String, Array e Main

Lezione 7

Scopo della Lezione

- Presentare la classe String ed il tipo Stringa;
- Presentare ed imparare ad usare gli array in Java;
- Approfondire la conoscenza con il metodo speciale main.

String

- Il tipo String contiene una stringa **costante** di caratteri
- Una stringa si denota delimitando i relativi caratteri con dei doppi apici, es. "Walter"
- Usando una variabile String come argomento di `println` o `print`, la corrispondente stringa viene visualizzata.

Sequenze di Escape

- Alcuni particolari elementi di una stringa non possono essere espressi tramite un carattere, es. l'andata a capo;
- Per inserire questi elementi è possibile usare delle descrizioni mnemoniche dette **sequenze di escape**;
- Le sequenze di escape sono costituite dal backslash (\) seguito da un carattere.

Carattere	Significato
`\n`	A capo (newline)
`\t`	Tabulazione
`\"`	Doppio apice
`\\`	Slash

Concatenamento

- L'operazione di concatenamento (+) giustappone una stringa a un'altra stringa,
  - es. "Walter " + "Cazzola" = "Walter Cazzola"
- L'esito dell'operazione consiste nella creazione di una nuova stringa.

- I tipi primitivi sono implicitamente promossi a stringa quando usati in combinazione con l'operatore di concatenazione.
  - Es."La radice quadrata di = " + 5

String è un oggetto

- String, come tutti gli oggetti, incorpora una o più funzioni che operano su di sé (sulla stringa che rappresenta)
- Per eseguire una funzione è necessario scrivere
  - il nome della variabile di tipo String;
  - il carattere di punto (.);
  - il nome della funzione seguito da parentesi tonde, contenenti eventuali argomenti separati da virgole

Lunghezza di una Stringa

- La funzione `length()` ritorna il numero di caratteri contenuti in una stringa
- Questa funzione non necessita di parametri

Esempio


```
import prog.io.*;
class Stringa {
  public static void main (String args[]) {
    ConsoleOutputManager video = new ConsoleOutputManager();
    String messaggio;
    messaggio = "The quick brown fox" + " jumps over the lazy dog";
    video.println(messaggio);
    video.println(messaggio.length());
  }
}
```

```
[18:15]cazzola@ulik:esercizi>java Stringa
The quick brown fox jumps over the lazy dog
43
```

Estrazione di Caratteri

- La funzione `charAt()` ritorna il carattere contenuto in una precisa posizione della stringa;
- È necessario passare il numero di posizione come argomento della funzione;
- La prima posizione ha come numero 0, la seconda 1 e così via.

Esempio


```
// Questo programma verrà compilato ma...
import prog.io.*;
class PosizioneSbagliata {
  public static void main (String args[]) {
    ConsoleOutputManager video = new ConsoleOutputManager();
    String mess;
    mess = "The quick brown fox" + " jumps over the lazy dog";
    video.println(mess.charAt(0));
    video.println(mess.charAt(mess.length()));
  }
}

[18:28]cazzola@ulik:esercizi>java PosizioneSbagliata
T
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String
    index out of range: 43
        at java.lang.String.charAt(String.java:444)
        at PosizioneSbagliata.main(PosizioneSbagliata.java:9)
```

Sottostringhe

- Si dice sottostringa una stringa che è parte di un'altra stringa
- La funzione `substring()` ritorna una sottostringa di una stringa
- È necessario passare come argomenti della funzione:
  - la posizione del primo carattere della sottostringa;
  - la posizione del suo ultimo carattere più uno (cioè la posizione del primo carattere che segue la sottostringa).

Esempio


```
import prog.io.*;
class Sottostringa {
  public static void main (String args[]) {
    ConsoleOutputManager video = new ConsoleOutputManager();
    String mess;
    mess = "The quick brown fox" + " jumps over the lazy dog";
    video.println(mess.substring(4,14));
  }
}
```

```
[18:33]cazzola@ulik:esercizi>java Sottostringa
quick brow
```

Ricerca di Sottostringhe

- La funzione `indexOf()` ritorna la posizione della prima occorrenza di una data sottostringa in una stringa;
- È necessario passare come argomento della funzione la sottostringa che si vuole ricercare;
- Quando la ricerca ha esito negativo, la funzione ritorna -1

Ricerca di Sottostringhe

- Una seconda versione di `indexOf()` effettua la ricerca a partire da una fissata posizione della stringa
- E' necessario passare come argomenti della funzione
  - La sottostringa che si vuole ricercare
  - La posizione a partire da cui effettuare la ricerca
- Se l'esito è negativo la funzione ritorna -1

Esempio


```
import prog.io.*;
class Ricerca {
  public static void main (String args[]) {
    ConsoleOutputManager video = new ConsoleOutputManager();
    String messaggio = "The quick brown fox" + " jumps over the lazy dog";
    video.println(messaggio.indexOf("the"));
    video.println(messaggio.indexOf("he"));
    video.println(messaggio.indexOf("he",5));
    video.println(messaggio.indexOf("Thi"));
  }
}

[18:37]cazzola@ulik:esercizi>java Ricerca
31
1
32
-1
```

Stringhe: Confronto

Le stringhe possono essere confrontate in base al loro ordine lessicografico. Caso speciale: la stringa vuota precede tutte le altre.

```
public boolean equals(Object anObject);       // ridefinisce Object.equals()
public boolean equalsIgnoreCase(String  anotherString)
public int compareTo(String  anotherString) // il segno mi dice l'ordine
```

Due stringhe sono uguali se hanno le stesse lettere nello stesso ordine:

```
String s1 = "hello", s2 = "Hello";
s1.equals(s2)                 // false
s1.equals("hello");           // true
```

Attenzione: s1 == s2 controlla se i due oggetti sono identici e non se contengono la stessa stringa.

Stringhe: Confronto (Continua)

Date le seguenti dichiarazioni:

```
String s1 = new String("hello");
String s2 = new String("hello");
String s3 = new String("Hello");
String s4 = s1;   // s1 ==  s4
String s5 = "hello";
String s6 = "hello";
```

s5 e s6 puntano allo stesso (identico) oggetto costante

Si ottengono i seguenti risultati:

Test di uguaglianza

```
s1.equals(s2) → true
s1.equals(s3) → false
s1.equals(s4) → true
s1.equals(s5) → true
s5.equals(s6) → true
```

Test di identità

```
s1 == s2 → false
s1 == s3 → false
s1 == s4 → true
s1 == s5 → false
s5 == s6 → true
```

StringBuffer

Le stringhe Java sono immutabili. Ogniqualvolta si assegna un nuovo valore ad una stringa, Java DEVE creare un nuovo oggetto di tipo stringa e distruggere quello vecchio.

Perciò nel seguente assegnamento:

- `resultStr = resultStr + ptr + " ";`

Java creerà un nuovo oggetto che verrà puntato da resultStr.

Gli oggetti istanza della classe `StringBuffer` sono stringhe modificabili.

ReverseString

Scrivere il programma ReverseString che inverta una stringa introdotta da tastiera.

- Usare StringBuffer;
- Non usare la funzione reverse.

ReverseString

```
import prog.io.*;   // java.lang.* è incluso automaticamente

class ReverseString {
  public static void main (String args[]) {
    ConsoleOutputManager video = new ConsoleOutputManager();
    ConsoleInputManager tastiera = new ConsoleInputManager();
    char aux;
    StringBuffer msg = new StringBuffer(tastiera.readLine("Stringa da Rovesciare: "));
    for(int i=0;i<=(msg.length()-1)/2; i++) {
      aux = msg.charAt(i);
      msg.setCharAt(i, msg.charAt(msg.length()-1-i));
      msg.setCharAt(msg.length()-1-i, aux);
    }
    video.println("Stringa Rovesciata: "+msg);
  }
}
[18:41]cazzola@ulik:esercizii>java ReverseString
Stringa da Rovesciare: Pippo
Stringa Rovesciata: oppiP
[18:41]cazzola@ulik:esercizi>java ReverseString
Stringa da Rovesciare: Walter
Stringa Rovesciata: retlaW
```

ReverseString


```
import prog.io.*;   // java.lang.* è incluso automaticamente

class ReverseString {
  public static void main (String args[]) {
    ConsoleOutputManager video = new ConsoleOutputManager();
    ConsoleInputManager tastiera = new ConsoleInputManager();
    char aux;
    StringBuffer msg = new StringBuffer(tastiera.readLine("Stringa da Rovesciare: "));
    for(int i=0;i<=(msg.length()-1)/2; i++) {
      aux = msg.charAt(i);  // variabile di appoggio
      msg.setCharAt(i, msg.charAt(msg.length()-1-i));     //scambio carattere i-esimo con (n-i)-esimo
      msg.setCharAt(msg.length()-1-i, aux);
    }
    video.println("Stringa Rovesciata: "+msg);
  }
}
[18:41]cazzola@ulik:esercizii>java ReverseString
Stringa da Rovesciare: Pippo
Stringa Rovesciata: oppiP
[18:41]cazzola@ulik:esercizi>java ReverseString
Stringa da Rovesciare: Walter
Stringa Rovesciata: retlaW
```

Array

Un array è una collezione di locazioni di memoria contigue contenenti dati dello stesso tipo.

- gli elementi degli array sono acceduti tramite la loro posizione nell'array piuttosto che usando il nome.
- gli n elementi di un array a sono riferiti come:

  a[0], a[1], a[2], ...,a[n-1].

Gli array sono (praticamente) trattati come oggetti:

- Vengono istanziati dall'operatore **new**.
- Hanno variabili di istanza (es. length).
- Le variabili di tipo Array sono dei riferimenti.
- Un riferimento ad un array è passato come parametro.

Ma …

- Non c'è nessuna classe. Quindi gli array non fanno parte della gerarchia delle classi di Java come i tipi primitivi.

Array: Terminologia

Un array vuoto contiene zero valori.

La lunghezza dell'array è il numero di elementi che lo compongono.

Ogni componente di un array ha lo stesso tipo.

Gli elementi di un array possono essere di un tipo qualsiasi.

Array: Dichiarazione e Creazione

Nel creare un array dobbiamo specificare sia il tipo dei suoi elementi che la sua lunghezza.

Dichiarazione e creazione di un array:

```
int arr[];              // Dichiara una variabile di tipo array
arr = new int[15];      // Crea l'array stesso
```

Combinando i due passi in uno:

```
int arr[] = new int[15];
```

Il nome dell'array è arr.

l'array può contenere 15 valori interi.

Array: Inizializzazione

Gli elementi di un array hanno un valore di default con cui vengono inizializzati:

- \0 per caratteri (NULL), interi e reali.
- false per i booleani.
- null per gli oggetti.

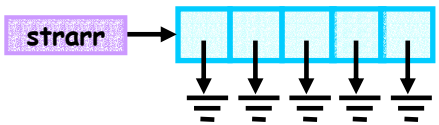
In alternativa, gli array possono essere inizializzati alla creazione:

```
int arr[] = { -2,8,-1,-3,16,20,25,16,16,8,18,19,45,21,-2 } ;
String strings[] = { "ciao", "mondo", "Walter"} ;
```

Array (Continua)
Es. Array di Stringhe

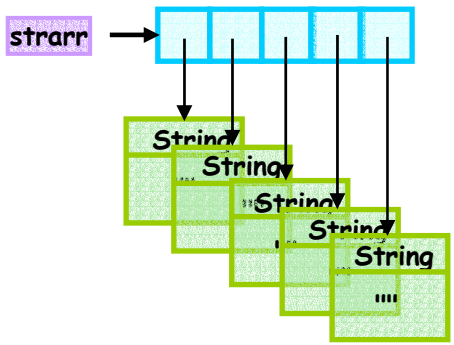

```
(a) String strarr[];  // array che punta a null
```

Dichiara un nome per l'array.




```
(b) strarr = new String[5];
    // crea un array di riferimenti
    // a stringhe nulle
```

Istanzia l'array.




```
(c) for (int k = 0; k < strarr.length; k++)
        strarr[k] = new String();

    // Crea 5 stringhe e le assegna all'array
```

Assegna 5 stringhe all'array.

Array: Uso degli Elementi

Le variabili di tipo array, una volta indiciate, sono usate esattamente come le altre variabili:

```
arr[0] = 5;
arr[2] = 3;
strings[0] = "ciao";
strings[1] = strings[2] = "mondo";
```

Cicli for possono essere usati per scorrere gli array:

```
for (int k = 0; k < arr.length; k++)
        arr[k] = (k+1) * (k+1);
```

length è una variabile di istanza non un metodo.

Array: Ordinamento

Bubble sort: Ad ogni scansione dell'array, l'elemento più grande non ancora ordinato "galleggia" verso "l'alto".

Per ordinare un array composto da N elementi sono necessarie N-1 passate:

```
1. per ognuna delle N-1 scansioni dell'intero array
2.   per ognuna delle N-1 coppie di elementi adiacenti nell'array
3.   se l'elemento di indice minore è più grande dell'elemento con indice maggiore
4.     scambia i due elementi
```

```
 21  20  27  24  19    // array non ordinato
[20  21] 27  24  19    // confronta gli elementi adiacenti
 20  21 [24  27] 19    // li scambia, quando necessario
 20  21  24 [19 |27]   // 1ª passata, 27 galleggia verso l'alto
 20  21  19 |24  27    // 2ª passata, 24 galleggia verso l'alto
 20  19 |21  24  27    // 3ª passata, 21 galleggia verso l'alto
 19 |20  21  24  27    // 4ª passata, l'array è ordinato
```

Array: Ordinamento (Continua)


```
import prog.io.*;
class BoubbleSort {
   public static void main (String args[]) {
     int arr[] = { 21, 20, 27, 24, 19 }, temp;  // inizializza
     ConsoleOutputManager video = new ConsoleOutputManager();
     for (int i=0; i<arr.length; i++) video.print( arr[i] + " ");
     video.println(); // stampa l'array prima che sia ordinato
     for (int pass=1; pass < arr.length; pass++)   // per ogni passo
        for (int pair = 1; pair < arr.length; pair++)    // per ogni coppia
           if (arr[pair-1] > arr[pair]) {                  // li confronta
             temp = arr[pair-1];                           // ed eventualmente scambia
             arr[pair-1] = arr[pair];
             arr[pair] = temp;
           } // if
     for (int i=0; i<arr.length; i++) video.print( arr[i] + " ");
     video.println(); // stampa l'array dopo che l'ordinamento
   }
}

[18:15]cazzola@ulik:esercizi>java BoubbleSort
21  20  27  24  19
19  20  21  24  27
```

Array: Ricerca Sequenziale

Problema: cercare un valore in un array. Se l'array non è ordinato, il valore va cercato sequenzialmente controllando ogni elemento dell'array.

```
import prog.io.*;

class SequentialSearch {
   public static void main (String args[]) {
     int k = 0, key, arr[] = { 21, 20, 27, 24, 19 }, temp;  // inizializza
     ConsoleOutputManager video = new ConsoleOutputManager();
     ConsoleInputManager tastiera = new ConsoleInputManager();
     key = tastiera.readInt("Chiave da cercare nell'array: ");
     while ((k < arr.length) && (arr[k] != key)) ++k;
     if (k < arr.length)
        video.println("La chiave "+key+" è stata trovata nella posizione: "+k);
     else video.println("La chiave "+key+" non è stata trovata");
   }
}
```

LA ricerca fallisce se si arriva alla fine dell'array senza aver trovato il valore.

Array: Ricerca Sequenziale

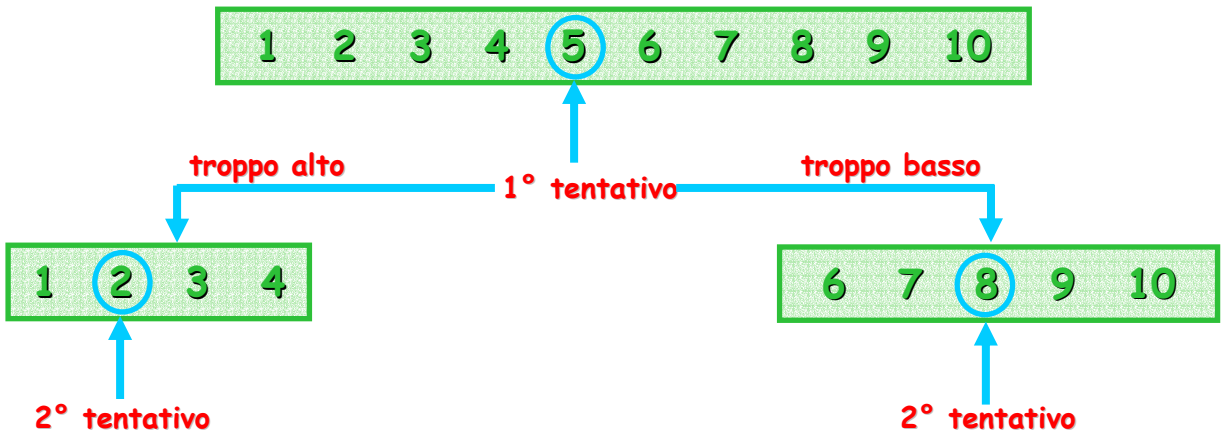
```
[19:05]cazzola@ulik:esercizi>java SequentialSearch
Chiave da cercare nell'array: 19
La chiave 19 è stata trovata nella posizione: 4
[19:05]cazzola@ulik:esercizi>java SequentialSearch
Chiave da cercare nell'array: 25
La chiave 25 non è stata trovata
[19:05]cazzola@ulik:esercizi>java SequentialSearch
Chiave da cercare nell'array: 27
La chiave 27 è stata trovata nella posizione: 2
```

Array: Ricerca Binaria

La ricerca binaria usa una strategia di tipo dividi e conquista su un array **ordinato**.

Ad ogni iterazione si divide l'array a metà, restringendo la ricerca alla metà che potrebbe contenere il valore cercato.

Esempio: tentare di indovinare un numero tra 1 e 10.



Array: Ricerca Binaria (Continua)


```
class BinarySearch {
   public static void main (String args[]) {
     ConsoleOutputManager video = new ConsoleOutputManager();
     ConsoleInputManager tastiera = new ConsoleInputManager();
     int key, arr[] = { 19, 21, 23, 25, 32, 33 };
     int low = 0;   // specifica la porzione di array da usare
     int high = arr.length - 1;
     key = tastiera.readInt("Chiave da cercare nell'array: ");
     while (low <= high) {              // finché non siamo in fondo
       int mid = (low + high) / 2;      // spezza l'array
       if (arr[mid] == key) {
          video.println("La chiave "+key+" è stata trovata nella posizione: "+mid);
          break;
       } else if (arr[mid] < key)
                 low = mid + 1;              // cerca nella parte alta
              else high = mid - 1;           // cerca nella parte bassa
     }
     if (low > high) video.println("La chiave "+key+" non è stata trovata");
   }  // se low diventa più grande di high allora il valore non è nell'array.
}
```

Array: Matrici

Gli array multi-dimensionali o matrici sono array i cui componenti sono a loro volta array e così via.

Esempio di array bi-dimensionale: il calendario.

```
double calendar[][] = new double[12][31];
```

indice dei mesi

indice dei giorni

calendar[7] è agosto.

Il primo array rappresenta i 12 mesi dell'anno. Ogni mese è rappresentato da un array di 31 giorni.

```
char c[][] = { {'a', 'b'}, {'c', 'd'} } ;
double d[][][] = { {1.0, 2.0, 3.0}, {4.0, 5.0}, {6.0, 7.0, 8.0, 9.0} } ;
```

array 2 x 2 di char.

array 3 x 2 x 4 di double.

Ogni dimensione di un array multidimensionale può avere una lunghezza diversa.

Battaglia Navale

Torniamo bambini e implementiamo la classe BattagliaNavale per il gioco suddetto.

Caratteristiche:

- campo di battaglia 5x5 predefinito;
- controllo che le righe e le colonne introdotte da tastiera siano ammissibili;
- finché manco le navi continuo a tentare

Battaglia Navale


```
import prog.io.*;

class BattagliaNavale {
   public static void main (String args[]) {
     ConsoleOutputManager video = new ConsoleOutputManager();
     ConsoleInputManager tastiera = new ConsoleInputManager();
     char campo[][] = new char[5][5]; // campo automaticamente inizializzato a '\0'
     int x, y;

       // introduco le navi sul campo di battaglia
     campo[0][1] = 'X'; campo[1][2] = 'X'; campo[3][3] = 'X'; campo[4][0] = 'X';
     campo[1][1] = 'X'; campo[2][4] = 'X'; campo[0][3] = 'X'; campo[4][1] = 'X';

       // mostro il campo
     for (int i=0;i<5;i++) {
       video.print("[ ");
       for (int j=0;j<5;j++) video.print(((campo[i][j]=='\0')?'·':campo[i][j])+" ");
       video.println("]");
     }
```

Battaglia Navale


```
    do {
        do {  // gestione introduzione coordinate corrette
          x = tastiera.readInt("Introduci Riga (da 0 a 4): ");
        } while ((x<0)||(x>4));
        do {
          y = tastiera.readInt("Introduci Colonna Y (da 0 a 4): ");
        } while ((y<0)||(y>4));
        if (campo[x][y]!='X') video.println("Acqua!!! Riprova!!!");
      } while (campo[x][y]!='X'); // gioco finché non affondo qualcosa
      video.println("Bravo!!! alle coordinate ("+x+","+y+") hai colpito una nave.");
    }
}
```

Battaglia Navale (Si Gioca!)


```
[13:09]cazzola@ulik:esercizi>java BattagliaNavale
[ · X · X · ]
[ · X X · · ]
[ · · · · X ]
[ · · · X · ]
[ X X · · · ]
Leggi Riga (da 0 a 4): 0
Leggi Colonna (da 0 a 4): 5
Leggi Colonna (da 0 a 4): 0
Acqua!!! Riprova!!!
Leggi Riga (da 0 a 4): 2
Leggi Colonna (da 0 a 4): 0
Acqua!!! Riprova!!!
Leggi Riga (da 0 a 4): 0
Leggi Colonna (da 0 a 4): 1
Bravo!!! alle coordinate (0,1) hai colpito una nave.
```

Main: Questo Sconosciuto

Ogni programma deve specificare un punto da cui inizierà la propria esecuzione.

La definizione del metodo main è uno dei modi che offre Java per informare la virtual machine del punto di inizio del programma.

Main: Questo Sconosciuto

```
public static void main (String args[]) {
                ...
}
```

Esaminiamo la sua definizione:

- la keyword **void** esprime il fatto che il main non ritorna alcun valore;
- il metodo main viene attivato prima di tutti gli altri compresi i metodi per la creazione di oggetti, quindi non puo` e non deve far riferimento ad istanze per poter essere attivato; deve pertanto essere un metodo statico;
- non deve avere limiti di accesso (metodo pubblico);
- è l'interfaccia del programma con l'esterno da cui prende degli argomenti (args).

Main: Questo Sconosciuto


```
import prog.io.*;

class ArgsFromCommandLine {
  public static void main (String args[]) {
    ConsoleOutputManager video = new ConsoleOutputManager();
    video.println("Il programma è stato eseguito con: "+args.length+" argomenti.");
    if (args.length>0)
      for (int i=0; i<args.length; i++) video.println("   · "+args[i]);
  }
}
```

```
[18:10]cazzola@ulik:esercizi>java ArgsFromCommandLine
Il programma è stato eseguito con: 0 argomenti.
[18:10]cazzola@ulik:esercizi>java ArgsFromCommandLine Qui Quo Qua 7 -0.7 e Paperino
Il programma è stato eseguito con: 7 argomenti.
Gli argomenti erano:
   · Qui
   · Quo
   · Qua
   · 7
   · -0.7
   · e
   · Paperino
```