



Introduzione al linguaggio Java



Java (breve storia)

- **1991**. Sun Microsystems creava *Green* per apparecchiature di consumo (accessori per la TV).
 - Scarso successo.
- **1995**. Browser Hot-Java sfruttava le caratteristiche di Green
 - neutralità rispetto all'architettura
 - esecuzione in tempo reale
 - affidabilità
 - sicurezza applicandole all'interazione client/server:
 - il browser Hot-Java poteva scaricare dal Web programmi (applet) scritti in Java ed eseguirli
- **1996**. Netscape ed Explorer supportavano Java (versione 1.0)
 -
- **2004**. Java versione 5.0
- **2006**. Java versione 6.0
- **2011**. Java versione 7.0
- **2014**. Java versione 8.0



JAVA: caratteristiche generali (1)

- E' object-oriented:
 - risponde all'esigenza di realizzare sistemi software facili da modificare e mantenere
 - consente alti livelli di *riutilizzabilità* del codice
- Ha una ricchissima libreria per lo sviluppo di interfacce utente e di applicazioni Internet impiegabili con relativa facilità
- E' robusto
 - Una delle principali cause di *crash* dei programmi scritti in C/C++ è l'uso scorretto dell'aritmetica dei puntatori:
 - non fornisce tipi puntatori, né tanto meno l'aritmetica dei puntatori



JAVA: caratteristiche generali (2)

- E' efficiente pur essendo un linguaggio interpretato:
 - i programmi Java sono mediamente meno di 10 volte più lenti dei corrispondenti programmi C++
 - riduzione di efficienza accettabile per tipiche applicazioni Java: programmi interattivi
 - meglio di altri linguaggi interpretati (Basic, PERL, etc.)
- E' sicuro:
 - esecuzione programmi confinata in un "firewall" da cui non è possibile accedere ad altre parti del computer
 - estremamente utile per l'esecuzione di programmi scaricati da internet

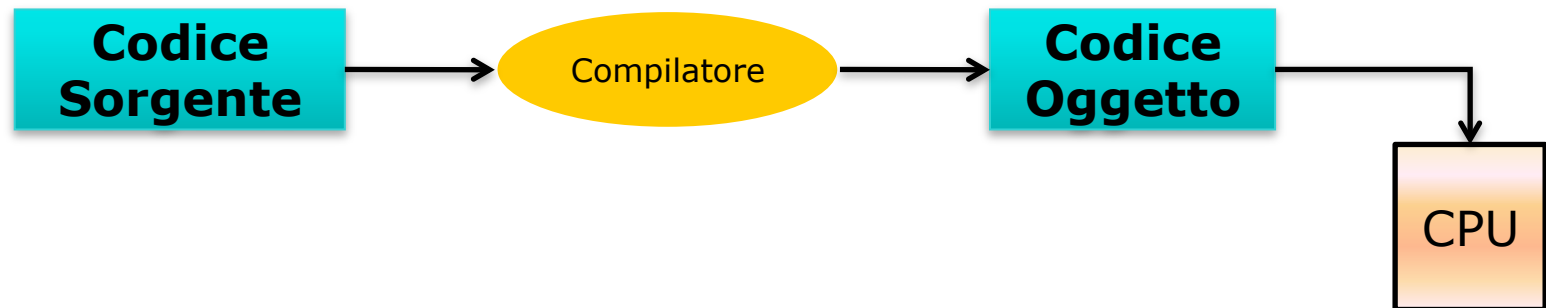


JAVA: caratteristiche generali (3)

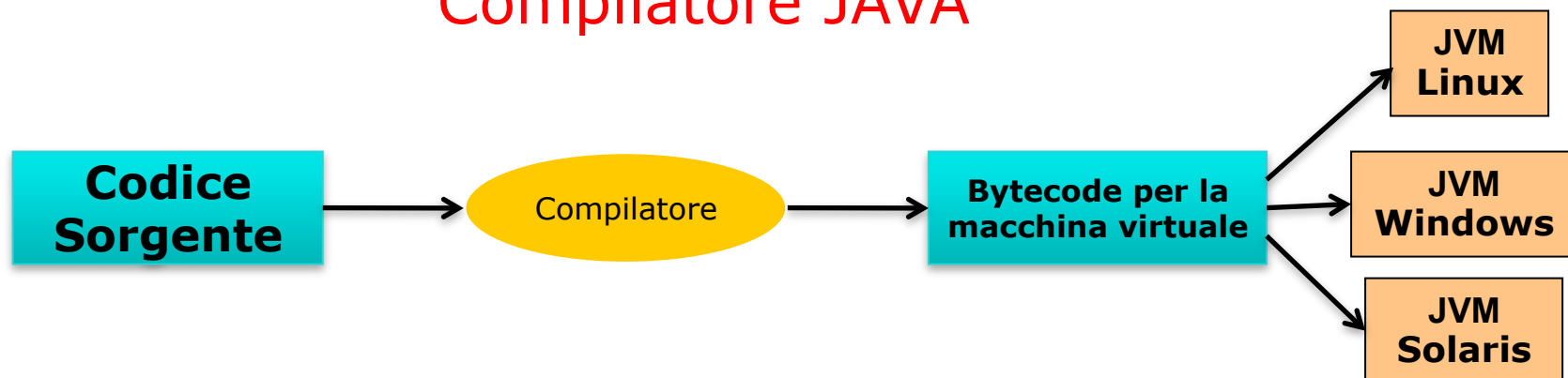
- E' portabile:
 - Programmi scritti in linguaggi convenzionali (es. C/C++) devono essere ricompilati per la nuova piattaforma
 - Programmi scaricabili da internet: bisogna predisporre l'eseguibile per ogni tipo di CPU o mettere a disposizione il codice sorgente
 - JAVA:
il compilatore genera un codice (Bytecode) eseguibile per una CPU virtuale, detta macchina virtuale (JVM)
 - La macchina virtuale viene poi simulata su una CPU reale

Bytecode e JVM

Compilatore convenzionale



Compilatore JAVA



- 

```
Reverse.class - Blocco note
File Modifica Formato Visualizza ?

Eb% . D Reverse; O java/lang/Object; O reverse_arr; C [C; O init; I V)
O code; I V) O O O O O O O O LineNumberTable LocalVariableTable this;
LReverse; O O O invertio; I C)V; O O C C C C C C printArr; O O O
O array_char; O java/lang/System; O out; Ljava/io/PrintStream; " # !
$ java/io/PrintStream; O print; C V) ( ) * main;
O Ljava/lang/String; O java/lang/String; . length; C I) O 0 1 / 2 O O O 40
charAt; I C) 6 7 / 8 O O O O args; Ljava/lang/String; O array_sto
Ljava/lang/String; O size; PreArray; SourceFile; Reverse.java ! O O O O O
O O H O O * % μ ±
O O R O O * * % d d d u ±
O O O O O O O O < / *. O += % N O d 6 O 6 $
O - O d + O 4 U n O O y i * q ± &
O / O O O O O $ O O O O O ! * * . ^ O O
O O % * % 4 + n O O * % y e ± O # O $ O
O , - O O 2 * 2 L + 3 = O Y O . 5 N O 6 $ O - O + O 9 l ; n O O O y i - q ±
O " O ( ) * + $ * - , 1 1 4 0 2 < = . . ? O )
@ O A O B O C
```

Java usa lo schema di codifica **Unicode**

- E' uno schema di codifica più ricco di ASCII/ANSI
- ASCII/ANSI infatti è un sotto-insieme dell'Unicode

range **Unicode (inizialmente)**: \u0000 \uFFFF

range **ASCII/ANSI:** \u0000 \u00FF

Bytecode e JVM

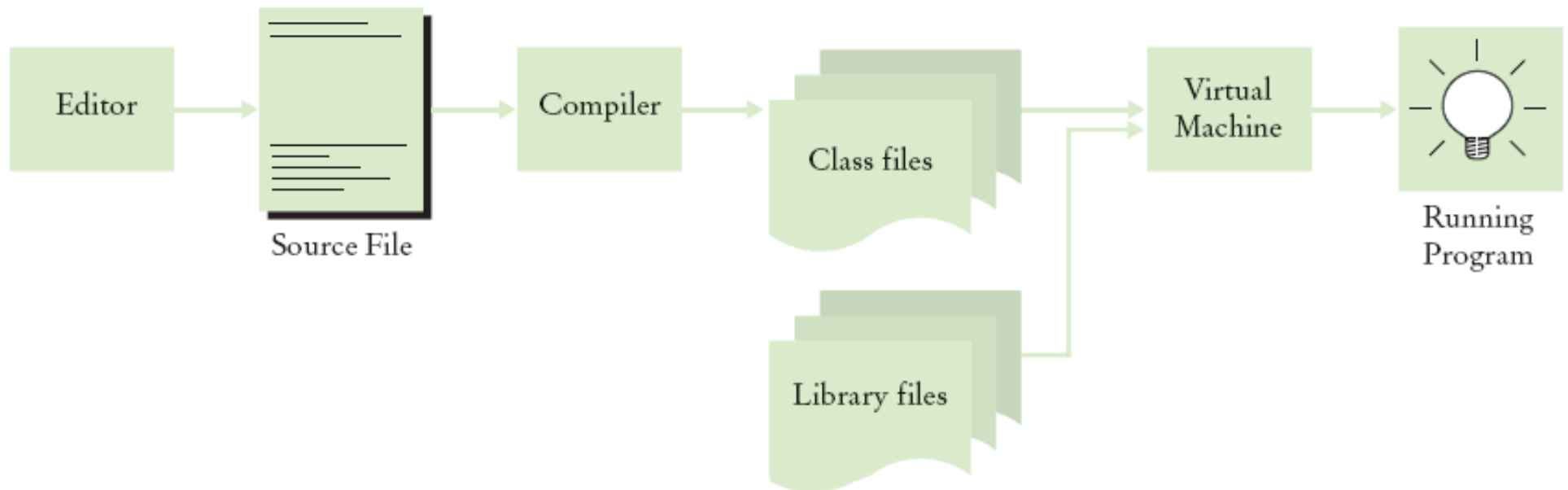


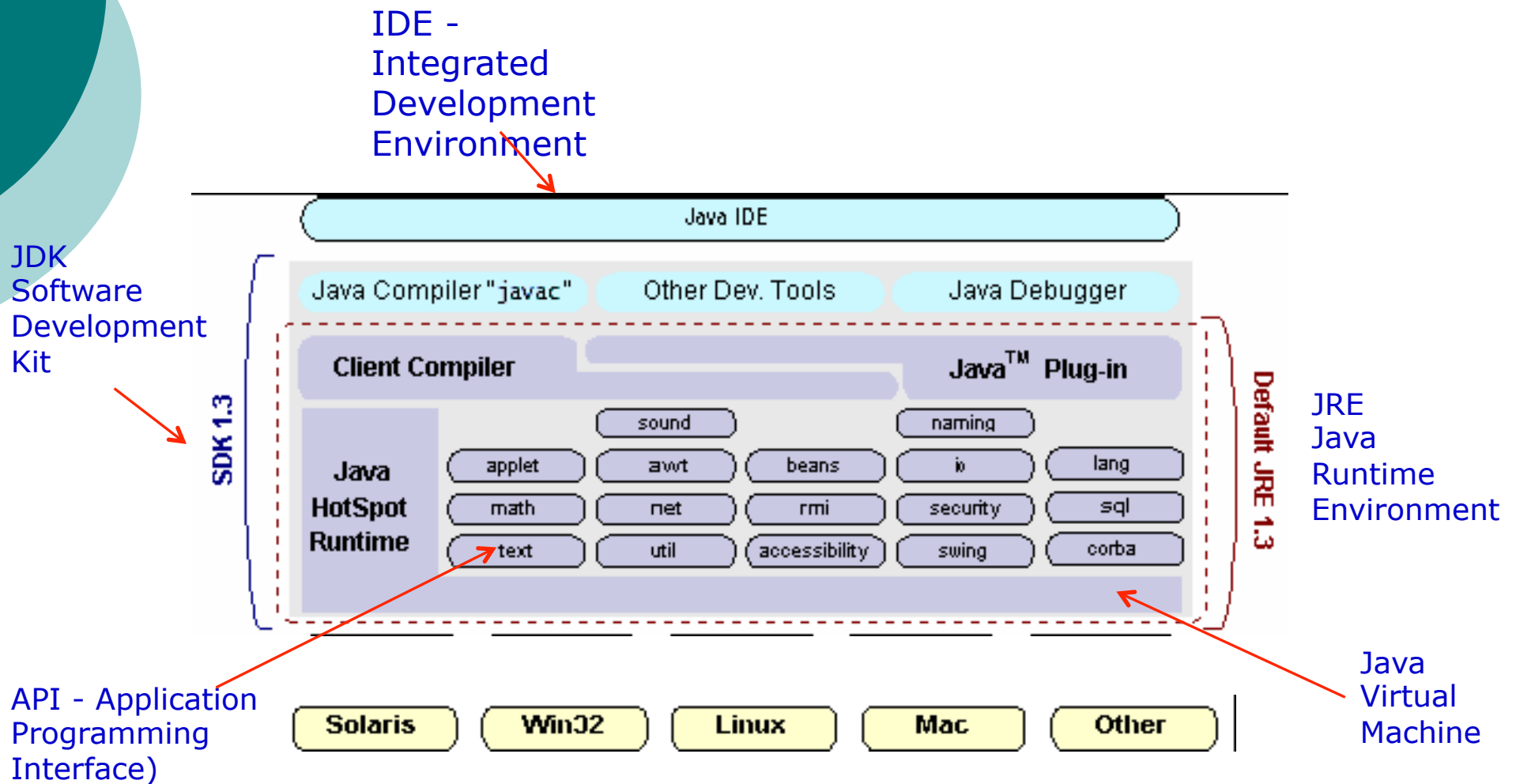
Figure 13 From Source Code to Running Program



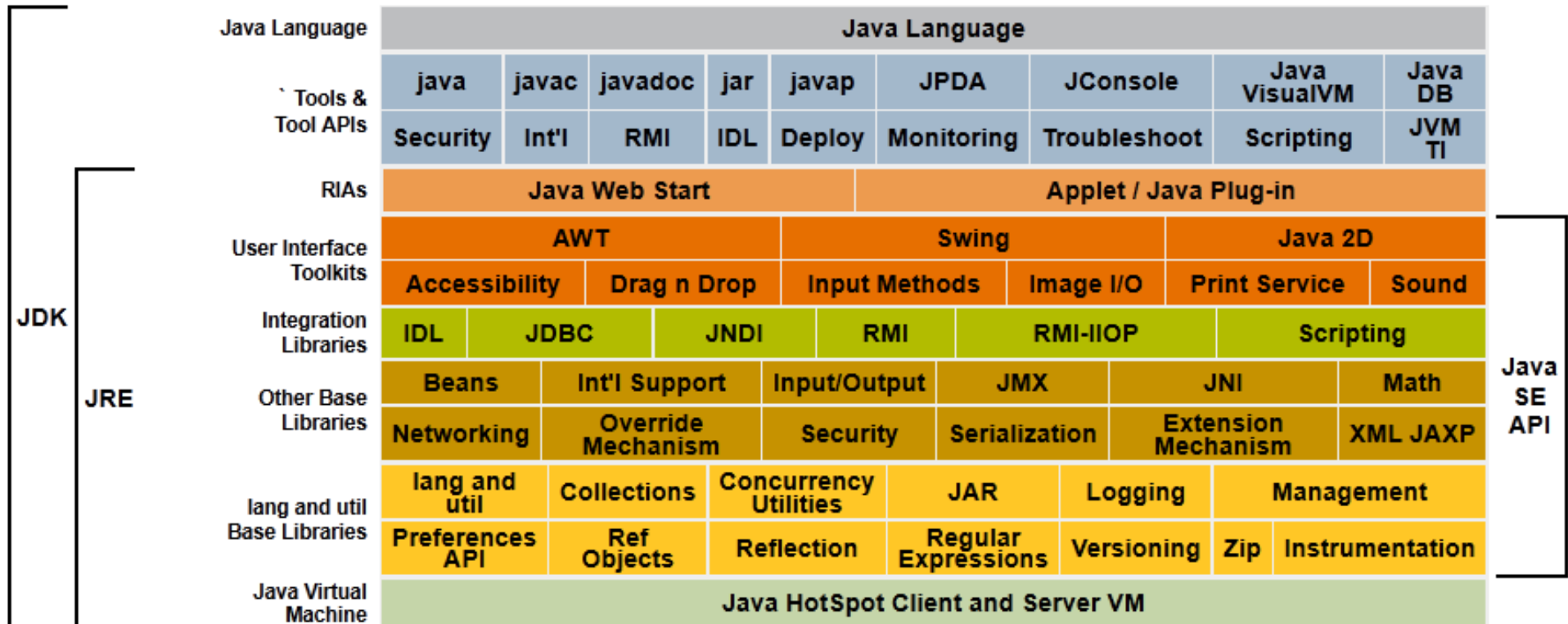
Java Platform

- La Java platform è solo software e viene eseguita al di sopra di altre piattaforme hardware
 - Java Virtual Machine (JVM)
 - Java Application Programming Interface (Java API)
 - collezione di software *components/artifacts* pronti per l'uso, es. per gestire Graphical User Interface (GUI)
 - organizzate in librerie di classi e interfacce correlate (packages)

Java Platform: più dettagliata



Java Platform 7





Java Platforms

- Tre edizioni principali:
 - **Standard Edition (Java SE™)**
 - fornisce ambiente runtime per esecuzione di applicazioni Java; API essenziali per sviluppare applicazioni di vario tipo (applet, applicazioni stand-alone)
 - **Enterprise Edition (Java EE™)**
 - Fornisce un framework per lo sviluppo di applicazioni server-side complesse. Adatta allo sviluppo di applicazioni Web-based a livello di impresa, e.g., per commercio elettronico
 - **Micro Edition (Java ME™)**
 - un Java runtime environment altamente ottimizzato indirizzato specificamente a “computer piccoli”: smart card, telefonini, PDA.



JDK

- **Java SE 8.0 – Java SE Development Kit 8**
- Scaricabile, collegandosi al link:
<http://www.oracle.com/technetwork/java/javase/downloads/index.html>
- Il **Java Development Kit (JDK)** è un insieme base di programmi che consente di far girare applicazioni scritte nel linguaggio Java.
- 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 8.0**
 - Esiste la versione per Windows:
 - **jdk-8u60-windows-i586.exe** (jdk-8u60-windows-x64.exe)
 - Linux RPM (RedHat Package Manager) in self-extracting file:
 - **jdk-8u60-linux-i586.rpm** (jdk-8u60-linux-x64.rpm)
- Documentazione sulla vasta raccolta di API Java si può scaricare in formato html gratuitamente da:
 - **<https://docs.oracle.com/javase/8/docs/>**
- La documentazione online è consultabile al link:
 - **<https://docs.oracle.com/javase/8/docs/api/>**



API

- Java Application Programming Interface (API)

- codice già scritto, strutturato in packages relativi a insiemi di argomenti comuni
- Java package

Un package è una collezione di classi correlate e interfacce, che forniscono accesso protetto e namespace management.

- Per usare una classe o un' interfaccia in un package

- usare il nome completo della classe **java.util.ArrayList**
- o importare la classe **import java.util.ArrayList;**
- o importare tutto il package **import java.util.*;**

API Documentation

Packages

Classi

Java™ Platform Standard Ed. 7

All Classes

Packages

- java.applet
- java.awt
- java.awt.color
- java.awt.datatransfer
- java.awt.dnd
- java.awt.event

All Classes

- AbstractAction
- AbstractAnnotationValueVisitor6
- AbstractAnnotationValueVisitor7
- AbstractBorder
- AbstractButton
- AbstractCellEditor
- AbstractCollection
- AbstractColorChooserPanel
- AbstractDocument
- AbstractDocument.AttributeContext
- AbstractDocument.Content
- AbstractDocument.ElementEdit
- AbstractElementVisitor6
- AbstractElementVisitor7
- AbstractExecutorService
- AbstractInterruptibleChannel
- AbstractLayoutCache
- AbstractLayoutCache.NodeDimensions
- AbstractList
- AbstractListModel
- AbstractMap
- AbstractMap.SimpleEntry
- AbstractMap.SimpleImmutableEntry
- AbstractMarshallerImpl
- AbstractMethodError
- AbstractOwnableSynchronizer
- AbstractPreferences
- AbstractProcessor
- AbstractQueue

Overview Package Class Use Tree Deprecated Index Help

Prev Next Frames No Frames

Java™ Platform, Standard Edition 7 API Specification

This document is the API specification for the Java™ Platform, Standard Edition.

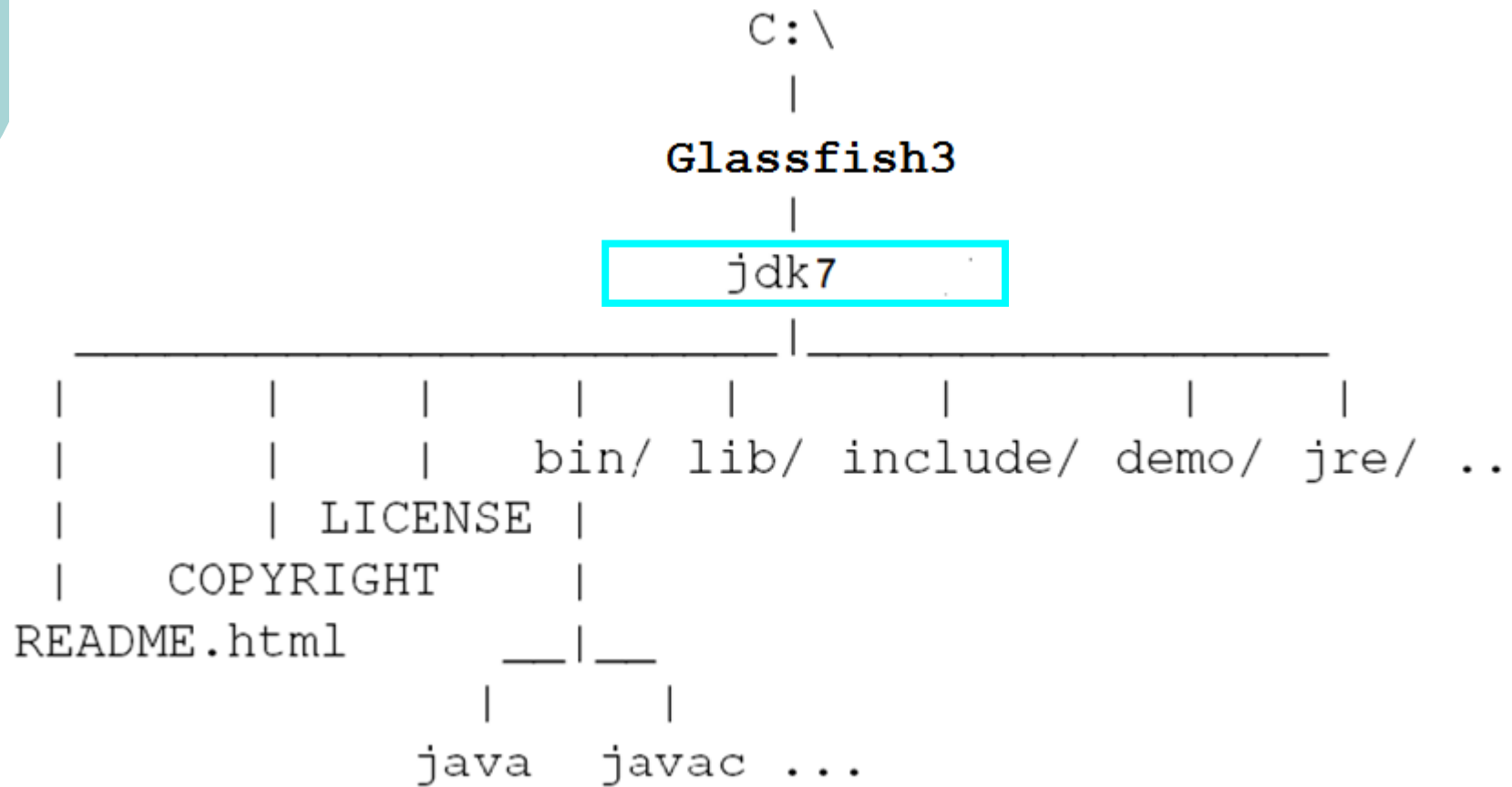
See: Description

Packages

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image	Provides classes for creating and modifying images.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent images.
java.awt.print	Provides classes and interfaces for a general printing API.
java.beans	Contains classes related to developing <i>beans</i> -- components based on the JavaBeans™ architecture.
java.beans.beancontext	Provides classes and interfaces relating to bean context.
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.

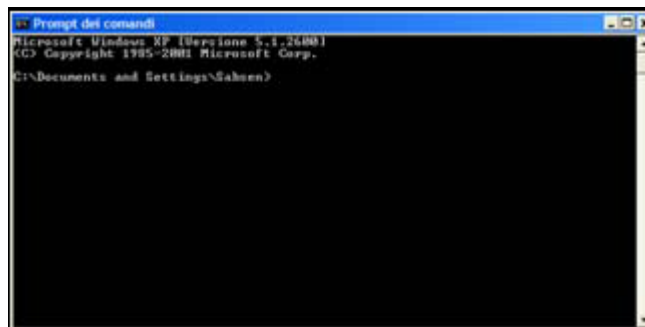
Dettaglio delle classi e relativi metodi

Albero delle directory Java



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.





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

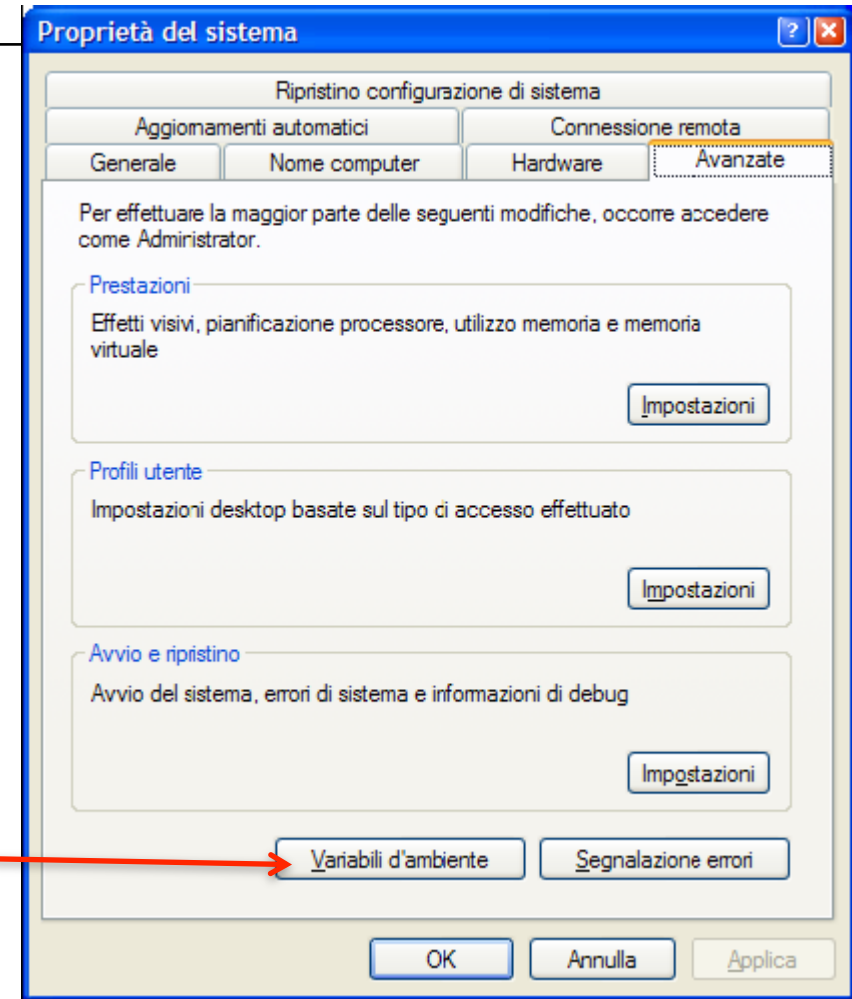
- Dopo aver installato JDK con la procedura guidata occorre settare le variabili di ambiente del proprio sistema per includere alcuni riferimenti al kit JDK.
- Per il sistema operativo Windows occorre modificare la variabile **Path** del sistema introducendo un riferimento alla directory **bin** dove si trova il programma **javac** (il compilatore java).
 - Se la cartella Java si chiama jdk8 e si trova installata nel percorso C:\Glassfish3\jdk8 bisogna scrivere:
 - **PATH= ...; C:\Glassfish3\jdk8\bin**

...usando Windows 7 e XP

- Selezionare:

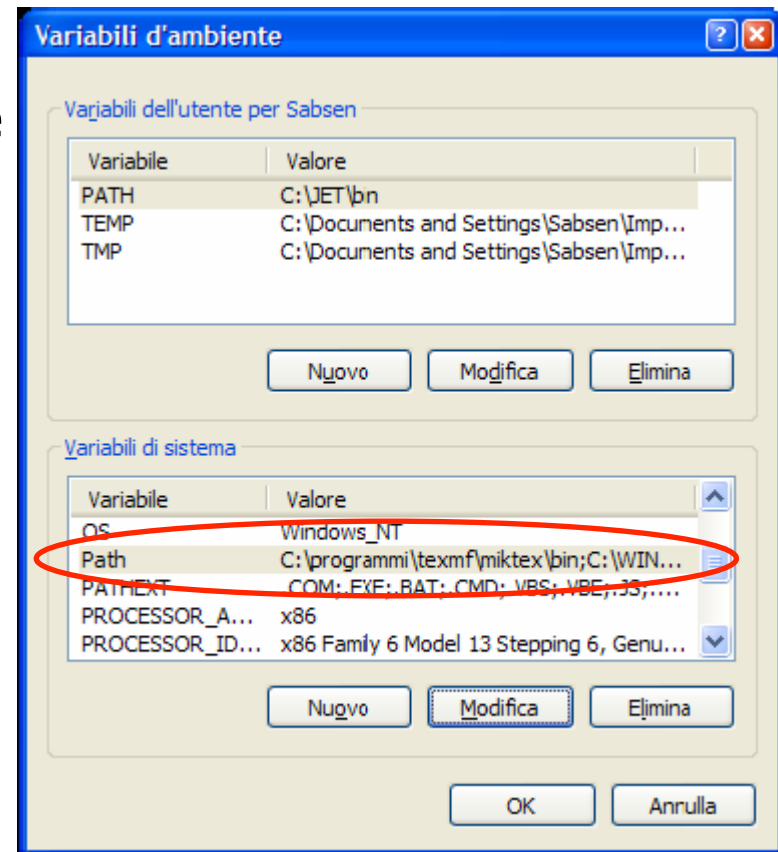
**Start →
Pannello di Controllo →
Sistema**

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



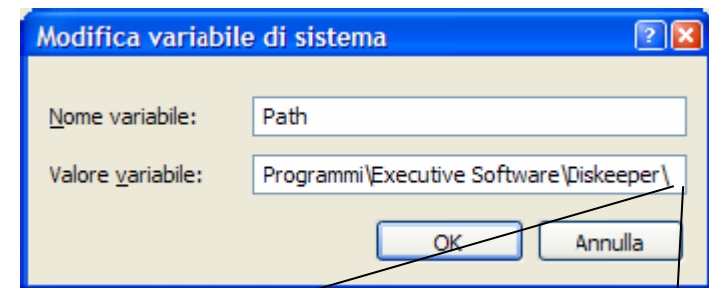
Con Windows 7 e XP

Tra le variabili di sistema, considerare la variabile Path e cliccare sul bottone *Modifica*



Con Windows 7 e XP

- Cliccare su *Valore variabile*
- Alla fine del testo già presente, inserire:
";" + *il percorso su cui si trova la JDK*
- Premere il bottone *OK*



; C:\Glassfish3\jdk8\bin



Il Classpath

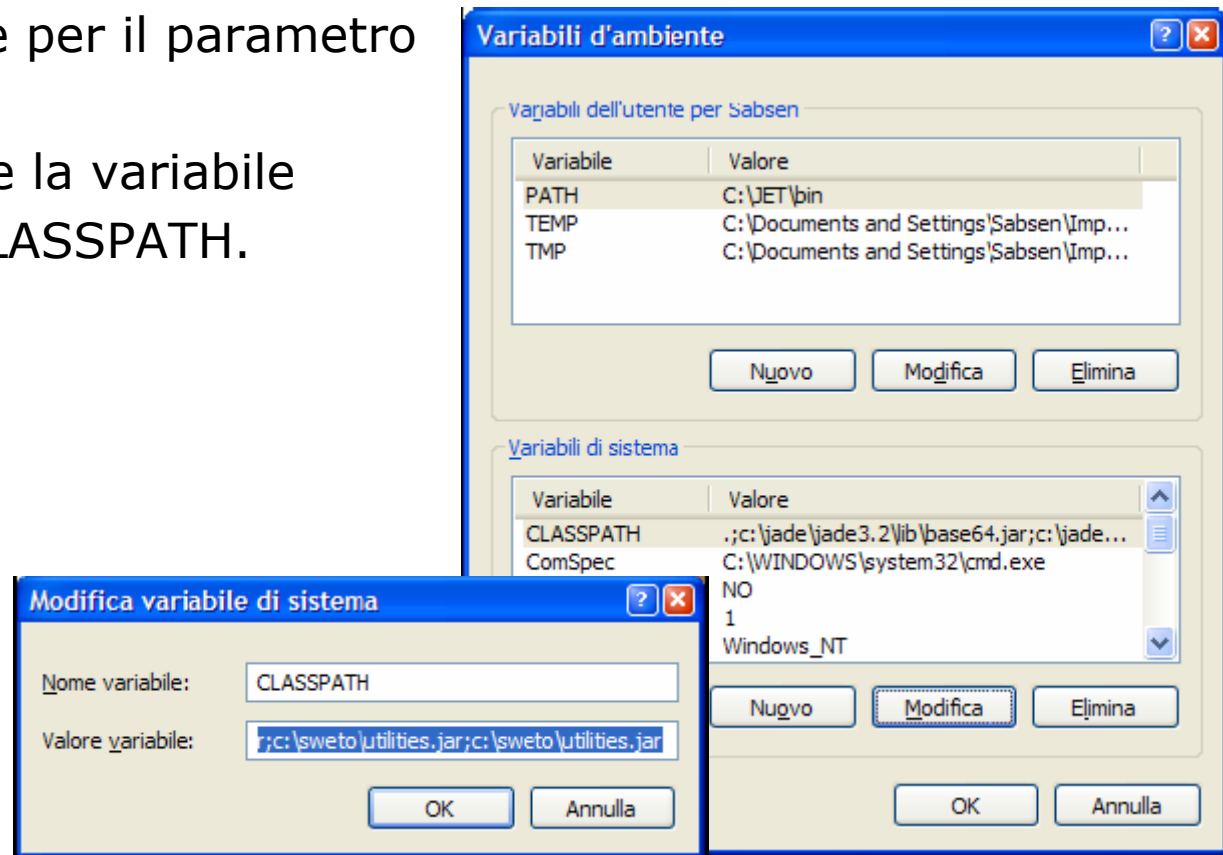
- Il parametro **classpath** serve per indicare i percorsi in cui ricercare i file **.class**.
- Il compilatore e la JVM sanno già dove trovare i package forniti con la distribuzione di Java.
- Ogni volta che dobbiamo usare nuovi package (di solito, file con estensioni jar) dobbiamo specificare il percorso 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

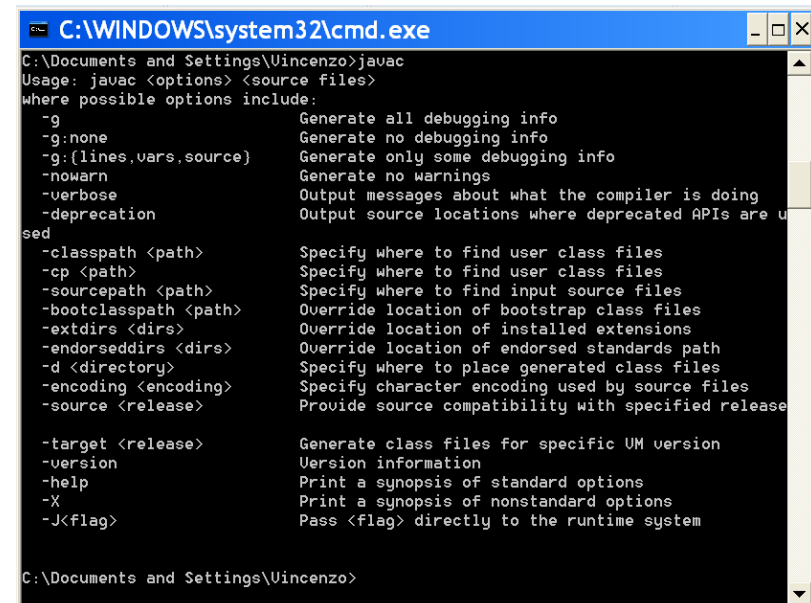
Il Classpath

- 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\Uincenzo>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 used
  -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\Uincenzo>
```



Per eseguire un programma Java

- Aprire NotePad (Windows) o un altro Editor di Testo
- 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 distingue tra maiuscole e minuscole



Primo programma Java

- Scrivere in un editore di testo il seguente codice:

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

- 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**