ARRAY RESTITUITI DA FUNZIONI

- **Gli array Java *possono essere restituiti* come risultato di funzioni, come qualunque altro oggetto**

ESEMPIO

```
int[] creaTabellaQuadrati(int n)
```

ESEMPIO DI ARRAY RESTITUITO

ESEMPIO

- **crea un array di interi di opportuna dimensione**
- **lo riempie con i quadrati richiesti**
- **lo restituisce al chiamante**

```
int[] creaTabellaQuadrati(int n){
  int[] v = new int[n];
  for (int i=0; i<v.length; i++)
     v[i] = i*i;
  return v;
}
```

ESERCIZIO 2 (1/4)

Scrivere una funzione che restituisca `true` se due array di interi sono uguali (cioè hanno tutti gli elementi corrispondentemente uguali).

Signature della funzione richiesta:

```
public static boolean idem(int[] a, int[] b){
  ...
}
```

ESERCIZIO 2 (2/4)

```
public static boolean idem(int[] a, int[] b){
   if (a.length != b.length) return false;
   for (int i=0; i<a.length; i++){
       if (a[i] != b[i]) return false;
   }
   return true;
}
```

Se la lunghezza è diversa, non sono certamente identici.

Non sono identici se *anche un solo elemento* differisce.

Progettare un main di prova

- **occorre creare tre array (uno di dimensione diversa dagli altri)**
- **i due della stessa dimensione devono avere contenuto diverso**
- **occorre fare almeno tre prove**

ESERCIZIO 2 (3/4)

```
public static void main(String args[]){
  int[] v1 = {2,3,4}, v2 = {2,3,5},
        v3 = {2,3},   v4 = {2,3,4};
  System.out.println( idem( v1, v3 ) );
  System.out.println( idem( v1, v2 ) );
  System.out.println( idem( v1, v1 ) );
  System.out.println( idem( v1, v4 ) );
}
```

ESPRESSIONI DI INIZIALIZZAZIONE di un array con costanti

Lunghezza diversa

Contenuto diverso

Stesso array

Contenuto uguale

Output:

Le ESPRESSIONI DI INIZIALIZZAZIONE causano la creazione di un OGGETTO ARRAY *esattamente come se fosse stata usata la new:* semplicemente, l'array ha quei valori come contenuto iniziale.

ESERCIZIO 2 (4/4)

```
public static void main(String args[]){
 System.out.println(
     idem( new int[]{2,3,4}, new int[]{2,3} ));
 System.out.println(
     idem( new int[]{2,3,4}, new int[]{2,3,5} ));
  System.out.println(
     idem( new int[]{2,3,4}, new int[]{2,3,4} ));
}
```

INIZIALIZZAZIONE "al volo" in fase di costruzione

Contenuto uguale, ma array distinti

Output:

ESERCIZIO 3 (1/3)

E se volessimo una funzione analoga per due array di OGGETTI, ad esempio di `Counter`?

Si vuole verificare se i due array contengono riferimenti uguali ad oggetti di tipo Conuter

Signature della funzione richiesta:

```
public static boolean
    idem(Counter[] a, Counter[] b){
  ...
}
```

ESERCIZIO 3 (2/3)

```
public static boolean idem(Counter[] a, Counter[] b){
  if (a.length != b.length) return false;
  for (int i=0; i<a.length; i++){
     if (!a[i].equals(b[i])) return false;
  }
  return true;
}
```

Come prima

Cambia il test: ora deve basarsi sul concetto di "uguale" *PROPRIO DELLA CLASSE `Counter`*

Adattare il main di prova

- **occorre creare ora array di Counter…**
- **…e ricordarsi di creare anche gli oggetti all'interno dell'array!**
- **Anche qui, possibile creare gli oggetti all'inizio, oppure "al volo"**

ESERCIZIO 3 (3/3)

```
public static void main(String args[]){
 System.out.println(idem( new Counter[]{
  new Counter(2),new Counter(3),new Counter(4)},
                          new Counter[]{
  new Counter(2),new Counter(3)} ));
 System.out.println(idem( new Counter[]{
  new Counter(2),new Counter(3), new Counter(4)},
                         new Counter[]{
  new Counter(2),new Counter(3), new Counter(5)} ));
 System.out.println(idem( new Counter[]{
  new Counter(2),new Counter(3), new Counter(4)},
                         new Counter[]{
  new Counter(2),new Counter(3), new Counter(4)} ));
}
```

E SE `equals()` cambiasse?

MATRICI come array di array (1/2)

Una matrice è un array di array (di reali o interi)

```
double[][] m;
```

Per crearla dobbiamo quindi

- <u>prima</u> creare l'array "esterno" (la "colonna di righe")

```
m = new double[3][];  // ipotesi: 3 righe
```

- <u>poi</u> creare i singoli array "interni" (le singole righe)

```
m[0] = new double[5]; // ipotesi: 5 col.
m[1] = new double[5];
m[2] = new double[5];
```

NB: si potrebbero definire anche matrici *irregolari*, con righe di lunghezza diversa le une dalle altre

MATRICI come array di array (2/2)

In alternativa, si può scrivere *più velocemente*:

```
m = new double[3][5];  // matrice 3 x 5
```

con lo stesso significato di cui sopra.

Adatto solo per matrici *regolari*

Dunque,

- **`m.length` è il numero di righe** ***(3 nell'esempio)***
- **`m[i].length` è il numero di colonne della riga i-ma**
  (per matrici regolari, i può essere qualsiasi)

Per accedere alle celle si usano ovviamente due indici:

```
m[0][0] = 1.2;   m[1][0] = -2.6;   ...
```

ESERCIZIO 4: somma di due matrici

```
public static double[][] sommaMatrici(
                    double[][] a, double[][] b) {
  double[][] c = new double[a.length][a[0].length];
  for (int i=0; i<a.length; i++)
        for (int j=0; j<a[0].length; j++)
            c[i][j] = a[i][j] + b[i][j];
  return c;
}
```

```
public static void main(String[] args){
  double[][] m = {{ 1.2, 2.3, 2.3 },
                  { 7.4, 5.1, 9.8 } };
  double[][] n = {{ 5.0, 4.0, 1.3 },
                  { 1.2, 0.3, 3.2 } };
  double[][] q = sommaMatrici(m,n);
  stampaMatrice(q);                 // da fare
}
```

ESERCIZIO 5: prodotto di due matrici

Analogo al precedente…

… MA attenzione alle dimensioni delle matrici!

fatelo voi!

Progetto del main di collaudo:

```
public static void main(String[] args){
  double[][] m = {{ 1.4, 4.3, 5.3 },      // 2 x 3
                  { 7.4, 5.0, 4.8 } };
  double[][] n = {{ 5.0, 3.0},            // 3 x 2
                  { 1.2, 0.6},
                  { 0.5, 8.1} };
  double[][] q = prodottoMatrici(m,n);    // 2 x 2
  stampaMatrice(q);                       // da fare
}
```

Array: funzionalità predefinite nei package standard

FUNZIONI DI UTILITÀ PREDEFINITE

- **Varie funzioni di utilità sugli array**
  package `java.util.Arrays`
  - **ordinamento: `sort`**
  - **ricerca binaria: `binarySearch`**
  - **riempimento: `fill`**
  - **e ovviamente: `equals`**

UN NUOVO ESERCIZIO

1. **Creare un array di interi**
2. **Riempirlo di valori casuali**
   → **libreria `java.util.Random`,**
   metodo `nextInt`
3. **Ordinarlo in senso crescente**
   → **libreria `java.util.Arrays`,**
   metodo `sort`
4. **Cercare al suo interno un certo valore**
   → **libreria `java.util.Arrays`,**
   metodo `binarySearch`

ESERCIZIO 2

```
public class SortAndSearchArray {
 ...
 public static void main(String args[]){
  int[] v = new int[20];
  java.util.Random gen = new java.util.Random();
  for (int i=0; i<v.length; i++)
      v[i] = gen.nextInt(30);
  show(v); // mia funzione che stampa tutto l'array
  java.util.Arrays.sort(v);
  show(v); // mia funzione che stampa tutto l'array
  int n = gen.nextInt(30);
  int pos = java.util.Arrays.binarySearch(v, n);
  System.out.println("cercato " + n +
                ": trovato alla posizione " + pos);
 }
}
```

ESERCIZIO 2

Output (caso di numero trovato):

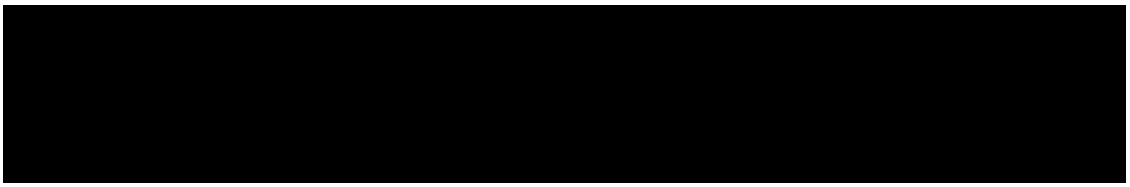
Output (caso di numero non trovato):

Un valore negativo restituito da binarySearch indica che il numero richiesto non è stato trovato (sarebbe da inserire alla posizione 20-1 = 19)

VARIANTE: array di stringhe

```
public class SortStringArray {
 ...
 public static void main(String[] args){
  show(args);
  java.util.Arrays.sort(args);
  show(args);
  int pos =
      java.util.Arrays.binarySearch(args, "Enrico");
  System.out.println("trovato alla posizione " + pos);
}
 }
}
```

Riga di comando:



Un esercizio completo: calcolo del codice fiscale

ESERCIZIO: CODICE FISCALE

Problema:

Scrivere un programma che, a partire dai dati anagrafici, calcoli il *codice fiscale* di una persona.

Il C.F. è composto di 5 parti:

- ***tre lettere* identificative del *cognome***
- ***tre lettere* identificative del *nome***
- ***due cifre, una lettera e altre due cifre*, indicanti *sesso e data di nascita***
- ***una lettera e tre cifre,* indicanti il *Comune di nascita***
- ***una lettera di controllo,* calcolata sulla base delle lettere e delle cifre precedenti.**

ESERCIZIO: CODICE FISCALE

Problema:

Scrivere u
anagrafici,

Il C.F. è composto

- ***tre lettere* identificat**
- ***tre lettere* identificat**
- ***due cifre, una lettera e***
  indicanti *sess*
- ***una le***
  di nas
- ***una le***
  lettere

Le prime tre consonanti (se non bastano, le vocali; se non bastano ancora, una X)

Esempi:
RO**SS**I **MAR**IO → RSS MRA
RO**SS**I **G**AS**T**O**N**E → RSS GTN

La prima, la terza e la quarta consonante (se non ci sono quattro consonanti, si usano le prime tre; se non bastano, si usano le vocali; se non bastano ancora, una X)

ESERCIZIO: CODICE FISCALE

Problema:

Scriv… anag… …na.

Il C.F. è…

- ***tre lettere* id… *…e del cognome*;**
- ***tre lettere* ide…**
- ***due cifre, una lette… indicanti *sesso e d…***
- ***una lettera e tre cifre,* indicanti il *Comune di nascita*;**
- ***una lettera di contr… calcolata sulla base delle lettere e del…***

Anno, mese e giorno di nascita
Per le donne, giorno aumentato di 40
Mese = A, B, C, D, E, H, L, M, P, R, S, T

Esempi:
01/02/1978 (maschio) → **78B01**
25/12/1969 (femmina) → **69T65**

Sigla che si trova su apposite tabelle
Bologna = A944, Reggio E. = H223, etc
→ *8276 diverse sigle, inclusi Stati esteri*

ESERCIZIO: CODICE FISCALE

Problema:

S… …na.

Il …

- **…**
- **…**
- **…**
- ***d…***
- ***una lettera …ontrollo,* calcolata sulla base delle lettere e delle cifre precedenti.**

Cifre 0 ÷ 9 → Lettere A ÷ J
Lettere di posto pari (2,4,..) → 0-25
Lettere di posto dispari (1,3,…)→ vedi tabella:

B	A	K	P	L	C	Q	D	R	E	V	O	S	F	T	G	U
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
H	M	I	N	J	W	Z	Y	X								
17	18	19	20	21	22	23	24	25								

Alla fine, si somma tutto e si prende il risultato modulo 26.

ESERCIZIO: CODICE FISCALE

Problema:

S… …ti

…na.

Esempio: Rossi Mario, 12/6/76, Bologna

```
R S S M R A 7 6 H 1 2 A 9 4 4 ?
R S S M R A H G H B C A J E E ?
  18  12   0   6   1   0   4    (pari)
8   12   8  17  17   5  21     (dispari)
Totale: 138 mod 26 = 8 → I
R S S M R A 7 6 H 1 2 A 9 4 4 I
```

Il …

- **…**
- **…**
- **…**
- **…**
  di nasc…
- ***una lettera di controllo,* calcolata sulla base delle lettere e delle cifre precedenti.**

UN PROBLEMA: OMOCODIA

L'algoritmo di calcolo del C.F. ha un difetto:
***può generare due codici identici* se due persone**

- **hanno "quasi" lo stesso <u>cognome e nome</u>**
- **sono nate nello <u>stesso giorno/mese/anno</u> e <u>comune</u>**
- **sono dello <u>stesso sesso</u>**

***Improbabile?* NO!**

- **al luglio 2000 c'erano già <u>24.000 casi</u>**
- **e più di 1000 nuovi ogni anno**

… ma per identificare l'omocodia bisogna accedere all' Anagrafe Tributaria!

Soluzione: le cifre numeriche sono via via sostituite da lettere, a partire da destra:

```
0→L  1→M  2→N  3→P  4→Q
5→R  6→S  7→T  8→U  9→V
```

OMOCODIA: CONSEGUENZE

- **In conseguenza del rischio di omocodia, per legge *solo l'Agenzia delle Entrate può generare un codice fiscale* perché solo l'Anagrafe Tributaria può verificare se un codice risulta già assegnato.**
  - in tal caso emette un NUOVO CODICE a ENTRAMBI i soggetti
- **Ogni altro soggetto può solo *verificare la correttezza formale di un codice,* "ovviamente" tenendo conto delle varianti da omocodia**
  - **programma Java disponibile sul sito dell'Agenzia delle Entrate:**
    Home → Servizi → Codice fiscale o Tessera Sanitaria → Codice Fiscale → Programma per verificare la correttezza formale del codice fiscale

ARCHITETTURA

Per definire l'architettura dell'applicazione bisogna rispondere ad alcune *domande fondamentali*:

- **Quali e quante classi? Chi usa cosa?**
- ***Componenti software (statici) o classi come ADT (oggetti dinamici)?***
- **Quale modello di interazione fra *l'applicazione nel suo complesso* e l'esterno?**
  - **da console (standard input)**
  - **dalla linea di comando**
  - in altro modo ancora...

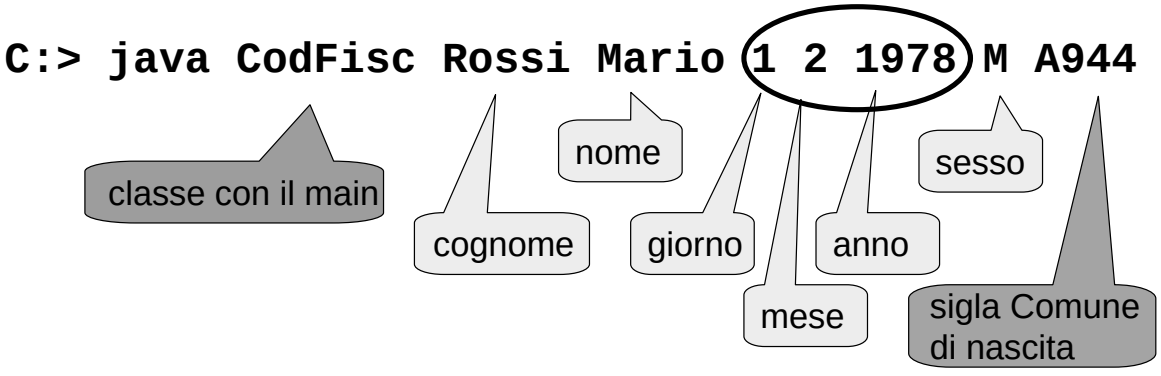
ARCHITETTURA: UNA POSSIBILE SCELTA

- **Quali e quante classi?**
  - **una sola pubblica: `CodFisc`**
- ***Componenti software (statici) o classi come ADT (oggetti dinamici)?***
  - **una componente software (statico) con una sola funzione (statica) pubblica: `public static String calcolaCodice(...)`**
- **Quale modello di interazione fra *l'applicazione nel suo complesso* e l'esterno?**
  - **argomenti dalla linea di comando del main**

INTERAZIONE CON L'ESTERNO

Ipotesi: argomenti sulla linea di comando del main

- ***Già.. ma FORMATTATI COME?***
- ***La data esiste come entità atomica (1/2/1978) o vengono invece fornite le singole entità componenti?***



Attenzione: *i caratteri numerici non sono numeri !*

INTERAZIONE CON L'ESTERNO

Sottoproblema: come trasformare stringhe in numeri?

- **dalla riga di comando otteniamo stringhe di caratteri come "1978", "2"….**
- **… *ma questi non sono gli interi millenovecentosettan-totto, due, etc: OCCORRE UNA TRASFORMAZIONE!***

```
C:> java CodFisc Rossi Mario 1 2 1978 M A944
```

La trasformazione di *stringhe* in *numeri (int, float, …)* viene svolta dalle funzioni statiche:

```
int Integer.parseInt(String)
float Float.parseFloat(String)
```

etc.; ad esempio:

```
int x = Integer.parseInt("1978");
```

LA CLASSE `CodFisc`

Struttura interna (funzioni statiche *private*):

- **una funzione *calcolaSiglaCognome* che calcola le prime tre lettere**
- **una funzione *calcolaSiglaNome* che calcola le seconde tre lettere**
- **una funzione *calcolaSiglaData* che calcola il blocco centrale (data di nascita e sesso)**
- **(la sigla del comune di nascita si assume data)**
- **una funzione *calcolaUltimaLettera* che calcola la lettera finale**

Interfaccia esterna (funzioni statiche *pubbliche*):

- ***la funzione calcolaCodice che si avvale dei servitori privati precedenti.***

LA FUNZIONE `calcolaSiglaCognome`

- **Interfaccia:**

```
private static
 String calcolaSiglaCognome(String cognome)
```

- **Specifica:**
  - occorre una stringa `sigla` per accumulare via via il risultato
  - esplorare `cognome` da sinistra verso destra: finché contiene consonanti, e purché non ne siano già state trovate tre, concatenare la consonante trovata in coda a `sigla`
  - se `sigla` non contiene ancora tre consonanti, ripetere l'esplorazione di `cognome` cercando le vocali: finché ce ne sono e fintanto che `sigla` non contiene tre lettere, accodare la vocale trovata a `sigla`
  - se dopo ciò in `cognome` manca ancora una lettera *(possibile solo con i cognomi di due sole lettere)*, accodare a `sigla` una 'X'
  - restituire `sigla`

COMPLETARE L'ESERCIZIO

Per finire l'esercizio:

- definire le altre funzioni
- definire un main che invochi `calcolaCodice` con i parametri appropriati

Osservazione (cambiamento dei requisiti):

- *Cosa cambierebbe* se i dati, invece che dalla linea di comando del main, provenissero
  - dallo standard input, da un file, ...
  - dalla rete, da un'interfaccia grafica… ??