# Programmazione Orientata agli Oggetti in Linguaggio Java

## Sintassi e Semantica: Convenzioni di Stile

versione 1.1



**G. Mecca – Università della Basilicata – mecca@unibas.it**



## Sommario

# Introduzione

- Convenzioni di stile
  - ⇨regole per la scrittura del codice sorgente
- Le convenzioni sono importanti
  - ⇨migliorano la leggibilità
  - ⇨migliorano la manutenibilità
  - ⇨migliorano la qualità del lavoro di gruppo
  - ⇨nello sviluppo industriale sono considerate indispensabili





# Introduzione

- Nel seguito
  - ⇨un elenco di regole basate sulle “Java Code Conventions” della Sun
  - ⇨vanno dalla posizione delle parentesi
  - ⇨ai nomi delle variabili
- Attenzione
  - ⇨possono sembrare aspetti secondari
  - ⇨ma in un gruppo di lavoro devono essere adottate da tutti

# Organizzazione dei File

- Un file .java
  - ⇨una classe pubblica (la prima)
  - ⇨eventuali altre classi private o friendly
- Organizzazione del file
  - ⇨commento iniziale (opzionale)
  - ⇨dichiarazione di package
  - ⇨istruzioni di import
  - ⇨dichiarazioni di classi





# Esempio

```
/*
    Classe di esempio
*/

package it.unibas.prova;

import java.util.ArrayList;
import java.util.Date;

public class ClasseDiProva {
    ...
}
```

# Organizzazione dei File

- Sezioni della classe
  - ⇨proprietà statiche costanti (public final)
  - ⇨proprietà statiche (private)
  - ⇨proprietà di oggetto (private)
  - ⇨costruttori
  - ⇨metodi statici
  - ⇨metodi di oggetto





# Esempio

```
public class ClasseDiProva {

    public static int x = 1;
    private static float y;

    private int a;
    private String s;

    public ClasseDiProva() {
       ...
    }

    public int getA() {
      ...
    }
}
```

# Spazi Bianchi

- Servono a rendere più leggibile il codice
- Utilizzo delle linee vuote
  - ⇨per separare le diverse sezioni di un file
  - ⇨per separare le diverse sezioni di una classe
- Utilizzo del carattere spazio
  - ⇨separare gli operatori binari dagli operandi
  - ⇨a = b + c; // OK
  - ⇨a=b+c; // NO





# Indentazione

- Dove usare l'indentazione
  - ⇨dove ci sono blocchi di istruzioni { }
- Livello di indentazione
  - ⇨4 caratteri
- Tipo di indentazione
  - ⇨è preferibile utilizzare spazi e non tabulazioni
- Lunghezza delle linee
  - ⇨evitare linee più lunghe di 80 caratteri
  - ⇨spezzare le linee in modo leggibile

# Esempio

```
public int modulo (int a) {
    int x;
    if (a > 0) {
        x = a;
    } else {
        for (int i = 0; i < a; i++) {
            System.out.println("Negativo");
        }
        x = -a;
    }
    System.out.println("Questa e' una linea"
                       + "lunga e viene spezzata");
    x = Math.sqrt(x * x * x) + -10 * 22
                             + Math.sin(a + b);
}
```





# Istruzioni

- Utilizzare una istruzione per linea
  - ⇨a = 1;
  - ⇨b++;
  - ⇨a = 1; b++; // NO
- Strutture di controllo
  - ⇨utilizzare sempre le parentesi graffe
  - ⇨anche quando il corpo contiene un'unica istruzione (si evitano errori)

# Istruzioni

- Posizione delle parentesi
  - ⇨la parentesi aperta è al livello dell'istruzione che comincia il blocco
  - ⇨il contenuto del blocco è indentato di un livello
  - ⇨la parentesi chiusa è a capo da sola





# Istruzione if: Esempi

```
if (condizione) {
    <istruzioni>;
}

if (condizione) {
    <istruzioni>;
} else {
    <istruzioni>;
}
```

```
if (condizione) {
    <istruzioni>;
} else if (condizione) {
    <istruzioni>;
} else {
    <istruzioni>;
}
```

# Istruzioni for e while: Esempi

```
for (int i = 0; i < 10; i++) {
    <istruzioni>;
}
int i = 0;
while (i < 10) {
    <istruzioni>;
    i++;
}
```





# Convenzioni sui Nomi

- Convenzione “cammello”
  - identificatore ottenuto da una sequenza di parole
  - togliendo spazi e caratteri speciali
  - con ciascuna iniziale intermedia maiuscola
- Esempi
  - nomeDellaCitta
  - getTotal
  - immagineInFormatoGif

# Convenzioni sui Nomi

- Metodi
  - ⇨iniziale minuscola e convenzione cammello
  - ⇨es: getNome
  - ⇨es: setNome
- Variabili
  - ⇨iniziale minuscola e convenzione cammello
  - ⇨es: nome
  - ⇨es: nomeECognome





# Convenzioni sui Nomi

- Nomi delle classi
  - ⇨iniziale maiuscola e convenzione cammello
- Esempi
  - ⇨class Controllore, class PuntoDiDomanda
- Non si utilizza la convenzione cammello
  - ⇨nei nomi delle costanti
  - ⇨nei nomi dei package

# Convenzioni sui Nomi

- Costanti
  - ⇨non si utilizza la convenzione cammello
  - ⇨nome tutto maiuscolo
  - ⇨spazi sostituiti da “underscore”
- Esempi
  - ⇨static final int DIM_MASSIMA = 3;
  - ⇨static final char PEDINA = ‘x’;





# Convenzioni sui Nomi

- Package
  - ⇨non si usa la convenzione cammello
  - ⇨nome completamente minuscolo
  - ⇨organizzato sotto forma di dominio Internet inverso
- Esempi
  - ⇨it.unibas.pinco
  - ⇨it.unibas.indovinasemplice
  - ⇨com.sun.eng

# Pratiche di Programmazione

- Pratica n. 1
  - ⇨evitare di utilizzare proprietà pubbliche nelle classi (incapsularle nei metodi di accesso)
  - ⇨metodi set e get
- Convenzione
  - ⇨per un attributo chiamato <nome>
  - ⇨setNome
  - ⇨getNome
  - ⇨il nome deve essere quello dell'attributo





# Pratiche di Programmazione

```
public class Prova {

    private int conta;

    public void setConta(int conta) {
        this.conta = conta;
    }

    public int getConta() {
        return conta;
    }
}
```

Evitare di usare nomi diversi; es: setContatore oppure getContatore

# Pratiche di Programmazione

- Inoltre
  - ⇨in un metodo set, è opportuno che il parametro si chiami come la proprietà
- Esempio

```
private int conta;
public void setConta(int conta) {
        this.conta = conta;
}
```

Evitare invece:

```
public void setConta(int c) {
      conta = c;
}
```





# Pratiche di Programmazione

- Pratica n. 2
  - ⇨utilizzare le parentesi nelle espressioni per rendere più evidente l'ordine di valutazione
- Pratica n. 3
  - ⇨evitare metodi eccessivamente lunghi
  - ⇨spezzarli in vari metodi indipendenti

# Pratiche di Programmazione

- Pratica n. 4
  - ⇨inserire sempre le classi in un package esplicito
  - ⇨non utilizzare mai il package standard anonimo
- Esempio

```
package esempio;
public class Prova {
    ...
}
```





# Riassumendo

- Introduzione
- Organizzazione dei File
- Spazi Bianchi
- Indentazione
- Istruzioni
- Convenzioni sui Nomi
- Pratiche di Programmazione

# Termini della Licenza