

Linguaggi di Programmazione

a.a. 14/15

docente: Gabriele Fici

gabriele.fici@unipa.it

6 - *Array* e *ArrayList*

6 - Array e ArrayList

- Un array è una sequenza di variabili (quindi tipi fondamentali o riferimenti a oggetti) dello stesso tipo indicizzata da un intero
- La dichiarazione si fa indicando il tipo seguito da parentesi quadre
- Un array è un oggetto, quindi la creazione si fa con `new`

Esempio:

```
int[] numeri; // dichiara un array di interi
Serbatoio[] serbatoi; // dichiara un array di
riferimenti a oggetti della classe Serbatoio

numeri = new int[10];
// crea un array di 10 interi
serbatoi = new Serbatoio[5];
// crea un array di 5 riferimenti a Serbatoio
```

6 - Array e ArrayList

- La lunghezza di un array è dichiarata in fase di costruzione e non può cambiare
- L'inizializzazione degli elementi è automatica in fase di costruzione (0 per i tipi numerici, `false` per i booleani, `null` per i riferimenti agli oggetti)
- Attenzione agli array dichiarati ma non inizializzati:

```
int[] numeri;  
numeri[0] = 5;           // errore!
```

6 - Array e ArrayList

- Un array si può inizializzare anche durante la costruzione

```
int[] numeri = new int[3];  
numeri[0] = numeri[1] = 3;  
numeri[2] = 4;  
  
//si può scrivere:  
int[] numeri = new int[]{3, 3, 4};  
  
//oppure ancora più semplicemente:  
int[] numeri = {3, 3, 4};
```

6 - Array e ArrayList

- In un array di n elementi il primo elemento ha indice 0 , e l'ultimo ha indice $n-1$
- Ogni array ha un attributo `length`, in cui è conservata la sua lunghezza

Esempio:

```
int[] numeri = new int[10];    // lunghezza 10
numeri[0] == 0;                  // vale true
numeri[numeri.length] = 5;       // errore!
numeri[numeri.length-1] = 5;     // ok
```

6 - Array e ArrayList

- La scansione degli oggetti di un array si può anche fare con un ciclo for esteso (*enhanced for*, chiamato anche *foreach*)

Esempio:

```
for (ContoBancario conto : conti2014) {  
    conto.deposita(1000.00);  
}
```

// equivale a:

```
for (int i=0; i<conti2014.length; i++) {  
    ContoBancario conto = conti2014[i];  
    conto.deposita(1000.00);  
}
```

6 - Array e ArrayList

- Attenzione: col `for` esteso non si possono assegnare valori agli elementi di un array, perché si modificherebbe solo la variabile locale usata per la scansione

Esempio:

```
for (ContoBancario conto : conti2014) {  
    conto = conti2014[0];  
    // assegna conti2014[0] a conto, non a conti2014[i]!  
}
```

```
for (int num : numeri) {  
    num = 5;  
    // non ha alcun effetto su numeri  
}
```


6 - Array e ArrayList

- La classe `java.util.Arrays` fornisce dei metodi (statici) per lavorare con gli array
- In particolare vedremo i metodi:
 - `equals`
 - `fill`
 - `sort`
 - `binarySearch`

6 - Array e ArrayList

- Il metodo `equals` confronta due array dello stesso tipo e restituisce un boolean

```
int[] p = {1,2,3};  
int[] q = {1,2,3,4};  
System.out.println(Arrays.equals(p,q));  
// stampa false
```

6 - Array e ArrayList

- Il metodo `fill` assegna agli elementi di un array (passato come argomento) uno stesso valore (anch'esso passato come argomento)

```
String[] p = new String[10];  
Arrays.fill(p, "ciao");
```

- E' possibile passare in argomento anche un range di indici

```
String[] p = new String[10];  
Arrays.fill(p, 3, 5, "ciao");
```

6 - Array e ArrayList

- Il metodo `sort` ordina un array
- Se si tratta di un array di stringhe l'ordine è quello lessicografico (usando l'ordine dei caratteri di Unicode)

```
int[] p = {4,2,3};  
Arrays.sort(p);  
for (int i=0 ; i<p.length ; i++)  
    System.out.print(p[i]);  
// stampa 2, 3, 4
```

6 - Array e ArrayList

- Per cercare se un elemento è presente in un array si può usare il metodo `binarySearch` che ha due parametri: l'array in cui cercare e l'elemento da cercare
- Restituisce la posizione dell'elemento cercato, oppure un valore negativo se l'elemento non è presente
- Attenzione: l'array deve essere ordinato prima della ricerca, altrimenti il comportamento è indeterminato
- In caso ci sia più di un'occorrenza dell'elemento cercato non c'è garanzia su quale occorrenza verrà restituita

```
int[] num = {2, 2, 5, 12, 12, 33};
```

```
System.out.println(Arrays.binarySearch(num, 12));  
// stampa 3
```

6 - Array e ArrayList

- In Java esiste una classe `ArrayList` (dentro il package `java.util`) che permette di gestire in modo più pratico ed efficiente gli array
- La lunghezza degli ArrayList può variare dinamicamente, e quindi non va specificata in fase di creazione
- La classe fornisce metodi per le operazioni più comuni (inserimento, cancellazione, etc.)
- Un ArrayList è anche chiamato *array dinamico* oppure *vettore dinamico*
- Attenzione: un ArrayList non può contenere tipi fondamentali, ma solo solo riferimenti a oggetti

6 - Array e ArrayList

- In realtà `ArrayList` è una classe parametrica: cioè `ArrayList<T>` è un array dinamico di oggetti di tipo T
- Ad esempio, `ArrayList<String>` è un array dinamico di oggetti di tipo Stringa, mentre `ArrayList<ContoBancario>` è un array dinamico di oggetti di tipo ContoBancario

Esempio:

```
ArrayList<String> nomi = new ArrayList<String>();  
// dichiara e costruisce un ArrayList di Stringhe
```

6 - Array e ArrayList

- Per aggiungere un elemento (in coda) si usa il metodo `add` con parametro l'elemento da aggiungere

```
ArrayList<String> nomi = new ArrayList<String>();  
nomi.add("Pippo"); //ora nomi ha dimensione 1  
nomi.add("Pluto"); //ora nomi ha dimensione 2
```

- Per avere la dimensione si usa il metodo `size`

```
System.out.println(nomi.size()); //stampa 2
```


6 - Array e ArrayList

- Per ottenere l'elemento in posizione `i` si usa il metodo `get` con parametro `i`

```
String s = nomi.get(1);    // s vale "Pluto"  
String t = nomi.get(2);    // errore, nomi ha size 2!
```

- Per assegnare un nuovo valore all'elemento in posizione `i` si usa il metodo `set` con parametro `i`

```
nomi.set(1, "Paperino");  
// "Pluto" è stato sovrascritto da "Paperino"
```

6 - Array e ArrayList

- Per aggiungere un nuovo elemento in posizione `i` si usa il metodo `add` con parametri `i` e l'elemento da aggiungere; gli altri elementi verranno spostati in avanti di una posizione

```
nomi.add(0, "Topolino");
```

- Per rimuovere l'elemento in posizione `i` si usa il metodo `remove` con parametro `i`; gli altri elementi verranno spostati all'indietro di una posizione

```
nomi.remove(1);
```

6 - Array e ArrayList

Esempio:

```
ArrayList<String> nomi = new ArrayList<String>();  
// ora nomi vale []  
nomi.add("Pippo");  
// ora nomi vale [Pippo]  
nomi.add("Pluto");  
// ora nomi vale [Pippo, Pluto]  
nomi.set(1, "Paperino");  
// ora nomi vale [Pippo, Paperino]  
nomi.add(0, "Topolino");  
// ora nomi vale [Topolino, Pippo, Paperino]  
nomi.remove(1);  
// ora nomi vale [Topolino, Paperino]
```

6 - Array e ArrayList

- Per avere la prima (risp. ultima) posizione di un elemento esistono i metodi `indexOf` e `lastIndexOf`

Esempio:

```
ArrayList<String> nomi = new ArrayList<String>();  
  
nomi.add("Pippo");  
nomi.add("Pluto");  
nomi.add("Pluto");  
  
System.out.println(nomi.indexOf("Pluto"));  
// stampa 1  
System.out.println(nomi.lastIndexOf("Pluto"));  
// stampa 2
```

6 - Array e ArrayList

- Il metodo `contains` permette di sapere se un oggetto è presente nell'ArrayList, e restituisce un boolean

Esempio:

```
ArrayList<String> nomi = new ArrayList<String>();  
  
nomi.add("Pippo");  
nomi.add("Pluto");  
nomi.add("Pluto");  
  
System.out.println(nomi.contains("Pluto"));  
// stampa true
```

6 - Array e ArrayList

- Per copiare un ArrayList su un array (passato a parametro) esiste il metodo `toArray`

Esempio:

```
ArrayList<ContoBancario> contiList =  
    new ArrayList<ContoBancario>();  
  
for (int i=0; i<5; i++)  
    contiList.add(new ContoBancario());  
  
ContoBancario[] contiArray =  
    new ContoBancario[contiList.size()];  
  
contiList.toArray(contiArray);  
  
for (ContoBancario conto : contiList)  
    System.out.println(conto.toString());  
  
for (ContoBancario conto : contiArray)  
    System.out.println(conto.toString());
```

6 - Array e ArrayList

- Gli ArrayList non possono contenere tipi primitivi, ma solo oggetti
- Tuttavia in Java esiste la possibilità di trasformare tipi primitivi in oggetti, tramite le classi wrapper (involucri)
- I wrapper sono: Byte, Short, Integer, Long, Float, Double, Character, Boolean
- Nota `int -> Integer` e `char -> Character`

6 - Array e ArrayList

- Le conversioni da tipo primitivo a oggetto e viceversa avvengono in modo automatico (si chiamano rispettivamente auto-boxing e auto-unboxing)

Esempio:

```
Double d = 3.14; // auto-boxing
               // equivale a Double d = new Double(3.14);

double x = d;    // auto-unboxing
               // equivale a double x = d.doubleValue();
```


6 - Array e ArrayList

- Le conversioni avvengono anche all'interno di espressioni aritmetiche

Esempio:

```
Double d = 3.14;  
Double e = d/2;
```

Valutazione: `d` viene auto-unboxed a un `double`, poi viene diviso per `2`, poi viene boxed a `Double` e il riferimento viene memorizzato nel wrapper `e`

6 - Array e ArrayList

- Attenzione: i wrapper sono oggetti, non tipi fondamentali, per cui non si possono usare le espressioni per questi ultimi
- Per confrontarli usare il metodo `equals`

Esempio:

```
Double d = 3.14, e = 3.14;

System.out.print(d==e);
/* Stampa "false" perché d ed e
   sono riferimenti a oggetti diversi */

System.out.println(d.equals(e));
// stampa "true"
```

6 - Array e ArrayList

Esempio:

```
ArrayList<Double> num = new ArrayList<Double>();  
// ArrayList<double> darebbe errore!  
num.add(3.14);  
// ora num vale [3.14]  
num.add(0, 8.00);  
// ora num vale [8.00, 3.14]  
num.set(1, num.get(1) + 1);  
// ora num vale [8.00, 4.14]  
num.set(num.size()-1, num.get(0)/2);  
// ora num vale [8.00, 4.00]
```

6 - Array e ArrayList

- La classe `java.util.Collections` fornisce dei metodi (statici) per lavorare con gli ArrayList
- In particolare vedremo i metodi:
 - `reverse`
 - `sort`
 - `max` e `min`
 - `shuffle`
 - `swap`
 - `binarySearch`

6 - Array e ArrayList

- Il metodo `reverse` inverte l'ordine degli elementi di un `ArrayList` (non ha valore di ritorno)

Esempio:

```
ArrayList<Integer> al = new ArrayList<Integer>();  
  
al.add(2);  
al.add(3);  
al.add(-2);  
  
Collections.reverse(al);  
System.out.println(al);  
// stampa [-2, 3, 2]
```

6 - Array e ArrayList

- Il metodo `sort` ordina gli elementi di un ArrayList (non ha valore di ritorno)
- Per le stringhe l'ordine è quello lessicografico

Esempio:

```
ArrayList<Integer> al = new ArrayList<Integer>();  
  
al.add(2);  
al.add(3);  
al.add(-2);  
  
Collections.sort(al);  
System.out.println(al);  
// stampa [-2, 2, 3]
```

6 - Array e ArrayList

- I metodi `max` e `min` restituiscono l'elemento più grande e l'elemento più piccolo, rispettivamente, di un `ArrayList`
- Per le stringhe l'ordine è quello lessicografico

Esempio:

```
ArrayList<Integer> al = new ArrayList<Integer>();

al.add(2);
al.add(3);
al.add(-2);

System.out.println("max = " + Collections.max(al));
// stampa 3
System.out.println("min = " + Collections.min(al));
// stampa -2
```

6 - Array e ArrayList

- Il metodo `shuffle` mischia casualmente gli elementi di un `ArrayList` (non ha valore di ritorno)

Esempio:

```
ArrayList<Integer> al = new ArrayList<Integer>();  
  
al.add(2);  
al.add(3);  
al.add(-2);  
  
Collections.shuffle(al);  
System.out.println(al);
```


6 - Array e ArrayList

- Il metodo `swap` scambia gli elementi di un `ArrayList` presenti nelle posizioni passate come parametri.

Esempio:

```
ArrayList<Integer> al = new ArrayList<Integer>();  
  
al.add(2);  
al.add(3);  
al.add(-2);  
  
Collections.swap(al, 0, al.size()-1);  
System.out.println(al);  
// stampa [-2, 3, 2]
```

6 - Array e ArrayList

- Il metodo `binarySearch` cerca un elemento in un `ArrayList` ordinato (più velocemente di `contains`)
- Restituisce la posizione dell'elemento trovato, oppure un valore negativo se l'elemento non è presente

Esempio:

```
ArrayList<Integer> al = new ArrayList<Integer>();  
  
al.add(2);  
al.add(3);  
al.add(-2);  
  
Collections.sort(al);  
System.out.println(Collections.binarySearch(al, -2));  
// stampa 0
```