Array

La nozione di algoritmo è strettamente legata a quella di struttura dati. Una struttura dati è un modo per organizzare dati correlati tra loro. Un array è una struttura di dati di lunghezza fissa composta di dati omogenei aventi lo stesso tipo

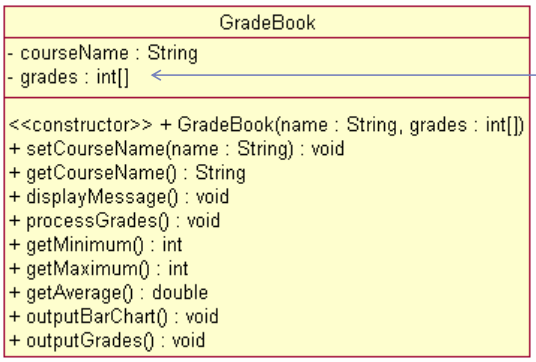
Sommario

- La nozione di array (vettore)
- Dichiarare e creare array
- Uso dell'istruzione **for** per iterare gli elementi di un array
- Gli array come parametri di metodi
- Uso di array nella classe GradeBook
- Esempio finale

Array: *premessa*

- Nello sviluppo del software è necessario disporre di strutture dati complesse e di potenti procedure di calcolo per l'elaborazione dei dati memorizzati in tali strutture
- Esempio:
  - strutture dati:
    - organizzate in modo sequenziale per memorizzare un numero non noto a priori di oggetti
  - procedure:
    - per inserire, cancellare e aggiornare oggetti nelle strutture
    - per ordinare e ricercare e oggetti nelle strutture
    - per fondere e dividere strutture; per estrarre sotto-strutture
- Nei linguaggi ad oggetti esistono potenti classi che realizzano tali concetti di strutture e procedure
- La struttura dati più semplice è rappresentata dalla nozione di Array

Array nella classe "GradeBook"



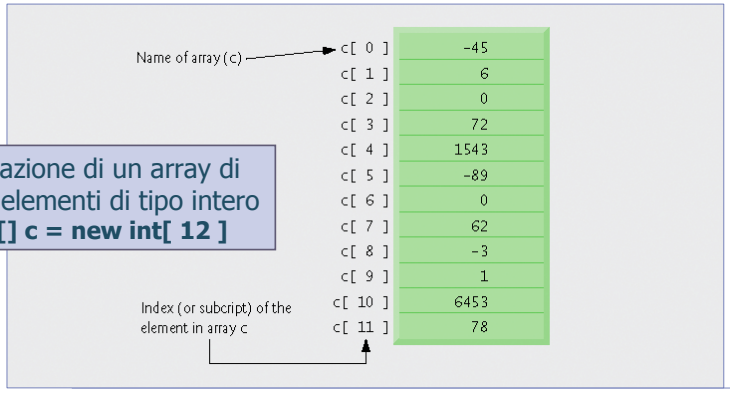
Array:
oggetto che rappresenta una "sequenza di oggetti". In questo caso "sequenza di interi", gli interi rappresentano i voti

→ Gli array in Java

questa lezione

Array: concetti di base (1)

- Un array è una struttura dati sequenziale che raggruppa un numero noto a priori di riferimenti ad oggetti o di tipi primitivi
- Gli oggetti all'interno dell'array sono chiamati elementi o componenti
- Gli elementi dell'array devono essere omogenei (tutti dello stesso tipo primitivo o della stessa classe)



creazione di un array di 12 elementi di tipo intero
int[] c = new int[12]

Array: concetti base (2)

- In Java un array è un oggetto (dunque ha attributi e metodi)
  - System.out.print(c.length) // stampa 12

attributo pubblico di c

- Il numero di posizione di un elemento nell'array viene chiamato indice dell'elemento
  - System.out.print(c[1]) // stampa 6 → l'elemento 6 ha // indice 1
- Per accedere ad un elemento dell'array bisogna specificare il nome del riferimento all'array e l'indice dell'elemento nell'array
  - c[1] = 7 // modifica dell'elemento di indice 1 nell'array c

Array: esempi di creazione

```
int MAX = 6;
boolean esiti[] = new boolean[MAX];
float[] valori = new float[2*3];
Utente[] listaUtenti = new Utente[6];
int[] numeri = {10, 9, 8, 7, 6, 5};
System.out.println( esiti[0] );     // stampa "false"
System.out.println( valori[3] );    // stampa "0.0"
System.out.println( numeri[1] );    // stampa "9"
listaUtenti[5].print();             // stampa "Mario 18"
```

```
public class Utente {
  private String nome;
  private int eta;
  Utente ( ) {
     nome = "Mario";
     eta = 18;
  }
  Utente ( String n,  int e ) {
     nome = n;
     eta = e;
  }
  public void print( ) {
     System.out.print( nome + " " + eta);
  }
}
```


Array: inizializzazione (1)

```
1  // Classe InitArray.java
2  // Crea un vettore di interi
3
4  public class InitArray
5  {
6     public static void main( String args[] )
7     {
8        int vettore[]; // dichiara un array denominato vettore
9
10       vettore = new int[ 10 ]; // crea spazio in memoria per il nuovo array
11
12       System.out.printf( "%s%8s\n", "Index", "Value" );
13
14       // visualizza gli elementi del nuovo array
15       for ( int counter = 0; counter < vettore.length; counter++ )
16          System.out.printf( "%5d%8d\n", counter, vettore[ counter ] );
17    } // fine main
18 } // fine classe InitArray
```

```
Index   Value
    0       0
    1       0
    2       0
    3       0
    4       0
    5       0
    6       0
    7       0
    8       0
    9       0
```

Uso dell'istruzione **for** per scandire tutti gli elementi del array

Array: inizializzazione (2)

```
1  // Classe InitArray.java
2  // Inizializza gli elementi di un vettore tramite un elenco di valori
3
4  public class InitArray
5  {
6     public static void main( String args[] )
7     {
8        // Un lista di interi specifica il valore
9        int array[] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };
10
11       System.out.printf( "%s%8s\n", "Index", "Value" );
12
13       // visualizza gli elementi del array
14       for ( int counter = 0; counter < array.length; counter++ )
15          System.out.printf( "%5d%8d\n", counter, array[ counter ] );
16    } // fine main
17 } // fine classe InitArray
```

Dichiara un array di interi denominato array e lo inizializza con un lista di interi. Il numero di elementi della lista determina la lunghezza del nuovo array

```
Index   Value
    0      32
    1      27
    2      64
    3      18
    4      95
    5      14
    6      90
    7      70
    8      60
    9      37
```


Array: inizializzazione (3)

```
1  // Classe InitArray.java
2  // Calcola il valore da attribuire agli elementi di un array
3
4  public class InitArray
5  {
6     public static void main( String args[] )
7     {
8        final int ARRAY_LENGTH = 10;
9        int vettore[] = new int[ ARRAY_LENGTH ]; // crea un nuovo array
10
11       // calcula il valore di ogni elemento del vettore
12       for ( int counter = 0; counter < vettore.length; counter++ )
13          vettore[ counter ] = 2 + 2 * counter;
14
15       System.out.printf( "%s%8s\n", "Index", "Value" );
16
17       // visualizza gli elementi del nuovo array
18       for ( int counter = 0; counter < vettore.length; counter++ )
19          System.out.printf( "%5d%8d\n", counter, vettore[ counter ] );
20    } // fine main
21 } // fine classe InitArray
```

Dichiara una costante tramite il *qualificatore* **final**. Le costanti devono essere inizializzate prima del loro uso e non possono essere modificate

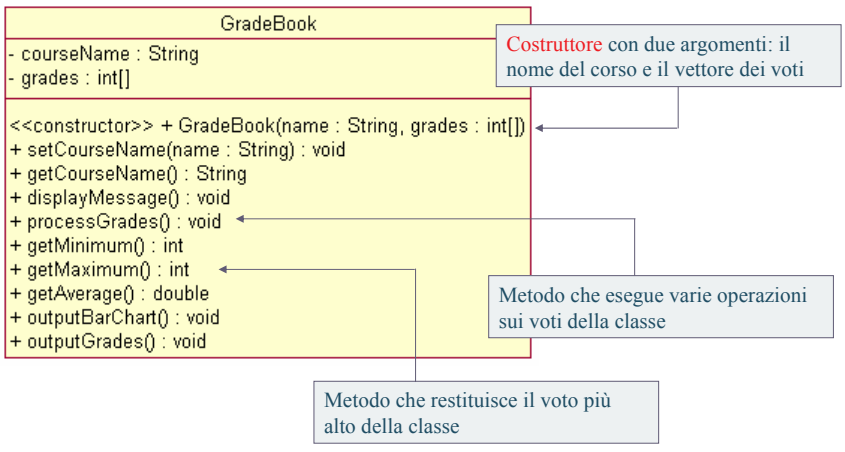
```
Index   Value
    0       2
    1       4
    2       6
    3       8
    4      10
    5      12
    6      14
    7      16
    8      18
    9      20
```

Somma degli elementi di un array

```
1  // Classe SumArray.java
2  // Calcola la somma degli elementi di un array
3
4  public class SumArray
5  {
6     public static void main( String args[] )
7     {
8        int vettore[] = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };
9        int total = 0;
10
11       // somma ogni elemento del vettore al totale
12       for ( int counter = 0; counter < vettore.length; counter++ )
13          total = total + vettore[ counter ];
14
15       System.out.printf( "Total of array elements: %d\n",   total );
16    } // fine main
17 } // fine classe SumArray
```

```
Total of array elements: 849
```


La classe GradeBook (1)



Vedremo solo i tre metodi evidenziati
gli altri sono lasciati per esercizio

La classe GradeBook (2)

```
1  // Classe GradeBook.java
2  // Registro di classe con i voti registrati in un array
3
4  public class GradeBook
5  {
6     private String courseName; // nome del corso
7     private int grades[]; // array dei voti
8
9     // Costruttore con due argomenti
10    public GradeBook( String name, int gradesArray[] )
11    {
12       courseName = name; // inizializza courseName
13       grades = gradesArray; // inizializza il vettore dei voti
14    }
15
16    // visualizza un messaggio di benvenuto
17    public void displayMessage()
18    {
19       // stampa il nome del corso
20       System.out.printf( "Welcome to the grade book for\n%s!\n\n",
21          getCourseName() );
22    }
23
```

Il costruttore prende due parametri: il nome del corso e l'array dei voti. Quando una applicazione crea un oggetto di tipo GradeBook, essa passa un array preesistente al costruttore, il quale a sua volta assegna il riferimento dell'array alla variabile di istanza grades (riga 13). La dimensione di grades è determinata dalla classe che passa l'array al costruttore.

La classe GradeBook (3)

```
24    // Trova il voto massimo
25    public int getMaximum()
26    {
27       int highGrade = grades[ 0 ]; // assume grades[ 0 ] come massimo
28
29       // cicla sull'array dei voti
30       for ( int x = 1; x < grades.length; x++ )
31       {
32          // calcola il nuovo voto massimo
33          if ( grades[ x ] > highGrade )
34             highGrade = grades[ x ];
35       } // fine for
36
37       return highGrade; // restituisci il voto piu alto
38    } // fine metodo getMaximum
39
40
```

Il metodo getMaximum() trova il voto più alto della classe. Assume all'inizio che il voto più alto sia quello nella posizione 0 dell'array dei voti (riga 27). Successivamente cicla sull'array dei voti e, ogni volta che trova un voto maggiore del massimo, aggiorna il massimo al valore di quel voto (righe 30-35)

La classe GradeBook (4)

```
41    // Esegue varie operazioni sui voti della classe
42    public void processGrades()
43    {
44       // visualizza l'array dei voti
45       outputGrades();
46
47       // chiama il metodo getAverage per calcolare la media dei voti
48       System.out.printf( "\nClass average is %.2f\n", getAverage() );
49
50       // chiama i metodi getMinimum e getMaximum
51       System.out.printf( "Lowest grade is %d\nHighest grade is %d\n\n",
52          getMinimum(), getMaximum() );
53
54       // chiama il metodo outputBarChart per visualizzare l'istogramma dei voti
55       outputBarChart();
56    }
57
58    // segue l'implementazione degli altri metodi della classe GradeBook
59    // ...
60 }  // fine classe GradeBook
```

Il metodo processGrade() esegue varie operazioni sui voti della classe.

Fa uso degli altri metodi di GradeBook

La classe GradeBookTest (1)

```
1  // Classe GradeBookTest.java
2  // Crea un oggetto GradeBook usando un array di voti
3
4  public class GradeBookTest
5  {
6     // metodo main
7     public static void main( String args[] )
8     {
9        // crea l'array dei voti degli studenti
10       int gradesArray[] = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };
11
12       GradeBook myGradeBook = new GradeBook(
13          "CS101 Introduction to Java Programming", gradesArray );
14       myGradeBook.displayMessage();
15       myGradeBook.processGrades();
16    } // fine main
17 } // fine classe GradeBookTest
```

Crea un oggetto di tipo GradeBoook passando al costruttore la stringa "CS101 Introduction to Java Programming" e l'array di interi gradesArray creato alla linea 10

Stampa il messaggio di benvenuto del corso CS101 Introduction to Java Programming

Chiama il metodo processGrades dell'oggetto myGradeBook che esegue varie elaborazioni sui voti degli studenti del corso CS101 Introduction to Java Programming

La classe GradeBookTest (2)

```
Welcome to the grade book for
CS101 Introduction to Java Programming!

The grades are:

Student  1:  87
Student  2:  68
Student  3:  94
Student  4: 100
Student  5:  83
Student  6:  78
Student  7:  85
Student  8:  91
Student  9:  76
Student 10:  87

Class average is 84.90
Lowest grade is 68
Highest grade is 100

Grade distribution:
00-09:
10-19:
20-29:
30-39:
40-49:
50-59:
60-69: *
70-79: **
80-89: ****
90-99: **
  100: *
```

Il metodo displayMessage dell'oggetto myGradeBook stampa il messaggio di benvenuto

Il metodo outputGrades di myGradeBook stampa i voti degli studenti contenuti nell'array grades di myGradeBook

I metodi getAverage, getMinimum e gatMaximum di myGradeBook stampano rispettivamente la media, il minimo e il massimo dei voti degli studenti contenuti nell'array grades di myGradeBook

Il metodo outputBarChart di myGradeBook stampa l'istogramma dei voti degli studenti contenuti nell'array grades di myGradeBook

Esempio finale (1)

- Progettare una applicazione Java che gestisce le bollette telefoniche di una certa utenza telefonica
- L'applicazione prevede la definizione di tre classi
  - BollettaTelefono: definisce le caratteristiche di una bolletta
  - UtenzaAnnualeTelefono: gestisce le bollette telefoniche di un intero anno relative ad una certa utenza
  - Test: utilizza oggetti di tipo BollettaTelefono e UtenzaAnnualeTelefono

Esempio finale (2)

- classe BollettaTelefono
  - attributi
    - mese
    - importo
  - metodi
    - costruttore (con validazione dati)
    - (altri metodi se ritenuti necessari)

Esempio finale (3)

```
1
2  public class BollettaTelefono  {
3
4      private int mese;
5      private double importo;
6
7      public BollettaTelefono ( int m, double i) {
8
9          if ( m > 0 && m < 13 ) // dati validi nell'intervallo 1..12
10             mese = m;
11         else
12             mese = 1;
13
14         if ( i >= 0.0 ) // validazione impo
15             importo = i;
16         else
17             importo = 0.0;
18     }
19
20     public int getMese() {
21         return mese;
22     }
23
24     public double getImporto() {
25         return importo;
26     }
27 }
28
```

Attributi che definiscono il mese e l'importo di una certa bolletta

Costruttore della classe BollettaTelefono. Assegna agli attributi mese e importo i valori dei parametri m e i rispettivamente dopo averli validati.

Metodi get per gli attributi privati della classe

Esempio finale (4)

- classe UtenzaAnnualeTelefono
  - attributi
    - struttura dati per memorizzare le bollette telefoniche ricevute ogni mese
  - metodi
    - costruttore (la struttura dati delle bollette è vuota)
    - `inserisciBolletta()`: permette di inserire una bolletta
    - `spesaAnnuale()`: calcola e ritorna la spesa telefonica complessiva nell'anno
    - `bollettaMax()`: stampa a video il mese e l'importo relativi alla bolletta avente importo massimo

Esempio finale (5)

- Classe Test
  - Il metodo main di questa classe fa quanto segue
    - Crea alcuni oggetti di tipo BollettaTelefono
    - Inserisce gli oggetti creati in un oggetto di tipo UtenzaAnnualeTelefono
    - Stampa l'importo totale delle bollette inserite
    - Calcola e stampa l'importo della bolletta più alta tra quelle inserite

Esempio finale (6)

```
1  public class UtenzaAnnualeTelefono  {
2
3      private BollettaTelefono[] bollette;
4
5      public UtenzaAnnualeTelefono ( ) {
6          bollette = new BollettaTelefono[12];
7      }
8
9      public void inserisciBolletta( BollettaTelefono b) {
10         int posizione = b.getMese() - 1;
11         bollette[posizione] = b;
12     }
13
14     public double spesaAnnuale() {
15         double totale = 0;
16
17         for ( int i=0; i<12; i++)  {
18             if ( bollette[i] != null )
19                 totale = totale + bollette[i].getImporto();
20         }
21
22         return totale;
23     }
24
25     public void bollettaMax() {
26         double maxCorrente = 0;
27         int indiceBollettaMax = 0;
28
29         for ( int i=0; i<12; i++)  {
30
31             if ( bollette[i] != null ) {
32                 double importo = bollette[i].getImporto();
33
34                 if ( importo > maxCorrente ) {
35                     maxCorrente = importo;
36                     indiceBollettaMax = i + 1;
37                 }
38             }
39         }
40
41         System.out.println( "Massimo importo = " + maxCorrente );
42         System.out.println( "Mese del massimo importo = " + indiceBollettaMax  );
43     }
44 }
```

In questo momento ogni componente dell'array bollette è un riferimento a null

Inserimento della bolletta nella posizione specificata dal mese. Es: la bolletta di febbraio è inserita come seconda componente, ovvero in bollette[1]. Attenzione, l'inserimento di più bollette riferite allo stesse mese non provoca errore, le bollette vengono sovrascritte!

Controllo necessario perché potrebbero esserci meno di 12 bollette inserite!

Memorizzo l'importo della bolletta corrente e calcola e memorizza il mese della bolletta corrente

Esempio finale (7)

```
1
2
3  public class Test  {
4
5      public static void main ( String[] args ) {
6
7          UtenzaAnnualeTelefono u = new UtenzaAnnualeTelefono();
8
9          BollettaTelefono a = new BollettaTelefono( 1, 15.25 );
10         BollettaTelefono b = new BollettaTelefono( 2, 20.20 );
11         BollettaTelefono c = new BollettaTelefono( 3, 14.55 );
12
13         u.inserisciBolletta( a );
14         u.inserisciBolletta( c );
15         u.inserisciBolletta( b );
16
17         System.out.println( "Spesa complessiva anno corrente = " + u.spesaAnnuale() );
18
19         u.bollettaMax();
20
21     }
22 }
```

Crea una nuova utenza annuale denominata u

Crea tre oggetti di tipo BollettaTelefono

Inserisce gli oggetti creati nell'utenza annuale u

Calcola e stampa l'importo della bolletta più alta tra quelle inserite

Stampa l'importo totale delle bollette inserite

Esempio finale (8)

Esercizio

- Modificare la classe UtenzaAnnualeTelefono come segue:
  - Aggiungere l'attributo privato posizioneDisponibile che indica dove inserire la prossima bolletta
  - Modificare i metodi della classe in modo da tenere conto della nuova vista privata della classe stessa
- Modificare di conseguenza la classe Test