are in mixed case with a lowercase first letter. Internal words start with capital letters. Variable names should not start with underscore _ or dollar sign $ characters, even though both are allowed.

- Variable names should be short yet meaningful. The choice of a variable name should be mnemonic- that is, designed to indicate to the casual observer the intent of its use. One-character variable names should be avoided except for temporary "throwaway" variables. Common names for temporary variables are i, j, k, m, and n for integers; c, d, and e for characters.

- ```
  int           i;
  char            c;
  float          myWidth;
  ```

- ```
  MyCalc   myCalculator;
  ```

Naming Conventions - Constants

- The names of variables declared class constants and of ANSI constants should be all uppercase with words separated by underscores ("_").
- static final int MIN_WIDTH = 4;
- static final int MAX_WIDTH = 999;
- static final int GET_THE_CPU = 1;

Java Doc

- Vedi il libro 7.8
- Sito della sun http://java.sun.com/j2se/javadoc/
- Eclipse suggerisce cosa inserire (usa ctrl+space)
- Usa qualche plugin se non ti ricordi
  - JdocEditor http://www.certiv.net/download/downloads.html
  - ...
- Cosa sono:
- Un modo standard per commentare classi e metodi per generare documentazione

Documentazione di classe

I commenti JavaDoc iniziano con /** e finiscono con */

Mettiamo del testo in cui spieghiamo cosa fa la classe.

Mettiamo i commenti della classe prima della dichiarazione della classe

```
/** Questa classe gestisce le iscrizioni on-line agli esami

@author Angelo Gargantini

@version 1.2

*/

public class GestioneEsami{

...

}
```

Usiamo poi all'interno dei commenti delle speciali parole per individuare alcune informazioni particolari

@author: per l'autore
@version: per la versione

Documentazione di metodo

I commenti JavaDoc iniziano con /** e finiscono con */

Mettiamo del testo in cui spieghiamo cosa fa il metodo

Mettiamo i commenti prima della dichiarazione del metodo

```
/** Questo metodo cancella il numero di matricola indicato dalla lista
    degli iscritti

    @param  matricola il numero di matricola da cancellare

    @return  0 se è andato a

*/

public int cancellaPrenotaz(

...

}..
```

Usiamo poi all'interno dei commenti delle speciali parole per individuare alcune informazioni particolari

@param *nomeprametro descrizione*
@return *descrizione*

Javadoc -> HTML

- Se il programma è documentato secondo queste convenzioni, è possibile utilizzare javadoc per generare automaticamente la documentazione:
- nella cartella MieiProgrammi, creiamo una nuova directory doc
- Dal prompt dei comandi, ci posizioniamo nella nuova directory (cd … \MieiProgrammi\doc)
- Lanciamo javadoc:
- Questo comando genera (sotto doc) una serie di file
- .html (+ un .css e un file package-list).
- Aprendo index.html con un normale browser (per es. Microsoft Explorer) viene visualizzata la documentazione

In eclipse

- Per l'inserimento scrivi @ e poi ctrl +space
- Per preview, apri la finestra javadoc
- Per generare html:
  - Export da file o con tasto destro
  - seleziona javadoc
    - Se non trova il comando javadoc devi cercarlo, ad esempio sotto c:\programmi\java\bin
  - Procedi
- select the class name and the press "Shift F2".

Javadoc delle liberire

- Le librerie java sono corredate dai loro documenti javadoc:
  - http://java.sun.com/j2se/1.5.0/docs/api/
- Puoi scaricarle
- Puoi dire a eclipse dove trovarle
  - Scarica il jdk-1.5.0-doc.zip e unzippalo
  - Su eclipse, tasto destro su rt.jar di JRE system libraries
    - Proprietà
    - Imposta Javadoc location: seleziona la directory dove hai scaricatola documentazione
  - Premi F2

Javadoc di java api in eclipse

- Per aver avere i Javadoc Tooltips che mostrano la documentazione di Java API o per avere una descrizione nella view javaDoc devi
  - Assicurati di avere il file src.zip
    - Linux: /opt/sun-jdk1.5.0/share/src.zip
    - Windows: C:/programmi/Java/jdk1.5.0.06/src.zip
      - Se non hai questo file devi installarlo dal pannello di controllo, installa applicazioni, modifica installazione jdk
- Da eclipse vai sotto Window->Preferences-> Java->Installed JREs->edit, spunta “usa default liberies”, seleziona rt.jar e “attach” i sorgenti src.zip