

Laboratorio di Informatica

Lezione 1

Cristian Consonni

16 settembre 2015

Outline

1 Intro

- Informazioni generali

2 Java

- Cos'è Java
- Altri linguaggi

3 Variabili

- Definizione ed esempi
- Dichiarazione e assegnamento

4 Operatori

- Definizione
- Operatori aritmetici e booleani

5 Eclipse

- Installazione
- Creazione progetto

6 Istruzioni Condizionali

- Definizione

7 Esercizi

- Esercizi

Outline for section 1

1 Intro

- Informazioni generali

2 Java

- Cos'è Java
- Altri linguaggi

3 Variabili

- Definizione ed esempi
- Dichiarazione e assegnamento

4 Operatori

- Definizione
- Operatori aritmetici e booleani

5 Eclipse

- Installazione
- Creazione progetto

6 Istruzioni Condizionali

- Definizione

7 Esercizi

- Esercizi

Cristian Consonni

- **DISI - Dipartimento di Ingegneria e Scienza dell'Informazione**
- **Pagina web** del laboratorio:
<http://disi.unitn.it/~consonni/teaching>
- **Email:** cristian.consonni@unitn.it
- **Ufficio:** Povo 2 - Open Space 9
 - Per domande: scrivetemi una mail
 - Ricevimento: su appuntamento via mail

Obiettivi del laboratorio

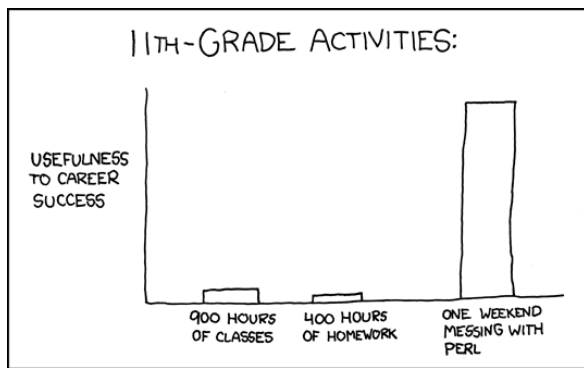
Obiettivi del laboratorio:

- Apprendere i fondamenti di un vero linguaggio di programmazione (Java)
- Svolgere il progetto

Obiettivi del laboratorio

- 1 Fare esperienza in laboratorio
- 2 Raggiungere una buona manualità nell'uso degli strumenti standard
- 3 Esercizi










Manualità (I)



<https://xkcd.com/519/>

Manualità (II)

How to Teach Yourself Programming:¹

<p>Days 1 - 10 Teach yourself variables, constants, arrays, strings, expressions, statements, functions,...</p> 	<p>Days 11 - 21 Teach yourself program flow, pointers, references, classes, objects, inheritance, polymorphism, ...</p> 	<p>Days 22 - 697 Do a lot of recreational programming. Have fun hacking but remember to learn from your mistakes.</p> 
<p>Days 698 - 3648 Interact with other programmers. Work on programming projects together. Learn from them.</p> 	<p>Days 3649 - 7781 Teach yourself advanced theoretical physics and formulate a consistent theory of quantum gravity.</p> 	<p>Days 7782 - 14611 Teach yourself biochemistry, molecular biology, genetics,...</p> 
<p>Day 14611 Use knowledge of biology to make an age-reversing potion.</p> 	<p>Day 14611 Use knowledge of physics to build flux capacitor and go back in time to day 21.</p> 	<p>Day 21 Replace younger self.</p> 

As far as I know, this
is the easiest way to
"Teach Yourself C++ in 21 Days".

¹<http://abstrusegoose.com/249>

Info sulle slide:

- le slide del corso saranno rese disponibili sul sito;
- segnalate pure eventuali errori;
- cercherò di pubblicare le slide in anticipo rispetto alla lezione;
- queste slide sono prodotte con \LaTeX Beamer (*usate \LaTeX !*);

Segnalazioni di materiale:

- Materiale da voi prodotto;
- Cose interessanti che trovate online;
- Possiamo valutare insieme se riutilizzarle;

Outline for section 2

- 1 Intro
 - Informazioni generali
- 2 Java
 - Cos'è Java
 - Altri linguaggi
- 3 Variabili
 - Definizione ed esempi
 - Dichiarazione e assegnamento
- 4 Operatori
 - Definizione
 - Operatori aritmetici e booleani
- 5 Eclipse
 - Installazione
 - Creazione progetto
- 6 Istruzioni Condizionali
 - Definizione
- 7 Esercizi
 - Esercizi



Cos'è Java (I)

Java Language Specification (788 pagg.)²

«*The Java[®] programming language is a general-purpose, [...] class-based, object-oriented language.*»

Java è:

- un **linguaggio** (grammatica, vocabolario, sintassi, ecc.);
- linguaggio di **programmazione**;
- general-purpose (vs domain-specific, e.g. *SQL*);
- orientato agli **oggetti** (attributi, metodi);
- class-based (**classe**, ereditarietà);

²<https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf>

Cos'è Java (II)

Altre caratteristiche di Java:

- **imperativo** (vs. funzionale vs. logico);

Paradigma imperativo

I linguaggi imperativi si fondano sull'idea di eseguire *comandi* che modificano lo *stato* del calcolatore.

Esempi:

C, C++, Java, C#, Python, Perl, Pascal, Ada, Fortran, JavaScript/ECMAScript, ...

- esistenza di uno stato, costantemente modificato
- il valore delle “variabili” può essere modificato dai comandi
- uso frequente dell'iterazione (ripetizione di un comando)
- spesso è possibile definire “funzioni”, che però assumono un significato un po' diverso da quello usuale
- ragionamento equazionale arduo: $f(x) + f(x) \neq 2 * f(x)$ in generale, perché $f(x)$ può modificare lo stato.
- molto usati nell'industria, in ogni ambito

Cos'è Java (II)

Altre caratteristiche di Java:

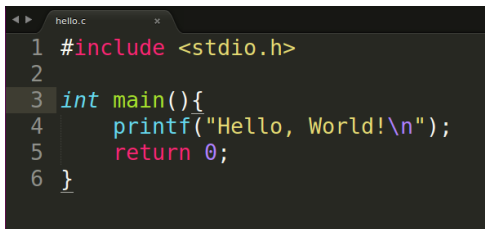
- **imperativo** (vs. funzionale vs. logico);
- **compilato** (vs. interpretato);
- **fortemente tipizzato**, *strongly typed* (vs. debolmente tipizzato)
- Molto usato in svariati ambiti;
- ...

Esistono moltissimi altri linguaggi:

- ad es. linguaggi di markup (e.g. HTML, XML, TeX)
- altri linguaggi **di programmazione**: C, C++, python, go, Scala, Prolog, Perl, ...

Altri linguaggi (II)

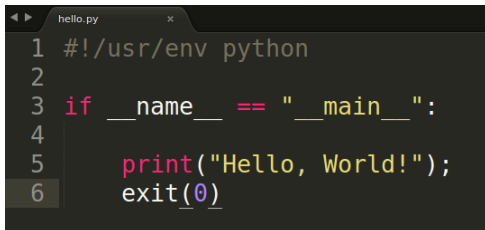
C:

A screenshot of a code editor window titled 'hello.c'. The editor shows a C program with six lines of code. Line 1: '#include <stdio.h>'. Line 2: an empty line. Line 3: 'int main(){'. Line 4: 'printf("Hello, World!\n");'. Line 5: 'return 0;'. Line 6: '}'. The code is color-coded: '#include' is pink, '<stdio.h>' is yellow, 'int' is green, 'main()' is green, 'printf' is blue, 'return' is pink, and '0' is blue. The cursor is at the end of line 6.

```
1 #include <stdio.h>
2
3 int main(){
4     printf("Hello, World!\n");
5     return 0;
6 }
```


Altri linguaggi (III)

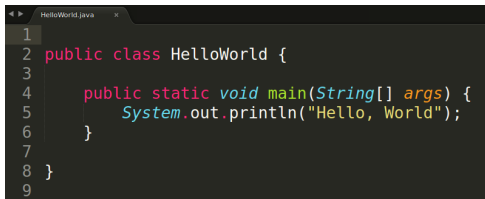
Python:

A screenshot of a code editor window titled 'hello.py'. The editor contains a Python script with six lines of code. Line 1 is a shebang line: '#!/usr/env python'. Line 2 is empty. Line 3 is an if statement: 'if __name__ == "__main__":'. Line 4 is empty. Line 5 is an indented print statement: ' print("Hello, World!");'. Line 6 is an indented exit statement: ' exit(0)'. The code is color-coded: 'if' is red, '==', 'print', and 'exit' are blue, and the string 'Hello, World!' is yellow. The line numbers 1 through 6 are on the left side of the editor.

```
1 #!/usr/env python
2
3 if __name__ == "__main__":
4
5     print("Hello, World!");
6     exit(0)
```

Altri linguaggi (IV)

Java:



```
1  
2 public class HelloWorld {  
3  
4     public static void main(String[] args) {  
5         System.out.println("Hello, World");  
6     }  
7  
8 }  
9
```

Altri linguaggi (V)

C:

```
[/tmp/hello]$ gcc -Wall hello.c -o hello  
[/tmp/hello]$  
[/tmp/hello]$ ./hello  
Hello, World!  
[/tmp/hello]$
```

Altri linguaggi (VI)

Python:

```
[/tmp/hello]$ python3 hello.py  
Hello, World!  
[/tmp/hello]$
```

Altri linguaggi (VII)

Java:

```
[/tmp/hello]$  
[/tmp/hello]$ javac HelloWorld.java  
[/tmp/hello]$  
[/tmp/hello]$ java HelloWorld  
Hello, World  
[/tmp/hello]$
```

Altri linguaggi (VIII)

C (bynary):

[illegible]

Altri linguaggi (IX)

Java (bytecode):

```
[/tmp/hello]$  
[/tmp/hello]$ cat HelloWorld.class  
000040: [00 00 00 00] init> [00 00] V [00 00] code [00 00] lineNumberTable [00 00] main [00 00] [Ljava/lang/String;)V [00 00]  
SourceFile [00 00] HelloWorld.java  
[00 00] [00 00]  
[00 00] [00 00] Hello, World [00 00]  
[00 00] [00 00]  
HelloWorld [00 00] java/lang/Object [00 00] java/lang/System [00 00] out [00 00] java/io/PrintStream; [00 00] java/io/PrintStream [00 00] printl  
n [00 00] [Ljava/lang/String;)V! [00 00] [00 00] [00 00]  
[00 00]  
[00 00] [00 00] % [00 00] 0 [00 00] [00 00]  
[00 00]  
[/tmp/hello]$
```

Pseudocode

Per esprimere un algoritmo senza adottare una sintassi legata ad un particolare linguaggio si usa lo **pseudocode**:

```
1:  $sum \leftarrow 0$ 
2: for  $i \leftarrow 1$  to  $N$  do
3:   for  $j \leftarrow 0$  to  $i$  do
4:     if  $i \bmod 2 = 0$  then
5:        $sum \leftarrow sum + 1$ 
6:     end if
7:   end for
8: end for
```


Pseudocode (II)

Per esprimere un algoritmo senza adottare una sintassi legata ad un particolare linguaggio si usa lo **pseudocode**:
(Esempio di dichiarazione di funzioni)

```
function INSERTIONSORT(Array  $x$ )  
  for  $i \leftarrow \text{length of } A$  do  
     $value \leftarrow A[i]$   
     $j \leftarrow i - 1$   
    while  $j \geq 0 \wedge A[j] > value$  do  
       $A[j + 1] \leftarrow A[j]$   
       $j \leftarrow j - 1$   
    end while  
  end for  
end function
```

Outline for section 3

- 1 Intro
 - Informazioni generali
- 2 Java
 - Cos'è Java
 - Altri linguaggi
- 3 Variabili
 - Definizione ed esempi
 - Dichiarazione e assegnamento
- 4 Operatori
 - Definizione
 - Operatori aritmetici e booleani
- 5 Eclipse
 - Installazione
 - Creazione progetto
- 6 Istruzioni Condizionali
 - Definizione
- 7 Esercizi
 - Esercizi

Variabili (I)

Variabile³:

*«In computer programming, a **variable** or scalar is a storage location paired with an associated symbolic name (an identifier), which contains some known or unknown quantity of information referred to as a value.»*

³da [https://en.wikipedia.org/wiki/Variable_\(computer_science\)](https://en.wikipedia.org/wiki/Variable_(computer_science))

Variabili (I)

Variabile³:

*«In computer programming, a **variable** or scalar is a **storage location** paired with an associated symbolic name (an identifier), which contains some known or unknown quantity of information referred to as a value.»*

³da [https://en.wikipedia.org/wiki/Variable_\(computer_science\)](https://en.wikipedia.org/wiki/Variable_(computer_science))

Variabili (I)

Variabile³:

«In computer programming, a **variable** or scalar is a *storage location* paired with an *associated symbolic name (an identifier)*, which contains some known or unknown quantity of information referred to as a value.»

³da [https://en.wikipedia.org/wiki/Variable_\(computer_science\)](https://en.wikipedia.org/wiki/Variable_(computer_science))

Variabili (I)

Variabile³:

«In computer programming, a **variable** or scalar is a *storage location* paired with an *associated symbolic name (an identifier)*, which contains some known or unknown quantity of information referred to as a *value*.»

³da [https://en.wikipedia.org/wiki/Variable_\(computer_science\)](https://en.wikipedia.org/wiki/Variable_(computer_science))

Variabili (II)

Esempi:

1 un intero:

- `int i = 0;`
- Pseudocodice $i \leftarrow 0$

2 un numero con la virgola:

- `double pi = 3.14;`
- (usate `Math.PI` per π)

3 un (singolo) carattere:

- `char k = 'c';`

4 un *valore booleano*:

- `bool val1 = true;`
- `bool val2 = false;`

5 una *stringa* di caratteri:

- `String yoda = "There is no try!";`

Variabili (II)

Esempi:

1 un intero:

- `int i = 0;`
- Pseudocodice $i \leftarrow 0$

2 un numero con la virgola:

- `double pi = 3.14;`
- (usate `Math.PI` per π)

3 un (singolo) carattere:

- `char k = 'c';`

4 un *valore booleano*:

- `bool val1 = true;`
- `bool val2 = false;`

5 una *stringa* di caratteri:

- `String yoda = "There is no try!";`

Variabili (II)

Esempi:

1 un intero:

- `int i = 0;`
- Pseudocodice $i \leftarrow 0$

2 un numero con la virgola:

- `double pi = 3.14;`
- (usate `Math.PI` per π)

3 un (singolo) carattere:

- `char k = 'c';`

4 un *valore booleano*:

- `bool val1 = true;`
- `bool val2 = false;`

5 una *stringa* di caratteri:

- `String yoda = "There is no try!";`

Variabili (II)

Esempi:

1 un intero:

- `int i = 0;`
- Pseudocodice $i \leftarrow 0$

2 un numero con la virgola:

- `double pi = 3.14;`
- (usate `Math.PI` per π)

3 un (singolo) carattere:

- `char k = 'c';`

4 un *valore booleano*:

- `bool val1 = true;`
- `bool val2 = false;`

5 una *stringa* di caratteri:

- `String yoda = "There is no try!";`

Variabili (II)

Esempi:

1 un intero:

- `int i = 0;`
- Pseudocodice $i \leftarrow 0$

2 un numero con la virgola:

- `double pi = 3.14;`
- (usate `Math.PI` per π)

3 un (singolo) carattere:

- `char k = 'c';`

4 un *valore booleano*:

- `bool val1 = true;`
- `bool val2 = false;`

5 una *stringa* di caratteri:

- `String yoda = "There is no try!";`

Variabili (II)

Esempi:

1 un intero:

- `int i = 0;`
- Pseudocodice $i \leftarrow 0$

2 un numero con la virgola:

- `double pi = 3.14;`
- (usate `Math.PI` per π)

3 un (singolo) carattere:

- `char k = 'c';`

4 un *valore booleano*:

- `bool val1 = true;`
- `bool val2 = false;`

5 una *stringa* di caratteri:

- `String yoda = "There is no try!";`

Variabili (II)

Esempi:

1 un intero:

- `int i = 0;`
- Pseudocodice $i \leftarrow 0$

2 un numero con la virgola:

- `double pi = 3.14;`
- (usate `Math.PI` per π)

3 un (singolo) carattere:

- `char k = 'c';`

4 un *valore booleano*:

- `bool val1 = true;`
- `bool val2 = false;`

5 una *stringa* di caratteri:

- `String yoda = "There is no try!";`

Variabili (II)

Esempi:

1 un intero:

- `int i = 0;`
- Pseudocodice $i \leftarrow 0$

2 un numero con la virgola:

- `double pi = 3.14;`
- (usate `Math.PI` per π)

3 un (singolo) carattere:

- `char k = 'c';`

4 un *valore booleano*:

- `bool val1 = true;`
- `bool val2 = false;`

5 una *stringa* di caratteri:

- `String yoda = "There is no try!";`

Variabili (II)

Esempi:

1 un intero:

- `int i = 0;`
- Pseudocodice $i \leftarrow 0$

2 un numero con la virgola:

- `double pi = 3.14;`
- (usate `Math.PI` per π)

3 un (singolo) carattere:

- `char k = 'c';`

4 un *valore booleano*:

- `bool val1 = true;`
- `bool val2 = false;`

5 una *stringa* di caratteri:

- `String yoda = "There is no try!";`

Variabili (II)

Esempi:

1 un intero:

- `int i = 0;`
- Pseudocodice $i \leftarrow 0$

2 un numero con la virgola:

- `double pi = 3.14;`
- (usate `Math.PI` per π)

3 un (singolo) carattere:

- `char k = 'c';`

4 un *valore booleano*:

- `bool val1 = true;`
- `bool val2 = false;`

5 una *stringa* di caratteri:

- `String yoda = "There is no try!";`

Variabili (II)

Esempi:

1 un intero:

- `int i = 0;`
- Pseudocodice $i \leftarrow 0$

2 un numero con la virgola:

- `double pi = 3.14;`
- (usate `Math.PI` per π)

3 un (singolo) carattere:

- `char k = 'c';`

4 un *valore booleano*:

- `bool val1 = true;`
- `bool val2 = false;`

5 una *stringa* di caratteri:

- `String yoda = "There is no try!";`

Variabili (II)

Esempi:

1 un intero:

- `int i = 0;`
- Pseudocodice $i \leftarrow 0$

2 un numero con la virgola:

- `double pi = 3.14;`
- (usate `Math.PI` per π)

3 un (singolo) carattere:

- `char k = 'c';`

4 un *valore booleano*:

- `bool val1 = true;`
- `bool val2 = false;`

5 una *stringa* di caratteri:

- `String yoda = "There is no try!";`

Variabili (II)

Esempi:

1 un intero:

- `int i = 0;`
- Pseudocodice $i \leftarrow 0$

2 un numero con la virgola:

- `double pi = 3.14;`
- (usate `Math.PI` per π)

3 un (singolo) carattere:

- `char k = 'c';`

4 un *valore booleano*:

- `bool val1 = true;`
- `bool val2 = false;`

5 una *stringa* di caratteri:

- `String yoda = "There is no try!";`

Una variable:

- Sono un “*contenitore*” di informazioni (= un certo numero di bytes allocati nella memoria volatile del computer (RAM));
- Contraddistinte da un identificatore, negli esempi di prima `i`, `pi`, `yoda`, ... (*case sensitive* `pippo` \neq `Pippo` \neq `PIPP0`);
- Nei linguaggi fortemente tipizzati hanno un **tipo**, negli esempi di prima `int`, `double`, `String`, ...;

Dichiarazione e assegnamento

Le Variabili possono essere create con la **dichiarazione**:

- **dichiarazione**: `int i;` (*specifica* il tipo di un identificatore);
- **assegnamento**: `i = 1;` (*assegna* un valore a un identificatore);
- **inizializzazione**: primo assegnamento `int i = 1;` (dichiarazione + inizializzazione);

Outline for section 4

- 1 Intro
 - Informazioni generali
- 2 Java
 - Cos'è Java
 - Altri linguaggi
- 3 Variabili
 - Definizione ed esempi
 - Dichiarazione e assegnamento
- 4 Operatori
 - Definizione
 - Operatori aritmetici e booleani
- 5 Eclipse
 - Installazione
 - Creazione progetto
- 6 Istruzioni Condizionali
 - Definizione
- 7 Esercizi
 - Esercizi

Definizione:⁴

«[Operator are] constructs which behave generally like functions, but which differ syntactically or semantically from usual functions»

⁴da [https://en.wikipedia.org/wiki/Operator_\(computer_programming\)](https://en.wikipedia.org/wiki/Operator_(computer_programming))

Operatori (II)

Gli operatori:

- funzioni disponibili in maniera predefinita all'interno di un linguaggio (ce ne sono molte altre);
- ritornano un risultato che avrà un certo **tipo**;
- **notazione infissa** (*infix notation*) operatore inserito tra gli operandi, ad es. $2 + 2$;
- **arietà** (o **adicità**) (numero di argomenti, v. *funzione variadica*):
 - **binari**: $+$ (addizione), $>$ (maggiore), $<=$ (minore uguale), $\&$ (*bitwise and*), $=$ (assegnamento), $[]$ (*bitwise and*);
 - **unari**: $-$ (sottrazione), (negazione), $++$ (decremento);
 - (in alcuni linguaggi **ma non in Java** tramite l'*overload* possono essere estese le funzionalità di un operatore);

Operatori aritmetici

Date tre variabili:

- `int x = 12;`
- `int y = 5;`
- `int z = 0;`

- `+`, `-`, `*` funzionano come vi aspettate
- `/` **attenzione!** `k = x / y;`
- `%` modulo (resto della divisione)

Operatori aritmetici

Date tre variabili:

- `int x = 12;`
- `int y = 5;`
- `int z = 0;`

- `+`, `-`, `*` funzionano come vi aspettate
- `/` **attenzione!** `k = x / y;`
- `%` modulo (resto della divisione)

Operatori aritmetici

Date tre variabili:

- `int x = 12;`
- `int y = 5;`
- `int z = 0;`

- `+`, `-`, `*` funzionano come vi aspettate
- `/` **attenzione!** `k = x / y;`
- `%` modulo (resto della divisione)

Operatori booleani

Anche in questo caso funziona tutto normalmente:

p	q	$\neg p$ (!p)	$p \wedge q$ (p && q)	$p \vee q$ (p q)
T	T	F	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	F

- **attenzione!** ai valori di verità delle variabili non booleane;
- `==` operatore di confronto;
- `a.equals(b)` confronto tra `stringhe (String)`;

Operatori booleani

Anche in questo caso funziona tutto normalmente:

p	q	$\neg p$ (!p)	$p \wedge q$ (p && q)	$p \vee q$ (p q)
T	T	F	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	F

- **attenzione!** ai valori di verità delle variabili non booleane;
- `==` operatore di confronto;
- `a.equals(b)` confronto tra `stringhe` (`String`);

Operatori booleani

Anche in questo caso funziona tutto normalmente:

p	q	$\neg p$ (!p)	$p \wedge q$ (p && q)	$p \vee q$ (p q)
T	T	F	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	F

- **attenzione!** ai valori di verità delle variabili non booleane;
- `==` operatore di confronto;
- `a.equals(b)` confronto tra stringhe (`String`);

Outline for section 5

- 1 Intro
 - Informazioni generali
- 2 Java
 - Cos'è Java
 - Altri linguaggi
- 3 Variabili
 - Definizione ed esempi
 - Dichiarazione e assegnamento
- 4 Operatori
 - Definizione
 - Operatori aritmetici e booleani
- 5 Eclipse
 - **Installazione**
 - **Creazione progetto**
- 6 Istruzioni Condizionali
 - Definizione
- 7 Esercizi
 - Esercizi

Eclipse (I)

Eclipse è un **ambiente di sviluppo integrato** (*IDE*) multi-linguaggio e multiplatforma.



Eclipse (II)

Eclipse è un **ambiente di sviluppo integrato (IDE)** multi-linguaggio e multiplatforma.

- software **libero** e **open source**;
- versione 1.0 rilasciata nel 2001, versione stabile 4.5.0 *Mars* (giugno 2015), voi avete 4.4.0 *Luna*;
- multiplatforma;
- estendibile con **plugins**;

Scaricare e installare Eclipse

- 1 Scaricare Java **JDK** (*Java Development Kit*) <http://www.oracle.com/technetwork/java/javase/downloads/index.html>;
- 2 Scaricare Eclipse <https://www.eclipse.org/downloads/>;
 - <https://www.cs.umd.edu/eclipse/>

Verificare che l'installazione di Java è andata a buon fine

Aprendo un **terminale** (o *shell*) (*nix) o “prompt dei comandi” (Windows):

```
$ java -version
```

```
java version "1.8.0_60"
```

```
Java(TM) SE Runtime Environment (build 1.8.0_60-b27)
```

```
Java HotSpot(TM) 64-Bit Server VM (build 25.60-b23, mixed mode)
```

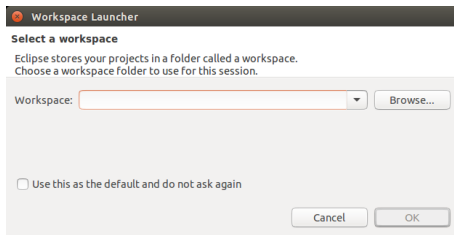
Esistono molte altre IDE:

- 1 **NetBeans**: <https://netbeans.org/>
- 2 **IntelliJ IDEA**: <https://www.jetbrains.com/idea/>

Avvio di Eclipse (I)



Workspace:





Eclipse Java EE IDE for Web Developers



Overview

Get an overview of the features



Tutorials

Go through tutorials



Samples

Try out the samples



What's New

Find out what is new



Quick Access

Java EE

Project Explorer



Outline

Task

An outline is not available.

Markers

Properties

Servers

Data Source Explorer

Snippets

0 items

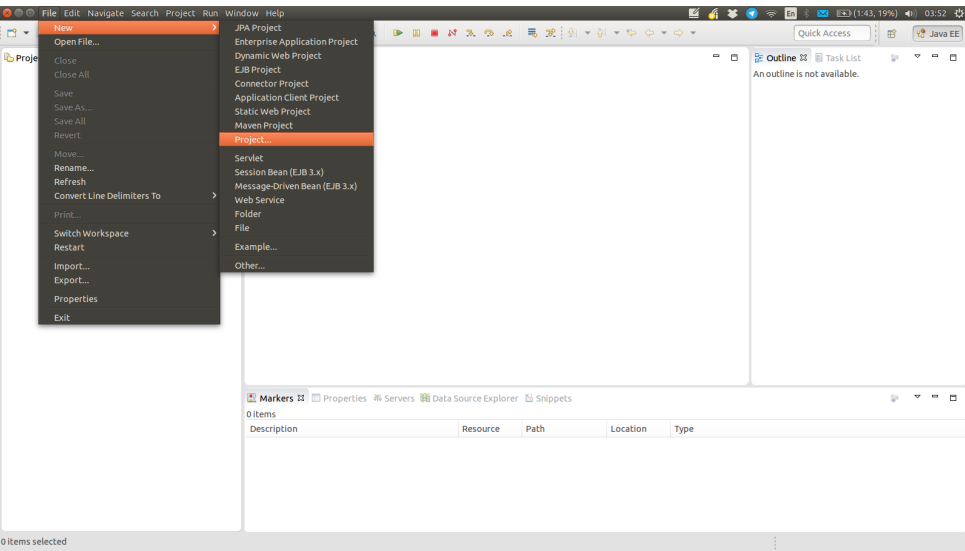
Description

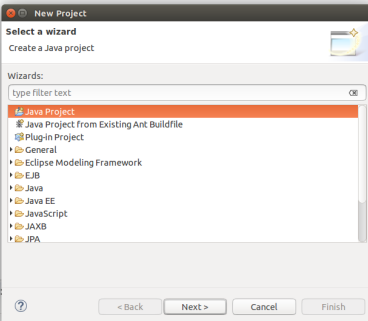
Resource

Path

Location

Type





Project Explorer

New Java Project

Create a Java Project
Create a Java project in the workspace or in an external location.

Project name:

☒ Use default location
Location: [Browse...](#)

JRE

☒ Use an execution environment JRE:

☐ Use a project specific JRE:

☐ Use default JRE (currently 'java-8-openjdk-amd64') [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

☐ Add project to working sets

Working sets: [Select...](#)

[?](#) [< Back](#) [Next >](#) [Cancel](#) [Finish](#)

Outline Task List

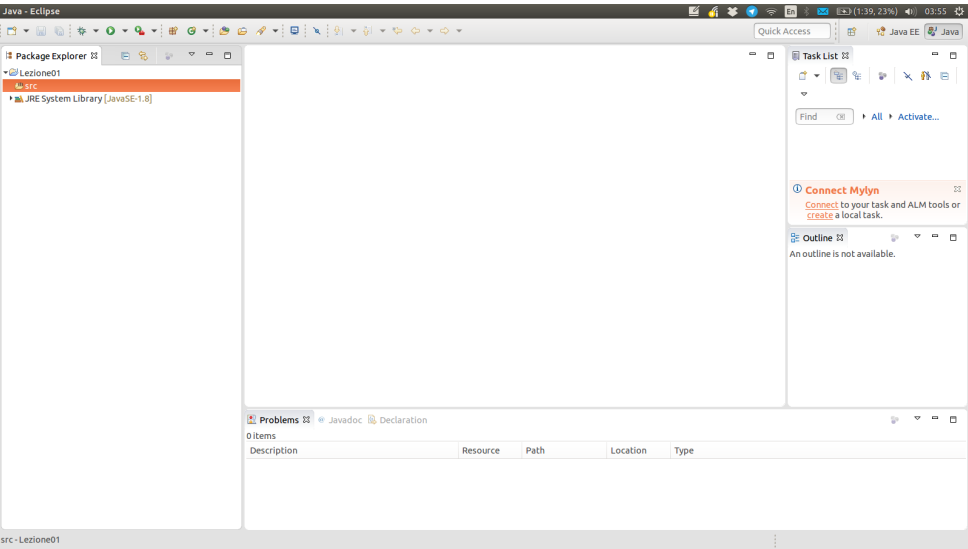
An outline is not available.

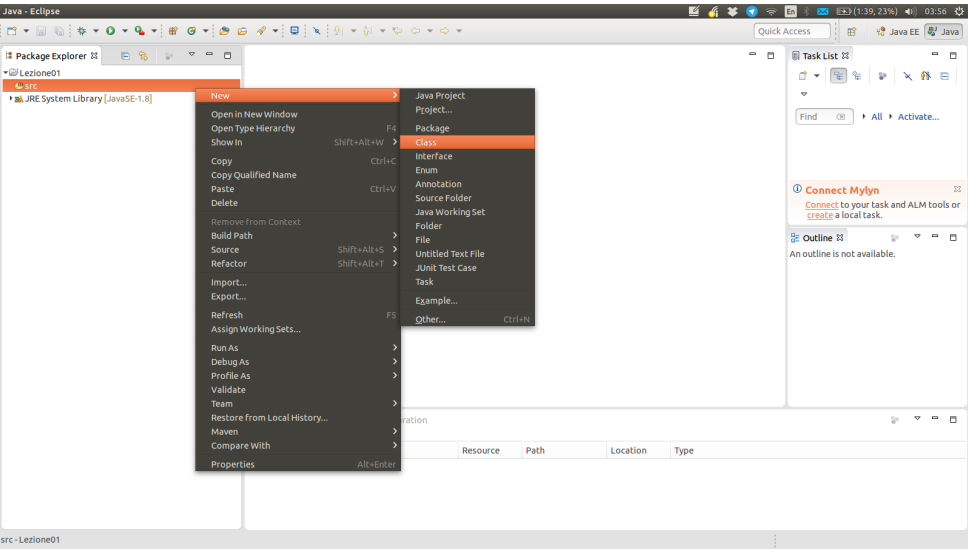
Markers

0 Items

Description

0 items selected





New Java Class

The use of the default package is discouraged.

Source Folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

- ☒ public static void main(String[] args)
- ☐ Constructors from superclass
- ☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

- ☐ Generate comments

The screenshot displays the Eclipse IDE with the following components:

- Package Explorer:** Shows the project structure with 'Lezione01' containing a 'src' package, which in turn contains 'CiaoMondo.java'.
- Editor:** Displays the source code of 'CiaoMondo.java':

```
1  
2 public class CiaoMondo {  
3  
4     public static void main(String[] args) {  
5         // TODO Auto-generated method stub  
6     }  
7  
8 }  
9  
10
```
- Task List:** Includes a 'Connect Mylyn' notification and an 'Outline' view showing the class 'CiaoMondo' and its 'main(String[]): void' method.
- Problems View:** Shows '0 items'.
- Bottom Panel:** Features a table with columns 'Description', 'Resource', 'Path', 'Local', 'Items', and 'Type'. The 'Items' column is currently empty.



Quick Access

Java EE Java

CiaoMondo.java

```
1
2 public class CiaoMondo {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6     }
7
8 }
9
10
```

Writable

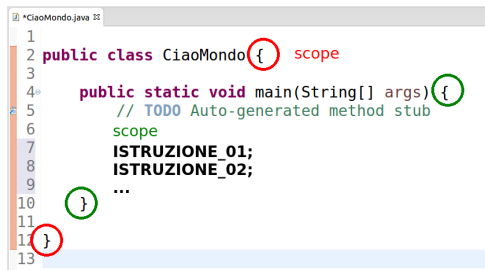
Smart Insert

1 : 1

*CiaoMondo.java

```
1
2 public class CiaoMondo {
3
4     public static void main(String[] args) {
5         TODO Auto-generated method stub method stub
6
7
8
9
10
11     }
12 }
13
```

Blocchi (I)



The image shows a code editor window titled "CiaoMondo.java". The code is as follows:

```
1
2 public class CiaoMondo { scope
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         scope
7         ISTRUZIONE_01;
8         ISTRUZIONE_02;
9         ...
10    }
11
12 }
13
```

Annotations in the image:

- A red circle highlights the opening curly brace of the class `CiaoMondo` on line 2, with the word "scope" written next to it.
- A green circle highlights the opening curly brace of the `main` method on line 4.
- A green circle highlights the closing curly brace of the `main` method on line 10.
- A red circle highlights the closing curly brace of the class `CiaoMondo` on line 12.

Blocchi (II)

```
*CiaoMondo.java
1
2 public class CiaoMondo {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         --> IL VOSTRO CODICE VA QUI <--
8
9
10    }
11
12 }
13 |
```

Outline for section 6

- 1 Intro
 - Informazioni generali
- 2 Java
 - Cos'è Java
 - Altri linguaggi
- 3 Variabili
 - Definizione ed esempi
 - Dichiarazione e assegnamento
- 4 Operatori
 - Definizione
 - Operatori aritmetici e booleani
- 5 Eclipse
 - Installazione
 - Creazione progetto
- 6 Istruzioni Condizionali
 - Definizione
- 7 Esercizi
 - Esercizi

Istruzione if

Le **istruzioni condizionali** permettono di effettuare operazioni diverse a seconda dei valori delle variabili.

1: **if** **condizione** **then**

2: istruzione 1

3: **else**

4: istruzione 2

5: **end if**

condizione deve essere una **espressione booleana**.

Istruzione if

In Java:

```
if (condizione) {  
    comando1  
} else {  
    comando2  
}
```

Outline for section 7

- 1 Intro
 - Informazioni generali
- 2 Java
 - Cos'è Java
 - Altri linguaggi
- 3 Variabili
 - Definizione ed esempi
 - Dichiarazione e assegnamento
- 4 Operatori
 - Definizione
 - Operatori aritmetici e booleani
- 5 Eclipse
 - Installazione
 - Creazione progetto
- 6 Istruzioni Condizionali
 - Definizione
- 7 Esercizi
 - Esercizi

- 1 Iniziate dai programmi più semplici;
- 2 I commenti sono importanti (ma ci sono 2 scuole di pensiero);
- 3 la leggibilità del codice è importante
(<https://www.python.org/dev/peps/pep-0020/>);

- 1 Dichiarare e inizializzare un intero e stampare a video se è pari o dispari;
- 2 Definire un programma che dati tre numeri a , b e x stabilisca se $x \in [a, b]$;
- 3 Definire un programma che dati tre numeri a , b e c stabilisca quale è il massimo;
- 4 Definire un programma che dati tre numeri a , b e c li stampi in ordine crescente;
- 5 Definire un programma che dati due numeri a , b restituisca la divisione (intera) a/b ed il resto di tale divisione;