Metodologie di Programmazione (M-Z)

II semestre - a.a. 2021 – 2022

Parte 3 – Istruzioni di Controllo e Array**

a cura di Stefano Faralli*

SAPIENZA
UNIVERSITÀ DI ROMA

**Le presenti slide sono una rielaborazione delle slide create dal prof. Roberto Navigli negli anni accademici passati per lo stesso corso.

Istruzioni di controllo

Istruzioni di controllo

```
public static void main(String[] args)
{
      int x = 5;
      int y = 7;

      x = y+1;
      y = x+1;
}
```

```
public static void main(String[] args)
{
      int x = 5;
      int y = 7;

      y = x+1;
      x = y+1;
}
```

- Finora abbiamo controllato il flusso di esecuzione del programma solo mediante la sequenza
- La sequenza è la struttura di controllo fondamentale che stabilisce l'ordine di esecuzione delle istruzioni

Prendere decisioni

- Come prendere decisioni in Java?
- Mediante le istruzioni di controllo **condizionali**
  - Alcune istruzioni possono essere o non essere eseguite sulla base di certe condizioni
- Mediante le istruzioni di controllo **iterative**
  - Permettono di specificare che un blocco di istruzioni deve essere eseguito ripetutamente sulla base di certe condizioni

L'istruzione if

- Per realizzare una decisione si usa l'istruzione if
- La sintassi è:

```
if (<espressione booleana>) <singola istruzione>
```

oppure:

```
if (<espressione booleana>)
{
      <istruzioni>
}
```

Eseguite se l'espressione booleana è vera

- Esempi:

```
if (x < 0) x = -x;

if (x < y)
{
      int t = x;
      x = y;
      y = t;
}
```

Codificare l'alternativa: l'istruzione else

- Per specificare l'alternativa da eseguire nel caso in cui il valore dell'espressione booleana sia falso
- Sintassi:

```
if (<espressione booleana>)
{
      <istruzioni caso true>
}
else
{
      <istruzioni caso false>
}
```

Eseguite se l'espressione booleana è vera

Eseguite se l'espressione booleana è falsa

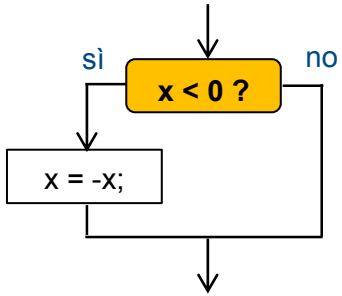
- Esempio:

```
if (x > y)  max = x;
else        max = y;
```

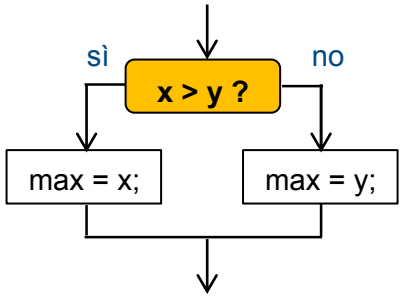
Enunciati if ed else mediante diagrammi di flusso

- Per capire il flusso del programma controllato mediante le istruzioni if ed else, possiamo utilizzare i diagrammi di flusso:

```
if (x < 0) x = -x;
```




```
if (x > y)   max = x;
else         max = y;
```



Esempi

valore assoluto

```
if (x < 0) x = -x;
```

ordina x e y

```
if (x < y)
{
      int t = x;
      x = y;
      y = t;
}
```

massimo di x e y

```
if (x > y)  max = x;
else        max = y;
```

controllo di divisione per zero

```
if (den == 0)     System.out.println("Division by zero");
else              System.out.println("Quoziente = " + num/den);
```

calcolo delle radici di un'equazione di secondo grado $x^2+bx+c = 0$

```
double discriminant = b*b - 4.0*c;
if (discriminant < 0.0) System.out.println("Non ci sono radici reali");
else
{
      System.out.println((-b + Math.sqrt(discriminant))/2.0);
      System.out.println((-b - Math.sqrt(discriminant))/2.0);
}
```

L'else "sospeso"

```
if (x > 0)
    if (y > 0)
        System.out.println("x e y sono > 0");
else
    System.out.println("x <= 0");
```

Errore: non si riferisce a if (x > 0) ma a if (y > 0)!!!

```
if (x > 0)
    if (y > 0)
        if (z < 5)
            if (w >= 3)
                System.out.println("x e y sono > 0, z < 5 e w >= 3");
            else
                System.out.println("x e y sono > 0, z < 5 e w < 3");
```

Corretto!

E' importante essere consapevoli che l'**else** si riferisce **SEMPRE** all'istruzione **if** immediatamente precedente

Forzare il riferimento dell'else all'istruzione if richiesta

- Utilizzando le parantesi graffe per specificare il corpo dell'if precedente

```
if (x > 0)
{
    if (y > 0)
        System.out.println("x e y sono > 0");
}
else
    System.out.println("x <= 0");
```

Tirare un moneta (non truccata) in due righe!

```
public class TiraMoneta
{
      public static void main(String[] args)
      {
            if (Math.random() < 0.5)      System.out.println("Testa");
            else                          System.out.println("Croce");
      }
}
```

- Esempio di esecuzione:
  - java TiraMoneta
    Testa
  - java TiraMoneta
    Croce
  - java TiraMoneta
    Croce
  - …

Operatore di selezione (o operatore condizionale)

- In Java (come in C) esiste un operatore di selezione (operatore condizionale) nella forma di espressione

  condizione ? valoreCasoVero : valoreCasoFalso

- Esempi:

```
int abs = x < 0 ? -x : x;

int max = x > y ? x : y;

String testaCroce = Math.random() < 0.5 ? "Testa" : "Croce";
```

Esercizio

- Progettare un metodo che estragga il carattere centrale da una stringa fornita in input
- Se la stringa ha un numero pari di caratteri, estrarre i due caratteri centrali (es. da “magico” deve estrarre “gi”)

• Fase 1: progettare l’interfaccia pubblica

```
public String estraiCarattereCentrale(String s)
{

}
```

• Fase 2: individuare la condizione per la diramazione:
  - La lunghezza della stringa è dispari?
  - s.length() % 2 == 1?

Esercizio

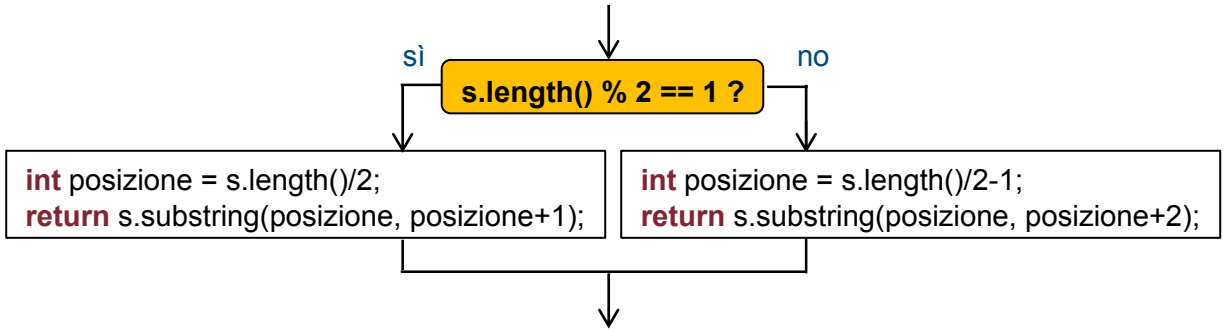
- Fase 3: progettare il codice (o lo pseudocodice) per le istruzioni da eseguire quando la condizione è soddisfatta

```
int posizione = s.length()/2;
return s.substring(posizione, posizione+1);
```

- Fase 4: progettare il codice (o lo pseudocodice) per le istruzioni
- da eseguire quando la condizione **NON** è soddisfatta

```
int posizione = s.length()/2-1;
return s.substring(posizione, posizione+2);
```

- Progetta il flusso di controllo:



Esercizio

- Una prima versione:

```
public String estraiCarattereCentrale(String s)
{
    if (s.length() % 2 == 1)
    {
        int posizione = s.length()/2;
        return s.substring(posizione, posizione+1);
    }
    else
    {
        int posizione = s.length()/2-1;
        return s.substring(posizione, posizione+2);
    }
}
```

Esercizio

- Una prima versione:

```
public String estraiCarattereCentrale(String s)
{
    if (s.length() % 2 == 1)
    {
        int posizione = s.length()/2;
        return s.substring(posizione, posizione+1);
    }
    else
    {
        int posizione = s.length()/2-1;
        return s.substring(posizione, posizione+2);
    }
}
```

- Fase 4: elimina le ripetizioni

```
public String estraiCarattereCentrale(String s)
{
    int posizione;
    int lunghezza;

    if (s.length() % 2 == 1)
    {
        posizione = s.length()/2;
        lunghezza = 1;
    }
    else
    {
        posizione = s.length()/2-1;
        lunghezza = 2;
    }

    return s.substring(posizione, posizione+lunghezza);
}
```

Alternative multiple

- In molte situazioni è necessario prendere più di una decisione di tipo if/else
- E' richiesta una sequenza di confronti

Esempio: descrizione del terremoto

- Progettare una classe che restituisca una descrizione del terremoto dato il valore di magnitudo della scossa su scala Richter

```
public class Terremoto
{
    /**
     * Valore sulla scala Richter
     */
    private double magnitudo;

    public Terremoto(double magnitudo)
    {
        this.magnitudo = magnitudo;
    }

    public String toString()
    {
        if (magnitudo >= 8.0) return "La maggior parte delle strutture saranno distrutte";
        else if (magnitudo >= 7.0) return "Molti edifici saranno distrutti";
        else if (magnitudo >= 6.0) return "Molti edifici saranno danneggiati, alcuni distrutti";
        else if (magnitudo >= 4.5) return "Danni solo a edifici con struttura debole";
        else if (magnitudo >= 3.5) return "Terremoto percettibile, ma nessuna distruzione";
        else if (magnitudo >= 0) return "Non percettibile";
        else return "Valore negativo non ammesso";
    }
}
```

Alternative multiple: l'istruzione switch

- Per confrontare il valore di un'espressione intera o convertibile a intero (o, da Java 7 in poi, un valore stringa), si può usare l'istruzione switch:

```
switch(<espressione intera>)
{
      case <valore1>: <istruzioni>; break;
      case <valore2>: <istruzioni>; break;
      ...
      case <valoren>: <istruzioni>; break;
}
```

Esercizio: Saluto casuale

- Progettare un metodo che emetta sullo standard output un saluto scelto casualmente tra “ciao”, “hello”, “bella”, “salve” e “buongiorno” (rendere quest’ultimo doppiamente più probabile)

```
import java.util.Random;

public class SalutoCasuale
{
    public void sayHello()
    {
        String hello = null;

        // new Random().nextInt(6 equivalente a (int)(Math.random()*6)
        switch(new Random().nextInt(6))
        {
            case 0:     hello = "ciao"; break;          // informale
            case 1:     hello = "hello"; break;         // inglese
            case 2:     hello = "bella"; break;         // romanesco
            case 3:     hello = "salve"; break;         // salute a te
            default:    hello = "buongiorno"; break;    // formale
        }

        System.out.println(hello);
    }

    public static void main(String[] args)
    {
        new SalutoCasuale().sayHello();
    }
}
```

caso di default

Switch compatti (java >=13)

- E' possibili utilizzare anche una notazione contratta con l'operatore -> che non richiede l'utilizzo del break per uscire:

```
switch(k % 7)
{
    case 0:
        iniziaLaSettimana();
        break;
    case 1:
        faiLaSpesa();
        break;
    /* ... */
    case 5:
        relax();
        break;
    case 6:
        gitaDellaDomenica();
        break;
}
```

java >=13

```
switch(k % 7)
{
    case 0 -> iniziaLaSettimana();
    case 1 -> faiLaSpesa();
    /* ... */
    case 5 -> relax();
    case 6 -> gitaDellaDomenica();
}
```

Espressioni switch (java >=13)

- E' possibile utilizzare lo switch con la notazione -> facendo restituire un'espressione:

```
String s;
switch(c)
{
    case 'k': s = "kappa"; break;
    case 'h': s = "acca"; break;
    case 'l': s = "elle"; break;
    case 'c': s = "ci"; break;
    case 'a': case 'e': case 'i':
    case 'u': case 'o':
        s = "vocale "+c; break;
    default: s = "non so leggerlo";
}
```

java >=13

```
String s = switch(c) {
    case 'k' -> "kappa";
    case 'h' -> "acca";
    case 'l' -> "elle";
    case 'c' -> "ci";
    case 'a', 'e', 'i',
         'u', 'o' -> "vocale "+c;
    default -> "non so leggerlo";
};
```

Espressioni switch (java >=13)

- La parola chiave yield permette di utilizzare lo switch classico come espressione e restituire il valore di interesse (interrompendo l'esecuzione dello switch):

```
String s;
switch(c)
{
    case 'k': s = "kappa"; break;
    case 'h': s = "acca"; break;
    case 'l': s = "elle"; break;
    case 'c': s = "ci"; break;
    case 'a': case 'e': case 'i':
    case 'u': case 'o':
        s = "vocale "+c; break;
    default: s = "non so leggerlo";
}
```

java >=13

```
String s = switch(c) {
    case 'k': yield "kappa";
    case 'h': yield "acca";
    case 'l': yield "elle";
    case 'c': yield "ci";
    case 'a': case 'e': case 'i':
    case 'u': case 'o':
        System.out.println("vocale!");
        yield "vocale "+c;
    default: yield "non so leggerlo";
};
```

La parola chiave var (java >=10)

- A partire da Java 10, è possibile **evitare di dichiarare il tipo** di una variabile locale

```
var k = 10;           // equivale a int k = 10;
var s = "ciao";       // equivale a String s = "ciao";
var p = new Pippo(); // equivale a Pippo p = new Pippo();
```

- Il tipo della variabile è sempre prefissato e **non può cambiare**

- E' possibile utilizzare var solo per le variabili locali
  - Non per i campi, né per i parametri di metodi

- **Consiglio:** la gestione statica dei tipi di Java è un punto di forza -> **utilizzate var solo per le variabili di comodo**

Istruzioni iterative

Istruzioni iterative

- Molti calcoli sono inerentemente ripetitivi
- La ripetizione (iterazione) è implementata in Java mediante le seguenti istruzioni:
  - while
  - do … while
  - for

L’istruzione while

- La sintassi dell’istruzione while è simile a quella dell’if:

```
while(<espressione booleana>)
{
      <istruzioni>
}
```

- La differenza è che le istruzioni nel corpo sono eseguite finché l’espressione booleana è vera
  - L’espressione booleana viene controllata all’inizio di ogni esecuzione del corpo
- Appena l’espressione booleana è falsa (eventualmente anche subito), il ciclo termina

Esempio: calcolare le potenze di 2 fino a 2^N

```
public class PotenzeDi2
{
    public static void main(String[] args)
    {
        // le prime potenze di 2 fino a 2^N
        final int N = Integer.parseInt(args[0]);

        int val = 1;
        int i = 0;

        while(i <= N)
        {
            System.out.println(i+" "+val);
            val *= 2;    // v = v*2;
            i++;         // i = i+1;
        }
    }
}
```

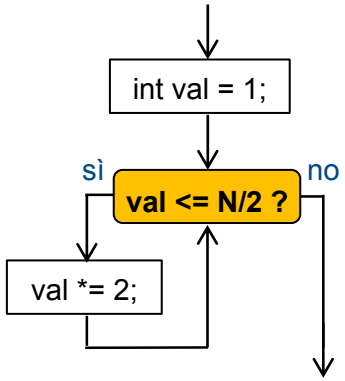
Nota che l'intero N è definito costante (final)

Perché se N = 31 otteniamo:
`31 -2147483648` ??

- Esecuzione: `java PotenzeDi2 4`
  - `0 1`
  - `1 2`
  - `2 4`
  - `3 8`
  - `4 16`

Esercizio: la più grande potenza di 2 <= N

- Scrivere le istruzioni per calcolare la più grande potenza di 2 che sia <= un intero positivo N




```
// piu' grande potenza di 2 <= N
int val = 1;
while (val <= N/2) val *= 2;
```

Il costrutto do … while

- Si comporta esattamente come il while ma la condizione di uscita viene verificata alla fine dell'esecuzione del corpo del ciclo (**invece** che all'inizio):

```java
public class PotenzeDi2DoWhile
{
    public static void main(String[] args)
    {
        // le prime N potenze di 2
        final int N = Integer.parseInt(args[0]);

        int val = 1;
        int i = 0;

        do
        {
            System.out.println(i+" "+val);
            val *= 2;    // v = v*2;
            i++;         // i = i+1;
        } while(i <= N);
    }
}
```

```java
public class PotenzeDi2
{
    public static void main(String[] args)
    {
        // le prime potenze di 2 fino a 2^N
        final int N = Integer.parseInt(args[0]);

        int val = 1;
        int i = 0;

        while(i <= N)
        {
            System.out.println(i+" "+val);
            val *= 2;    // v = v*2;
            i++;         // i = i+1;
        }
    }
}
```

L’istruzione for

- E’ un costrutto alternativo al while che fornisce più flessibilità nella realizzazione di cicli
- La sintassi:

```
for (<inizializzazione>; <espressione booleana>; <incremento>)
{
      <istruzioni>
}

for (<inizializzazione>; <espressione booleana>; <incremento>)
      <istruzione>;
```

- Lo schema è il seguente:
  - Inizializza la variabile “di controllo”
  - Esegui il test d’uscita sull’espressione booleana
  - Esegui il corpo del for
  - Alla fine di ogni ciclo incrementa/decrementa il valore della variabile di controllo come specificato

Equivalenza dell'istruzione for con il while

```
for (<inizializzazione>; <espressione booleana>; <incremento>)
{
      <istruzioni>
}
```

è equivalente a:

```
<inizializzazione>;
while (<espressione booleana>)
{
      <istruzioni>
      <incremento>;
}
```

Ma è più elegante!

Equivalenza dell'istruzione for con il while

calcola 1+2+…+N

```
int somma = 0;
for (int k = 1; k <= N; k++) somma += k;
System.out.println(somma);
```

calcola 1*2*…*N

```
int prodotto = 1;
for (int k = 1; k <= N; k++) prodotto *= k;
System.out.println(prodotto);
```

stampa una tabella di valori di funzione

```
for (int r = 0; r <= N; r++)
    System.out.println(r + " " + 2*Math.PI*r);
```

stampa i valori di un array di stringhe

```
String[] s = new String[] { "a", "b", "c", "d" };
for (int k = 0; k < s.length; k++)
    System.out.println(k + " " + s[k]);
```

calcola l'armonica $H_N = 1+1/2+\ldots+1/N$

```
double somma = 0.0;
for (int k = 1; k <= N; k++) somma += 1.0/k;
```

Equivalenza dell'istruzione for con il while

Stampa 3 righe, ciascuna con 4 asterischi

```
for (int i = 1; i <= 3; i++)
{
      for (int j = 1; j <= 4; j++)
            System.out.print("*");
      System.out.println();
}
```

Stampa 4 righe, ciascuna con 3 asterischi

```
for (int i = 1; i <= 4; i++)
{
      for (int j = 1; j <= 3; j++)
            System.out.print("*");
      System.out.println();
}
```

Stampa 4 righe, di lunghezza 1,2,3 e 4

```
for (int i = 1; i <= 4; i++)
{
      for (int j = 1; j <= i; j++)
            System.out.print("*");
      System.out.println();
}
```

Stampa asterischi nelle colonne pari e $ nelle colonne dispari

```
for (int i = 1; i <= 3; i++)
{
      for (int j = 1; j <= 5; j++)
            if (j % 2 == 0) System.out.print("*");
            else System.out.print("$");
      System.out.println();
}
```

Stampa una scacchiera

```
for (int i = 1; i <= 3; i++)
{
      for (int j = 1; j <= 5; j++)
            if ((i+j) % 2 == 0) System.out.print("*");
            else System.out.print(" ");
      System.out.println();
}
```

Incremento non unitario / non crescente / multiplo

```
for (int i=0; i<=10; i+=3)
{

}
```

L'incremento non deve essere necessariamente unitario, dipende da cosa si intende realizzare

```
for (int i=10; i>0; i--)
{

}
```

La variabile di controllo non deve essere necessariamente crescente

```
for (int k = 0, i = 0; i <= 10; i++, k += 5)
{
    // codice dell'iterazione
}
```

Le istruzioni di inizializzazione e incremento possono riferirsi a più variabili

Esercizio: for annidati

- Scrivere un metodo che, dato un intero N, stampi una matrice NxN il cui elemento (i, j) vale:
  - 1 se i è un divisore di j (o viceversa);
  - 0 altrimenti.

Variabili visibili solo all'interno del ciclo for

```
public void printMatrix(final int N)
{
    for (int i = 1; i <= N; i++)
    {
        for (int j = 1; j <= N; j++)
        {
            if ((i % j == 0) || (j % i == 0))
                System.out.print("1 ");
            else
                System.out.print("0 ");
        }

        System.out.println();
    }
}
```

printMatrix(10)

```
1 1 1 1 1 1 1 1 1 1
1 1 0 1 0 1 0 1 0 1
1 0 1 0 0 1 0 0 1 0
1 1 0 1 0 0 0 1 0 0
1 0 0 0 1 0 0 0 0 1
1 1 1 0 0 1 0 0 0 0
1 0 0 0 0 0 1 0 0 0
1 1 0 1 0 0 0 1 0 0
1 0 1 0 0 0 0 0 1 0
1 1 0 0 1 0 0 0 0 1
```

Esercizio: ContaParola

- Scrivere un metodo che, presi in ingresso un testo sotto forma di stringa e una parola w, trasformi il testo in parole (token) e conti le occorrenze di w nel testo

Esercizio: StringaVerticale

- Scrivere un metodo che legge una stringa da console (ovvero da input args) e la stampa in verticale un carattere per linea
- Ad esempio, dato in input “ciao”, viene stampato:

```
c
i
a
o
```

Esercizio: StringheVerticali

- Scrivere un metodo che riceve tre stringhe e le stampa in verticale una accanto all'altra
- Ad esempio: date "ciao", "buondì", "hello", stampa:

```
cbh
iue
aol
onl
 do
 ì
```

Esercizio: ContaVocali

- Scrivere un metodo che riceve una stringa e stampa a video il conteggio delle vocali in essa contenute
- Ad esempio: data la stringa “le aiuole sono pulite”, il metodo stampa:

```
a=1 e=3 i=2 o=3 u=2
```

Esercizio: Divisori

- Scrivere un metodo che, dato un intero positivo n in ingresso, stampi i divisori propri di n (ovvero i divisori < n)
- Ad esempio, dato l'intero 20, il metodo stampa:

```
1, 2, 4, 5, 10
```

Esercizio: SommaNumeriPrecedenti

- Scrivere un metodo che, dati in ingresso due interi a e b e un terzo intero N, stampi a e b e gli N numeri della sequenza in cui ogni numero è la somma dei due precedenti
- Ad esempio, dati gli interi 2, 3 e 6, il metodo stampa:

```
2, 3, 5, 8, 13, 21, 34, 55
```

Esercizio: ConversioneNumeri

- Progettare una classe per la conversione di base dei numeri interi
- Ogni oggetto della classe viene costruito con un intero o con una stringa che contiene l'intero
- La classe è in grado di convertire l'intero nella base desiderata (restituito sotto forma di stringa)

Esercizio: FrasePalindroma

- Una stringa è palindroma se rimane uguale quando viene letta da sinistra verso destra o da destra verso sinistra
- Scrivere un metodo che dica che una stringa è palindroma
- Scrivere anche una classe di collaudo che testi il metodo in diverse situazioni
- Ad esempio, data in ingresso le stringhe “angelalavalalegna” o “itopinonavevanonipoti”, il metodo deve segnalare che la stringa è palindroma

Esercizio: Cornice

- Scrivere un metodo che, dato un intero N in ingresso, stampi una cornice NxN
- Ad esempio: dato l'intero 5 in ingresso il metodo stampa:

```
*****
*   *
*   *
*   *
*****
```

Esercizio: CorniceAvanzata

- Scrivere un metodo che, dato un intero N e una stringa in ingresso, stampi una cornice NxN all'interno della quale venga visualizzata la stringa (eventualmente suddivisa per righe)
- Ad esempio: dato l'intero 6 e la stringa "Cornici in Java" in ingresso il metodo stampa:

```
******
*Corn*
*ici *
*in J*
*ava *
******
```

Esercizio: TernePitagoriche

- Una terna pitagorica è una tripla di numeri interi a, b, c tali che
  1 <= a <= b <= c e $a^2+b^2 = c^2$
  - Ovvero a e b sono i lati di un triangolo rettangolo e c l'ipotenusa
- Scrivere un metodo che legge un intero N e stampa tutte le triple pitagoriche con c <= N
- Ad esempio: dato N=15 il metodo stampa:

a=3 b=4 c=5
a=6 b=8 c=10
a=5 b=12 c=13
a=9 b=12 c=15

Esercizio: StampaTriangoli

- Scrivere un metodo che, dato un intero positivo dispari N>1, stampi un triangolo isoscele la cui base è costituita da N caratteri e l'altezza da N-2
- Ad esempio, dato l'intero 5, il metodo stampa:

```
  *
 ***
*****
```

Esercizio: la classe RettangoloDiCaratteri

- Progettare una classe RettangoloDiCaratteri che rappresenta un rettangolo riempito con caratteri *
- Un oggetto della classe viene costruito fornendo la posizione x, y e la lunghezza e altezza del rettangolo
- Il metodo draw si occupa di stampare il rettangolo a video, partendo dalla posizione (0,0)
- Ad esempio, dato il rettangolo (2, 2, 4, 3), il metodo draw() stampa (␣ rappresenta una spazio):

```
␣
␣
␣****
␣****
␣****
```

Esercizio: ampliare la classe RettangoloDiCaratteri

- Aggiungere alla classe RettangoloDiCaratteri i seguenti metodi:
  - setCarattere(): Permette di specificare il carattere da utilizzare per stampare i rettangoli
  - drawVerticalStripes(): stampa il rettangolo a strisce verticali usando anche un secondo carattere, ad esempio:

```
*$*$
*$*$
*$*$
```

  - drawHorizontalStripes(): stampa il rettangolo a strisce orizzontali
  - drawChessboard(): stampa il rettangolo a mo' di scacchiera:

```
*$*$
$*$*
*$*$
```

  - Una seconda versione di setCarattere() che permetta di specificare entrambi i caratteri da utilizzare per la stampa
  - Un metodo che permetta di modificare la posizione del rettangolo
  - Un metodo che permetta di accedere ai due caratteri usati per la stampa

Esercizio: calcolo di media, mediana e moda

- Progettare una classe Sequenza i cui oggetti si costruiscono con un array di interi
- La classe espone i seguenti metodi:
  - getMediana() che restituisce l'elemnto centrale dell'array ordinato (si utilizzi Arrays.sort per ordinare l'array e Arrays.copyOf per effettuarne una copia)
  - getMedia() che restituisce la media degli elementi dell'array
  - getModa() che restituisce il valore più frequente nella sequenza

Esercizio: da numeri romani a numeri interi

- Progettare una classe NumeroRomano i cui oggetti si costruiscono con una stringa contenente un numero romano
  - M = 1000, D = 500, C = 100, L = 50, X = 10, V = 5, I = 1
- La classe espone il metodo toInteger() che restituisce il valore intero corrispondente
- Ad esempio, new

  NumeroRomano(“MMXIX”).toInteger() restituisce 2019

Uscire da un ciclo

- Indipendentemente dal tipo di ciclo (while, do…while, for), può essere necessario uscire dal ciclo durante l'esecuzione del suo corpo
- Mediante l'istruzione break
  - Utilizzabile solo all'interno di un ciclo

```
public class StampaPrimeNCifre
{
    public void stampaPrimeNCifre(final String s, final int N)
    {
        int conta = 0;
        for (int k = 0; k < s.length(); k++)
        {
            char c = s.charAt(k);
            if (Character.isDigit(c))
            {
                System.out.print(c);
                conta++;
                if (conta == N) break;
            }
        }
    }

    public static void main(String[] args)
    {
        new StampaPrimeNCifre().stampaPrimeNCifre("abc3ddfed5aafwewr94423948", 3);
    }
}
```

Metodo statico della classe Character: true se è una cifra

Esce dal ciclo

Uscire da cicli annidati

- L'istruzione break permette di uscire solo dal ciclo corrente **se non si specifica un'etichetta**
- Come fare nel caso in cui si voglia uscire da una sequenza di cicli annidati?
- Si può specificare un'etichetta prima di un ciclo e uscire da quel ciclo con break <etichetta>

```
outer:
for (int i = 0; i < h; i++)
{
    for (int j = 0; j < w; j++)
    {
        // codice qui
        // ...
        if (j == i) break outer;
    }
}
```

Esce dal for esterno

Break vs. return

- L'istruzione return interrompe l'esecuzione del metodo
- L'istruzione break interrompe l'esecuzione di un ciclo for, while o do…while.

Ancora sulle stringhe: un ChatBot appena appena intelligente…

Esempio di output

```
ChatBotAppenaAppenaIntelligente al tuo servizio...
ciao bello!
ciao, come ti chiami?
mi chiamo roberto
piacere di conoscerti, roberto
mi piacerebbe proprio andare a fare una bella nuotata!
perché non lo fai?
con questo tempo forse è meglio andare al mare!
hai visto che tempo che fa oggi?!
```

```java
import java.util.Scanner;

public class ChatBotAppenaAppenaIntelligente
{
    public static void main(String[] args)
    {
        Scanner scanner = new Scanner(System.in);

        System.out.println("ChatBotAppenaAppenaIntelligente al tuo servizio...");

        while(true)
        {
            String input = scanner.nextLine().toLowerCase();

            if (input.contains("esci")) break;
            else if (input.startsWith("ciao")) System.out.println("ciao, come ti chiami?");
            else
            {
                final String miChiamo = "mi chiamo";
                int index = input.indexOf(miChiamo);
                if (index != -1) System.out.println("piacere di conoscerti, "+input.substring(index+miChiamo.length()+1));
                else if (input.startsWith("mi piacerebbe")) System.out.println("perché non lo fai?");
                else System.out.println("hai visto che tempo che fa oggi?!");
            }
        }
    }
}
```

Ciclo infinito

Esce dal ciclo

Saltare all'iterazione successiva

- Può essere utile anche saltare all'iterazione successiva

```
public class StampaPrimeNCifre2
{
    public void stampaPrimeNCifre(final String s, final int N)
    {
        int conta = 0;
        for (int k = 0; k < s.length(); k++)
        {
            char c = s.charAt(k);
            if (!Character.isDigit(c)) continue;

            System.out.print(c);
            conta++;
            if (conta == N) break;
        }
    }

    public static void main(String[] args)
    {
        new StampaPrimeNCifre2().stampaPrimeNCifre("abc3ddfed5aafwewr94423948", 3);
    }
}
```

Salta questa iterazione

Coding horror: troppi casi, troppi “corpi” { … }

```
public boolean contiene (int posizione, int numero)
{
    if(mioArray.length > 0)
    {
        if(mioArray[posizione] == numero)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    else
    {
        return false;
    }
}
```

Coding horror: stampare invece di restituire

```java
/**
  * Calcola il resto dovuto al cliente.
  */

public void calcolaResto()
{
    System.out.println(totPagato-totPagare);
}
```

Coding horror: campi non privati

```
public class RegistratoreDiCassa
{
    Double FullPrice;
    Double Payed;

    public RegistratoreDiCassa()
    {
        FullPrice = 0.0;
        Payed = 0.0;
    }

    public void addItem(Double ItemPrice)
    {
        FullPrice += ItemPrice;
    }

    public void pay(Double Price)
    {
        Payed = Price;
    }

    public double change()
    {
        return FullPrice - Payed;
    }
}
```

Coding horror: metodi statici provano a modificare campi d'istanza

```
public class Segmento
{
    private Punto inizioA;
    private Punto inizioB;

    public Segmento(Punto A, Punto B)
    {
        inizioA = A;
        inizioB = B;
    }

    public static void modificaVerticeA(Punto puntoA)
    {
        inizioA = puntoA;
    }

    public static void modificaVerticeB(Punto puntoB)
    {
        inizioB = puntoB;
    }

    public static void main()
    {
        Punto startA = new Punto(1, 3, 8);
        Punto startB = new Punto(4, 4, 7);
        Segmento segmento1 = new Segmento(startA, startB);
    }
}
```

Coding horror: x=x+y

```
public void addPrice(double prezzo)
{
    price = price+prezzo;
}
```

usa l'operatore +=

Coding horror: troppe righe di codice

```java
public boolean contiene(int posizione, int numero)
{
    if (posizione >= arrayInteri.length)
        return false;
    else
    {
        if (arrayInteri[posizione] == numero)
            return true;
        else return false;
    }
}
```

```java
public boolean contiene(int posizione, int numero)
{
    return posizione < arrayInteri.length && arrayInteri[posizione] == numero;
}
```

Coding horror: oggetti senza dignità

```
public class Segmento
 {
    private Punto startPoint;
    private Punto endPoint;

    public Segmento()
    {
        startPoint = new Punto();
        endPoint = new Punto();
    }

    public Segmento(Punto startP, Punto endP)
    {
        startPoint = startP;
        endPoint = endP;
    }
}
```

Non può esistere un punto senza coordinate, né un segmento con punti non definiti

Coding horror: variabili locali inutili

```
public void registra(double ammontare)
{
    double restituisco = val+ammontare;
    val = restituisco;
}

public double paga(double contanti)
{
    double importoVersato=contanti;
    return importoVersato;
}
```

Coding horror: parentesi inutili

```
public boolean contiene(int pos, int value)
{
    boolean b=false;
    if (pos>(a.length-1)) return b;
    if (a[pos]==value) b=true;
        return b;
}
```

- L'operatore di confronto ha precedenza inferiore agli operatori aritmetici:
  - if (pos > a.length-1) return b;

```
if (pos < (a.length)) return (a[0]+a[1]);
if (pos>(a.length-1)) return b;
```

- Return non richiede parentesi
  - return a[0]+a[1];

Coding horror: si può fare in una riga!

```java
public int maxTripla()
{
    if(array[0] > array[array.length / 2] && array[0] > array[array.length - 1])
        return array[0];
    else if(array[0] < array[array.length / 2] && array[array.length / 2] > array[array.length - 1])
        return array[array.length / 2];
    return array[array.length - 1];

}
```

```java
public int maxTripla()
{
    if (array.length < 3)
        return 0;
    else
        return Math.max(Math.max(array[0], array[array.length-1]), array[array.length/2]);
}
```

Coding horror: usare float invece di double

```
public class RegistratoreDiCassa
{
    private float prezzo;
    private float resto;
    private float pagamento;

    public void insertPrezzo(float value)
    {

        float prezzo = value;

    }

    public void sommaPagamento(float contanti)
    {

        pagamento = pagamento + contanti;

    }

    public void calcolaResto()
    {
        resto = resto + (pagamento - prezzo);
    }
}
```

Coding horror: campi pubblici

```
public class Punto
{
    public double x;
    public double y;
    public double z;

    public Punto (double x,double y,double z)
    {
    }

    public double getX()
    {
        return x;
    }

    public double getY()
    {
        return y;
    }

    public double getZ()
    {
        return z;
    }
}
```

Coding horror: il costruttore non salva i parametri

```
public class Segmento
{
    private Punto p1;
    private Punto p2;

    public Segmento(Punto p1, Punto p2)
    {

    }

    public static void main(String[] args)
    {
        Punto a = new Punto(1, 3, 8);
        Punto b = new Punto(4, 4, 7);
        Segmento s = new Segmento(a, b);
    }
}
```

Coding horror: inizializza due volte i campi

```
public class Segmento
{
    //Campi
    private Punto p1 = new Punto();
    private Punto p2 = new Punto();

    // Costruttore
    public Segmento(Punto pd1, Punto pd2)
    {
        p1=pd1;
        p2=pd2;
    }

    // Metodi

    // Main
    public static void main(String[] args)
    {
        Punto u1 = new Punto(1,3,8);
        Punto u2 = new Punto(4,4,7);
        Segmento s1 = new Segmento(u1,u2);
    }

}
```

Prima volta (inutile)

Seconda volta (corretto)

Coding horror: variabili di appoggio come campi!

```
public class MioArray
{
    int[] array;
    int varAppoggio = 0;
    int max = 0;
    int lunghezza = 0;  /* Instance variables */
    int posMax;
    int[] arrayFirstLast = null;

    public MioArray(int[] array)
    {
        this.array = array;
    }
```

Coding horror: variabili locali a un corpo

```
public int Somma()
{
    if (array.length > 2);
    {
        int somma = array[0] + array[1];
    }
    if (array.length == 1);
    {
        int somma = array[0];
    }
    if (array.length == 0);
    {
        int somma = 0;
    }
    return somma;
}
```

Inaccessibili all'esterno dell'if

Inaccessibili all'esterno dell'if

Inaccessibili all'esterno dell'if

Array

Array

- Un array rappresenta un gruppo di variabili (chiamate elementi) tutte dello stesso tipo
- Gli array **sono oggetti**
  – Quindi le variabili di array contengono il **riferimento** all'array
- Gli elementi di un array possono essere tipi primitivi (interi, double, ecc.) oppure riferimenti a oggetti (inclusi altri array!)

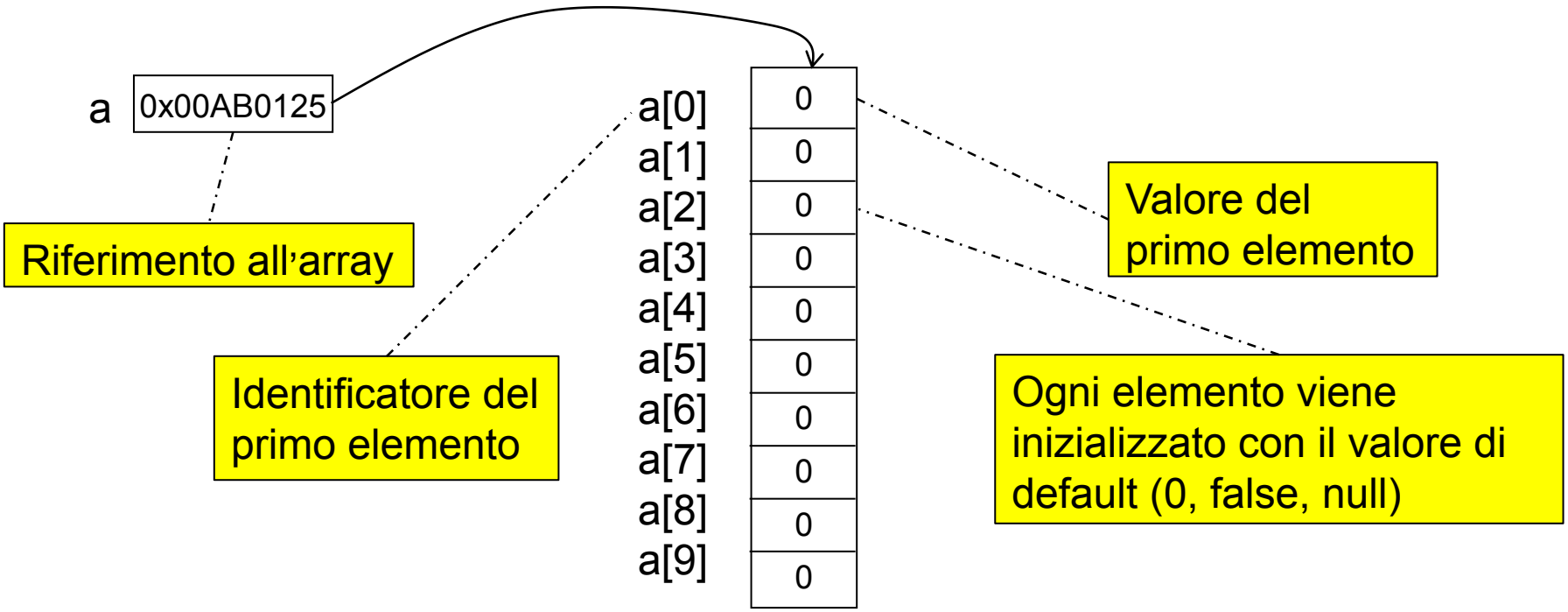
Un esempio di array: creazione senza valori

- Dichiarazione: `int[] a;`

Nome dell'array

- Creazione senza valori:

```
a = new int[10];
```



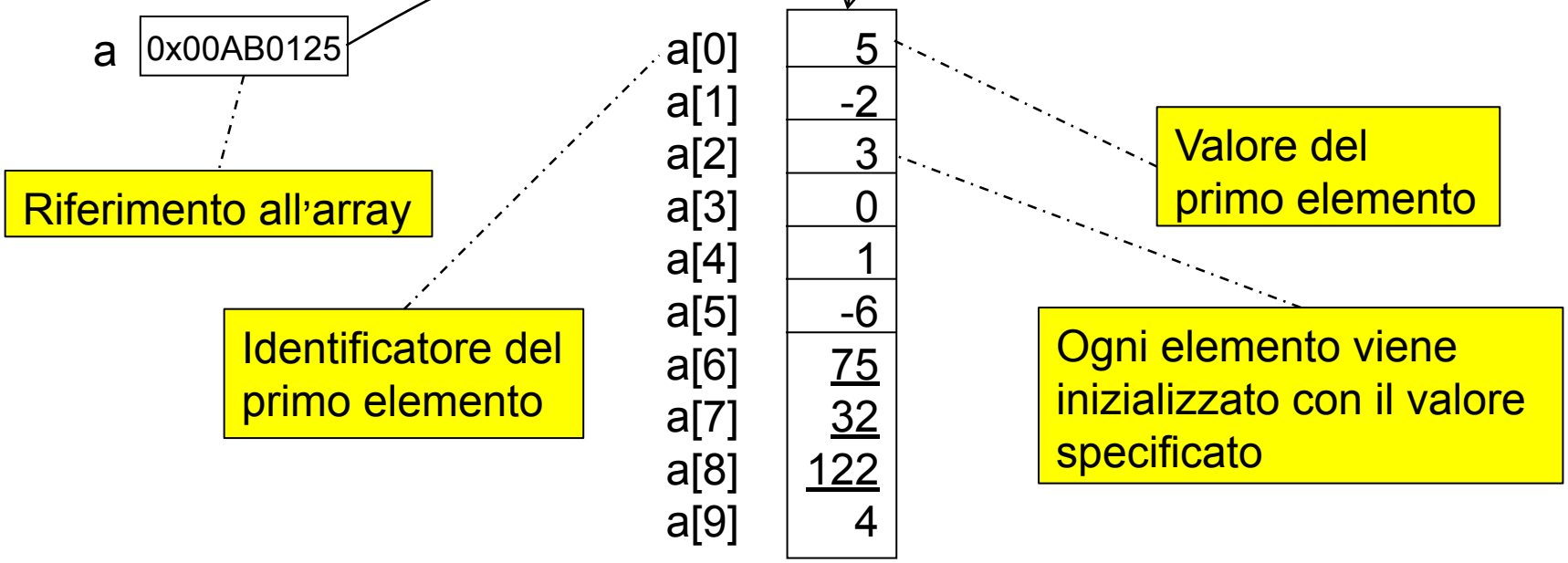
Un esempio di array: creazione con valori

- Dichiarazione: `int[] a;`

  Nome dell'array

- Creazione con valori:

```
a = new int[] { 5, -2, 3, 0, 1, -6, 75, 32, 122, 4 };
```



Dichiarazioni valide di array

Un array di 10 interi. Tutti inizializzati a 0.

```
int[] numeri = new int[10];
```

Meglio utilizzare una COSTANTE per la dimensione di un array.

```
final int NUMERO_DI_CIFRE = 10;
int[] numeri = new int[NUMERO_DI_CIFRE];
```

La lunghezza di un array può essere specificata con una variabile

```
int numeroDiCifre = new Scanner(System.in).nextInt();
int[] numeri = new int[numeroDiCifre];
```

Un array di 10 interi costruito specificando i valori

```
int[] numeri = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
```

Un array di 3 riferimenti a stringa, tutti inizializzati a null

```
String[] nomi = new String[3];
```

Un array di 3 riferimenti a stringa, con valori assegnati

```
String[] nomi = { "mario", "luigi", "wario" };
```

Errori tipici con gli array

- Al contrario del linguaggio C, in Java non è possibile specificare la dimensione accanto al nome dell'array:

```
int a[10];
```

- Né specificare la dimensione e allo stesso tempo inizializzare i valori:

```
int[] a = new int[10] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
```

Accedere agli elementi di un array

- Si accede a un elemento dell'array specificando il nome dell'array seguito dalla posizione (indice) dell'elemento tra parentesi quadre

```
// stampa il sesto elemento dell'array
System.out.println(a[5]);

// memorizza la somma dei primi 3 elementi
int k = a[0] + a[1] + a[2];
```

- L'indice è sempre positivo, compreso tra 0 e la dimensione dell'array-1
- L'indice può essere un'espressione:

```
int i = 2;
int j = 3;

a[i+j*2] += 2;
```

Esercizio: stampa di un array

- Scrivere un metodo che, dato un array di stringhe, ne stampi i valori in sequenza
- Ad esempio, dato l'array { "son", "buoni", "almeno?", "assaggi", "il", "vino" } stampi: ["son", "buoni", "almeno", "assaggi", "il", "vino"]

```
public void stampaArray(String[] a)
{
    System.out.print("[ ");

    for (int k = 0; k < a.length; k++)
        System.out.print("\"" + a[k] + "\""
                         + (k == a.length-1 ? " ]" : ", "));

    System.out.println();
}
```

Lunghezza dell'array (**attenzione:** .length **SENZA PARENTESI**)

Esercizio: somma dei valori di un array

- Scrivere un metodo che, dato un array di interi, restituisca la somma dei suoi elementi
- Ad esempio, dato l'array { 10, 12, 12, 8 } il metodo deve restituire: 42

```
public int sommaArray(int[] a)
{
    int val = 0;

    for (int k = 0; k < a.length; k++) val += a[k];

    return val;
}
```

Esercizio: media dei valori di un array

- Utilizzando il metodo scritto per l'esercizio precedente, scrivere un metodo che, dato un array di interi, restituisca la media (double) dei suoi elementi
- Ad esempio, dato l'array { 10, 12, 12, 8 } il metodo deve restituire: 10.5

```
public double mediaArray(int[] a)
{
    return sommaArray(a)/a.length;
}
```

occhio: divide intero per intero (cosa serve?)

RIUSO DEL CODICE!!!

- Scrivete anche la versione per array di double!

Altri esercizi molto semplici con gli array

- Scrivere un metodo che, dato un array di stringhe e una stringa in input, restituisca true se l'array contiene la stringa, false altrimenti
- Scrivere una seconda versione del metodo che restituisca la posizione della stringa trovata, -1 altrimenti
- Scrivere un metodo che, dato un array di double, restituisca il valore massimo dell'array

Esercizio: reimplementa conta vocali

- Scrivere un metodo che riceve una stringa e stampa a video il conteggio delle vocali in essa contenute
  - Utilizzare un array per il conteggio separato delle 5 vocali
- Ad esempio: data la stringa “le aiuole sono pulite”, il metodo stampa: `a=1 e=3 i=2 o=3 u=2`
- Come l’avevate implementato **SENZA** array?

Esercizio: array con fattoriale

- Scrivere un metodo che, dato un intero n in ingresso, restituisca un array di dimensione n contenente k! nella k-esima cella, per ogni valore di k

```
public int[] getArrayFattoriali(final int n)
{
    int[] fatt = new int[n];

    fatt[0] = 1;
    for (int k = 1; k < n; k++) fatt[k] = fatt[k-1]*k;

    return fatt;
}
```

Esercizio: “l’array risponde”

Progettare una classe MioArray i cui oggetti vengono costruiti con un array di interi (int[ ])

- La classe implementa i seguenti metodi:

– contiene, che dati in ingresso una posizione e un intero, restituisce true o false se l’intero è contenuto in quella posizione nell’array

– somma2, che restituisce la somma dei primi due elementi dell’array. Se l’array è di lunghezza inferiore (**info**: la lunghezza dell’array a si ottiene con il campo speciale length, quindi a.length), restituisce il valore del primo elemento oppure 0 se l’array è vuoto

– scambia, che date in ingresso due posizioni intere, scambia i valori presenti nelle due posizioni dell’array (es. scambia(1, 3) trasforma l’array { 1, 2, 3, 4, 5 } in { 1, 4, 3, 2, 5 })

– maxTripla: che restituisce il valore massimo tra il primo, l’ultimo e il valore in posizione intermedia dell’array (es. restituisce 3 se l’oggetto è costruito con { **1**, 7, 5, **3**, 0, 2, **2** }, le posizione esaminate sono in grassetto)

– falloInDue: che restituisce un array di due interi, il primo è il primo elemento dell’array dell’oggetto, il secondo è l’ultimo elemento dell’array dell’oggetto

Esercizio: stampa di istogrammi

- Progettare una classe Istogramma che rappresenta la distribuzione di dati (es. voti degli studenti) in un intervallo da i a j fornito in input (es. da 0 a 31 (trenta e lode))
- La classe permette di incrementare il conteggio in corrispondenza di ciascun elemento dell'intervallo (es. memorizzando così un nuovo voto di uno studente)
- La classe può stampare a video l'istogramma corrispondente
  - Più facile in orizzontale
  - Provate a stampare in verticale!!!

Esercizio: mescolare e distribuire un mazzo di carte da gioco

- Progettare una classe Carta che rappresenti una singola carta da gioco (con seme e valore)
  - La classe deve restituire su richiesta la propria rappresentazione sotto forma di stringa
- Progettare quindi una classe MazzoDiCarte che rappresenti un intero mazzo da 52 carte
- La classe deve implementare i seguenti metodi:
  - mescola il mazzo di carte
  - distribuisci la prossima carta
- Infine si progetti una classe di collaudo che crea un mazzo, mescoli le carte e ne distribuisca carte fino ad esaurimento del mazzo

Esercizio: DaCifreALettere

- Scrivere un metodo che prenda in ingresso una stringa contenente cifre e restituisca una stringa in cui ciascuna cifra è stata trasformata nella parola corrispondente
- Ad esempio, data in input la stringa “8452”, il metodo restituisce “otto quattro cinque due”
- Viceversa, scrivere un metodo che prenda in ingresso una stringa contenente cifre scritte a lettere e restituisca una stringa contenente le cifre corrispondenti
- Ad esempio, data in input la stringa “otto quattro cinque due”, il metodo restituisce “8452”
- **Nota:** è conveniente utilizzare gli array di stringhe String[]!

Esercizio: da cifre a lettere e viceversa (avanzato!!!)

- Come l'esercizio precedente, ma stampando (o leggendo) a lettere tenendo conto della posizione delle cifre (occhio ai **casi speciali**: undici, dodici, ecc…)
- Ad esempio, "8452" viene trasformato in "ottomila quattrocento cinquanta due"

Modificare le dimensioni di un array

- Un array ha dimensioni prefissate che **NON** possono essere modificate
- Tuttavia è possibile creare un nuovo array con nuove dimensioni a partire da un array preesistente

Con il metodo statico copyOf della classe java.util.Arrays

```
import java.util.Arrays;

public class MyArrays
{
    public static void main(String[] args)
    {
        // array di dimensione 9
        int[] array = { 1, 5, 8, 2, 3, 4, 7, 6, 9 };

        System.out.println(Arrays.toString(array));

        // restringe l'array a dimensione 5
        array = Arrays.copyOf(array, 5);
        System.out.println(Arrays.toString(array));

        // allarga l'array a dimensione 8 (gli ultimi 3 valori sono inizializzati a 0)
        array = Arrays.copyOf(array, 8);
        System.out.println(Arrays.toString(array));
    }
}
```

Crea un nuovo array con i primi 5 elementi dell' array in input

Crea un nuovo array di 8 elementi (i primi elementi dall' array in input)

Metodo per la formattazione a stringa di un array

Output

```
[1, 5, 8, 2, 3, 4, 7, 6, 9]
[1, 5, 8, 2, 3]
[1, 5, 8, 2, 3, 0, 0, 0]
```

Esercizio: implementare un filtro

- Progettare una classe Filtro costruita con un array di interi
- La classe implementa operazioni che permettono di ottenere nuovi sotto-array dell'array iniziale:
  - passaBasso: restituisce tutti gli elementi <= k nell'ordine iniziale
  - passaAlto: restituisce tutti gli elementi >= k nell'ordine iniziale
  - filtra: restituisce l'array iniziale da cui sono state eliminate tutte le occorrenze dell'intero passato in input
  - filtra: una seconda versione del metodo che restituisce l'array iniziale da cui vengono eliminate tutte le occorrenze di interi presenti nell'array passato in input
- Se Filtro viene costruito con l'array { 1, 2, 10, 2, 42, 7, 8 }:
  - passaBasso(8) restituisce { 1, 2, 2, 7, 8 }
  - passaAlto(9) restituisce { 10, 42 }
  - filtra(2) restituisce { 1, 10, 42, 7, 8 }
  - filtra(new int[] { 2, 7, 42 }) restituisce { 1, 10, 8 }

Esercizio: reimplementare Arrays.copyOf() e Arrays.toString()

- Implementare un metodo statico copyOf che, analogamente a java.util.Arrays.copyOf, copi un array in un nuovo array delle dimensioni specificate (troncando l'array in input, se più grande)

Esercizio: sequenza di cifre estensibile

- Progettare una classe SequenzaDiCifre che espone un metodo che, data in input una stringa e un intero N, aggiunga alla sequenza inizialmente vuota (rappresentata mediate un array) le prime N cifre contenute nella stringa (si assuma che ne contenga comunque almeno N). La classe espone anche un metodo toString che fornisce una rappresentazione sotto forma di stringa della sequenza.
- Ad esempio:
  - SequenzaDiCifre s = new SequenzaDiCifre();
  - s.aggiungiCifre("abc1--23", 2);
  - s.aggiungiCifre("xx0a8b76543100", 4);
  - System.out.println(s.toString());
- stampa: [1,2,0,8,7,6]

Esercizio: implementare una lista mediante array

- Che cos'è una lista? E' una sequenza di oggetti
- Implementare una classe ListaDiInteri che permetta le seguenti operazioni:
  - Restituisce l'elemento i-esimo della lista
  - Restituisce l'indice della posizione dell'intero fornito in input
  - Restituisce una stringa formattata contenente la lista di interi
  - Restituisce la dimensione della lista
  - Contiene un determinato intero (true o false)?
  - Aggiungi un intero in coda alla lista
  - Aggiungi un intero nella posizione specificata
  - Elimina la prima occorrenza di un intero dalla lista
  - Elimina l'elemento i-esimo della lista

Metodi con un numero di parametri variabile (Java >=5)

- Si possono dichiarare metodi con un numero variabile di parametri (a partire da Java 5)
- Mediante la sintassi: tipo…

```
public class SommaDouble
{
    public static double sum(double... valori)
    {
        double somma = 0.0;
        for (int k = 0; k < valori.length; k++) somma += valori[k];
        return somma;
    }

    public static double sumFirstN(final int N, double... valori)
    {
        double somma = 0.0;
        for (int k = 0; k < valori.length && k < N; k++) somma += valori[k];
        return somma;
    }

    public static void main(String[] args)
    {
        System.out.println(sum());
        System.out.println(sum(1, 2, 3, 4));
        System.out.println(sumFirstN(3, 1, 2, 3, 4));
    }
}
```

Zero, uno o più valori double

Di fatto è un riferimento ad array

E' possibile specificare altri parametri, ma **PRIMA** dell'**UNICA** sequenza variabile di parametri

Output:

```
0
10
7
```

Array a 2 dimensioni

- Possiamo specificare un array a 2 dimensioni (o matrice) semplicemente specificando due coppie di parentesi quadre []:

```
String[][] matrice = new String[RIGHE][COLONNE];
```

- L'accesso avviene specificando le due dimensioni dell'array:

```
System.out.println(matrice[y][x]);
```

Esercizio: la tavola pitagorica

- Scrivere una classe che rappresenti la tavola pitagorica NxN (dove l'intero N è un parametro di costruzione della classe)
- La classe deve, su richiesta, restituire il valore della tabella in corrispondenza della posizione (i, j)

La classe deve poter stampare l'intera tavola

x	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

Esercizio: il gioco del tris

- Progettare una classe ScacchieraTris che implementi la scacchiera del gioco del tris
- La classe deve memorizzare la scacchiera i cui elementi possono essere:
  - " " (se non è stata ancora occupata la casella)
  - "X" oppure "O" (secondo il giocatore che ha occupato la casella)
- La classe deve stampare in qualsiasi momento la situazione della scacchiera
- Deve permettere di occupare una casella con un simbolo "X" o "O"
- Progettare quindi una classe Tris che implementi il gioco utilizzando la scacchiera appena progettata