

Il Linguaggio **JAVA**

Prima parte

Capitolo 1

Il Linguaggio JAVA (generalità)

Il Linguaggio JAVA ...

- ▶ Il linguaggio Java è il linguaggio di programmazione più usato del pianeta
- ▶ Con il termine “Java” ci si riferisce sia al linguaggio di programmazione, sia alla tecnologia
- ▶ Il termine di tecnologia include “sotto-tecnologie” affermate in diversi ambiti
- ▶ Ambito Software (Enterprise) e Hardware

Il Linguaggio JAVA ...

- ▶ Fu sviluppato nel 1995 da Sun Microsystems
- ▶ Nel 2010 la società è stata assorbita da Oracle
- ▶ Insieme a Sun, Oracle ha assorbito numerose società produttrici di tecnologie java-oriented
- ▶ Bea, Mysql ecc...
- ▶ Tutto ha avuto inizio cercando di creare un linguaggio che uniformasse la comunicazione tra diversi dispositivi consumer (VCR e TV ad esempio)

Il Linguaggio JAVA ...

- ▶ Nasce con il nome Oak
- ▶ La piattaforma fu respinta dalle diverse case produttrici di dispositivi
- ▶ Nel frattempo si afferma il WWW, il *Green Team* ha riconosciuto che il linguaggio Oak era perfetto per lo sviluppo di componenti web multimediali per migliorare le pagine web. Queste piccole applicazioni, chiamate applet, sono diventate il primo utilizzo del linguaggio Oak
- ▶ I programmatori che utilizzavano Internet hanno adottato quello che divenne il linguaggio di programmazione Java.
- ▶ Il punto di svolta per Java è venuto nel 1995, quando Netscape ha incorporato Java nel suo browser

Il Linguaggio JAVA ...

► Punti di forza :

- E' Object Oriented
- Gratuito
- Robusto (gestione delle eccezioni chiara e funzionale, meccanismo automatico di gestione della memoria – Garbage Collector)
- Indipendente dall'architettura (compilo 1 volta eseguo ovunque) grazie alla JVM
- Aperto : dal tool di sviluppo all'interazione con altri linguaggi e tecnologie (SQL,XML,Framework,ecc..)

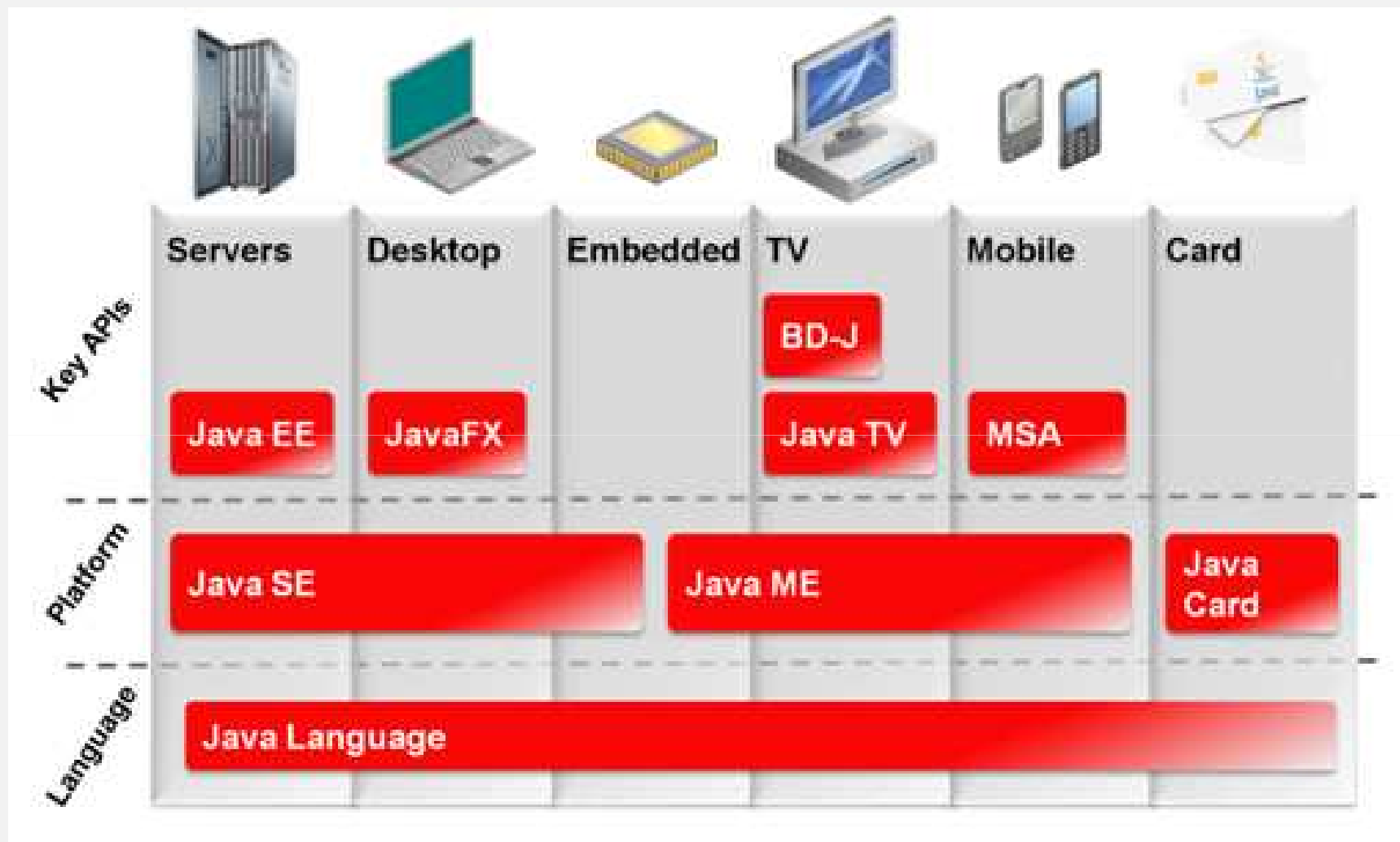
Il Linguaggio JAVA ...

- *Distribuito* : il linguaggio fornisce il supporto per tecnologie di rete distribuiti, come Remote Method Invocation (RMI), Common Object Request Broker Architecture (CORBA), e Universal Resource Locator (URL).
- *Multi-Threaded* : supporta il multithreading. Questo permette l'esecuzione contemporanea di diversi compiti

Il Linguaggio JAVA ...

- ▶ La slide successiva illustra il concetto che i termini tecnologia Java e linguaggio di programmazione Java non si riferiscono alla stessa cosa.
- ▶ Tecnologia Java si riferisce a una famiglia di prodotti di tecnologia Java, di cui il linguaggio di programmazione è solo una parte

Il Linguaggio JAVA ...

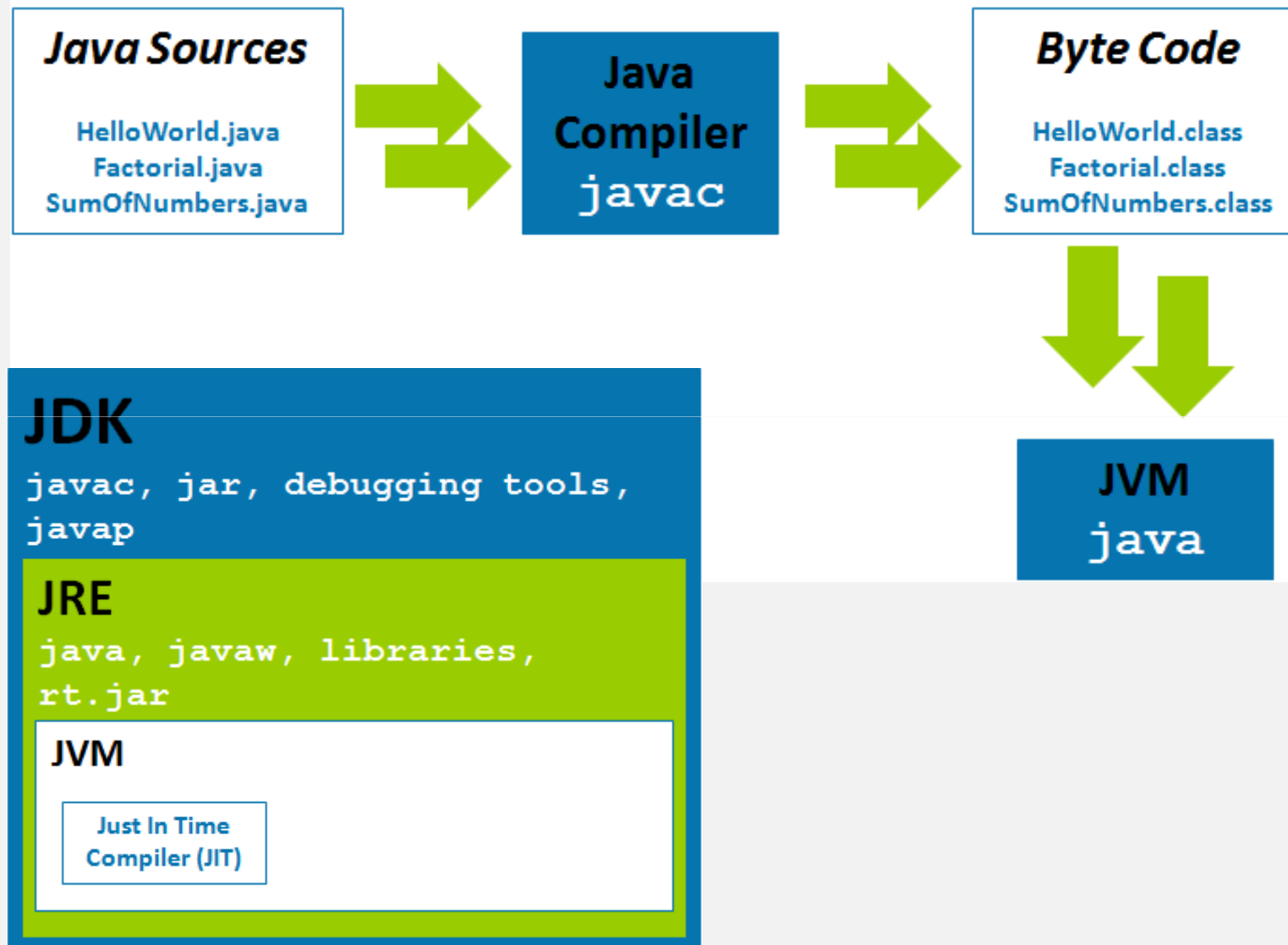


La tecnologia JAVA

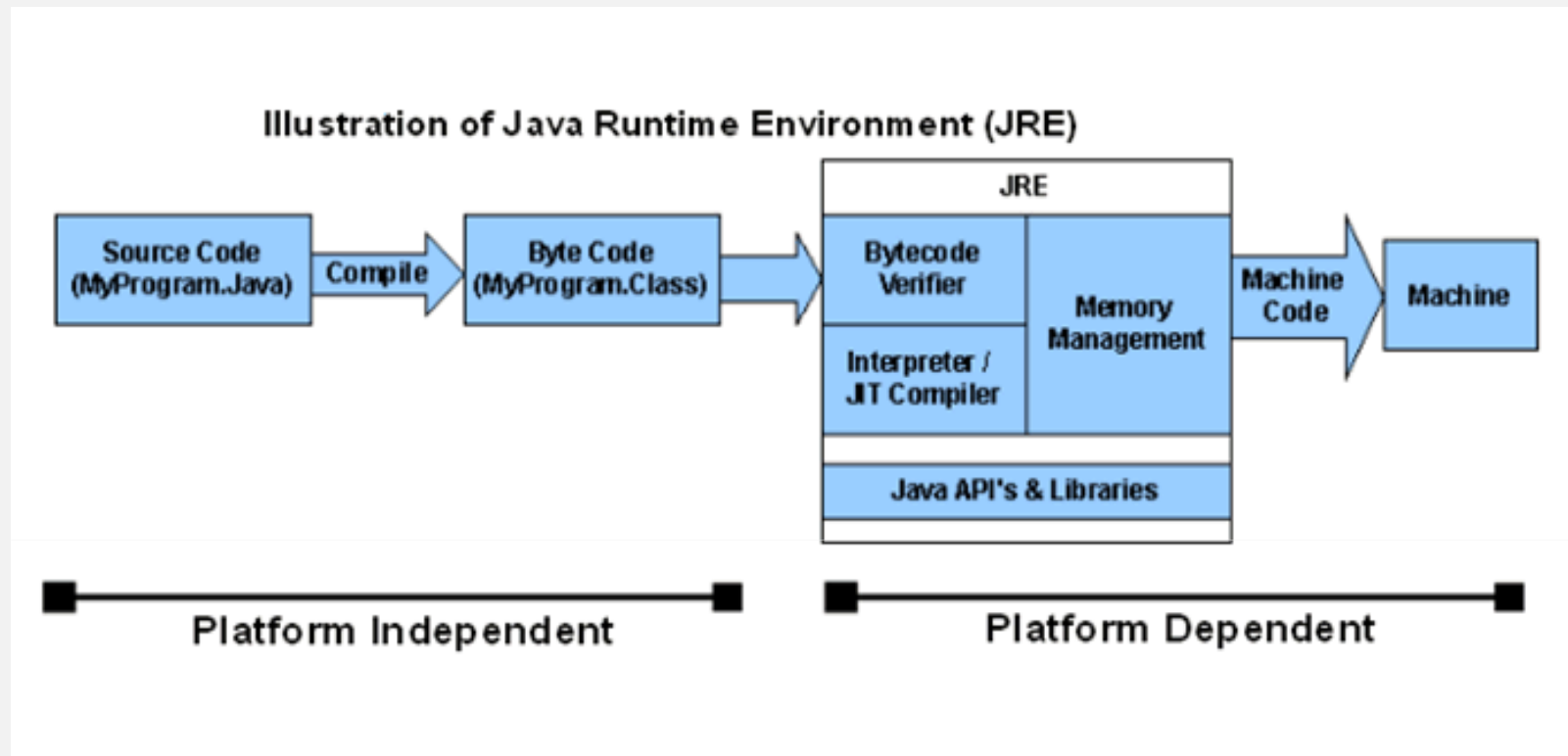
Il Linguaggio JAVA ...

- ▶ **JVM** – Java Virtual Machine
- ▶ E' il sw che rende indipendente l'applicazione dalla piattaforma
- ▶ Dopo aver scritto il testo Java (*nomefile.java*), anche con un *Blocco note*, lo si compila e otteniamo il bytecode (*nomefile.class*) linguaggio molto vicino al linguaggio macchina
- ▶ Sarà la JVM che interpreterà il bytecode ed il nostro programma potrà essere eseguito

Il Linguaggio JAVA ...



Il Linguaggio JAVA ...



Il Linguaggio JAVA ...

- ▶ **J**ava **D**evelopment **K**it (SE – **S**tandard **E**dition)
- ▶ Per iniziare si ha bisogno di un compilatore, una **JVM** ed un ambiente di esecuzione (ambiente di runtime)
- ▶ Il JDK contiene tutto ciò !
- ▶ Scaricabile da www.oracle.com gratuitamente

Il Linguaggio JAVA ...

► Struttura di JDK :

- ❑ *bin* : contiene tutti gli eseguibili – javac, java ecc
- ❑ *Include* e *lib* : librerie utilizzate dalla JDK
- ❑ **J**ava **R**untime **E**nviroment : affinché un'applicazione java risulti eseguibile, basta installare il solo Jre.
- ❑ Si tratta della JVM con il supporto alle librerie che afferiscono alla versione corrente di Java
 - JRE è installato automaticamente quando viene installato il JDK, nella stessa cartella

Il Linguaggio JAVA

► Differenze tra JVM, JRE, JDK :

- ❑ **JVM** : è la macchina virtuale che simula un hw capace di interpretare ed eseguire un testo scritto in bytecode
- ❑ **JRE** : ambiente di esecuzione java. Contiene la JVM
- ❑ **JDK** : kit completo di sviluppo java. Contiene la JRE e quindi la JVM
- ❑ **JDK** serve per **sviluppare**, **JRE** per **eseguire**

Capitolo 2

Preparazione dell'Ambiente di Sviluppo

Preparazione dell'Ambiente di Sviluppo ...
















- ▶ Per il corso utilizzeremo la versione 7 della SE
- ▶ Dal link è possibile scegliere la versione se a 64 o 32 bit

Preparazione dell'Ambiente di Sviluppo ...

Java SE Development Kit 7u71

You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

☐ Accept License Agreement ☒ Decline License Agreement

Product / File Description	File Size	Download
Linux x86	119.44 MB	 jdk-7u71-linux-i586.rpm
Linux x86	136.76 MB	 jdk-7u71-linux-i586.tar.gz
Linux x64	120.81 MB	 jdk-7u71-linux-x64.rpm
Linux x64	135.63 MB	 jdk-7u71-linux-x64.tar.gz
Mac OS X x64	185.84 MB	 jdk-7u71-macosx-x64.dmg
Solaris x86 (SVR4 package)	139.36 MB	 jdk-7u71-solaris-i586.tar.Z
Solaris x86	95.48 MB	 jdk-7u71-solaris-i586.tar.gz
Solaris x64 (SVR4 package)	24.68 MB	 jdk-7u71-solaris-x64.tar.Z
Solaris x64	16.36 MB	 jdk-7u71-solaris-x64.tar.gz
Solaris SPARC (SVR4 package)	138.74 MB	 jdk-7u71-solaris-sparc.tar.Z
Solaris SPARC	98.62 MB	 jdk-7u71-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	23.94 MB	 jdk-7u71-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	18.35 MB	 jdk-7u71-solaris-sparcv9.tar.gz
Windows x86	127.78 MB	 jdk-7u71-windows-i586.exe
Windows x64	129.52 MB	 jdk-7u71-windows-x64.exe



Preparazione dell'Ambiente di Sviluppo ...

▶ **JavaDoc**

- ▶ La JDK è corredata dalla documentazione
- ▶ Il formato è in HTML quindi portabile e consultabile online
- ▶ E' scaricabile sempre dal sito Oracle

Preparazione dell'Ambiente di Sviluppo ...


[Overview](#) [Downloads](#) [Documentation](#) [Community](#) [Technologies](#) [Training](#)


Java SE Development Kit 7 Documentation

Java SE Development Kit 7u71 Documentation

You must accept the [Java SE Development Kit 7 Documentation License Agreement](#) to download this software.

☐ Accept License Agreement ☒ Decline License Agreement

Product / File Description	File Size	Download
Documentation	58.29 MB	 jdk-7u71-docs-all.zip



Preparazione dell'Ambiente di Sviluppo ...

- ▶ **Windows 7**
- ▶ Premere il tasto “*Start*”
- ▶ Premere tasto destro del mouse su “*Risorse del Computer*” o “*Computer*”
- ▶ Selezionare “*Proprietà*”
- ▶ Selezionare “*Impostazioni di Sistema Avanzate*”
- ▶ Selezionare “*Variabili d'Ambiente*”

Preparazione dell'Ambiente di Sviluppo ...

- ▶ Tra le *Variabili di Sistema* (o di Utente)
- ▶ Cercare la variabile *PATH* ed andare a Modifica
- ▶ In coda ai riferimenti aggiungere un “;” e il riferimento alla cartella “***bin***” della JDK appena installata (recuperabile da risorse del computer)
- ▶ Es : C:\Program Files\Java\jdk1.7.0_01\bin

Preparazione dell'Ambiente di Sviluppo ...

- ▶ Non sempre necessario
- ▶ Sempre tra le variabili di sistema
- ▶ Selezionare “Nuova”
- ▶ Aggiungere la variabile **JAVA_HOME**
- ▶ Con valore il path della JDK
- ▶ Es: *C:\Program Files\Java\jdk1.7.0_01*

Preparazione dell'Ambiente di Sviluppo ...

- ▶ Da Start eseguire il comando “**cmd**”
- ▶ Segue apertura della console DOS
- ▶ Eseguire il comando : *java -version*
- ▶ Testare anche il compilatore : *javac*

Preparazione dell'Ambiente di Sviluppo ...

- Risultato del comando `"java -version"`

```
C:\Users\Windows7>java -version
java version "1.7.0_01"
Java(TM) SE Runtime Environment (build 1.7.0_01-b08)
Java HotSpot(TM) 64-Bit Server VM (build 21.1-b02, mixed mode)

C:\Users\Windows7>
```

Preparazione dell'Ambiente di Sviluppo ...

► Risultato del comando “java”

```
C:\Users\Windows7>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 and annotation processors
  -cp <path>              Specify where to find user class files and annotation processors
  -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
  -proc:{none,only}       Control whether annotation processing and/or compilation is done.
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; bypasses default discovery process
  -processorpath <path>    Specify where to find annotation processors
  -d <directory>          Specify where to place generated class files
  -s <directory>          Specify where to place generated source files
  -implicit:{none,class}  Specify whether or not to generate class files for implicitly referenced 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
  -Akey[=value]           Options to pass to annotation processors
  -X                      Print a synopsis of nonstandard options
  -J<flag>                Pass <flag> directly to the runtime system
  -Werror                 Terminate compilation if warnings occur
  @<filename>             Read options and filenames from file

C:\Users\Windows7>
```

Preparazione dell'Ambiente di Sviluppo ...

- ▶ Principali IDE (**I**ntegrated **D**evelopment **E**nvironment) :
Eclipse, Netbeans, Jdeveloper ...
- ▶ <https://www.eclipse.org/downloads/packages/release/Kepler/SR2>
- ▶ Anche in questo caso è possibile scegliere tra la versione a 32 o a 64 bit
- ▶ La versione utilizzata è la Kepler

Preparazione dell'Ambiente di Sviluppo

[GETTING STARTED](#)[MEMBERS](#)[PROJECTS](#)[MORE ▾](#)[DOWNLOAD](#)[HOME](#) / [DOWNLOADS](#) / [PACKAGES](#) / [ECLIPSE KEPLER SR2 PACKAGES](#)

Releases

[Mars Packages](#)[Luna Packages](#)[Kepler Packages](#)[Juno Packages](#)[Indigo Packages](#)[Helios Packages](#)[Galileo Packages](#)[Ganymede Packages](#)[Europa Packages](#)[All Releases](#)

Eclipse Standard 4.3.2

201 MB - Downloaded 5,630,148 Times



Eclipse IDE for Java EE Developers

250 MB - Downloaded 3,904,175 Times



Eclipse IDE for Java Developers

153 MB - Downloaded 1,210,493 Times



Eclipse IDE for C/C++ Developers

144 MB - Downloaded 858,277 Times

Windows 32-bit 64-bit
Mac Cocoa 32-bit 64-bit
Linux 32-bit 64-bit

Windows 32-bit 64-bit
Mac Cocoa 32-bit 64-bit
Linux 32-bit 64-bit

Windows 32-bit 64-bit
Mac Cocoa 32-bit 64-bit
Linux 32-bit 64-bit

Windows 32-bit 64-bit
Mac Cocoa 32-bit 64-bit
Linux 32-bit 64-bit

Capitolo 3

Il primo programma JAVA

Il Primo Programma Java ...

- ▶ Si tratta di un semplice programma presentato per iniziare
- ▶ Ci occuperemo della sua scrittura, compilazione ed esecuzione
- ▶ 2 i comandi coinvolti :
 - ▶ ***javac*** per compilare
 - ▶ ***java*** per eseguire

Il Primo Programma Java ...

► HelloWorld.java

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

Il Primo Programma Java ...

- ▶ Commento sul testo
- ▶ *public class HelloWorld*
- ▶ Ogni applicazione java è costituita da **classi**
- ▶ La classe è stata dichiarata accessibile in maniera **pubblica** (modificatore public)
- ▶ Le parentesi graffe { } delimitano **blocchi** di codice
- ▶ *public static void main (String args[])*
 - args **non** è un nome imposto, potrebbe essere anche un altro
- ▶ Si tratta della dichiarazione del metodo ***main***

Il Primo Programma Java ...

▶ Il metodo *main* ...

- ▶ Un metodo speciale che la JVM riconosce come **punto di partenza** per ogni programma Java
- ▶ Il metodo main contiene 2 modificatori entrambi necessari : **public** e **static**
- ▶ Il metodo main **non** restituisce alcun valore (tipo restituito **void**)
- ▶ Il metodo main accetta 0 o più oggetti di tipo String (String[] args)

Il Primo Programma Java

- ▶ *System.out.println(" Hello World") ;*
- ▶ Comando che mostrerà sul video (output) il messaggio

Hello World

- ▶ Per ora basti sapere che si sta invocando un metodo della libreria standard di java, `println()`, passandogli come parametro di ingresso la stringa da mostrare
- ▶ Quando viene salvato, il file DEVE avere lo stesso nome della Classe dichiarata
- ▶ Nel nostro caso `HelloWorld.java`

Capitolo 4

Variabili, tipi

Variabili e Tipi ...

Un po' di convenzioni ...

- ▶ Java è a schema libero : si potrebbe scrivere un intero programma su un'unica riga ! Il compilatore non avrebbe problemi
- ▶ Si potrebbe anche andare a capo ad ogni parola. Si perderebbe solo in leggibilità del codice
- ▶ Java identifica uno statement, un'istruzione con il “ ; ”
 - nel caso di dichiarazione di **classi** e **metodi** con le **parentesi** ()

Variabili e Tipi ...

Errori tipici :

- ▶ Dimenticare di chiudere una parentesi
- ▶ Dimenticare il ; dopo un'istruzione
- ▶ Dimenticare le parentesi () nell'invocazione di un metodo
- ▶ Il linguaggio Java e' **case sensitive**

Variabili e Tipi ...

- ▶ **Commento** su **singola** riga si esprimono con **//** per ogni riga
- ▶ **Commento** su **più** righe **/*** ***/**
- ▶ **Commento Javadoc** **/**** ***/** consente la produzione di documentazione in HTML

Variabili e Tipi ...

Regole per gli identificatori

- ▶ I nomi delle classi, degli oggetti, dei metodi, delle variabili, delle costanti hanno **2** semplici regole
- 1. Un **identificatore** non può coincidere con una parola chiave
- 2. In un identificatore il **primo** carattere può essere **A - Z, a - z, _, \$** ed il secondo potrà essere **A - Z, a - z, _, \$, 0 - 9**

Variabili e Tipi ...

Convenzioni sui nomi

- ▶ L'identificatore di una **variabile** è composto da 1 o più sostantivi (in CamelCase) es : numeroLati, pesoInKg, peso
- ▶ Gli identificatori delle **costanti** avranno, per **consuetudine**, tutte le lettere maiuscole
 - ▶ Es : PI_GRECO, GIORNI_SETTIMANA
- ▶ Gli identificatori dei **metodi** è bene contengano verbi es : stampa_Numero(), somma()
- ▶ L'identificatore di una **classe**, per **consuetudine**, inizia con una lettera maiuscola
 - ▶ Es : Persona, Figura_Geometrica

Variabili e Tipi ...

- ▶ Java definisce **8** tipi di dato primitivi
- ▶ Tipi interi :
 - ▶ **byte**
 - ▶ **short**
 - ▶ **int**
 - ▶ **long**
- ▶ Tipi floating point
 - ▶ **float**
 - ▶ **double**
- ▶ Tipo carattere : **char**
- ▶ Tipo logico booleano : **boolean**

Variabili e Tipi ...

Tipo	Lunghezza	Range	Esempio
byte	8 bits	-2^7 to $2^7 - 1$ (-128 to 127, 256 valori possibili)	2 -114 0b10 (binary number)
short	16 bits	-2^{15} to $2^{15} - 1$ (-32,768 to 32,767, 65,535 valori possibili)	2 -32699
int	32 bits	-2^{31} to $2^{31} - 1$ (-2,147,483,648 to 2,147,483,647, 4,294,967,296 valori possibili)	2 147334778 123_456_678

Occupazione **spazio** e **range** possibili valori

Variabili e Tipi ...

Tipo	Lunghezza	Range	Esempio
long	64 bits	-2^{63} to $2^{63} - 1$ (– 9,223,372,036,854,7 75,808 A 9,223,372,036,854,7 75,807, OPPURE 18,446,744,073,709 ,551,616 valori possibili)	2 –2036854775808L 1L

Tipo	Lunghezza	Esempi
float	32 bits	99F –327456,99.01F 4.2E6F (notazione esponenziale per $4.2 * 10^6$)
double	64 bits	–1111 2.1E12 99970132745699.999

Variabili e Tipi ...

- ▶ Tipo **char** utilizzato per memorizzare un carattere singolo racchiuso tra **apici** singoli es : 'c'
- ▶ Molti linguaggi di programmazione utilizzano l' American Standard Code for Information Interchange (**ASCII**), a **8** bit
- ▶ Il linguaggio Java utilizza **Unicode** (a **16** bit) che può memorizzare tutti i caratteri visualizzabili necessari nella stragrande maggioranza delle lingue usate nel mondo
- ▶ Quindi i programmi possono essere scritti in modo da funzionare correttamente e visualizzare la lingua corretta per la maggior parte dei paesi
- ▶ I primi 128 caratteri di Unicode **coincidono** con l' ASCII

Variabili e Tipi ...

- ▶ Tipo **boolean** : 1 bit e 2 possibili valori **true** o **false**
- ▶ Può contenere anche il risultato di un'espressione logica in modo da poter essere utilizzata in un contesto decisionale
 - ▶ Es boolean b = (33 < 44)
 - b conterrà il valore **true**

Variabili e Tipi ...

- ▶ Dichiarazione di una variabile

tipo identificatore [= valore];

- ▶ Es : `int i; // dichiaro una variabile intera`
- ▶ L'operatore di **assegnazione** in Java è “ = ”
- ▶ Dichiarazione **ed** inizializzazione contestuale
 - ▶ `int i = 3;`
- ▶ E' possibile dichiarare **più variabili** su una sola istruzione, però devono essere dello stesso tipo
 - ▶ `int x,y,z;`

Variabili e Tipi ...

- ▶ Promotion
- ▶ Un'operazione e la successiva **assegnazione** possono portare ad una **mancata corrispondenza** tra i **tipi**
 - ▶ `int num1 = 53; // 32 bit di memoria per contenere il valore`
 - ▶ `int num2 = 47; // 32 bit di memoria per contenere il valore`
 - ▶ `byte num3; // 8 bit di memoria riservati`
 - ▶ `num3 = (num1 + num2); // Causa errore del compilatore`

Variabili e Tipi ...

- ▶ In alcune circostanze, il compilatore **cambia** il tipo di una variabile ad un tipo che supporta un valore di dimensione più grande
- ▶ Questa azione viene definita **promozione implicita**
- ▶ Le promozioni che sono fatte automaticamente dal compilatore includono :
 - ▶ Assegnazione di un tipo più piccolo (a **destra** delle **=**) ad un tipo più grande (a **sinistra** delle **=**)

Variabili e Tipi ...

- ▶ Es. `long k = 6;`
- ▶ 6 è un `int` ma nell'assegnazione viene **promosso** a `long`
- ▶ **Prima** di essere assegnato ad una variabile, il **risultato** di un'espressione è collocato in una posizione temporanea in memoria
- ▶ La dimensione della locazione è **sempre** pari alla dimensione di un **`int`** o alle dimensioni del tipo **più grande**
- ▶ Es. se l'espressione moltiplica fra loro 2 tipi `int`, la dimensione del contenitore sarà un tipo `int` (32 bit)

Variabili e Tipi ...

- ▶ Per risolvere il problema si utilizza il **casting**
- ▶ Il cast **abbassa** “la portata” di un dato o **forza** una conversione
- ▶ Tornando a prima :
 - ▶ `int num1 = 53; // 32` bit di memoria per contenere il valore
 - ▶ `int num2 = 47; // 32` bit di memoria per contenere il valore
 - ▶ `byte num3; // 8` bit di memoria riservati
 - ▶ `num3 = (byte) (num1 + num2); // Senza perdita di dati`

Variabili e Tipi ...

- ▶ Sintassi :
- ▶ `identificatore = (target_type) dato`
- ▶ dove:
 - ▶ ***identificatore*** è il nome assegnato ad una variabile
 - ▶ ***dato*** è il valore che si desidera assegnare all'identificatore
 - ▶ ***(target_type)*** è il tipo a cui si desidera portare il valore. Il `target_type` **deve** essere posto tra **parentesi**

Variabili e Tipi

Operazione	Operatore
Somma	+
Differenza	-
Prodotto	*
Divisione	/
Modulo o resto	% (es 31%6=1)

Operator e	Utilizzo	Esempio	Note
++	Pre-incremento (++ <i>variabile</i>)	<pre>int i = 6; int j = ++i; i è 7, j è 7</pre>	Prima incrementato i e poi assegnato a j
	Post-incremento (<i>variabile</i> ++)	<pre>int i = 6; int j = i++; i è 7, j è 6</pre>	Il valore di i prima è assegnato a j e poi viene incrementato i

Capitolo 5

Operatori relazionali e controllo del flusso

Operatori relazionali e controllo del flusso ...

Condizione	Operatore	Esempio
E' uguale a	<code>=</code>	<pre>int i=1; (i == 1)</pre>
Non è uguale a	<code>!=</code>	<pre>int i=2; (i != 1)</pre>
Minore	<code><</code>	<pre>int i=0; (i < 1)</pre>
Minore o uguale	<code><=</code>	<pre>int i=1; (i <= 1)</pre>
Maggiore	<code>></code>	<pre>int i=2; (i > 1)</pre>
Maggiore o uguale	<code>>=</code>	<pre>int I = 1; (i >= 1)</pre>

Operatori relazionali e controllo del flusso

Operazione	Operatore	Esempio
Una condizione e un'altra condizione	&&	<pre>int i = 2; int j = 8; ((i < 1) && (j > 6))</pre>
Una condizione o un'altra condizione		<pre>int i = 2; int j = 8; ((i < 1) (j > 10))</pre>
NOT	!	<pre>int i = 2; (!(i < 3))</pre>
Operazione	Operatore	Esempio
Se condizione è vera produci come risultato value1 altrimenti value2	?:	<pre>condizione ? value1 : value2</pre>

Costrutto if

```
if (boolean_expression)  
{  
    code_block  
}
```

- ▶ Se espressione booleana è **vera** viene eseguito il blocco di codice
- ▶ **Altrimenti** si riprende dall'istruzione successiva alla parentesi graffa chiusa

Costrutto if / else

```
if (boolean_expression)
```

```
{
```

```
    code_block1
```

```
} // end of if construct
```

```
else
```

```
{
```

```
    code_block2
```

```
} // end of else construct
```

```
// program continues here
```

Costrutto if /else if ... Else ...

```
if (boolean_expression) {  
    code_block1  
} // end of if construct
```

```
else if (boolean_expression) {  
    code_block2  
} // end of else if construct
```

```
else {  
    code_block3  
}
```

.....

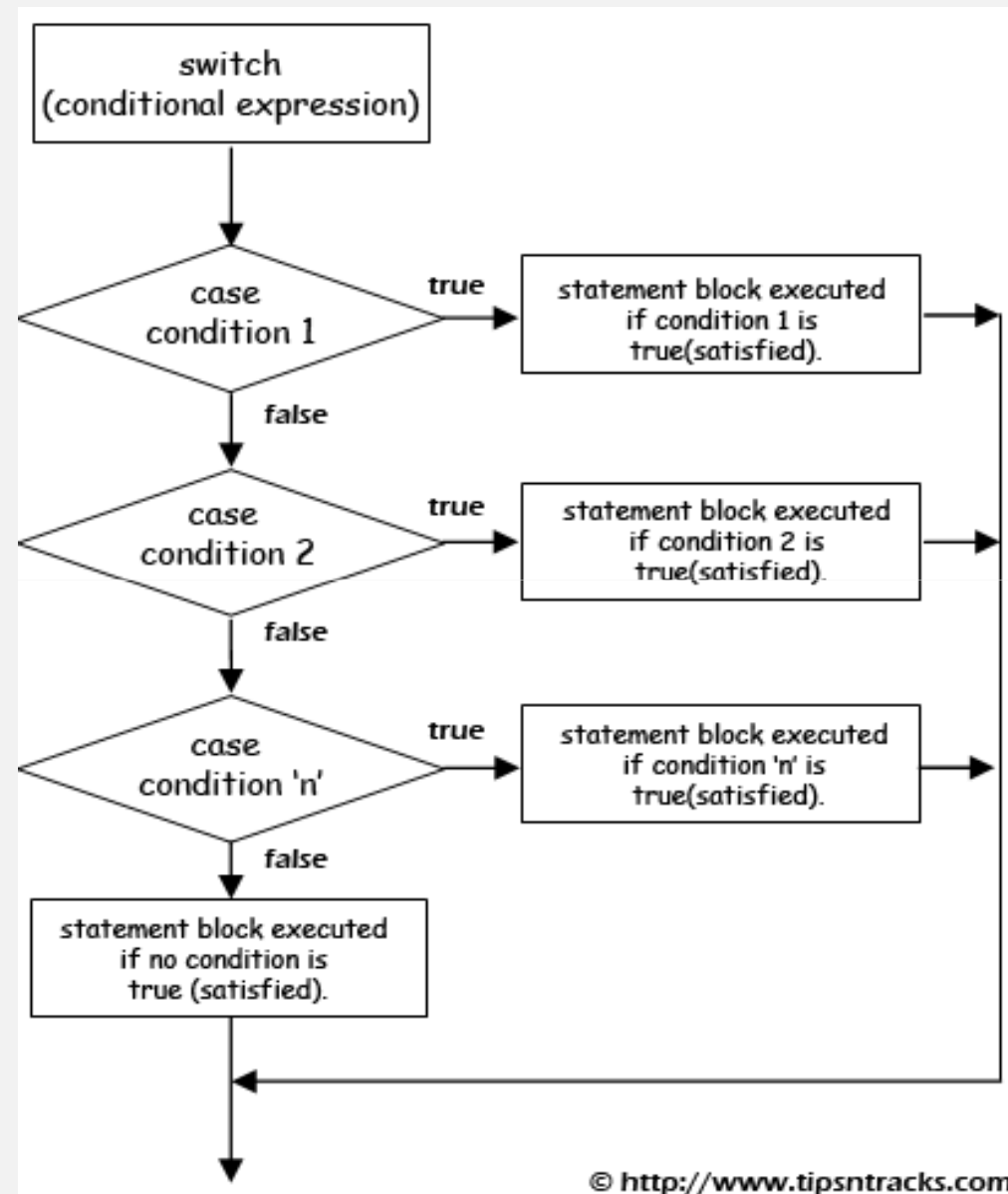
.....

Costrutto if /else if ... Else esempio

```
public void calculateNumDays( ) {  
    if ( month == 1 || month == 3 || month == 5 || month == 7 || month  
        == 8 || month == 10 || month == 12 ) {  
        System.out.println( "There are 31 days in that month." );  
    } else if ( month == 2 ) {  
        System.out.println( "There are 28 days in that month" );  
    } else if ( month == 4 || month == 6 || month == 9 || month == 11 ) {  
        System.out.println( "There are 30 days in that month." );  
    } else { System.out.println( "Invalid month" );  
    }  
}
```

Switch ...

```
switch (variable) {  
    case cond1:  
        code_block  
        [break;]  
    case cond2:  
        code_block  
        [break;]  
    [default:  
        code_block ]  
}
```



© <http://www.tipsntracks.com>

Switch ...

- ▶ La variabile da testare può essere :
 - int, short, byte
 - char
 - String
- ▶ Attenzione alla clausola break

Switch

- E' possibile che ad una stessa azione corrispondano più valori :

```
switch ( x )
```

```
{
```

```
    case 'A' :
```

```
    case 'a' :
```

```
        code_block
```

```
    case 'S' :
```

```
    case 's' :
```

```
        code_block
```

```
}
```

Operatore ternario ...

- ▶ E' una forma **sintetica** di scrivere un **if**
- ▶ Presenta **3** parti, **tutte** obbligatorie : **condizione**, azione se **vera**, azione se **falsa** suddivise da **separatori** (**? :**)
- ▶ Testa la condizione ed esegue la conseguente azione

.....

```
int x, y = 1, z = 6;
```

```
x = y > z ? 1 : -1;
```

```
System.out.println( x );
```

.....

Operatore ternario

- E' possibile **innestare** un operatore ternario in un altro :

.....

```
int x, y = -10, z = 6;
```

```
char zz = 'A';
```

```
x = y > z ? ( zz == 'A' ? (int) zz : 100 ) : -1;
```

```
// Parentesi ( ) non obbligatorie
```

```
System.out.println(x);
```

.....

Capitolo 6

Costrutti Ciclici

Costrutti Ciclici

- ▶ In Java esistono diversi costrutti per esprimere operazioni ripetute :
- ▶ **While** (*condizione*) : opera finchè la condizione è **vera**
- ▶ **Do / while** (*condizione*) : esegue **almeno** 1 volta le operazioni e continua finchè la condizione è **vera**
- ▶ **For** (; *condizione* ;) : opera finchè la condizione è **vera**. E' molto usata con gli **arrays**

While ...

while (*condizione*)

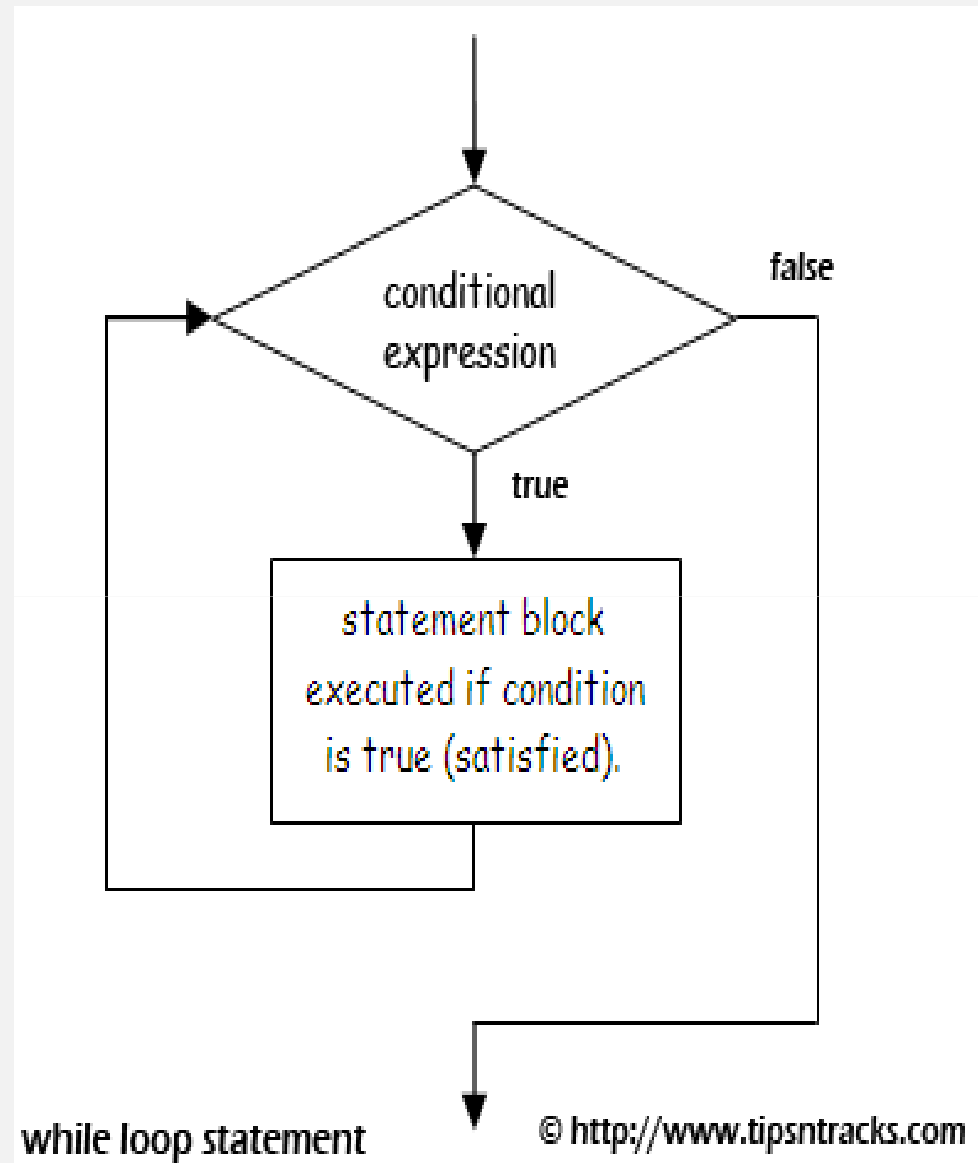
{

.....

code_block

} // end of while

.....



While ...

- ▶ Il blocco di codice viene eseguito **se e soltanto se** la condizione risulta vera
- ▶ Ad **ogni** iterazione (passaggio) viene testata la condizione
- ▶ Nel caso in cui la condizione risulti essere falsa il flusso del programma continuerebbe **immediatamente dopo** il blocco di codice che afferisce al ciclo

While esempio ...

```
int  initialSum = 500, interest = 7, years = 0;
    // Conversione in centesimi
int  currentSum = initialSum * 100;
while ( currentSum <= 100000 )
{
    currentSum += currentSum * interest/100;
    years++;
    System.out.println("Year " + years + ": " + currentSum/100);
}
```

- ▶ Il codice calcola il numero di anni che occorrono affinché una somma raddoppi ad un determinato tasso di interesse

Do ...

do

{

.....

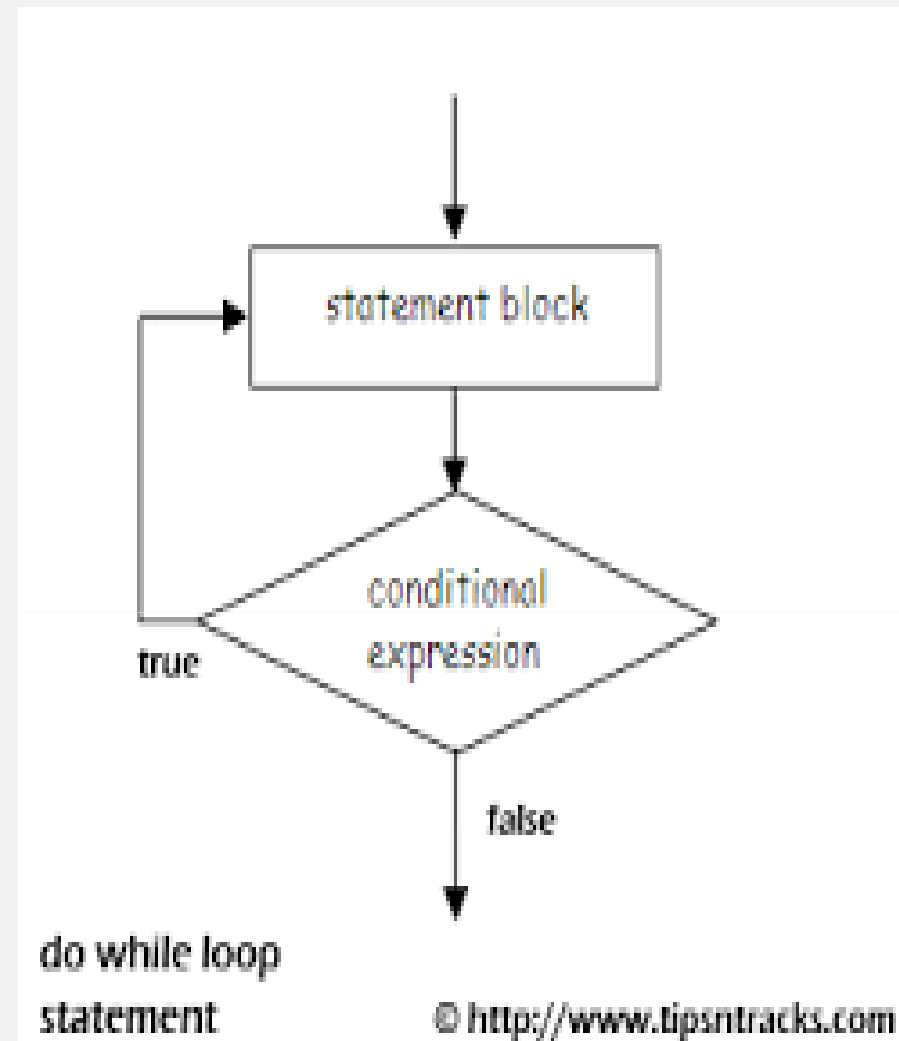
code_block;

.....

} **while**

(*boolean_expression*);

- ▶ Il codice fra le parentesi viene eseguito almeno una volta
- ▶ Il “;” che afferisce alla clausola while è **obbligatorio**



Do esempio ...

do

{

if (*currentFloor* < *desiredFloor*) {

 goUp() ;

 }

else if (*currentFloor* > *desiredFloor*) {

 goDown() ;

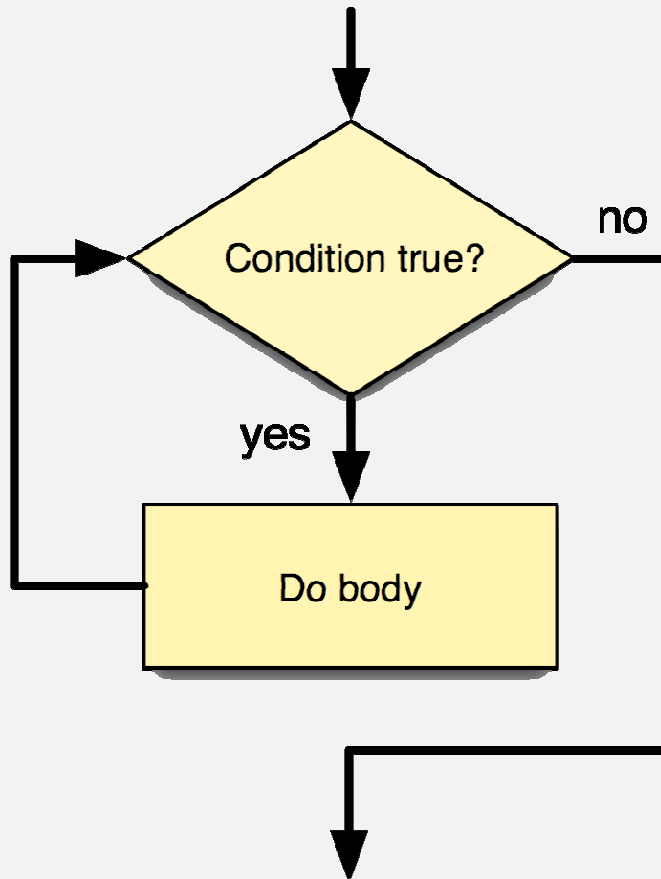
 }

} **while** (*currentFloor* != *desiredFloor*) ;

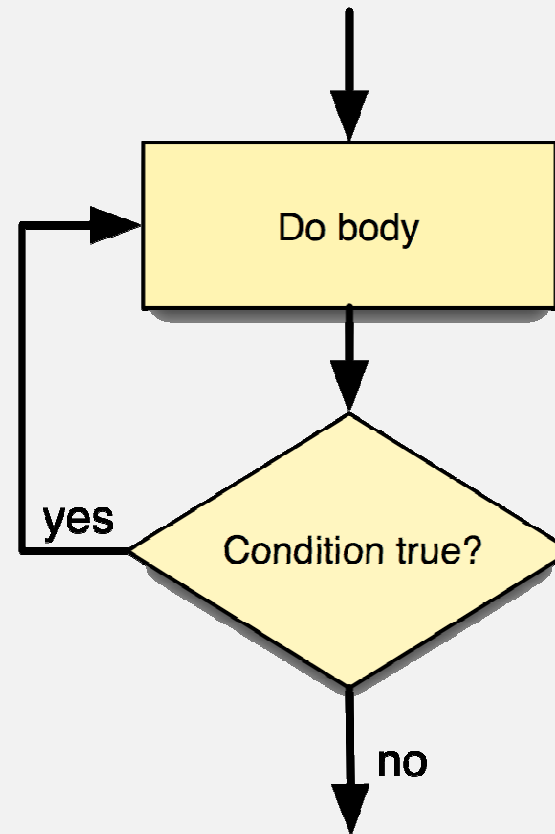
Do esempio

- ▶ L'esempio precedente simula il comportamento di un ascensore
- ▶ Se il piano corrente è minore del piano desiderato => vai SU
- ▶ Se il piano corrente è maggiore del piano desiderato => vai GIU'
- ▶ Se piano corrente è diverso da piano desiderato reitera

Costrutti Ciclici



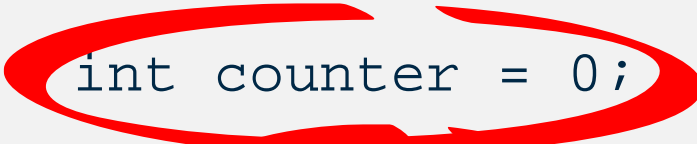

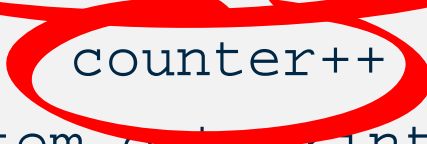
while flowchart



do/while flowchart

For

```
for (initialize[,initialize];  
      boolean_expression;   update[,update])  
  
    {  
  
      code_block  
  
    }
```


```
for(  int counter = 0;  counter < 4;  
     counter++ ) {  
    System.out.println(counter);  
  
}
```

Costrutti Ciclici ...

- ▶ Ciclo for : **break** e **continue**
- ▶ È possibile uscire anticipatamente (se si verifica una **condizione**) da un ciclo for con il comando **break**
- ▶ E' possibile saltare un blocco di codice (al verificarsi di una **condizione**) con il comando **continue**

Costrutti Ciclici ...

```
for (int i=0; i< 10;i++)  
{  
    .....  
    if (a < b)  
    {  
        .....  
        break;  
    }  
    .....  
} //fine for  
System.out.println("fuori dal ciclo");
```

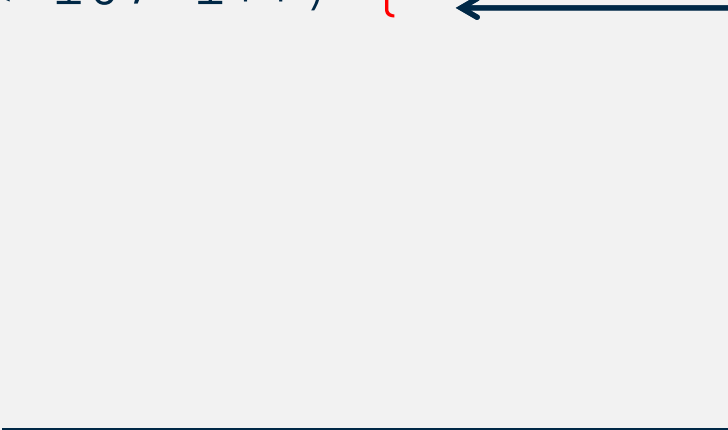


Costrutti Ciclici ...

- ▶ Si sono impostate 10 iterazioni
- ▶ Ad ogni iterazione viene testata anche la condizione $(a < b)$
- ▶ Se risulta **vera** in una **qualsiasi** delle iterazioni, si esegue anche il comando **break** che porta il flusso di esecuzione del programma direttamente alla prima istruzione fuori dal ciclo for

Costrutti Ciclici ...

```
for (int i=0; i< 10; i++) {  
    if ( a < b )  
    {  
        .....  
        continue;  
    }  
    Istruzione1;  
    Istruzione2;  
}
```



Costrutti Ciclici

- ▶ Sono state impostate 10 iterazioni
- ▶ All'interno del ciclo se risulta verificata la condizione $(a < b)$
- ▶ Si esegue il comando **continue** che porta all'iterazione successiva ma **senza** eseguire *Istruzione1* ed *Istruzione2*

Capitolo 7

Object Oriented Programmmin

Object Oriented Programming ...

OOP – **O**bject **O**riented **P**rogramming (Programmazione Orientata agli Oggetti)

E' un *paradigma di programmazione*, quindi :

- ▶ Stile di programmazione
- ▶ Insieme di strumenti concettuali
- ▶ Metodologie di astrazione per rappresentare i dati

Object Oriented Programming ...

- ▶ Si basa su **2** principali strumenti :

- Classe

- Oggetto

- ▶ E su **3** concetti :

- ☐ Incapsulamento

- ☐ Ereditarietà

- ☐ Polimorfismo

- ▶ **Classe :**

- E' un' **astrazione**. Un insieme di elementi che condividono le **stesse** caratteristiche e le **stesse** funzionalità

- ▶ **Oggetto :**

- E' un' **istanza** (creazione fisica) della classe

Object Oriented Programming ...

- ▶ Un classe consente all'utente di definire **TIP**
- ▶ Tutti gli elementi della classe hanno le **stesse** proprietà e comportamenti (azioni) **comuni**
- ▶ Es : astrazione del concetto *ContoCorrente*
 - tutti i conti hanno un *numero*
 - tutti i conti hanno un *saldo*

Object Oriented Programming ...

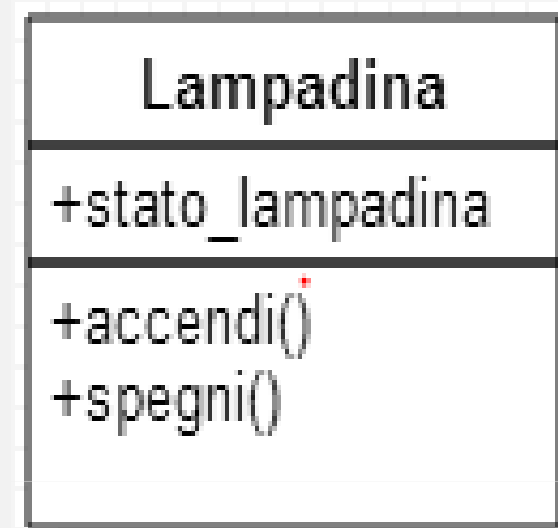
- ▶ Allo stesso tempo :
 - ☐ Il “particolare” conto ha IL PROPRIO SALDO e il PROPRIO NUMERO (ha un proprio stato)
- ▶ Su tutti I conti correnti è possibile effettuare operazioni di :
 - ☐ Prelievo ()
 - ☐ Versamento ()

Object Oriented Programming ...

- ▶ Le **richieste** che si possono fare ad un oggetto sono definite nell'**interfaccia**, ed è il **tipo** (e quindi la **classe**) a determinare l'interfaccia
- ▶ L'interfaccia definisce **quali** richieste possono essere rivolte alla particolare istanza della classe (cioè all' **oggetto**)
- ▶ In questo modo gli oggetti vengono visti come **fornitori di servizi**

Object Oriented Programming ...

- ▶ L'insieme di tutte le lampade
- ▶ Ogni lampadina avrà uno stato (attributo)
- ▶ Due funzionalità (comportamento):
 - ▶ *Accendi*
 - ▶ *Spegni*
- ▶ Il box descrive l' **interfaccia** della classe



Object Oriented Programming ...

▶ ***Incapsulamento*** o ***Information Hiding***

- ▶ Tramite l'HI una classe acquisisce caratteristiche di **robustezza**, **indipendenza** e **riusabilità** e **manutenibilità**
- ▶ Costruire una classe pubblicando (rendendo **visibile**) **solo** quello che è necessario occultandone l'implementazione
- ▶ L'eventuale **modifica** della parte nascosta sgrava dal preoccuparsi degli **effetti** che la modifica stessa potrebbe avere sugli utilizzatori della classe
- ▶ Un esempio è il *telefono* :
 - L' **interfaccia** è costituita da cornetta e tastiera
 - Tutti siamo in grado di effettuare una telefonata **ignorando** totalmente **come** avvenga, ossia i circuiti che rendono possibile telefonare

Object Oriented Programming ...

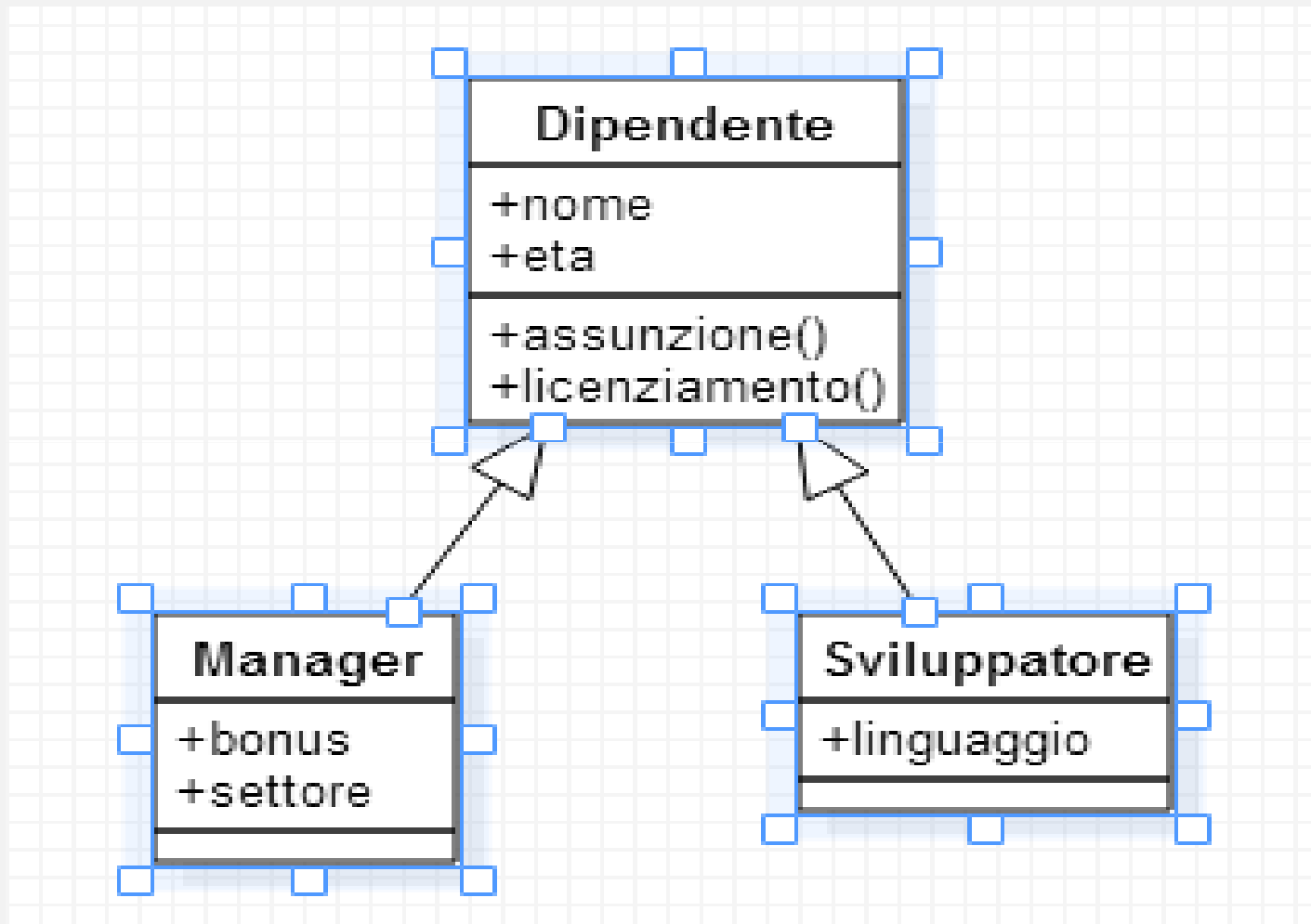
► *Ereditarietà*

- L' ereditarietà consente di **non scrivere** nuove classi che hanno che differiscono solo per un qualche dettaglio da una **già esistente**
- In fase di analisi, nel momento in cui ci si accorge che più entità del dominio hanno **caratteristiche comuni** ma **differiscono** per qualcosa (es. proprietà), si procede creando una classe (detta **superclasse** o **classe padre** o **classe base**) e poi creando da essa le altre (**sottoclassi**) specificando **solo** gli elementi **distintivi**
- Si sta così **riusando** il sw già scritto !!

Object Oriented Programming ...

- ▶ La superclasse racchiude tutte le caratteristiche e i comportamenti **comuni**
- ▶ Le classi sottoclassi (dette anche **classi figlie** o **classi derivate**) **ereditano** dalla superclasse proprietà e comportamenti e conterranno **solo** le implementazioni delle caratteristiche **distintive**

Object Oriented Programming ...



Esempio di superclasse e sottoclassi

Object Oriented Programming ...

▶ ***Polimorfismo***

- ▶ Consente di riferirci con un **unico** termine ad entità **diverse**
- ▶ Meccanismo molto potente che consente di **ridurre** il codice da scrivere e di poterlo **ampliare** per **derivazione**

Object Oriented Programming ...

- ▶ Es: calcolo area di un poligono
- ▶ Un Quadrato, un Rettangolo, un Triangolo, un Rombo
.... sono poligoni !
- ▶ La formula può **variare** a seconda del poligono (del TIPO)
- ▶ Sarà al **run - time** che l'utente specificherà di quale figura si vuole calcolare l'area

Fine prima parte