



UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA
DIPARTIMENTO DI ECCELLENZA

Università di degli Studi di Salerno

Dipartimento di Informatica

Programmazione ad Oggetti

a.a. 2023-2024

Istallazione JDK

Docente: Prof. Massimo Ficco

E-mail: *micco@unisa.it*

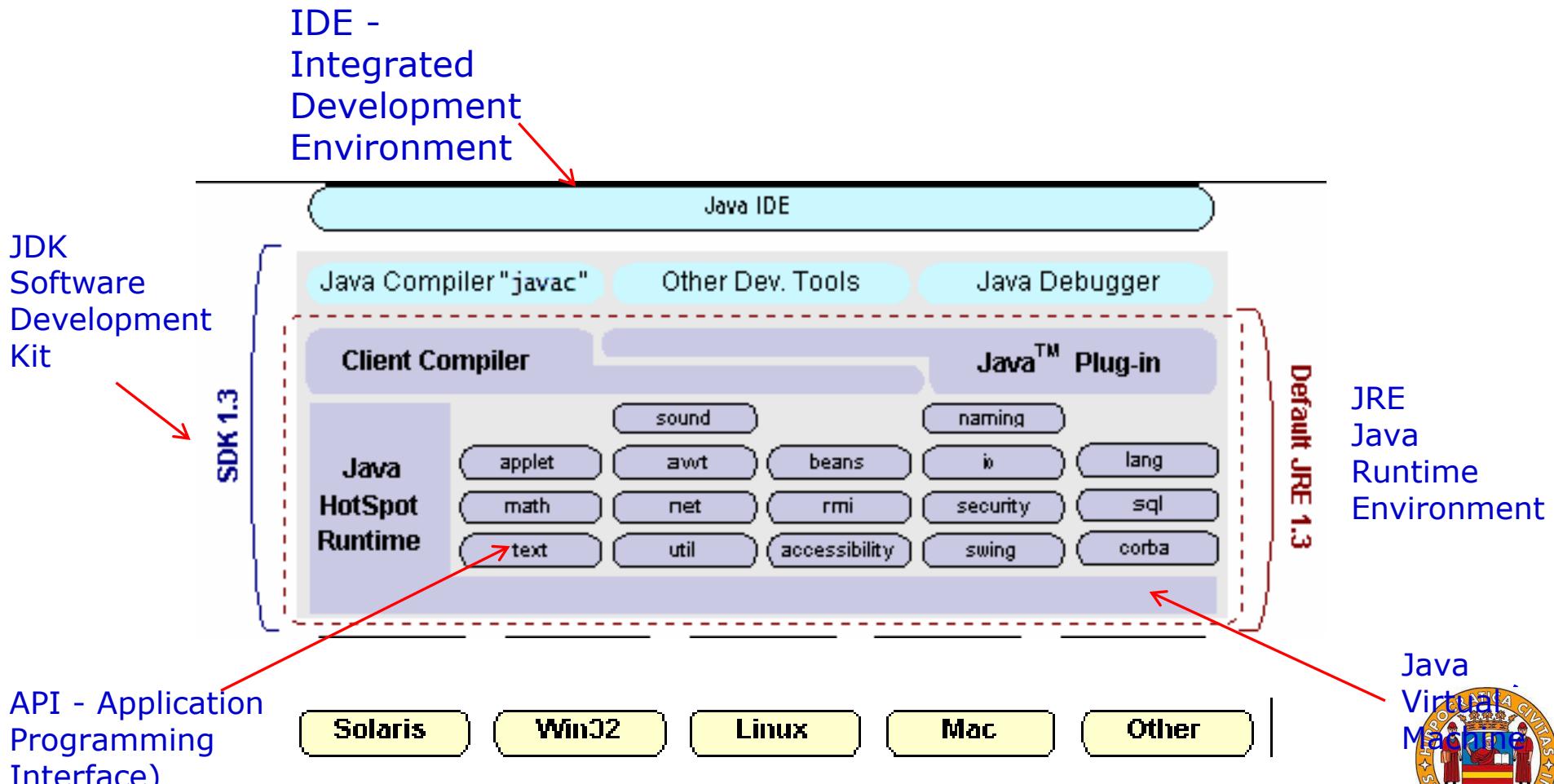
Argomenti affrontati

Nella lezione si descriveranno brevemente le procedure necessarie per l'installazione del kit di sviluppo per Java sotto i sistemi operativi Windows.

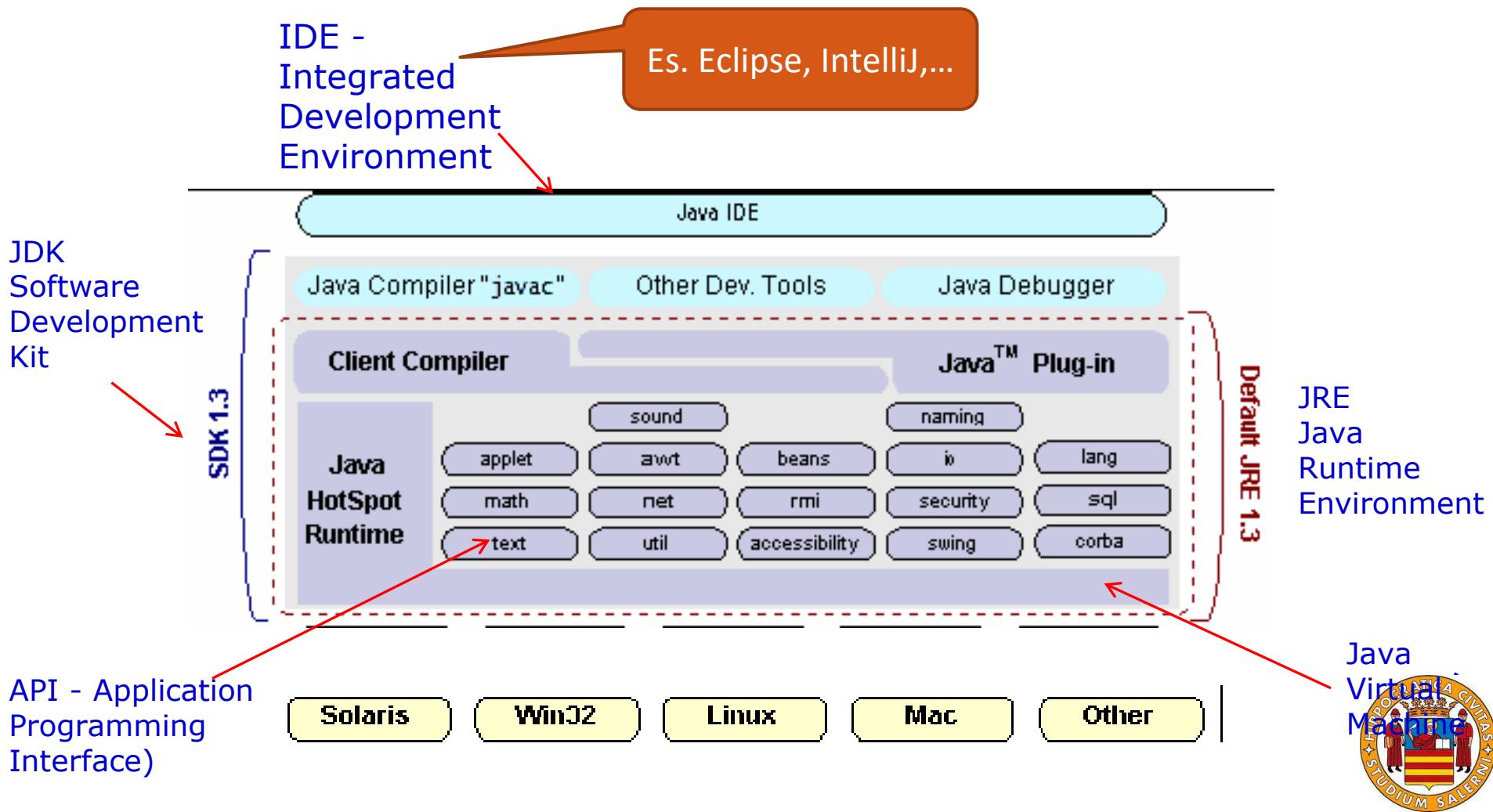
Si vedrà inoltre come scrivere e compilare un semplice programma Java.



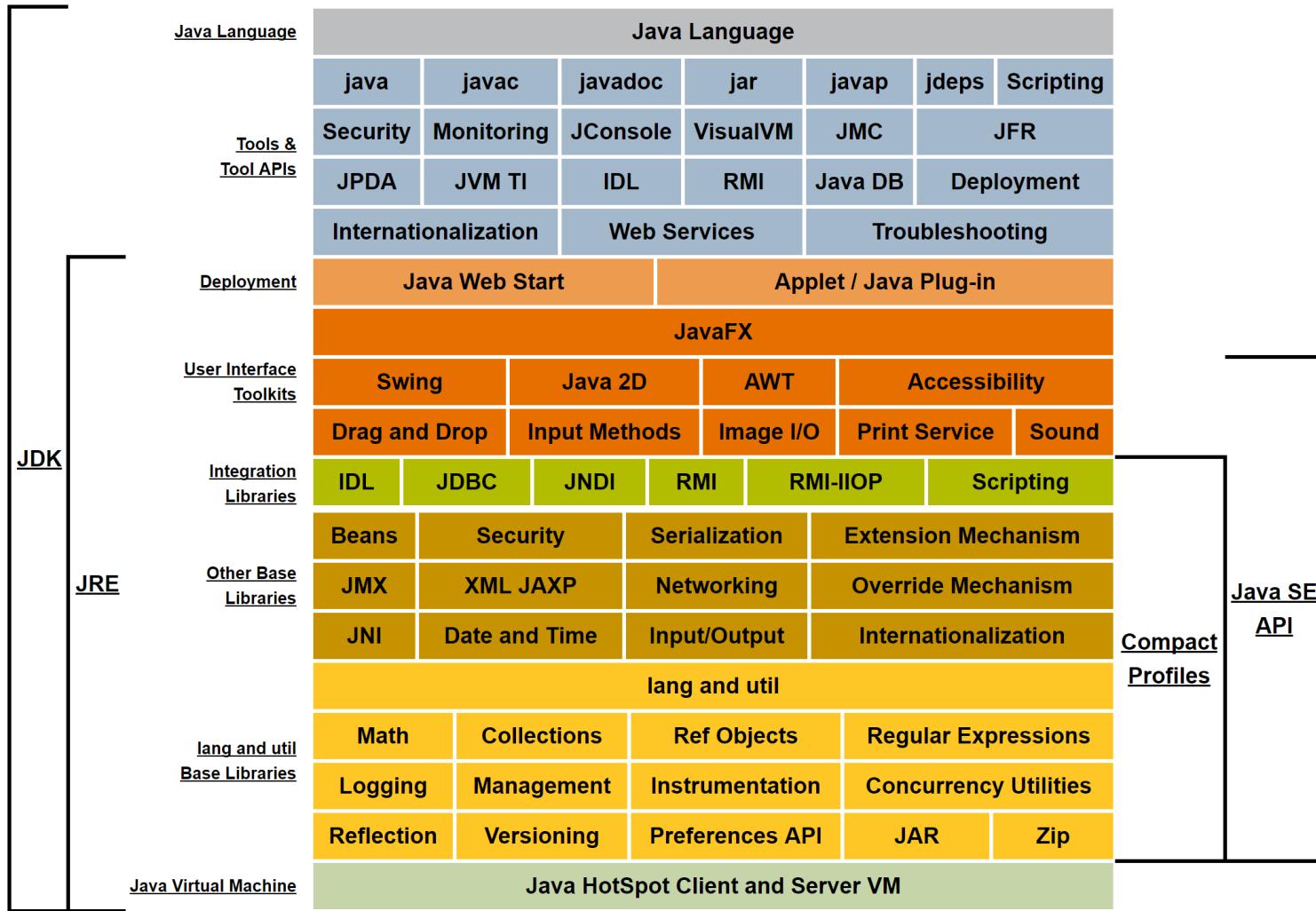
La Piattaforma Java



La Piattaforma Java



Java Platform Standard Edition 8



JDK

- ▶ Java SE 16.0.2 – Java SE Development Kit 16
 - ▶ Scaricabile, collegandosi al link:
 - ▶ <https://www.oracle.com/java/technologies/javase/jdk16-archive-downloads.html>
 - ▶ Il Java Development Kit (JDK) è un insieme base di programmi che consente di far girare applicazioni scritte nel linguaggio Java.
 - ▶ <https://www.oracle.com/java/technologies/downloads/>
- ▶ I programmi più importanti sono il compilatore Java (programma `javac`) che traduce il sorgente Java in codice eseguibile dalla macchina virtuale Java, e l'esecutore (programma `java`), che implementa la vera e propria macchina virtuale.



JDK

- ▶ Occorre scaricare solo JDK 16
 - ▶ Esiste la versione per Windows:
 - ▶ jdk-16.0.1_windows-x64_bin.exe (oppure jdk SE 17)
 - ▶ macOS Installer
 - ▶ jdk-16.0.1_osx-x64_bin.dmg
 - ▶ Linux RPM (RedHat Package Manager) in self-extracting file:
 - ▶ jdk-16.1_linux-x64_bin.rpm
 - ▶ **Attenzione, per Ubuntu non utilizzate questo**
 - ▶ **<https://ubuntuhandbook.org/index.php/2021/03/oracle-java-16-released-install-ubuntu-20-04/>**
- ▶ Documentazione sulla vasta raccolta di API Java si può consultare al link:
 - ▶ **<https://docs.oracle.com/en/java/javase/16/docs/api>**





Products Industries Resources Customers Partners Developers Events Company



[View Accounts](#)

Java / Technologies / JavaSE /
Java SE 16 Archive Downloads

Java SE 16 Archive Downloads

Go to the [Oracle Java Archive page](#).

The JDK is a development environment for building applications using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java™ platform.

Important Oracle JDK License Update

The Oracle JDK License has changed for releases starting April 16, 2019.

The new [Oracle Technology Network License Agreement for Oracle Java SE](#) is substantially different from prior Oracle JDK licenses. The new license permits certain uses, such as personal use and development use, at no cost -- but other uses authorized under prior Oracle JDK licenses may no longer be available. Please review the terms carefully before downloading and using this product. An FAQ is available [here](#).

Commercial license and support is available with a low cost [Java SE Subscription](#).

Oracle also provides the latest OpenJDK release under the open source [GPL License](#) at [jdk.java.net](#).

WARNING: These older versions of the JRE and JDK are provided to help developers debug issues in older systems. **They are not updated with the latest security patches and are not recommended**



Installazione del JDK sul disco fisso

Se l'installazione va a termine correttamente sul disco fisso sarà presente la directory **jdkx.x.x** con le seguenti sotto directory:

- **bin**, contiene il compilatore e gli altri eseguibili;
- **lib**, etc.

JRE: contiene l'interprete, estensioni, ed altre informazioni di configurazione

docs: contiene la documentazione (*N.B.: da scaricare separatamente*)



API Documentation

OVERVIEW MODULE PACKAGE CLASS USE TREE DEPRECATED INDEX HELP Java SE 16 & JDK 16

SEARCH: Search X

Java® Platform, Standard Edition & Java Development Kit Version 16 API Specification

This document is divided into two sections:

Java SE
The Java Platform, Standard Edition (Java SE) APIs define the core Java platform for general-purpose computing. These APIs are in modules whose names start with `java`.

JDK
The Java Development Kit (JDK) APIs are specific to the JDK and will not necessarily be available in all implementations of the Java SE Platform. These APIs are in modules whose names start with `jdk`.

All Modules	Java SE	JDK	Other Modules
Module	Description		
<code>java.base</code>	Defines the foundational APIs of the Java SE Platform.		
<code>java.compiler</code>	Defines the Language Model, Annotation Processing, and Java Compiler APIs.		
<code>java.datatransfer</code>	Defines the API for transferring data between and within applications.		
<code>java.desktop</code>	Defines the AWT and Swing user interface toolkits, plus APIs for accessibility, audio, imaging, printing, and JavaBeans.		
<code>java.instrument</code>	Defines services that allow agents to instrument programs running on the JVM.		
<code>java.logging</code>	Defines the Java Logging API.		

Moduli →



API Documentation

OVERVIEW MODULE PACKAGE CLASS USE TREE DEPRECATED INDEX HELP Java SE 16 & JDK 16

MODULE: DESCRIPTION | MODULES | PACKAGES | SERVICES SEARCH: Search X

Exports

Package	Description
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.
java.lang.annotation	Provides library support for the Java programming language annotation facility.
java.lang.constant	Classes and interfaces to represent <i>nominal descriptors</i> for run-time entities such as classes or method handles, and classfile entities such as constant pool entries or <code>invokedynamic</code> call sites.
java.lang.invoke	The <code>java.lang.invoke</code> package provides low-level primitives for interacting with the Java Virtual Machine.
java.lang.module	Classes to support module descriptors and creating configurations of modules by means of resolution and service binding.
java.lang.ref	Provides reference-object classes, which support a limited degree of interaction with the garbage collector.
java.lang.reflect	Provides classes and interfaces for obtaining reflective information about classes and objects.
java.lang.runtime	The <code>java.lang.runtime</code> package provides low-level runtime support for the Java language.
java.math	Provides classes for performing arbitrary-precision integer arithmetic (<code>BigInteger</code>) and arbitrary-precision decimal arithmetic (<code>BigDecimal</code>).

Packages



API Documentation

OVERVIEW MODULE PACKAGE CLASS USE TREE DEPRECATED INDEX HELP		Java SE 15 & JDK 15
SEARCH: <input type="text"/> <input type="button"/>		
Class Summary		
Class	Description	
Boolean	The Boolean class wraps a value of the primitive type <code>boolean</code> in an object.	
Byte	The Byte class wraps a value of primitive type <code>byte</code> in an object.	
Character	The Character class wraps a value of the primitive type <code>char</code> in an object.	
Character.Subset	Instances of this class represent particular subsets of the Unicode character set.	
Character.UnicodeBlock	A family of character subsets representing the character blocks in the Unicode specification.	
Class<T>	Instances of the class <code>Class</code> represent classes and interfaces in a running Java application.	
ClassLoader	A class loader is an object that is responsible for loading classes.	
ClassValue<T>	Lazily associate a computed value with (potentially) every type.	
Compiler	Deprecated, for removal: This API element is subject to removal in a future version. <i>JIT compilers and their technologies vary too widely to be controlled effectively by a standardized interface.</i>	
Double	The Double class wraps a value of the primitive type <code>double</code> in an object.	
Enum<E extends Enum<E>>	This is the common base class of all Java language enumeration types.	
Enum.EnumDesc<E extends Enum<E>>	A nominal descriptor for an enum constant.	
Float	The Float class wraps a value of primitive type <code>float</code> in an object.	
InheritableThreadLocal<T>	This class extends <code>ThreadLocal</code> to provide inheritance of values from parent thread to child thread: when a child thread is created, the child receives initial values for all inheritable thread-local variables for which the parent has values.	
Integer	The Integer class wraps a value of the primitive type <code>int</code> in an object.	
Long	The Long class wraps a value of the primitive type <code>long</code> in an object.	
Math	The class <code>Math</code> contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.	
Module	Represents a run-time module, either <code>named</code> or <code>unnamed</code> .	
ModuleLayer	A layer of modules in the Java virtual machine.	
ModuleLayer.Controller	Controls a module layer.	
Number	The abstract class <code>Number</code> is the superclass of platform classes representing numeric values that are convertible to the primitive types <code>byte</code> , <code>double</code> , <code>float</code> , <code>int</code> , <code>long</code> , and <code>short</code> .	

Classi



Prompt dei comandi

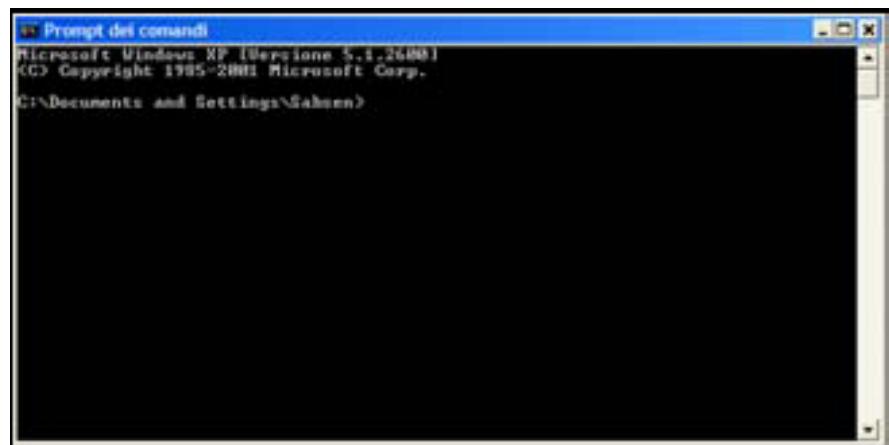
I programmi (tools) forniti con Java presenti nella directory bin non prevedono un'interfaccia grafica e quindi devono essere eseguiti dall'interprete dei comandi (shell) di Windows (detto Prompt dei comandi)

L'interprete dei comandi è in genere eseguibile dal menu

- Avvio -> Programmi -> Accessori -> Prompt dei comandi

Appare una finestra con un cursore lampeggiante dopo una stringa del tipo
C:\WINDOWS>

Tale stringa indica la directory corrente.



PATH e CLASSPATH

Cosa è una variabile di ambiente:

- Una variabile del sistema operativo e visibile da tutte le applicazioni
- Può essere settata temporaneamente in una finestra
- Può essere settata per l'intero sistema

PATH: contiene la lista delle directory dove il sistema operativo cerca i comandi da eseguire

CLASSPATH: contiene la lista delle directory dove il compilatore java e l'interprete cercano le classi

JAVA_HOME: per far sapere ad applicazioni che utilizzano JAVA dove si trova la cartella di Java



La variabile di sistema PATH

- ▶ Indica all'interprete dei comandi le directory dove cercare i programmi eseguibili
- ▶ Per utilizzare in modo efficiente e pratico i programmi presenti nella directory bin si configura la variabile di sistema PATH
- ▶ Essa contiene un elenco di directory separate da un carattere:
 - ▶ ; nei sistemi Windows
 - ▶ : nei sistemi Unix (Linux)



Come rendere operativo l'ambiente JAVA

Aprite il prompt dei comandi e provate a lanciare il comando `java -version` se avete l'errore «Comando non trovato» c'è bisogno di qualche piccola configurazione

Dopo aver installato JDK con la procedura guidata occorre settare le variabili di ambiente del proprio sistema per includere alcuni riferimenti al kit JDK.

Per il sistema operativo Windows occorre modificare la variabile Path del sistema introducendo un riferimento alla directory bin dove si trova il programma javac (il compilatore java).

- Se la cartella Java si chiama jdk16 e si trova installata nel percorso C:\jdk16 bisogna scrivere:
- PATH= ...; C:\jdk16\bin



...usando Windows

Saltate le prossime slide se `java -version` funziona e restituisce una versione ≥ 16

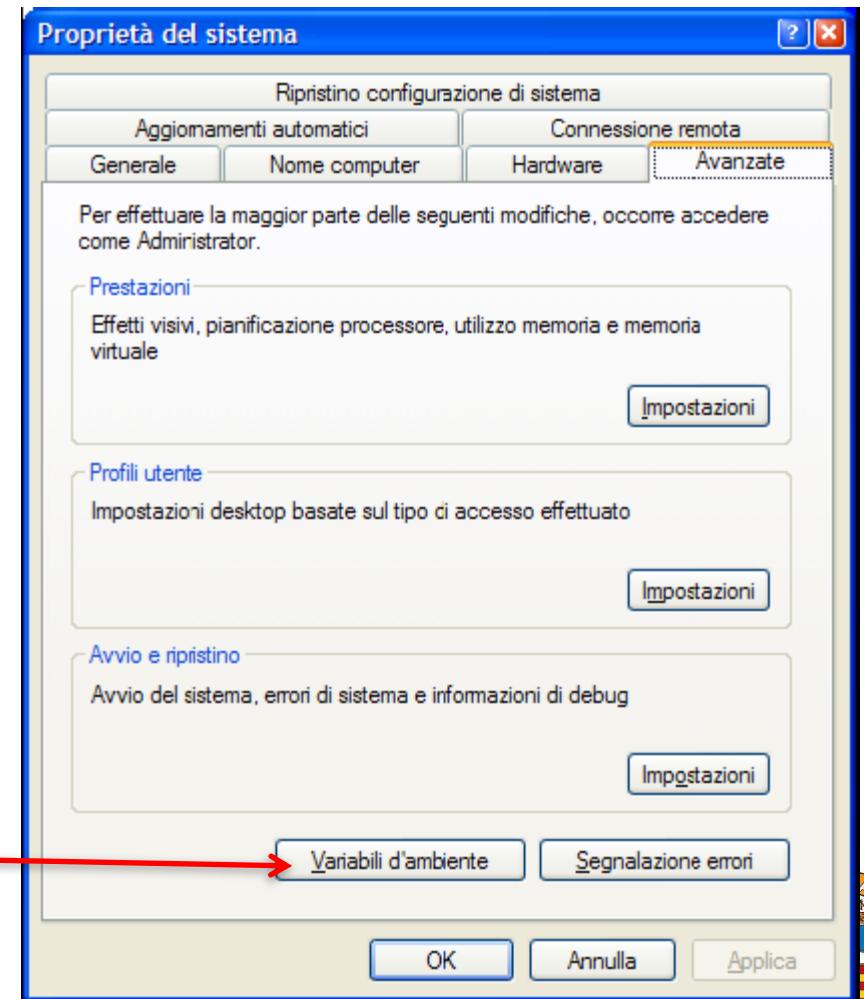
Selezionare in Windows 8 o precedente:

- Start →
- Pannello di Controllo →
- Sistema

O, in Windows 10 e 11,

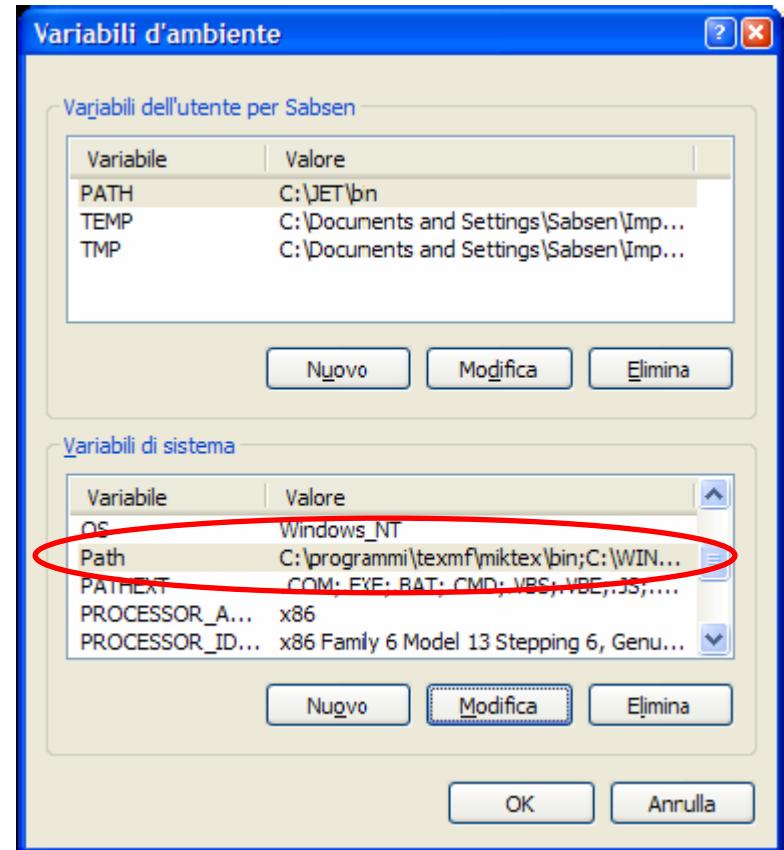
- Tasto windows + R
- `sysdm.cpl`

*Cliccare su **Variabili d'ambiente***



...usando Windows

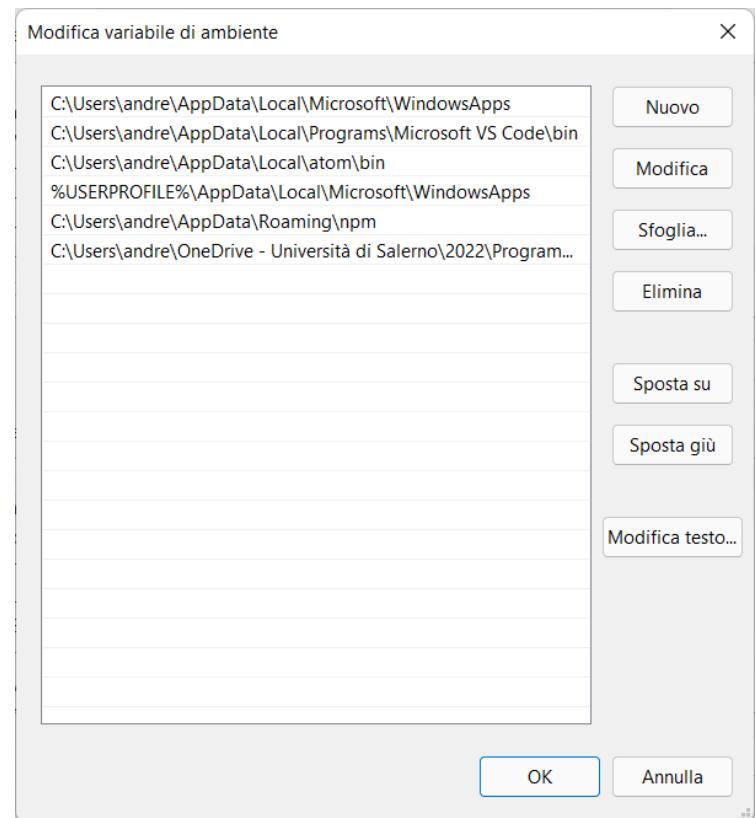
- ▶ Tra le variabili di sistema,
 - ▶ Selezionare la variabile **Path** e
 - ▶ cliccare sul bottone **Modifica**



...usando Windows

Cliccate su «Sfoglia...»

Selezzionate il path degli eseguibili
java (es. C:\jdk16\bin)



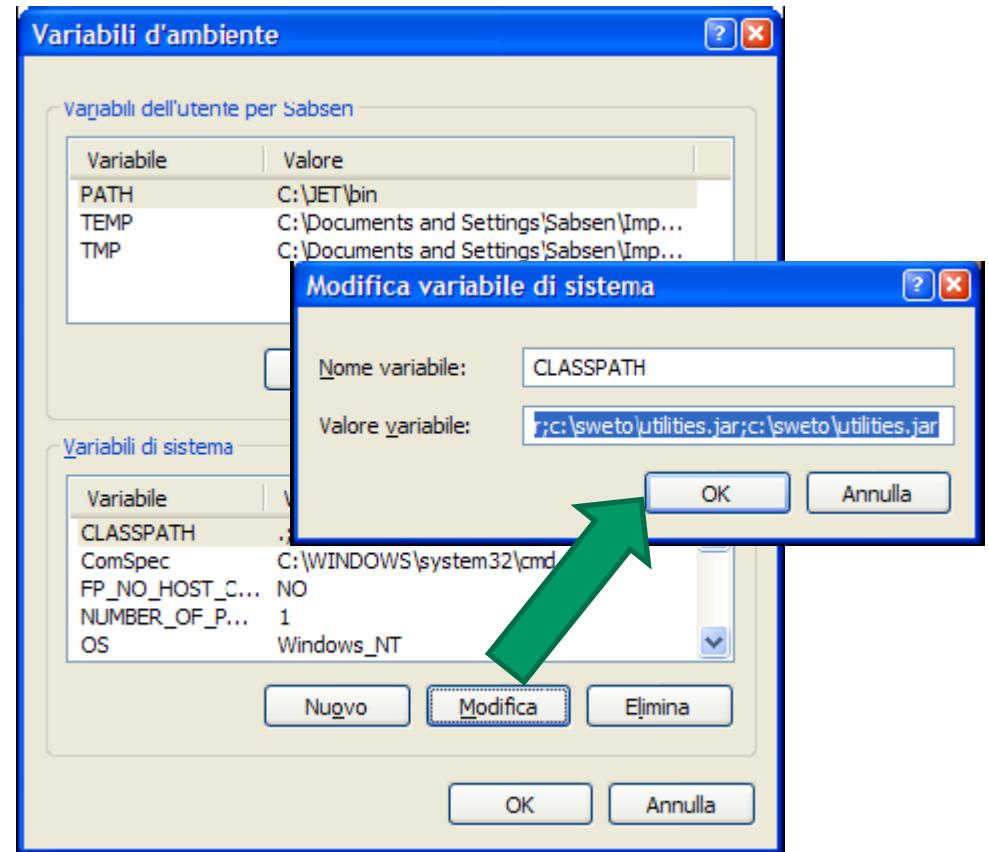
Il Classpath

- ▶ Il parametro classpath serve per indicare i percorsi in cui ricercare i file **.class**.
- ▶ Il compilatore e la JVM sanno già dove trovare i package forniti con la distribuzione di Java.
- ▶ Ogni volta che dobbiamo usare nuovi package (di solito, file con estensioni jar) dobbiamo specificare il percorso in cui si trovano. Ad esempio se la directory C:\MyProject ha i file importati allora potremmo definire:
 - ▶ CLASSPATH = .; C:\MyProject ;
- ▶ In questo modo stiamo dicendo al compilatore e alla JVM che quando importiamo una classe devono:
 - ▶ iniziare a cercarla in . (la directory corrente)...
 - ▶ ... se non la trova deve provare a cercarla in C:\MyProject



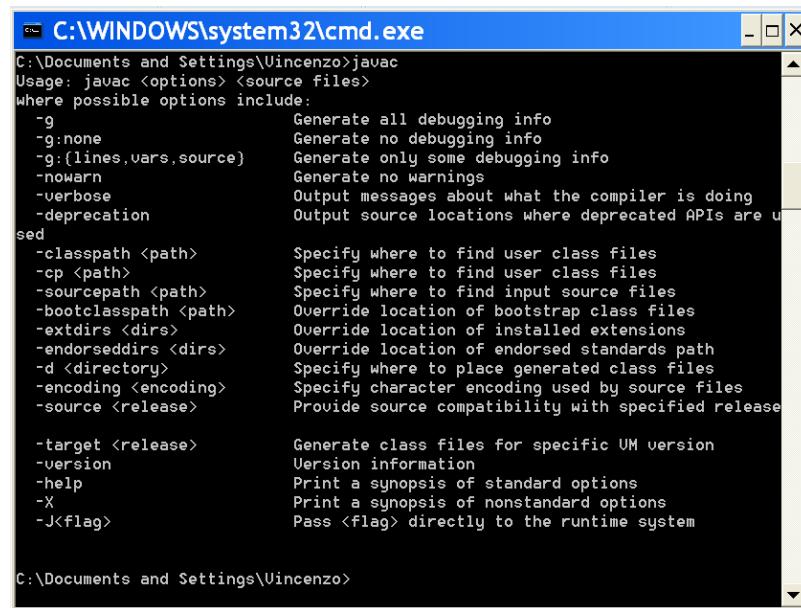
Il Classpath

- Si opera come per il parametro Path.
- Si va a settare la variabile d'ambiente CLASSPATH.



Per controllare la correttezza...

- ▶ Aprire la shell di DOS
- ▶ (Start → Programmi → Accessori → Prompt dei comandi)
- ▶ Es.: Digitare javac
 - ▶ Informazioni per i comandi di compilazione
- ▶ Digitare java –version
 - ▶ Informazioni sulla versione corrente della JVM



```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\Vincenzo>javac
Usage: javac <options> <source files>
Where possible options include:
  -g                         Generate all debugging info
  -g:none                     Generate no debugging info
  -g:{lines,vars,source}       Generate only some debugging info
  -nowarn                     Generate no warnings
  -verbose                    Output messages about what the compiler is doing
  -deprecation                Output source locations where deprecated APIs are u
sed
  -classpath <path>           Specify where to find user class files
  -cp <path>                  Specify where to find user class files
  -sourcepath <path>          Specify where to find input source files
  -bootclasspath <path>       Override location of bootstrap class files
  -extdirs <dirs>             Override location of installed extensions
  -endorseddirs <dirs>        Override location of endorsed standards path
  -d <directory>              Specify where to place generated class files
  -encoding <encoding>        Specify character encoding used by source files
  -source <release>           Provide source compatibility with specified release

  -target <release>          Generate class files for specific VM version
  -version                    Version information
  -help                       Print a synopsis of standard options
  -X                         Print a synopsis of nonstandard options
  -J<flag>                   Pass <flag> directly to the runtime system

C:\Documents and Settings\Vincenzo>
```



Per eseguire un programma Java

- ▶ Aprire NotePad (Windows) o un altro Editor di Testo
 - ▶ Provate Notepad++ (<https://notepad-plus-plus.org/>)
- ▶ Scrivere il codice e salvare il file, aggiungendo l'estensione **.java**.
- ▶ Salvarlo in una cartella, ad esempio c:\es_java
- ▶ Dalla shell di DOS, raggiungere la cartella con il path c:\es_java (attraverso il comando cd di MS-DOS)
- ▶ Eseguire il comando (compilazione):
 - ▶ javac <nome_file>.java
- ▶ Eseguire il comando (esecuzione):
 - ▶ java <nome_file>



Errori

- ▶ **Errore di sintassi**
 - ▶ violazione delle regole del linguaggio di programmazione
 - ▶ rilevati dal compilatore
- ▶ **Errore logico**
 - ▶ il programma esegue un'azione che non era nelle intenzioni del programmatore
 - ▶ rilevati nella fase di test del programma
- ▶ NOTA:
 - ▶ Java è **case sensitive**, distingue tra maiuscole e minuscole



Primo programma Java

- ▶ Scrivere in un editore di testo il seguente codice di fianco
- ▶ Salvare il file con nome **Program1.java**
- ▶ Dalla shell di DOS, raggiungere la cartella con il file (attraverso il comando cd di MS-DOS)
- ▶ Compilare il file eseguendo il comando a riga di comando
 - ▶ javac Program1.java
- ▶ Eseguire il programma con il comando
 - ▶ java Program1

```
▶ public class Program1 {  
▶     public static void main(String[] args){  
▶         System.out.println("Benvenuti al corso");  
▶     }  
▶ }
```



Primo programma Java

DEVONO ESSERE UGUALI!!!

- ▶ Scrivere in un editore di testo il seguente codice di fianco
- ▶ Salvare il file con nome **Program1.java**
- ▶ Dalla shell di DOS, raggiungere la cartella con il file (attraverso il comando cd di MS-DOS)
- ▶ Compilare il file eseguendo il comando a riga di comando
 - ▶ javac Program1.java
- ▶ Eseguire il programma con il comando
 - ▶ java Program1

```
▶ public class Program1 {  
▶     public static void main(String[] args){  
▶         System.out.println("Benvenuti al corso");  
▶     }  
▶ }
```



Scrittura del programma di esempio

Il modo più semplice per editare un file sorgente Java è quello di adoperare un programma di scrittura come Notepad o WordPad di Windows.

Supponiamo di voler scrivere il seguente programma Java, che stampa a video un messaggio di saluto:



Un editor Avanzato

A metà strada tra un editor standard ed un ambiente di sviluppo integrato vi sono alcuni editor come **Texpad**, **Jcreator**, **Jedit** o **Kawa**.

Il vantaggi rispetto ad un editor standard sono due:

Le parole chiave del linguaggio sono evidenziate, questo permette una maggiore leggibilità del codice;

E' possibile compilare ed eseguire un programma direttamente dall'editor.



HELLO WORLD



L'esempio più semplice HELLO WORLD

Affinché un applicazione ad oggetti “parta”, devo avere una classe con un metodo statico e pubblico di nome **main**

Hello.java

```
class Hello {  
    public static void main (String args [ ]) {  
        System.out.println("Hello World!");  
    }  
}
```

obbligatorio in questa forma



SPIEGAZIONI

public static void main (String args [])

- **void** indica che *main* non ritorna nulla, il che è necessario per superare il type-checking del compilatore
- *args[]* sono gli argomenti passati a *main* dalla shell quando si digita:
java Hello arg1 arg2 ... argn
- *String* dice che gli argomenti sono di classe *String*
- **public** rende il metodo main visibile alle altre classi - e al comando *java* (interprete)
- **static** associa *main* alla classe *Hello*, e non alle sue istanze

System.out.println("HelloWorld!")

- invoca il metodo *println* dell'oggetto *out* della classe *System*, che stampa la stringa sul file *stdout*



Come scrivere un Programma Java

- Creare un file testo e salvarlo con estensione **.java** (es. **Shirt.java**)
- Creare una classe il cui nome sia lo stesso del file
- Compilare il file (es. a linea di comando tramite il compilatore **javac**) viene creato un file **Shirt.class**

Command Prompt

```
C:\sun\examples\module3>javac.exe Shirt.java
Shirt.java:1 : 'class' or 'interface' expected
public Class Shirt {
    ^
1 error

C:\sun\examples\module3>
```

- Lanciare la Java Virtual Machine specificando il programma da eseguire

java Shirt



Passare argomenti all'applicazione da linea di comando

Per eseguire un applicazione java e passargli un argomento, basta far seguire al nome dell' applicazione il valore desiderato: verrà interpretato come una stringa e memorizzato nell'array di stringhe args[] definito in main

Esempio: **java Shirt verde**

```
// Determina se passata una stinga sulla riga di comando

public class Shirt {
    public static void main(String args[]) {
        System.out.println("Stringa digitata: " + args[0]);
    }
}
```



Passare argomenti all'applicazione da linea di comando

Per eseguire un applicazione java e passargli un argomento, basta far seguire al nome dell' applicazione il valore desiderato: verrà interpretato come una stringa e memorizzato nell'array di stringhe args[] definito in main

Esempio: **java Shirt verde**

```
// Determina se passata una stinga sulla riga di comando

public class Shirt {
    public static void main(String args[]) {
        if (args.length < 1) {
            System.out.println("Nessun argomento");
        }
        else {
            System.out.println("Stringa digitata: " + args[0]);
        }
    }
}
```



Un altro esempio

In questo secondo esempio si adopererà la libreria grafica di Java, creando una dialog box per l'immissione di alcuni dati. Lo scopo del programma è solo quello di mostrare un primo esempio di output grafico, non è necessario comprendere tutte le istruzioni.

Il testo sorgente è:



Testo secondo esempio

```
import javax.swing.*;  
  
public class InputTest {  
  
    public static void main(String[] args) {  
        String nome;  
        nome=JOptionPane.showInputDialog ("Come ti chiami? ");  
        System.out.println("Salve "+nome);  
    }  
}
```



Compilazione ed esecuzione del secondo esempio

Il file va salvato con lo stesso nome della classe con in più l'estensione .java (InputTest.java). Per la generazione del bytecode il comando sarà:

```
javac InputTest.java
```

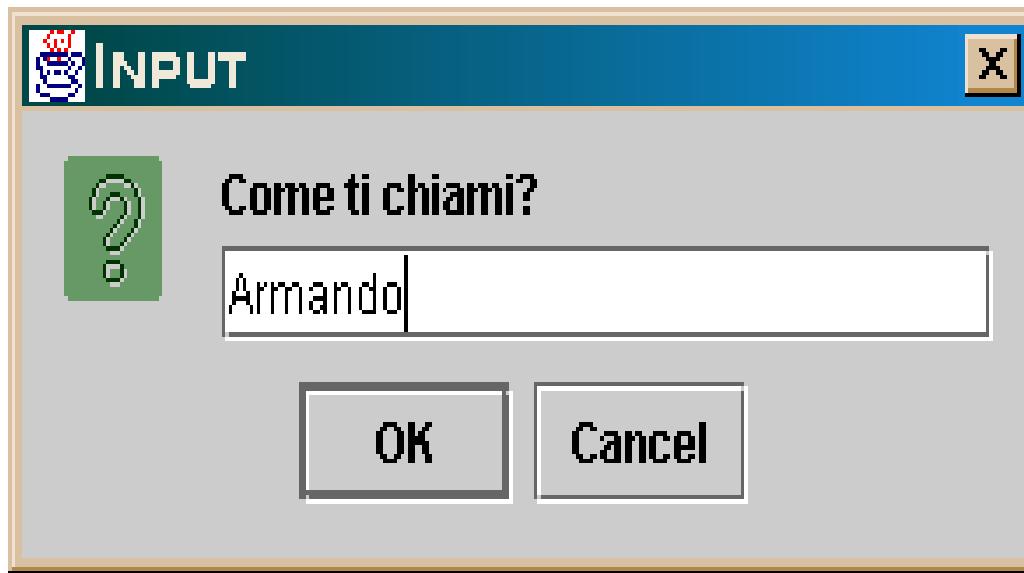
Per l'esecuzione:

```
java InputTest
```



Esecuzione del programma

Eseguendo il programma compare la seguente finestra di input:



Esecuzione del programma

Immettendo dei caratteri e premendo OK si otterrà un output su console.

