Lezione 1

JAVA

Basi e Fondamenti

R-DeV
Revolution Development
learning@r-dev.it

Argomenti della lezione 1

- Programmazione ad oggetti
- Tipi : linguaggi fortemente e debolmente tipati
- Java : concetti base, storia ed evoluzione
- Installazione Java e ambiente di sviluppo

R-DeV
Revolution Development
learning@r-dev.it

Programmazione ad oggetti

OOP → Object Oriented Programming

La Programmazione Orientata agli Oggetti (OOP) è un paradigma di programmazione che consiste nella definizione di **oggetti**, in grado di interagire tra di loro attraverso lo scambio di messaggi.

Un **paradigma di programmazione** è un insieme di concetti e astrazioni che definiscono un programma e i passi che compongono un calcolo.

Cos’è una CLASSE?

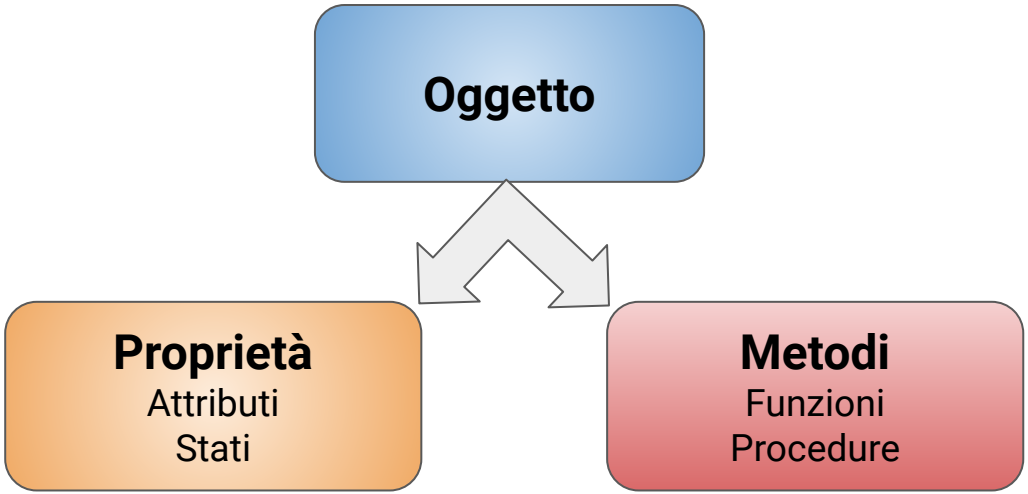
La programmazione orientata agli oggetti prevede l'organizzazione di strutture dati e procedure, in contenitori chiamati classi.

Una classe è un'astrazione di un modello reale, ed è definita come un tipo di dato astratto.

In una classe troviamo **attributi** (dati) e **metodi**(comportamenti).

Cos'è un oggetto in java?

Un oggetto è un'**istanza** di una classe, ovvero un singolo esemplare di una certa categoria. Possono essere presenti più istanze della stessa classe, quindi più oggetti dello stesso tipo.



Principi della OOP

Ereditarietà

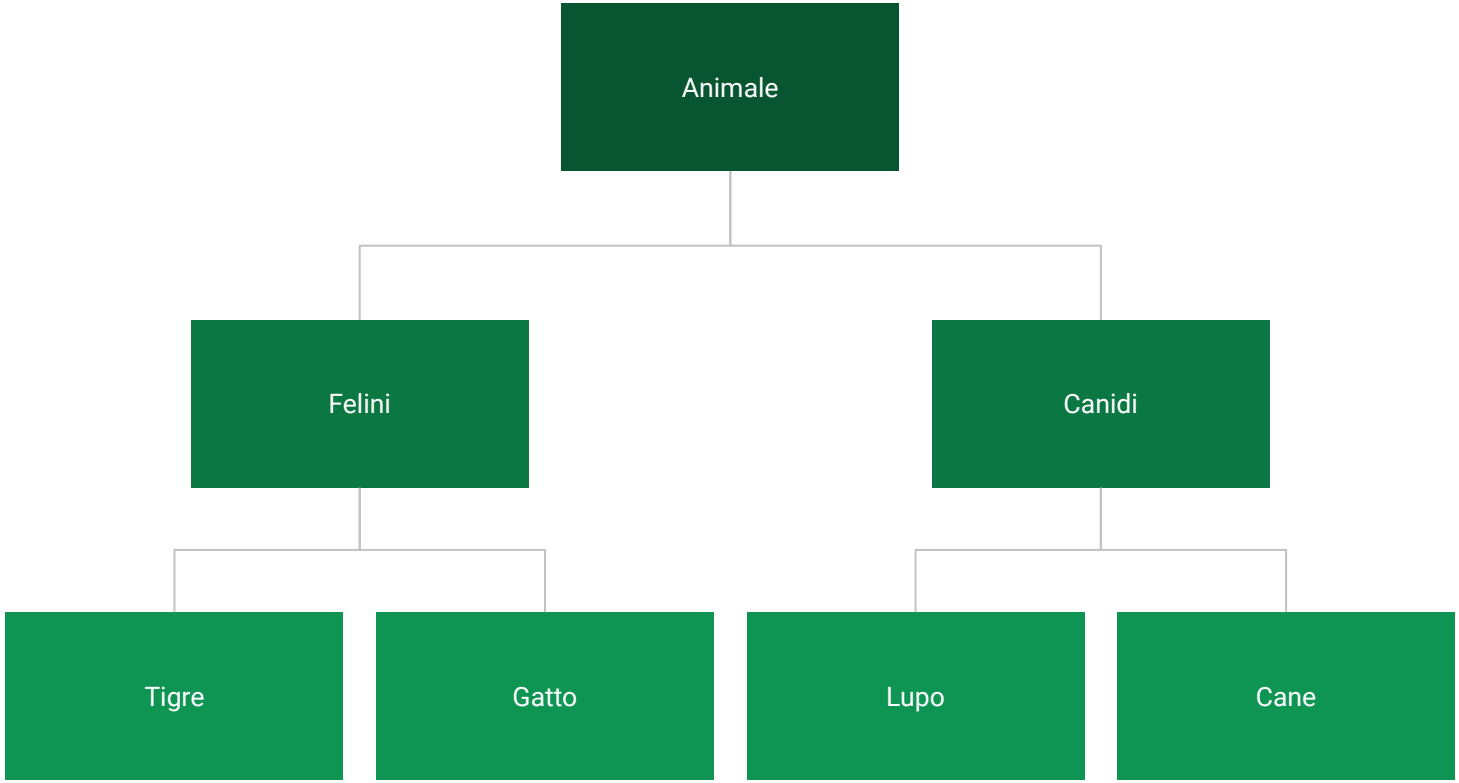
Incapsulamento

Polimorfismo

Ereditarietà

Permette di estendere una classe, facendole ereditare le proprietà di un'altra classe e ridefinendo altri. Da questo segue che possiamo definire tipi e sottotipi.

L'ereditarietà ci permette di vedere le relazioni di parentela tra classi che ereditano dalla stessa superclasse, come un albero radicato.



Incapsulamento

E’ un meccanismo del linguaggio di programmazione atto a limitare l'accesso diretto agli elementi dell'oggetto.

```
public class Cane {
  private String nome;
  public String getNome(){
     return this.nome;
  }
  public void setNome(String nome){
     this.nome = nome;
 }
}
```

Attributi → PRIVATI

Metodi → PUBBLICI

Polimorfismo

Proprietà per cui un oggetto può assumere la forma di un oggetto più generico che lo descrive (per questo poliforme). Questo concetto è legato all'ereditarietà, e permette la creazione di un oggetto A, istanziandolo da una sua sottoclasse B (cioè che eredita).

```
Animale pippo = new Cane();
```

Vantaggi della programmazione OOP

Rispetto alla programmazione tradizionale, la programmazione orientata agli oggetti (OOP) offre vantaggi in termini di:

- **modularità**: le classi sono i moduli del sistema software;
- **coesione dei moduli**: una classe è un componente software ben coeso in quanto rappresentazione di una unica entità;
- **disaccoppiamento dei moduli:** gli oggetti hanno un alto grado di disaccoppiamento in quanto i metodi operano sulla struttura dati interna ad un oggetto; il sistema complessivo viene costruito componendo operazioni sugli oggetti;
- **information hiding:** sia le strutture dati che gli algoritmi possono essere nascosti alla visibilità dall'esterno di un oggetto;
- **riuso**: l'ereditarietà consente di riusare la definizione di una classe nel definire nuove (sotto)classi; inoltre è possibile costruire librerie di classi raggruppate per tipologia di applicazioni;
- **estensibilità**: il polimorfismo agevola l'aggiunta di nuove funzionalità, minimizzando le modifiche necessarie al sistema esistente quando si vuole estenderlo

Tipizzazione

In un **linguaggio fortemente tipizzato**, il programmatore è tenuto a specificare il tipo di ogni elemento sintattico che durante l'esecuzione denota un valore, e il linguaggio garantisce che tale valore sia utilizzato in modo coerente con il tipo specificato.

Java è fortemente tipizzato

Storia di Java

Java fu inventato intorno al **1991** (presentato ufficialmente il 1995) da un gruppo di ingegneri con **James Gosling** come leader, chiamato **Green Team** il quale faceva parte della **Sun Microsystem**, un'azienda che vendeva televisori ed altri dispositivi.

La specifica è assolutamente chiara: produrre un linguaggio **leggero**, svincolato dalla particolare architettura hardware ed in grado di funzionare con risorse esigue, soprattutto poca memoria. In particolare la necessità di rendere il linguaggio **indipendente** dall'hardware spinge i progettisti ad adottare una **Virtual Machine**.

Il codice compilato che viene eseguito su una piattaforma non deve essere ricompilato per essere eseguito su una piattaforma diversa. Infatti il prodotto della compilazione è in un formato chiamato **bytecode** che può essere eseguito da una qualunque implementazione di un processore virtuale detto Java Virtual Machine.

L’inizio del successo

Intanto, agli inizi degli anni ‘90, il mercato degli apparecchi elettronici “intelligenti” non è ancora sufficientemente sviluppato e l’intero progetto rischiava il fallimento. Però, il **World Wide Web** sta crescendo a ritmi elevatissimi e il team si rende conto che la nascente tecnologia dei browser può essere notevolmente potenziata proprio da un linguaggio con le caratteristiche di Java (indipendente dall’architettura hardware, real-time, affidabile e sicuro).

Nel 1995 alla conferenza SunWorld viene presentato il browser **HotJava**, scritto interamente in Java e in grado di eseguire codice Java contenuto in pagine web (ciò che oggi è chiamato applet). Agli inizi del 1996, Sun rilascia la prima versione di Java e il linguaggio iniziò a suscitare un interesse anche se non è ancora adeguato per lo sviluppo serio di applicazioni. Ma da allora il linguaggio, attraversando parecchie versioni, è stato via via arricchito e le sue librerie sono state enormemente potenziate ed ampliate.

Versioni

A partire dal 2017, solo Java 8 e 10 sono supportati pubblicamente. Non verranno più rilasciate security update per Java 9.

- JDK 1.0 (21 gennaio 1996)
- JDK 1.1 (19 febbraio 1997)
- J2SE 1.2 (8 dicembre 1998)
- J2SE 1.3 (8 maggio 2000)
- J2SE 1.4 (6 febbraio 2002)
- J2SE 5.0 (30 settembre 2004)
- Java SE 6 (11 dicembre 2006)
- Java SE 7 (28 luglio 2011)
- Java SE 8 (18 marzo 2014)
- Java SE 9 (21 settembre 2017)
- Java SE 10 (20 marzo 2018)
- Java SE 11 (25 settembre 2018)
- Java SE 12 (19 Marzo 2019)

Java oggi

Oggi Java è di proprietà della Oracle Corporation ed è il linguaggio più richiesto e usato nel mondo del lavoro, soprattutto dalle Enterprise.

È molto versatile, viene usato in: app, software, programmazione web, big data, sistemi embedded.

Java Virtual Machine

La macchina virtuale Java, detta anche Java Virtual Machine o **JVM**, è il componente della piattaforma Java che esegue i programmi tradotti in **bytecode** dopo una prima fase di compilazione in bytecode.

La JVM è definita da una specifica, mantenuta da **Oracle**. Qualsiasi sistema che si comporti in modo coerente con tale specifica viene considerato come una particolare implementazione della JVM. Esistono implementazioni software per praticamente tutti i sistemi operativi moderni, sia gratuite che commerciali..

La disponibilità di implementazioni della macchina virtuale Java per diversi ambienti operativi è la chiave della portabilità di Java, proclamata nello slogan ***write once, run everywhere*** **(**"scrivi una volta, esegui dappertutto"). La macchina virtuale realizza infatti un ambiente di esecuzione omogeneo, che nasconde al software Java (e quindi al programmatore) qualsiasi specificità del sistema operativo sottostante.

JDK

Il **JDK** (**java development kit**) è l'insieme degli strumenti per sviluppare programmi da parte dei programmatori Java. È un prodotto della Oracle Corporation, e fin dall'introduzione di Java è sempre stato l'ambiente di sviluppo più utilizzato dai programmatori Java soprattutto per applicazioni desktop. Per applicazioni più complesse (es. applicazioni web) oggi sempre più spesso si utilizzano per lo sviluppo ed esecuzione programmi IDE a cui è possibile agganciare la JRE.

JRE

Il **JRE**, o **Java Runtime Environment**, è un ambiente di esecuzione per applicazioni scritte in linguaggio Java. In quanto tale contiene la Java Virtual Machine, le librerie standard (API Java) e un *launcher* per le applicazioni Java, necessario per avviare i programmi scritti in linguaggio Java e già compilati in bytecode; è dunque necessario quando si hanno programmi scritti in Java forniti già compilati in bytecode. Materialmente JRE è un plugin per browser in quanto è impiegato per applicazioni web che contengono componenti scritte in Java. Non costituisce un ambiente di sviluppo software e non contiene strumenti di sviluppo (compilatori e/o *debugger*): per poter sviluppare in Java a monte, a partire dal codice sorgente, è necessario infatti il Java Development Kit (che tipicamente contiene anche il JRE).

Installazione Java su Windows

Verificare l'architettura di sistema aprendo la finestra della proprietà del sistema:

- click con il tasto destro del mouse sull'icona *Questo PC*
- selezionare la voce *Proprietà*

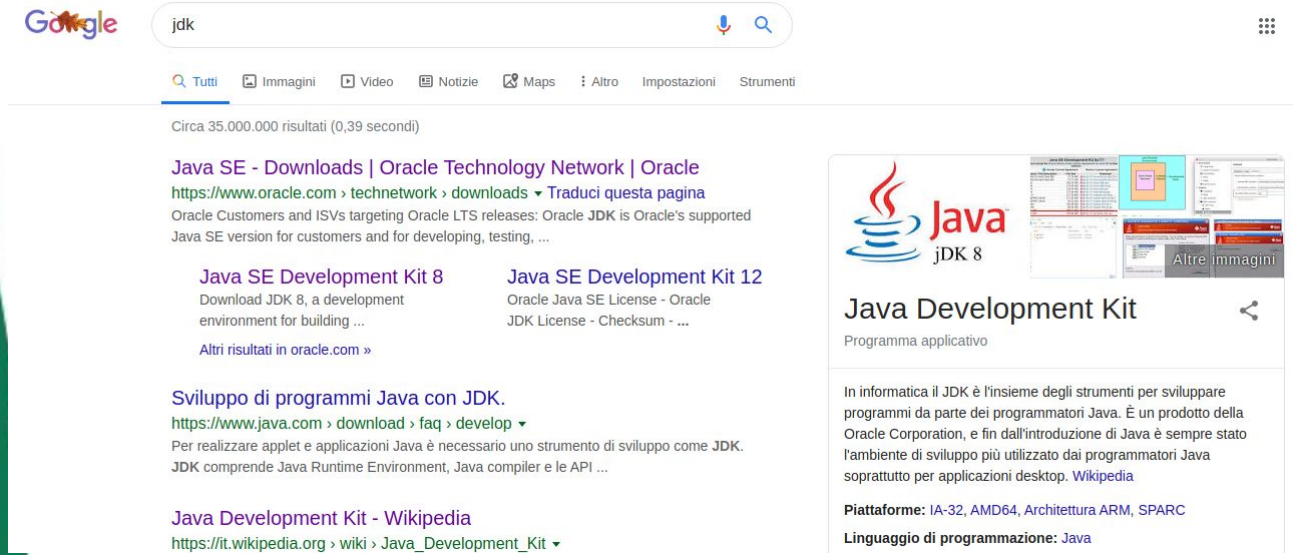
Oppure:

- aprire il *pannello di controllo*
- cliccare su *sistema e Sicurezza*
- cliccare su *Sistema*

Per capire se il nostro pc è a **32** oppure **64 bit** basta osservare la voce *Tipo sistema*

Download JDK 1.8

Controllata l'architettura possiamo scaricare il pacchetto JDK corrispondente (quello a 32 oppure a 64 bit). Colleghiamoci al sito ufficiale di **Oracle** per il download. La pagina web è facilmente raggiungibile da Google, basta cercare *jdk* e cliccare sul primo risultato che appare:



R-DeV
Revolution Development
learning@r-dev.it

Configurazione

Per poter utilizzare il JDK bisogna prima aggiungere la variabile d'ambiente al sistema.

- aprire le impostazioni di sistema avanzate
- cliccare sul pulsante *Variabili d'ambiente*

JAVA_HOME

- Click sul pulsante *Nuova*
- In nome inseriamo : JAVA_HOME
- in valore va inserito il path dove abbiamo installato il JDK

PATH

- Selezionare la variabile *Path*
- click su *modifica*
- click su *modifica testo*
- aggiungere un punto e virgola
- aggiungere alla fine il path della cartella *bin* in JDK

Verifica

Per verificare che tutto funzioni aprire il prompt dei comandi
Menù start -> tutti i programmi -> Accessori -> Prompt dei comandi
e digitare il comando:

```
java –version
```

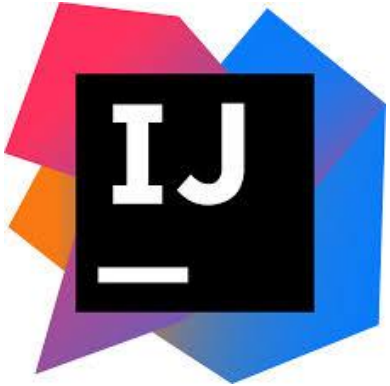
il sistema dovrebbe rispondere con tre righe simili a:

```
java version "1.8.0_221"
Java(TM) SE Runtime Environment (build 1.8.0_221-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.221-b11, mixed mode)
```

IDE di sviluppo

NetBeans

Eclipse



IntellliJ IDEA

Fine lezione 1

R-DeV
Revolution Development
learning@r-dev.it