



# Laboratorio di Programmazione

## Laurea in Bioinformatica

Web: <http://www.scienze.univr.it/fol/main?ent=oi&id=28023&lang=it>

Docente: *Carlo Drioli*

Email: *drioli@sci.univr.it*

Lucidi a cura di  
Nicola Drago      Carlo Drioli      Federico Fontana

*Lezione 1*

# Sommario

---

- Compilare ed eseguire codice JAVA
- Editor e ambienti di sviluppo
- Scrittura di codice
- Primi esempi

# Il linguaggio JAVA

---

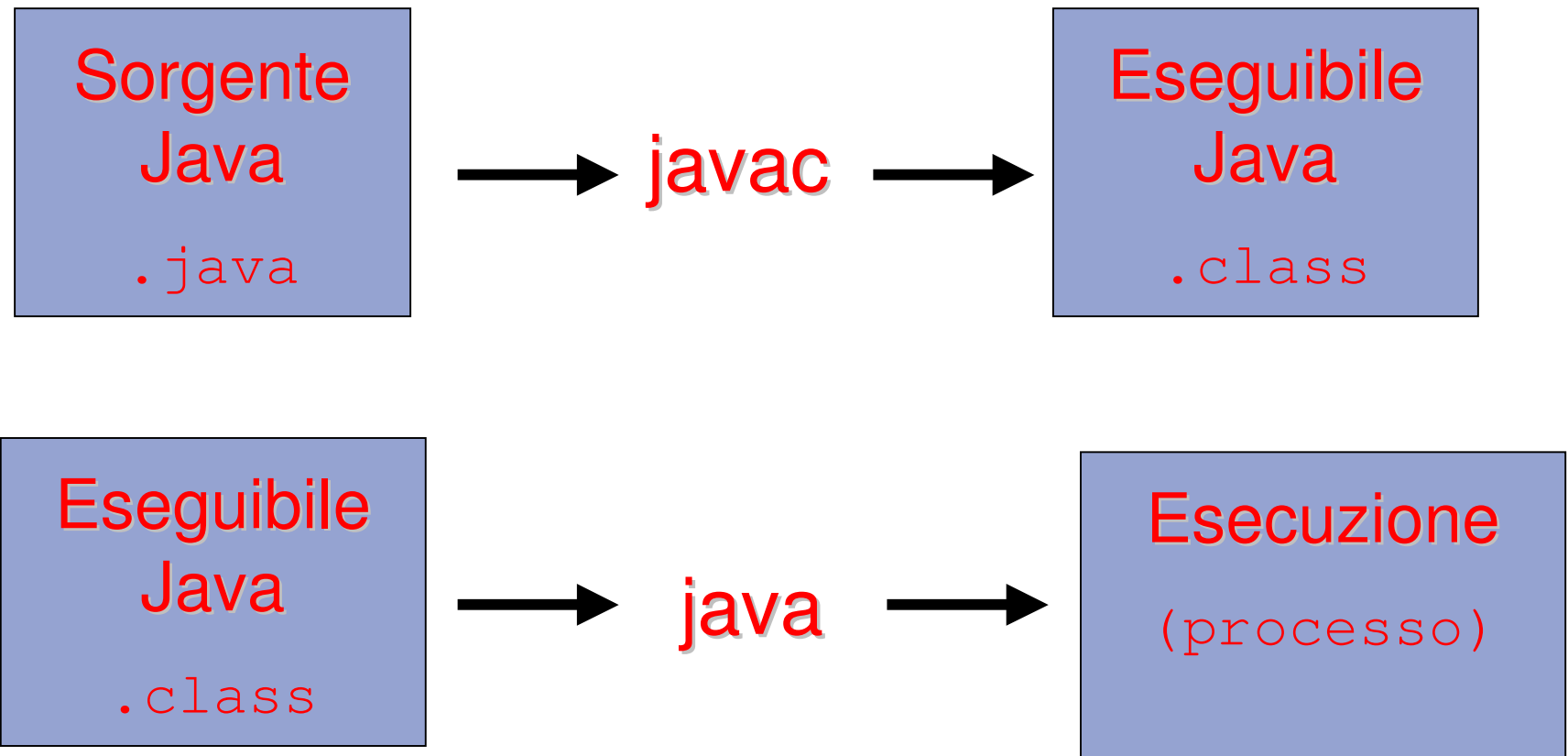
- Java è stato creato da *Sun Microsystems* ed è disponibile liberamente su <http://java.sun.com/>
- E' un linguaggio di programmazione orientato agli oggetti, derivato dal C++
- E' indipendente dalla piattaforma: un programma Java è compilato in un formato intermedio (bytecode) uguale per tutte le piattaforme.
- Il bytecode è interpretato da una Java Virtual Machine (JVM).
- Per lo sviluppo, Sun mette a disposizione un System Developer Kit (SDK)

# Compilare ed eseguire codice JAVA

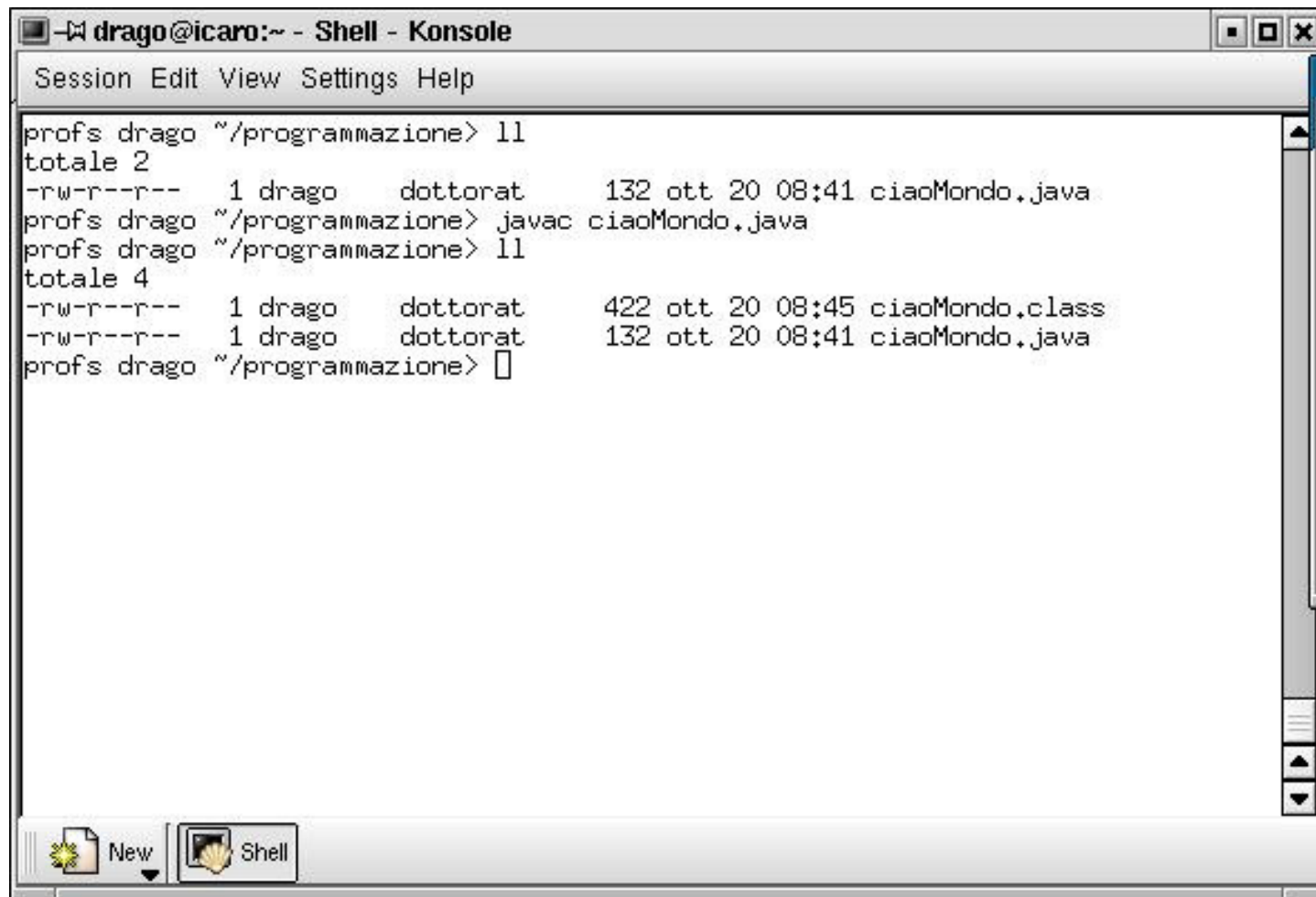
---

- “javac”: compila il file sorgente producendo un file eseguibile in formato bytecode
- “java”: esegue il bytecode
- I sorgenti hanno estensione .java
- Gli eseguibili hanno estensione .class
- I file di libreria hanno estensione .jar

# Compilare ed eseguire codice JAVA



# Compilare ed eseguire codice JAVA



The screenshot shows a terminal window titled "drago@icaro:~ - Shell - Konsole". The window contains the following text:

```
Session Edit View Settings Help
profs drago ~/programmazione> ll
totale 2
-rw-r--r--  1 drago  dottorat  132 ott 20 08:41 ciaoMondo.java
profs drago ~/programmazione> javac ciaoMondo.java
profs drago ~/programmazione> ll
totale 4
-rw-r--r--  1 drago  dottorat  422 ott 20 08:45 ciaoMondo.class
-rw-r--r--  1 drago  dottorat  132 ott 20 08:41 ciaoMondo.java
profs drago ~/programmazione> 
```

The terminal window has a menu bar with "Session", "Edit", "View", "Settings", and "Help". At the bottom, there are buttons for "New" and "Shell".

# Tool di elaborazione testi in Unix

---

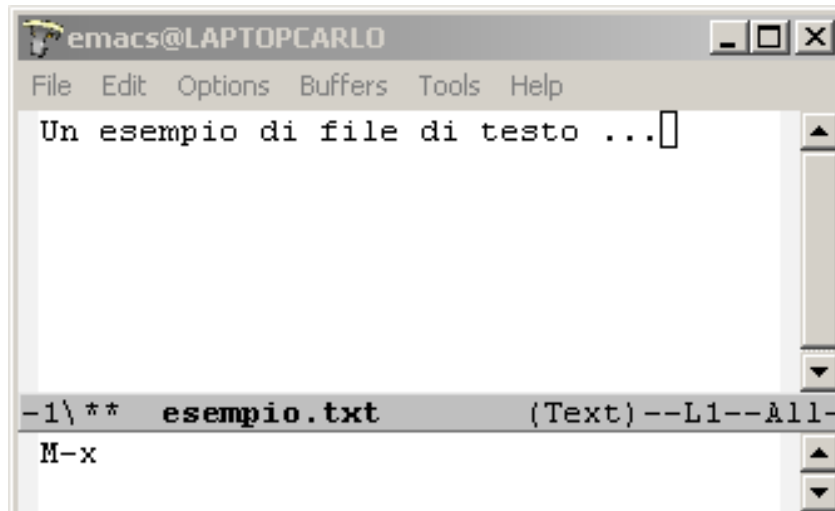
- In ambiente Unix è particolarmente importante disporre di strumenti efficaci per poter leggere, modificare e scrivere file di testo.
- Molte operazioni di **configurazione e manutenzione** del sistema richiedono la modifica di file testuali.
- Lo **sviluppo** di **applicazioni** e di **nuove componenti** avviene attraverso la manipolazione di file di testo contenenti istruzioni dei linguaggi di programmazione
- I programmi di elaborazione di file di testo più diffusi in ambiente Unix sono **vi** ed **emacs**.

# Elaborazione di testi con *emacs*

- Creare o modificare un documento

**\$ emacs nomefile**

- Lo spazio dello screen editor è diviso in tre parti
  - Area di testo
  - Riga di stato
  - Area di comando



← Area di testo

← Riga di stato

← Area di comando



# Elaborazione di testi con *emacs* (2)

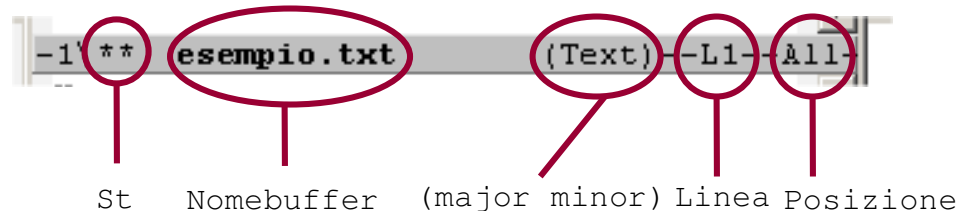
---

- Emacs opera su tre componenti principali:
  - **File:** è un file memorizzato sul disco. Non viene mai manipolato direttamente, tutte le operazioni vengono eseguite copiando i file in dei buffer di memoria e salvando il risultato delle manipolazioni sui buffer in un file.
  - **Buffer:** è una struttura interna che contiene il testo da elaborare. Possono esserci più buffer attivi allo stesso tempo.
  - **Finestre:** una finestra corrisponde alla visualizzazione di un buffer. E' possibile visualizzare uno o più buffer per volta aprendo e chiudendo finestre durante una sessione di elaborazione del testo.

# La riga di stato di *emacs*

- Visualizza informazioni relative al testo corrente.

- Struttura:



- **St**: indica se il file è stato salvato dopo l'ultima modifica.  
“\* \*” (non salvato), “--” (salvato), “% %” (file di sola lettura)
- **Nomebuffer**: indica il nome del buffer corrente
- **(major minor)**: modalità di editing del file.  
**major** fa riferimento a configurazioni di editing per linguaggi particolari (es. Lisp, C, testo semplice, etc.)  
**minor** fa riferimento a modalità di inserimento testo particolari
- **Linea**: numero di linea su cui è posizionato il cursore
- **Posizione**: posizione del cursore in relazione all'inizio del file

# Comandi principali di *emacs*

- In Emacs i comandi vengono invocati attraverso la combinazione dei tasti CTRL o ALT con altri tasti.  
Ad esempio per **uscire** da Emacs si può usare la sequenza **CTRL-x CTRL-c**

## Comandi di manipolazione dei file

CTRL-x	CTRL-f	apre un file esistente
CTRL-x	CTRL-s	salva il file corrente
CTRL-x	CTRL-w	salva il file con nome

# Comandi principali di *emacs* (2)

## Comandi di manipolazione dei buffer

CTRL-x b	seleziona un buffer attivo o crea un buffer nuovo
CTRL-x CTRL-b	elenca i buffer attivi
CTRL-x k	elimina un buffer

## Comandi di manipolazione delle finestre

CTRL-x o	seleziona un'altra finestra tra quelle attive
CTRL-x 0	chiudi la finestra corrente
CTRL-x 1	chiudi tutte le finestre eccetto quella corrente
CTRL-x 2	divide la finestra del buffer corrente in 2 (vert.)
CTRL-x 3	divide la finestra del buffer corrente in 2 (orizz.)
CTRL-v	scorrimento del testo in avanti
ALT-v	scorrimento del testo all'indietro

# Comandi principali di *emacs* (3)

## Comandi di spostamento del cursore

CTRL-a	sposta il cursore a inizio riga
CTRL-b	sposta il cursore a sinistra di 1 carattere
CTRL-n	sposta il cursore alla riga sottostante
ESC 6 CTRL-b	cursore a sinistra di 6 caratteri
ESC <	sposta il cursore a inizio buffer
ESC >	sposta il cursore a fine buffer

# Comandi principali di *emacs* (4)

## Comandi di selezione di blocchi

CTRL-barra spazio	segna l'inizio del blocco
ESC h	definisce come blocco il paragrafo corrente
CTRL-x CTRL-p	definisce come blocco la pagina
CTRL-w	cancella un blocco
ESC w	copia un blocco in un buffer di memoria

# Comandi principali di *emacs* (5)

## Comandi di cancellazione

CTRL-d

cancella il carattere a destra del cursore

BACKSPACE

cancella il carattere a sinistra del cursore

## Comandi di cancellazione con memorizzazione

CTRL-k

cancella la parte della riga a destra del cursore

ESC d

cancella parola dopo il cursore

ESC BACKSPACE

cancella parola prima del cursore

CTRL-y

inserisce dopo il cursore il testo cancellato

CTRL-\_

annulla il comando precedente

# Esempi

- **Esempio:** Sequenza che sposta la riga corrente in alto di tre righe.

```
CTRL-k  
ESC 3 CTRL-p  
CTRL y
```

- **Esempio:** Operazioni su paragrafi.

ESC h CTRL-w	cancella un paragrafo
ESC h ESC w	copia un paragrafo

- **Esempio:** Copia e incolla. Sequenza che copia la pagina corrente nel buffer di memoria e la incolla all'inizio del file.

```
CTRL-x CTRL-p ESC w  
ESC <  
CTRL-y
```



# Comandi principali di *emacs* (6)

## Comandi di ricerca di stringhe

CTRL-s	cerca un stringa in avanti
ESC CTRL-s	cerca un'espressione regolare in avanti
ESC x replace-string	esegue una sostituzione globale
ESC x replace-regexp	esegue una sostituzione con espressioni regolari
ESC %	esegue una sostituzione condizionale (query-replace)
ALT-x occur	cerca un'espressione regolare e salva il risultato

# Comandi principali di *emacs* (6)

## Espressioni Regolari

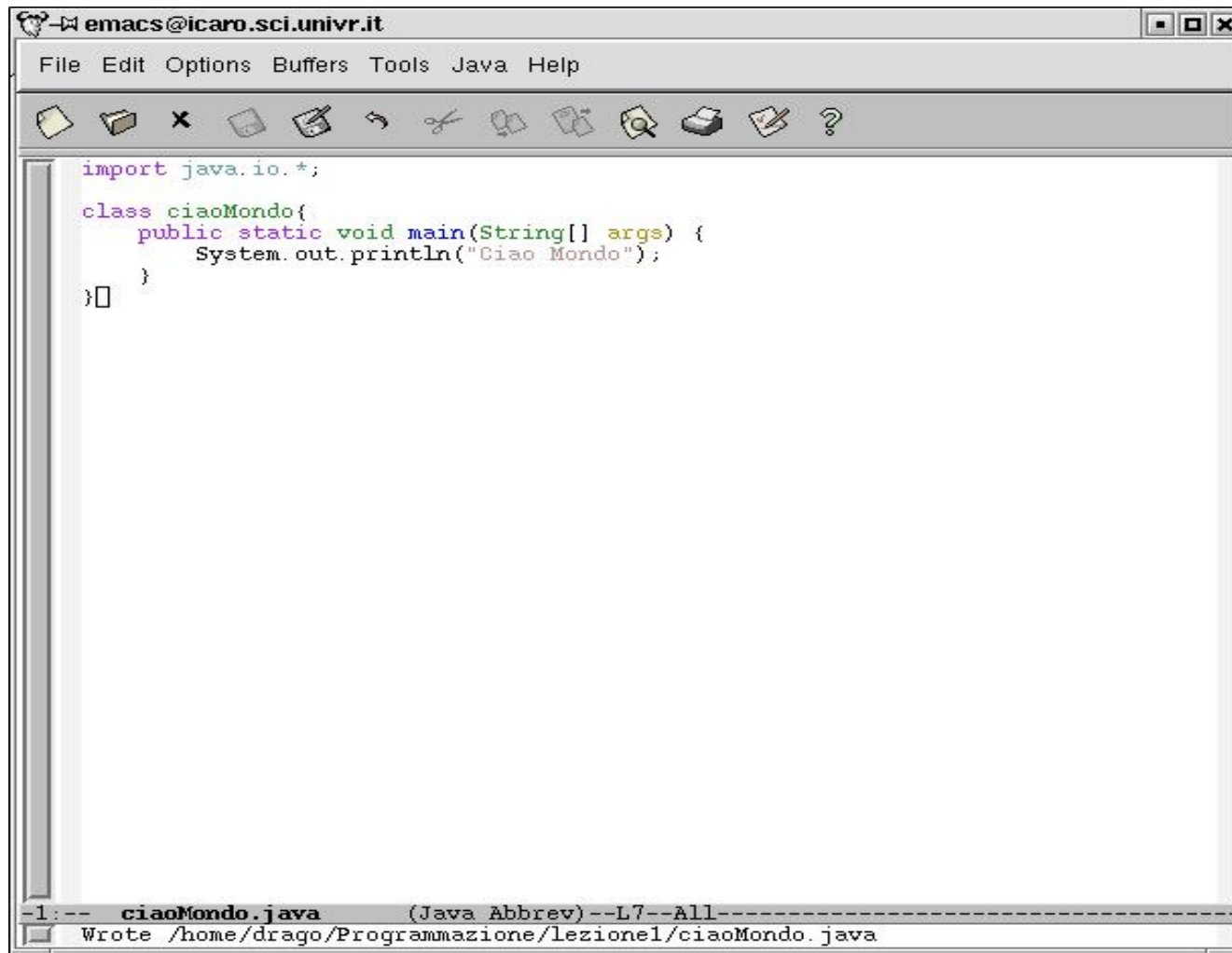
.	Un singolo carattere qualsiasi
*	Occorrenza del carattere precedente zero o piu' volte
+	Occorrenza del carattere precedente una o piu' volte
?	Occorrenza del carattere precedente zero o una volta
{n,m}	Occorrenza del carattere precedente almeno n e al max. m volte
[...]	Insiemi di caratteri
[^...]	Nega l'insieme di caratteri fra parentesi
^	Inizio della riga
\$	Fine della riga
\	Annulla il significato speciale del metacarattere seguente

# Esempio

```
ESC % stringa_1 [Invio] stringa_2 [Invio] opzione
```

- Questa sequenza di comandi permette di sostituire le occorrenze nel testo di **stringa\_1** con **stringa\_2**, con una procedura interattiva. Dopo il secondo comando di [Invio], all'utente viene chiesto di selezionare per l'occorrenza corrente un'opzione fra le seguenti:
  - **Y o barra spazio**  
Sostituisce e passa alla prossima occorrenza
  - **N o Canc**  
Non sostituisce e passa alla prossima
  - **^**  
Salta all'occorrenza precedente
  - **.**  
Sostituisce l'occorrenza ed esce

# Emacs: editing di sorgenti java



The screenshot shows the Emacs editor window with the title bar "emacs@icaro.sci.univr.it". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "Java", and "Help". The toolbar contains icons for file operations and editing. The main text area displays the following Java code:

```
import java.io.*;

class ciaoMondo{
    public static void main(String[] args) {
        System.out.println("Ciao Mondo");
    }
}
```

The status bar at the bottom shows the file name "ciaoMondo.java" and the line number "1". The message "Wrote /home/drago/Programmazione/lezionel/ciaoMondo.java" is displayed in the status bar.

# Scrivere il codice

---

Il codice deve essere:

- leggibile (!)
  - Ordinamento verticale delle istruzioni
  - Indentazione delle righe
- sintatticamente corretto
  - Aiuto dall'editor (o ambiente di sviluppo)
  - Messaggi dal compilatore
- semanticamente corretto
  - Warning del compilatore
  - Esperienza del programmatore

# Uso dei commenti

Per aumentare la leggibilità il codice va commentato:

- `//` Commento in riga  
`/*` Commento di un'area `*/`
- `//` Dichiarazione di variabile  
`Int I;`
- `/*`  
Programma per calcolare  
l'area del rettangolo  
`*/`  
`class rettangolo{`  
`.....`  
`}`

# Il primo programma

---

Per creare un programma è necessario:

- Dare un nome al nostro programma
- Dare lo stesso nome al file .java che contiene la classe (vedremo che cos'è) che definisce il programma

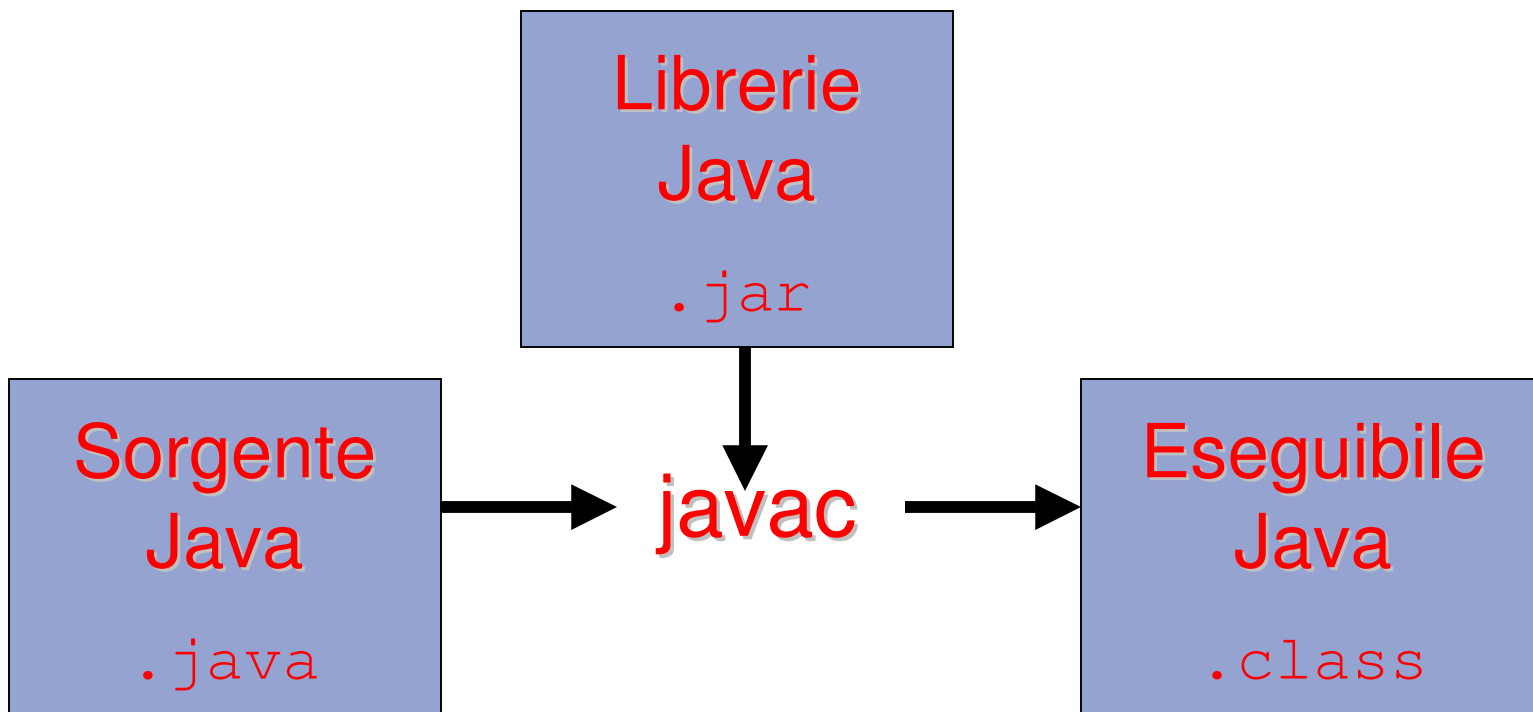
```
class ciaoMondo {  
...  
//corpo della classe  
...  
}
```



file ciaoMondo.java

# Il primo programma

Le librerie contengono programmi utili, che per semplicità il sistema ci mette a disposizione.





# Il primo programma

- Per compilare anche un programma elementare occorrono funzionalità di libreria:

```
import java.io.*
```

- E' necessario dichiarare un metodo di nome "main"

```
public static void main(String args[]){
```

```
...
```

```
//corpo del programma
```

```
...
```

```
}
```

- Scrivere le istruzioni necessarie per far svolgere al programma il compito desiderato

```
System.out.println("CiaoMondo");
```

# Il primo programma

---

I comandi per la compilazione e l'esecuzione di un programma sono:

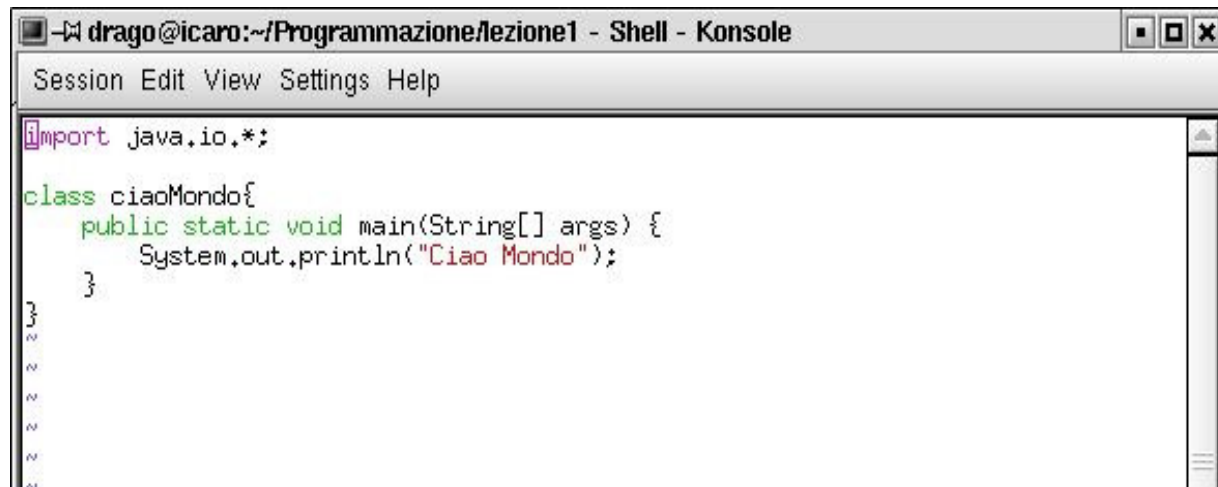
- **Compilazione del programma**

```
javac CiaoMondo.java
```

- **Esecuzione del bytecode**

```
java CiaoMondo
```

# Il primo programma



The image shows a terminal window titled "drago@icaro:~/Programmazione/lezione1 - Shell - Konsole". The window contains a Java program with the following code:

```
import java.io.*;

class ciaoMondo{
    public static void main(String[] args) {
        System.out.println("Ciao Mondo");
    }
}
```

The code is color-coded: `import` is purple, `java.io.*` is black, `class` is green, `ciaoMondo` is black, `public static void` is green, `main` is black, `(String[] args)` is black, `{` is black, `System.out.println` is black, `("Ciao Mondo")` is black, `;` is black, `}` is black, and `}` is black. The terminal window has a menu bar with "Session", "Edit", "View", "Settings", and "Help".

# Esercizi

---

- Scrivere ed eseguire un programma che quando viene eseguito visualizzi:  
  
Ciao sono il programma di <Nome Cognome>
- Modificare il programma precedente ed eseguirlo in modo che quando viene eseguito visualizzi:

Ciao sono il programma di <Nome Cognome>  
Questo è il mio secondo programma

# Esercizi

---

3. Scrivere ed eseguire un altro programma di nome “secondo” che quando viene eseguito visualizzi:

Questo è il terzo programma!  
Realizzato da <Nome Cognome>  
Buona Giornata!