Java

Capitolo 2 – Caratteristiche di base

Prof. Ivan Gentile

Struttura di un programma

- Ogni programma deve essere a oggetti (OOP = Object Oriented Programming)
- Almeno una «**classe**»
- Un file può contenere **una sola classe** pubblica
- La classe pubblica deve avere lo *stesso* nome del file (a parte l'estensione .java)

Primo programma (esercizio 1)

HelloWorld.java

```
1
2  public class HelloWorld {
3      public static void main(String[] args) {
4          System.out.print("Hello World!");
5      }
6
7  }
```

Output di una variabile/costante

- **`System.out.print(variabileOCostante)`**
- **`System.out.println(variabileOCostante)`**
- Convertono l'argomento in stringa e lo stampano sullo standard output (tipicamente lo schermo)

Esercizio 2

- `02-FraseWatson`
- Realizzare un progetto che stampi a video la frase indicata in figura
- Rispettare le «andate a capo»

```
Credo che in tutto il mondo ci sia mercato per forse cinque computer.
Thomas Watson, IBM, 1943.
```

Sequenze di escape

Sequenza di escape	Significato
\b	Backspace
\n	New line
\t	Tabulatore
\r	Ritorno carrello
****	Backslash
\"	Virgolette (dritte)
\’	Apostrofo

Esercizio 3

- `03-Tabella`
- Sfruttando le sequenze di escape e in particolare opportune tabulazioni realizzare un programma che stampi a video la seguente tabella

```
Persona Altezza (cm)    Misura di scarpe
=====================================
Luisa   160             36
Marco   180             43
Giulia  168             38
```

Tipi primitivi

- `nomeTipo nomeVar1, nomeVar2, …;`
  - `int somma;`
  - `float media;`
- Costanti: **`final`**
  - `final nomeTipo NOME_COST = valore;`
  - `final double PI = 3.14;`
- Caratteri tra singoli apici
  - `char c = 'Z';`
- Per le costanti reali usare sempre il punto
  - `double period = 99.0; // con 99 si ha errore`

Nome	Dim. (bit)	Val. default
int	32	0
short	16	0
long	64	0
byte	8	0
float	32	0.0f
double	64	0.0
char	16	'\u0000'
boolean		false

Esercizio 04-BMI: soluzione

- `04-BMI`
- Si definisce BMI (Body Mass Index), il rapporto tra la massa (in kg) di una persona e il quadrato dell'altezza (in m).
- Gli individui con BMI < 18.5 sono sottopeso, quelli con BMI > 25 sono sovrappeso.
- Calcolare il BMI mettendo come valori i propri, supponendo però di esprimere rispettivamente massa e altezza in grammi e centimetri
  - Fare quindi nel programma le dovute conversioni

Esercizio 04-BMI: soluzione

```java
public class BMI {

    public static void main(String[] args) {
        final int GRAMMI_PER_CHILO = 1000;
        final int CENTIMETRI_PER_METRO = 100;

        // Massa e peso propri in grammi e centimetri
        int massa = 75000;  // gr
        int altezza = 190;  // cm

        double massaKg = ((double)massa) / GRAMMI_PER_CHILO;
        double altezzaM = ((double)altezza) / CENTIMETRI_PER_METRO;

        double bmi = massaKg / (altezzaM * altezzaM);

        System.out.println("Una persona con peso " + massaKg + "kg, ");
        System.out.println("altezza " + altezzaM + "m, ");
        System.out.println("ha un BMI pari a  " + Math.round(bmi));
    }

}
```

Lettura dallo standard input

- Più complicata …
- Importare il pacchetto **`java.io`** (prima della definizione della classe)
  - `import java.io.*;`
  - Aggiungere prima della parentesi graffa di apertura di main il lancio dell'eccezione I/O
    - **throws IOException**
  - Creare un oggetto BufferedReader sullo standard Input
    - **`BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));`**
  - Leggere una riga dallo standard input
    - **`stdin.readLine()`**
  - Convertirla nel tipo atteso usando le classi Wrapper, esempio
    - **int altezza = Integer.*parseInt(stdin.readLine());***

Lettura dallo standard input

05-BMI_Input

```
import java.io.*;

public class BMI_Input {

    public static void main(String[] args) throws IOException {
        final int GRAMMI_PER_CHILO = 1000;
        final int CENTIMETRI_PER_METRO = 100;

        BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));

        System.out.println("CALCOLATORE BMI");

        System.out.print("Inserire la massa (in grammi): ");
        int massa = Integer.parseInt(stdin.readLine());  // gr

        System.out.print("Inserire l'altezza (in centimetri): ");
        int altezza = Integer.parseInt(stdin.readLine());  // gr


        double massaKg = ((double)massa) / GRAMMI_PER_CHILO;
        double altezzaM = ((double)altezza) / CENTIMETRI_PER_METRO;

        double bmi = massaKg / (altezzaM * altezzaM);

        System.out.println("Una persona con peso " + massaKg + "kg, ");
        System.out.println("altezza " + altezzaM + "m, ");
        System.out.println("ha un BMI pari a  " + Math.round(bmi));
    }

}
```

Es: 30-Operazioni Aritmetiche

- Leggere due numeri double dallo standard input
- Stampare **con una sola operazione di stampa** il risultato delle quattro operazioni aritmetiche (addizione, sottrazione, moltiplicazione, divisione) come in figura
- *Facoltativo*: se i numeri sono interi stampare anche il resto modulo (ovvero il resto della divisione del primo per il secondo)

```
Inserisci i numeri uno per riga
14
3
n1 + n2 = 17.0
n1 - n2 = 11.0
n1 * n2 = 42.0
n1 / n2 = 4.666666666666667
n1 % n2 = 2
```

```
Inserisci i numeri uno per riga
4
2.3
n1 + n2 = 6.3
n1 - n2 = 1.7000000000000002
n1 * n2 = 9.2
n1 / n2 = 1.7391304347826089
```

Classe Scanner (1/2)

- Introdotta in Java 1.5 per semplificare la lettura dei dati
- Creare un oggetto Scanner
  - **`Scanner sc = new Scanner(System.in);`**
- A seconda della variabile che si vuole leggere si usa il metodo `next`
  - `int x = sc.nextInt();`

- `boolean nextBoolean()`
- `byte nextByte()`
- `double nextDouble()`
- `float nextFloat()`
- `int nextInt()`
- `String nextLine()`
  - Prende tutta riga tranne il newline che lo scarta
- `long nextLong()`
- `short nextShort()`
- I next tranne nextLine() lasciano nel buffer il newline

Classe Scanner (2/2)

- Se l'utente non fornisce il tipo atteso si genera un'eccezione.
- Per evitare l'eccezione si può verificare prima se il valore dell'utente è del tipo atteso, per farlo usare has…

- `boolean hasNextLine() // Returns true if se il file non è finite, false se è EOF.`
- `boolean hasNextBoolean() // Returns true if the next value entered by the user is a valid boolean value.`
- `boolean hasNextByte() // Returns true if the next value entered by the user is a valid byte value.`
- `boolean hasNextDouble() // Returns true if the next value entered by the user is a valid double value.`
- `boolean hasNextFloat() // Returns true if the next value entered by the user is a valid float value.`
- `boolean hasNextInt() // Returns true if the next value entered by the user is a valid int value.`
- `boolean hasNextLong() // Returns true if the next value entered by the user is a valid long value.`
- `boolean hasNextShort() // Returns true if the next value entered by the user is a valid short value`

Utilità di Scanner (1/2)

- **Scanner è utile per le stringhe soprattutto quando ogni riga del file testo contiene più dati**
  - se volessimo usare substring dovremmo sapere con certezza dove iniziano e dove finiscono questi dati
- Scenner chiama **token** una sottostringa di una riga delimitata da spazi, tab o newline

Utilità di Scanner (2/2): estrazione dei token

```
Scanner in = new Scanner(System.in);
String line = in.nextLine();
Scanner t = new Scanner(line);
```

e poi si fanno tante invocazioni del metodo next su tale oggetto fino a che il metodo `hasNext()` è `true`,

```
while (t.hasNext()) {
      String token = t.next();
      // elabora token
}
```

Stringhe

- Classe **`String`**
- Inclusa nel pacchetto **`java.lang`**
  - Importato automaticamente in Java
- `Due modi per la dichiarazione`
  - **`String nomeStringa = "valore";`**
  - **`String nomeStringa = new String("valore");`**
- Non sono proprio equivalenti…

Operatori

- + concatenamento
- Da stringa a tipo primitivo: metodi **parse** della classe wrapper
  - `int x = Integer.parseInt(“100”);`
  - `short x = Short.parseShort(“100”);`
  - `long x = Long.parseLong(“100”);`
  - `byte x = Byte.parseByte(“100”);`
  - `float x = Float.parseFloat(“19.95”);`
  - `double x = Double.parseDouble(“19.95”);`
  - `boolean x = Boolean.parseBoolean(“true”);`

Metodi (1/3)

`s.length()`	Restituisce il numero di caratteri di s
`s.charAt(ind)`	Restituisce il carattere che si trova nella posizione ind, che si conta a partire da zero.
`char[] toCharArray()`	Converte la stringa su cui è invocata in un vettore di caratteri
`String toLowerCase()`	Converte la stringa su cui è invocata in minuscolo
`String toUpperCase()`	Converte la strina su cui è invocata in maiuscolo
`s.indexOf('x')`	Restituisce la prima occorrenza (a partire da zero) del carattere x in s, o -1 se non è presente. Si può passare anche una sottostringa. Si può opzionalmente passare un secondo parametro intero che è la posizione da cui iniziare la ricerca
`s.lastIndexOf(p)`	Come indexOf ma restituisce l'ultima occorrenza
`s1.equals(s2)`	restituisce true se le stringhe hanno stesso contenuto
`s1.equalsIgnoreCase(s2)`	come equals ma ignora la differenza tra maiuscola e minuscola

Metodi (2/3)

s1.compareTo(s2)	restituisce 1 se s1 alfabeticamente viene dopo s2, -1 se viene prima e 0 se uguale
s1.compareToIgnoreCase(s2)	come compareTo ma ignora le maiuscole
boolean contains(CharSequence)	Restituisce true se la stringa su cui è invocato contiene il parametro passato. Il parametro può essere String, StringBuffer o StringBuilder
s2 = s1.substring(2,8)	estrae la sottostringa da s1 con i caratteri da 2 a 8-1=7 e la mette in s2. Se si mette un solo parametro prende tutti i caratteri da quella posizione fino alla fine.
s2 = s1.replace ('A', 'a')	restituisce in s2 la stringa s1 in cui tutti i caratteri 'A' sono stati sostituiti da 'a'
String replaceAll(String old, String new)	Restituisce la stringa su cui è invocata ma dopo aver rimpiazzato tutte le occorrenza del primo parametro con il secondo. Il primo parametro può essere un'espressione regolare
String replaceFirst(String old, String new)	Come replaceAll ma rimpiazza solo la prima occorrenza.

Metodi (3/3)

```
String s= "Elettronica Informatica Telecomunicazioni";
String v[] = s.split("\\s+");
for (String e: v)
    System.out.println(e);
```

```
Elettronica
Informatica
Telecomunicazioni
```

String [] split(String)	Divide la stringa per ogni match nell'espressione regolare passata come parametro e restituisce un array.
Boolean startsWith(String)	Restituisce true se la stringa su cui è invocato inizia con quella passata come parametro. Accetta anche un secondo parametro intero opzionale che indica da che posizione iniziare la ricerca
s.trim()	Restituisce la stringa s senza eventuali spazi iniziali e finali.
String valueOf(primitiveType)	Restituisce una rappresentazione in strina di un qualsiasi tipo primitivo passato come parametro
s1.endsWith(s2)	restituisce true se s1 termina con la stringa s2

Es: 06-ConversioneDate

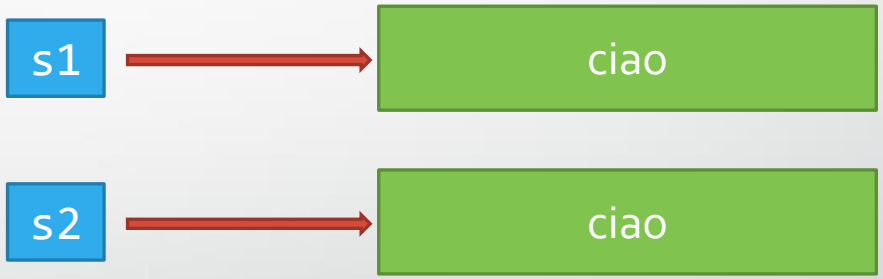
- `06-ConversioneDate`
- Convertire una data dal formato americano (es. Luglio 4, 1776) al formato internazionale (es. 1779-Luglio-4)

Stringhe e riferimenti (1/3)

- Il nome di una stringa è un **riferimento**
  - In generale sono *diverse* anche se hanno lo stesso *contenuto*

```
String s1 = new String (“ciao”);
```

```
String s2 = new String (“ciao”);
```




```
System.out.println((s1==s2));
```

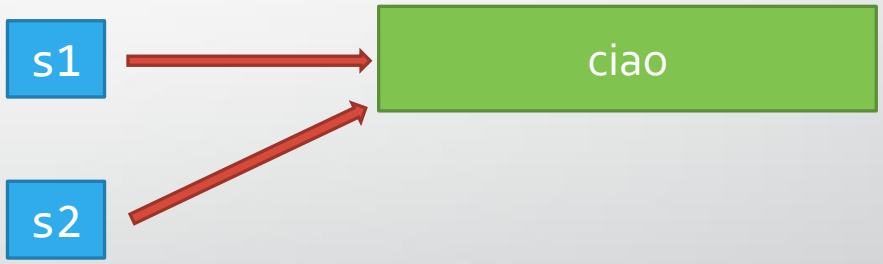
false

Stringhe e riferimenti (2/2)

- Nel caso in cui sono definite come tipo primitivo ed hanno lo stesso valore si fa un'ottimizzazione: puntano alla stessa locazione di memoria

```
String s1 = "ciao";

String s2 = "ciao";
```




```
System.out.println((s1==s2));
```

true

Stringhe e riferimenti (3/3)

- Per verificare l'uguaglianza usare il metodo **equals** (e non ==)
- In questo modo si verifica il contenuto
- **`System.out.println((s1.equals(s2));`**
  - Restituisce true in entrambi i casi

Esercizio

- `07-ConfrontoStringhe`
- Date due stringhe lette in input, stabilire se le stringhe sono uguali e nel caso in cui non lo sono quale delle due è minore (secondo l'ordine lessicografico)

Stringhe: oggetti immutabili

- Una stringa una volta creata non può variare il suo valore
- Cambiare valore a un suo reference significa creare un'altra
- Se quella precedente non è più referenziata viene cancellata dal garbage collector

```
String a = "ciao";
String b = "hello";
a = b;
```

```
String a = "alfa";
String b = a;
a = "beta";
```

Stringhe: oggetti immutabili (2/2)

- Le operazione di modifica delle stringhe (come il concatenamento) sono molto onerose
  - Si creano e distruggono oggetti di continuo
- Le classi `StringBuffer` e `StringBuilder` permettono di modificare una stringa

Esercizio sul costrutto swtich

- `08-GiorniMese`
- Realizzare un programma che dato un anno e un mese stabilisce il numero di giorni di quel mese.
- Usare il costrutto switch e tenere conto degli anni bisestili

Esercizio (facoltativo)

- `09-DiagnosiFebbre`
- Realizzare un programma che (attraverso una serie di selezioni) a partire da un serie di domande proposte a un paziente provi a stabilire la causa della sua febbre.
- Le domande da fare al paziente e le risposte da fornirgli sono indicate nel file `DiagnosiFebbre.pdf`
- Suggerimenti: trattare una sola volta il blocco che parte dalla domanda «Ha `dolori alle ossa e/o alle giunture?`», sfruttando la funzione `System.exit(0);`

Es: 10-MediaAritmetica

- `10-MediaAritmetica`
- Chiedere all'utente di inserire un elenco di numeri non negativi, uno per riga. La fine dell'elenco va indicata inserendo un numero negativo. Riportare la media dei numeri non negativi positivi.
  - Suggerimento: usare un ciclo while

Es: 11-TestoMinuscolo

- `11-TestoMinuscolo`
- Dato un testo in input, stampare lo stesso testo in tutte lettere minuscole
- Suggerimenti:
  - Inserire il testo in una sola stringa
  - Digitare EOF per la fine del testo di input
  - Nel caso di EOF `readLine()` restituisce la costante `null`
  - La stringa vuota è `""`

Es: 12-TavolaPitagorica

- `12-TavolaPitagorica`
- Stampare la tavola pitagorica come in figura
- Suggerimento:
  - 2 cicli innestati
  - `System.out.println(); // Stampa solo un newline`

TAVOLA PITAGORICA

*	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10
2	0	2	4	6	8	10	12	14	16	18	20
3	0	3	6	9	12	15	18	21	24	27	30
4	0	4	8	12	16	20	24	28	32	36	40
5	0	5	10	15	20	25	30	35	40	45	50
6	0	6	12	18	24	30	36	42	48	54	60
7	0	7	14	21	28	35	42	49	56	63	70
8	0	8	16	24	32	40	48	56	64	72	80
9	0	9	18	27	36	45	54	63	72	81	90
10	0	10	20	30	40	50	60	70	80	90	100

Funzioni (1/2)

- Dette anche **metodi**
- Fino ad ora sappiamo usare solo metodi `static`
  - Detti **metodi di classe**
- Fino ad ora sappiamo usare solo variabili locali
- Per richiamare un sottoprogramma da `main` bisogna dichiararlo anch'esso static
  - Metodo di classe

Funzioni (2/2)

- Un metodo statico si richiama come `NomeClasse.nomeMetodo(..)`
  - Nel caso è un metodo della classe stessa si può evitare di mettere `NomClasse.` avanti.
- Parametri:
  - Per valore se primitivi
  - Per indirizzo se oggetti

11-Combinazioni

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Combinazioni {

    static long fattoriale(int n) {
        long nFattoriale = 1;
        for (int i = 2; i <= n; i++)
            nFattoriale *= i;
        return nFattoriale;
    }
    public static void main(String[] args) throws IOException {
        BufferedReader stdin = new BufferedReader(new InputStreamReader (System.in));

        System.out.println("CALCOLO COMBINAZIONI SEMPLICI di n elementi di classe k");
        System.out.println("Attenzione deve essere n<=20 (perché la formula è calcolata come n!/[(n-k)!k!] e"
                + "quando l'argomento è troppo grande il fattoriale non riesce a stare in un long)");

        System.out.print("Inserisci il numero di elementi n = ");
        int n = Integer.parseInt(stdin.readLine());
        System.out.print("Inserisci la classe k = ");
        int k = Integer.parseInt(stdin.readLine());

        long c = fattoriale(n) / (fattoriale(n-k) * fattoriale(k));

        System.out.println("Numero di combinazioni c = " + c);

    }
}
```

Esercizi

- Realizzare attraverso funzioni statiche i seguenti programmi
  - `14-Max2Num`: Calcolo del massimo tra due numeri
  - `15-MaxNNum`: Massimo tra n numeri dati in ingresso
  - `16-Celsius2Fahrenheit`: Conversione tra gradi celsius a fahrenheit e viceversa

Vettori (1/2)

- Dichiarazione
  - `tipo[] nomeArray;`
  - `tipo nomeArray[];`
- Definizione
  - `nomeArray = new tipo[dimensione];`
  - Valore iniziale di ogni elemento: `null` (se oggetto)
- Dimensione può essere a tempo di esecuzione
- Attributo `length`

Vettori (2/2)

- Inizializzazione
  - `int[] primes = { 2, 3, 5, 7, 11, 13, 17 };`
  - `int[] primes = new int[] { 2, 3, 5, 7, 11, 13, 17 };`

Esercizi

- 17-LetturaStampaArray: leggere un vettore di stringhe e stamparlo
- 18-MassimoVettore: dato un vettore di numeri reali calcolare il massimo
- 19-VettoreDeiMassimi: dati due vettori v1 e v2 letti in ingresso di numeri interi, creare un nuovo vettore max il cui generico elementi max[i] è il massimo tra v1[i] e v2[i]
  - Usare una funzione per la lettura dei due vettori

Esercizi

- `20-RicercaSequenziale`: realizzare e provare un programma che dato un vettore di caratteri e un carattere letto in ingresso verifica attraverso una ricerca sequenziale se il carattere è presente nel vettore e in caso affermativo ne restituisce la posizione, se non è presente (-1)
- `21-RicercaBinaria`: effettuare un programma per la ricerca binaria in un vettore (ordinato) di caratteri.

Esercizi

- `22 – VettoreContrario1`: Dato un vettore di n elementi costruirne un altro al contrario (in cui cioè il primo elemento del primo vettore diventa l'ultimo del secondo vettore, il secondo il penultimo etc.) e stampare quest'ultimo
- `23- OperazioniVettori`: Dati due vettori di n elementi reali e uno scalare reale, calcolare il prodotto tra lo scalare e il primo vettore, la somma dei vettori, il prodotto naturale dei vettori (cioè la somma dei prodotti delle componenti omologhe).

Esercizi

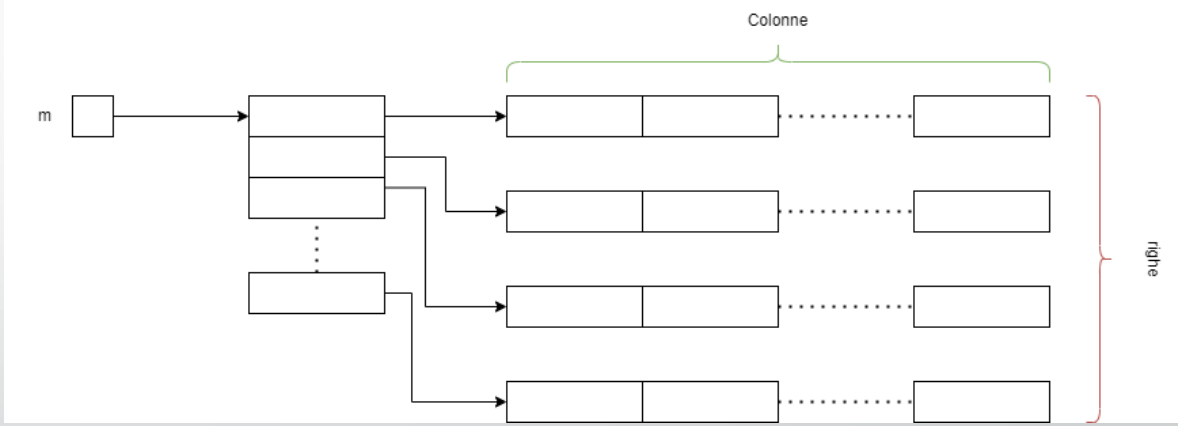
- `24 – CambioPrimoNome`: Dato un array di nomi di persone letti in input, costruire un nuovo array che ha gli stessi nomi tranne per quelli che iniziano con Anna che vanno sostituiti con Maria. Inoltre gli spazi vanno tolti e il secondo nome va messo in minuscolo. Esempio
  - `Anna Rita - > Mariarita`
  - `Annarosa -> Mariarosa`
  - `Anna Rosaria -> Mariarosaria`
  - `Luisa - > Luisa`

Array a più dimensioni

- Dichiarazione
  - `tipo [][]…[] nomeArray;`
  - `Esempio:`
  - `int [nRighe][nColonne] matrice;`
- Definizione
  - `nomeArray = new tipo [dim1][dim2]…[dimN];`

Matrice

- Gli array a due dimensioni sono detti anche **matrici** o **tabelle**
- Una matrice di m righe e n colonne ha m x n elementi
- Internamente è memorizzata come un array di array



Dimensione di una matrice

- Come per tutti gli array le dimensioni (righe e colonne) possono essere stabilite a tempo di esecuzione
- Non è neccessario portarsi dietro (per esempio nel passaggio dei parametri) le dimensioni di una matrice
- **`mat.length`** è il numero di righe
- **`mat[0].length`** è il numero di colonne
  - Ovviamente al posto di `0` si può usare qualsiasi numero di riga tanto tutte le righe hanno stesso numero di colonne

Lettura matrice

Vedere progetto
`40-LetturaMatrici`

Per righe

```
for(int i = 0; i<m; i++)
    for(int j = 0; j <n; j++)
        mat[i][j] = sc.nextInt();
```

```
1 2 3
4 5 6
```

Se `m = 2, n = 3`
E in input abbiamo: `1 2 3 4 5 6`

Per colonne

```
for(int j = 0; j<n; j++)
    for(int i = 0; i <m; i++)
        mat[i][j] = sc.nextInt();
```

```
1 3 5
2 4 6
```

Lettura/stampa di una matrice con le funzioni

Lettura

```
public static int [][] leggiMatrice(int m, int n){...}
```

Stampa

```
public static void stampaMatrice(int m[][]) {...}
```

Si può aggiungere un altro parametro che rappresenta dove leggere/stampare (da Standard Input/Output, da file,..)

Esercizio: prodotto tra matrici

- Date due matrici m1 e m2 calcolare il prodotto (se possibile)
- Due matrici sono **compatibili rispetto al prodotto** se il numero di colonne della prima è pari al numeri di righe della seconda
  - Supponiamo che m1 ha dimensione m x **n**
  - La matrice m2 deve avere **n** x p

Esercizio: prodotto tra matrici

- Se le matrici sono compatibili il **prodotto mp = m1 x m2** sarà una matrice che il numero di righe di m1 e il numero di colonne di m2, cioè **m x p**
- Il generico elemento (i, j) della matrice mp sarà il **prodotto naturale** della riga i di m1 per la colonna j di m2
- Il prodotto naturale è la somma dei prodotti delle componenti omologhe
  - Esempio `[1, 2, -4], [0, 3, 5]` il prodotto naturale è
  - `1 * 0 + 2 * 3 + (-4)*5 = 0 + 6 -20 = -14`

Esempio

```
m1 =
[2 4
 6 8]
```

```
m2 =
[1 0 4
-2 6 -2]
```

```
mp =
[ mp(1,1) mp(1,2) mp(1,3)
  mp(2,1) mp(2,2) mp(2,3)]
```

```
mp(1,1) = [2 4] * [1 -2] = 2 * 1 + 4 *(-2) = -6
mp(1,2) = [2 4] * [0 6] = 2 * 0 + 4 * 6 = 24
mp(1,3) = [2 4] * [4 -2] = 2 * 4 + 4 * (-2) = 0
mp(2,1) = [6 8] * [1 -2] = 6 * 1 + 8 * (-2) = -10
mp(2,2) = [6 8] * [0 6] = 6 * 0 + 8 * 6 = 48
mp(2,3) = [6 8] * [4 -2] = 6 * 4 + 8 * (-2) = 8
```

```
mp =
[ -6 24 0
  -10 48 8]
```

Prodotto tra matrici: passaggi del programma

```
1) Leggere le matrici m1 e m2 da file
// Parametri
// nomefile
// intestazione (vero o falso)
// righe e colonne della matrice (indifferente se intestazione è vera)
public static int[][] leggiMatrice(String nomeFile,boolean intestazione, int r, int c)

1.1) Stampo le matrici m1 e m2 su schermo (opzionale)
// nomefile è null se voglio stampare su schermo
public static void stampaMatrice(String nomeFile, int[][] mat)


2) Controllare se le matrici sono compatibili
public static boolean isCompatible(int[][] mat1, int[][] mat2)
Se falso il programma termina (con un messaggio)
Se vero procede

3) Prodotto tra matrici
public static int[][] prodotto(int[][] m1, int[][] m2)


4) Stampo la matrice  del prodotto su file
```

Approfondimento sulla funzione prodotto tra matrici

```
3) Prodotto matrici

a) Doppio ciclo innestato (su quante righe e su quante colonne??)

b) per il calcolo del prodotto tra riga i di m1 e colonna j di m2
creo un sottoprogramma che mi estrae un array riga o colonna da una matrice
public static int [] estrai(int[][] mat, int indice, char tipo)

c) Faccio il prodotto naturale tra riga e colonna estratte per calcolare mp[i][j]
public static int prodotto(int[] aRiga, int[] aColonna)
```