Fondamenti di Informatica

Lezione 4

Danilo Amendola - daniloamendola.github.io

27 marzo 2018

University of Trieste

Java: imperativo, tipizzato, Oggetti e metodi

Java è un linguaggio:

- imperativo, concetti base: le variabili ed l'assegnazione delle varibili;
- fortemente tipato, in Java esistono esitono: tipi primitivo, array o Oggetto;
- ad oggetti: Java è basato sugli oggetti. Un Oggetto è un'entità software costituita da dati interni e da procedure che operano su tali dati, e fonisce un'"interfaccia" d'uso all'esterno;
- basato su metodi (procedure o funzioni).

Gli Oggetti, le Classi ed i metodi i

Java è un linguaggio ad Oggetti:

- Un Oggetto è un entità dotata di uno stato, dei suoi campi e di un comportamento;
- Il comportamento di un oggetto è definito attraverso l'uso dei metodi, i quali possono modificare lo stato o produrre risultati sulla base dei campi dell'oggetto o di valori estreni;
- Un programma Java è composto da oggetti che interagiscono tra di loro;
- In Java esistono due specie di metodi: quelli associati agli oggetti e quelli associati alla classe (metodi definiti *static*);

Attenzione a capire le differenze tra Classi, Oggetti e metodi

Un'Oggetto è un'istanza di Classe.

Gli Oggetti, le Classi ed i metodi ii

Programmare ad Oggetti:

- Agli oggetti sono associati dati, gli oggetti possono eseguire azioni;
- I dati sono contenuti nelle variabili di istanza (o campi). Le azioni vengono definite dai metodi di istanza;
- Gli oggetti vengono istanziati da variabili di tipo classe. Una classe è la definizione di un tipo di oggetto;
- La classe specifica i metodi dei suoi oggetti;
- La classe specifica il nome ed il tipo delle variabili di istanza degli oggetti, ma non specifica il loro valore;
- Un oggetto di una classe è una istanza della classe stessa;
- Il valore delle variabili di istanza è specifico delle singole **istanze** (ogni istanza possiede una propria copia delle variabili).
- Tutte le istanze di una determinata classe hanno gli stessi metodi.

Librerie Standard di Java: un introduzione

Per il momento abbiamo visto solo alcuni oggetti della libreri (Scanner, etc):

Le classi della Libreria standard di Java sono molto ricche e sono organizzate in *package*, cioè pacchetti. Alcuni pacchetti sono:

- java.util: varie classi di utilità, tra le quali Scanner (immediatamente disponibili al programmatore);
- java.lang: le classi più comunemente usate (considerate "di base");
- java.io: contiene classi per l'input/output (tramite console, files, ...);
- java.security: contiene le classi per la crittografia e altri aspettidi sicurezza
- ... e tante altre.

Per importarte una libreria:

```
import nome_package;
```

La documentazione ufficiale della Libreria Standard di Java è disponibile qui: http://docs.oracle.com/javase/7/docs/api/

Oggetti e Classi

Una semplice classe di esempio:

```
class  Quadrato{
  int lato;

  public Quadrato(int lato){
    this.lato = lato;
  }

  public int Area(){
    return lato*lato;
  }
}
```

Ed ecco quindi delle istanze/oggetti della Classe Quadrato:

```
  Quadrato q1 = new Quadrato(5);
  Quadrato q2 = new Quadrato(10);
```

Oggetti e Classi: Metodi ed invocazione di metodi

Da ricordare: **si definiscono Classi; si Creano Oggetti;**

Con la parola chiave *new* istanzia un Oggetto:

```
new Scanner(System.in)
```

Un metodo di un'altra classe si invoca cos'ì:

```
<nomeclasse>.<nomemetodo>(<parametri>)
```

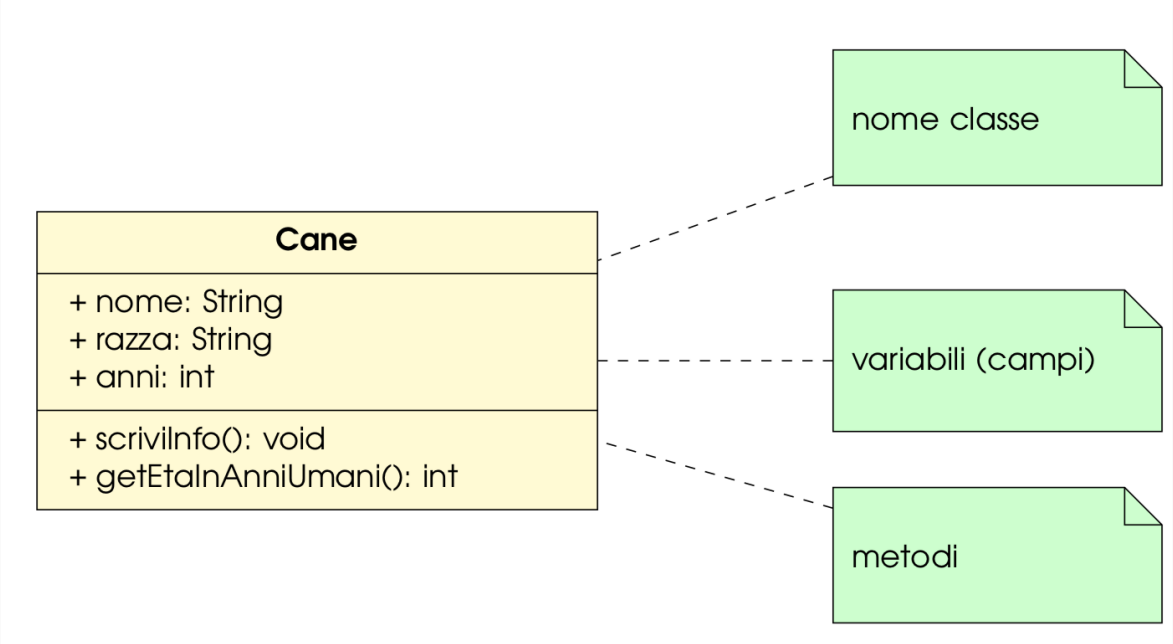
Esempio Classe e Oggetto/Istanza: Cane i

Vogliamo descrivere un Cane.

Utilizzando UML (Universal Modelling Language):

Proprietà di un Cane:

- nome;
- razza;
- anni (o età).



Le proprietà della Classe rappresentano lo stato dell'oggetto.

I metodi ne descrivono il comportamento.

Esempio Classe e Oggetto/Istanza: Cane ii

```java
public class Cane {
  // varibili di istanza, non e' buona prassi definirle pubbiche
  public String nome;
  public String razza;
  public int anni;

  // metodi di istanza
  public void scriviInfo() {
    System.out.println("Nome: " + nome);
    System.out.println("Razza: " + razza);
    System.out.println("Eta': " + anni);
  }

  public int getEtaInAnniUmani() {
    if (anni <= 2)
      return anni * 11;
    else
      return 22 + ((anni - 2) * 5);
  }
}
```

Esempio Classe e Oggetto/Istanza: Cane iii

Vediamo alcune Istanze/Oggetti di classe: esempio della classe Cane
Istanze della Classe Cane:

- caneDiShaggy[nome: Scooby-Doo, razza: alano, anni: 9]
- caneDiTopolino[nome: Pluto, razza: bracco, anni: 7]
- caneRandaggio[nome: Balto, razza: husky mezzo-lupo, anni: 8]

Modficatori di visibilità

I modificatori di visibilità in Java sono i seguenti:

- **public**: proprietà e metodi dichiarati public sono accessibili sia dall'interno che dall'esterno della classe e dalle classi derivate da essa;
- **protected** (di default): proprietà e metodi dichiarati protected, possono essere utilizzati all'interno della classe stessa e all'interno delle classi derivate, ma non sono accessibili dall'esterno della classe;
- **private**: membri e metodi dichiarati private, possono essere utilizzati soltanto all'interno della classe stessa.

Il modificatore static i

Il modificatore **static**:

- **static**: viene usato per indicare un membro statico. Una proprietà o metodo statico viene condiviso da tutte le istanze della classe, quindi un membro statico può essere utilizzato anche senza istanziare la classe (cioè non serve creare l'oggetto con la *new*);
- l'uso del modificatore **static** comporterà il caricamento in memoria della classe contenente il membro in questione.
- Un esempio di metodo statico è il metodo Math.sqrt(), che viene chiamato sulla classe Math tramite la sintassi: **Math.sqrt(numero)**;
- Math è il nome della classe e non il nome di un'istanza di essa;
- Una variabile statica, essendo condivisa da tutte le istanze della classe, assumerà lo stesso valore per ogni oggetto di una classe.

Il modificatore static ii

Vediamo un esempio di una classe che fa uso di stati:

```
class ClasseDiEsempio {
  public static int a = 0;
}

public class EsempioStatic {
  public static void main (String args[]) {
    System.out.println("a = " + ClasseDiEsempio.a); // a = 0

    ClasseDiEsempio ogg1 = new ClasseDiEsempio();
    ClasseDiEsempio ogg2 = new ClasseDiEsempio();

    ogg1.a = 10;
    System.out.println("ogg1.a = " + ogg1.a+ " ogg2.a = " + ogg2.a);
        // ogg1.a=10 ogg2.a=10
    ogg1.a = 20;
    System.out.println("ogg1.a = " + ogg1.a+" ogg2.a ="+ ogg2.a);
        // ogg1.a=20 ogg2.a=20
  }
}
```

Costruttore dell'Oggetto

Un modo più corretto di costruire l'Oggetto, Costruttore dell'Oggetto:

```
public class Cane {
  private String nome;
  private String razza;
  private int anni;

  // Costruttore di default dell'Oggetto Cane
  public Cane(){
    nome = "";
    razza = "";
    anni = 0;
  }

  // Costruttore overloaded (sovraccarico) dell'Oggetto Cane
  public Cane(String nome, String razza, int anni){
    this.nome = nome;
    this.razza = razza;
    this.anni = anni;
  }
  // .... OMISSIS
}
```

Metodi Getter e Setter

Un modo più corretto di costruire l'Oggetto, metodi ausiliari Getter e Setter dell'Oggetto:

```
public class Cane {
  private String nome;
  private String razza;
  private int anni;

  // ... OMISSIS

  //metodo getter per il nome di Oggetto Cane
  public String getNome(){
    return nome;
  }

  //Metodo setter per il nome di Oggetto Cane
  public void setNome(String nome){
    this.nome = nome;
  }

  // .... OMISSIS
}
```

La Classe Math i

Esempio: la classe Math

contiene numerosi metodi che eseguono calcoli matematici

Di seguito alcuni tra i metodi più comuni della classe Math:

- *Math.abs*(a) restituisce il valore assoluto del numero a;
- *Math.sqrt*(a) restituisce la radice quadrata di a;
- *Math.round*(a) restituisce l'arrotondamento di a all'intero più vicino;
- *Math.floor*(a) restituisce l'arrotondamento di a per difetto;
- *Math.ceil*(a) restituisce l'arrotondamento di a per eccesso;
- *Math.sin*(a) restituisce il seno di a;
- *Math.pow*(a, b) restituisce l'elevamento a potenza ab;
- *Math.random*() restituisce un numero frazionario casuale compreso tra 0.0 (compreso) e 1.0 (escluso). Esiste anche la classe Random che fa cose più sofisticate.)

La Classe Math ii

Qualche esempio con la classe Math:

```
 // generiamo un numero casuale tra 0.0 e 1.0
double numeroCasuale = Math.random();

// trasformiamo il valore di numeroCasuale in un valore tra 0.0 e
    10.0
numeroCasuale *= 10;

// trasforma in un numero tra 1.0 e 10.0 senza decimali (arrotonda
    per eccesso)
double interoCasuale = Math.ceil(numeroCasuale);

// calcola il quadrato di un numero interoCasuale
int quadrato = (int) Math.pow(interoCasuale ,2)
```

Un oggetto particolare: la classe String i

La classe String:

- contiene metodi che permettono l'elaborazione di stringhe;
- non è un tipo primitivo.

La parola riservata **new** istanzia l'oggetto attravererso il suo costruttore.

```
String nome = new String("Dario");
String cognome = new String("Brunori");
```

Esempio di creazione di due varibili: nome e cognome:

- sono due oggetti di tipo String;
- sono due istanze della classe String;
- hanno due stati diversi tra loro, ma la classe è la stessa.

Un oggetto particolare: la classe String ii

Particolarità della classe String è la seguente modalità di creazione:

```
String nome = "Dario"; // equivalente a new String("Dario");
String cognome = "Brunori"; // equivalente a new String("Brunori");
```

La classe String mette a disposizione numerosi metodi per elaborare i propri oggetti

La chiamata di un metodo su un oggetto ha la seguente sintassi:

```
<nomeoggetto>.<nomemetodo>(<parametri>)
```

Metodi comuni della classe String

Supponendo che **str** sia un oggetto di tipo String, ecco alcuni tra i metodi più comuni che si possono chiamare su esso:

- **str.length()** resitiuisce la lunghezza della stringa (numero di caratteri);
- **str.isEmpty()** restituisce true se la stringa è vuota (0 caratteri) o false altrimenti;
- **str.substring(i,j)** restituisce la porzione di str che va dal carattere in posizione i al carattere in posizione j (escluso);
- **str.substring(i)** restituisce la porzione di str che va dal carattere in posizione i alla fine della stringa.
- **charAt()**: vedi descrizione sulla documentazione!
- **indexOf()**: vedi descrizione sulla documentazione!

Esempio: uso di **substring**:

```
String saluto = "Hello , World!";
System.out.println(saluto.substring(7,12)); // stampa World
```

Esempio Conta Caratteri

Conta le occorrenze di un carattere scelgo in una stringa data.

```
import java.util.Scanner;

public class ContaCaratteri{
  public static void main(String args[]){
    char carattereScelto = ' ';
    Scanner input = new Scanner(System.in);

    System.out.println("Inserisci una riga di testo");
    String s = input.nextLine();
    int cont = 0; // contatore degli spazi
    int i = 0; // indice usato per scandire il testo
    while (i!=-1) {
      // cerca il prossimo spazio a partire dalla posizione i+1
      i = s.indexOf(carattereScelto,i+1);
      if (i!=-1)
        cont++;
  }
}
```

Confronto tra stringhe

Per confrontare due interi o due tipi nativi di Java abbiamo visto l'operatore di confronto $==$; con le stringhe e gli oggetti in generale questo **non vale**.

Per gli Oggetti l'operatore $==$ confronta l'indirizzo di memoria dei due oggetti a destra e sinistra dello stesso.

La classe **String** fornisce metodi per confrontare le stringhe:

- **s1.equals(s2)** restituisce *true* se s1 ed s2 sono uguali, i.e. s1 ed s1 sono composte dalla stessa sequanza di caratteri;
- s1.compareTo(s2) restitsce:
  - un valore minore di zero (< 0) se s1 precede lessicograficamente s2;
  - un valore maggiore di zero (> 0) se s1 segue lessicograficamente s2;
  - il valore 0 se s1 ed s2 sono uguali lessicograficamente.

  Si intende per ordine lessicografico

ATTENZIONE: è utile utilizzare i metodi toUpperCase() / toLowerCase() di String nei problemi di confronto tra stringhe.

Esempio Stringhe e cicli: Indovina la Parola

```
import java.util.Scanner;
public class IndovinaParola {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    String parolaMisteriosa = "ciao";
    int tentativi =0;     // contatore dei tentativi
    boolean indovinato=false; // "flag" che dice quando terminare

    do {
      String s = input.next();
      if (s.equals(parolaMisteriosa)) {
        System.out.println("INDOVINATO!!!");
        indovinato=true; // interrompe il ciclo
      } else {
        System.out.println("Sbagliato..."); tentativi ++;
      if (parolaMisteriosa.compareTo(s)<0)
        System.out.println("La parola misteriosa precede " + s);
      else
        System.out.println("La parola misteriosa segue " + s); }
    } while (! indovinato && tentativi <10); // due condizioni di
     uscita
    if (!indovinato)  // verifica il motivo dell'uscita dal ciclo
      System.out.println("Tentativi esauriti");
  }
}
```

FINE

Riferimenti

Riferimenti:

- Laboratorio di Programmazione - Cristian del Fabbro;
- Elementi di programmazione in Java, Rosatio Culmoneù;
- Fondamenti di Informatica, Luca Cabibbo;
- Elementi di programmazione in Java, R.Culmone;
- Programmazione in Java, Damiano Macedonio