3. La sintassi di Java

3.1 I tipi di dati statici

In Java, come in Pascal, esistono tipi di dati statici predefiniti e sono i seguenti:

byte	8 bit da -128 a 127
short	16 bit coincide con l'integer di Pascal da -32768 a 32767
int	32 bit coincide con il longint del Pascal
long	64 bit
float	32 bit e un real in semplice precisione
double	64 bit è un real in doppia precisione
char	16 bit è esteso rispetto il Pascal: deve contenere i car. UNICODE
boolean	true o false (è oscurata all'utente, la dimensione dipende dal sistema)

3.2 Le stringhe

Le stringhe in java sono oggetti e sono allocate in memoria in modo dinamico come segue :

String s = new String ("banana");

oppure anche in una forma apparentemente statica come:

String s = "banana";

Le due dichiarazioni sono equivalenti: generano in memoria lo stesso oggetto.

Per ora la trattazione si limiterà a questo breve accenno: la conoscenza della classe String sarà approfondita in un paragrafo successivo.

3.3 Gli array

In java gli array, mono e bidimensionali, sono allocati in memoria in modo dinamico.
Ad esempio, la dichiarazione di un array d'interi di 10 elementi può essere effettuata in modo esplicito secondo una delle seguenti sintassi:

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

oppure in modo implicito tramite l' assegnazione dei valori:

```
int v[ ]={1,5,6,7,8,11,12,13,14, -1};
int[ ] v={1,5,6,7,8,11,12,13,14, -1};
```

Analogamente per un array bidimensionale di caratteri:

```
char a[ ][ ]=new char[2][2];
```

N.B. Gli elementi sono numerati a partire da 0, perciò per il vettore **v** l'indice va da 0 a 9

3.4 Istruzioni di sequenza, selezione e ciclo

Java è un linguaggio imperativo e, come tutti i linguaggi imperativi, è dotato delle strutture di controllo: SEQUENZA, SELEZIONE, ITERAZIONE.

Istruzioni sequenziali:

```
int a,c;                                  // dichiarazione di variabili intere di tipo int
a = 2;                                    // assegnazione di valore
c = a*a+7;                                // assegnazione di valore tramite espressione
System.out.println("a vale "+a+"  c vale "+c);      // istruzione di stampa
```

Istruzioni di selezione:

```
if (<condizione>)  {
        <blocco>
        }
```

```
if (<condizione>)  {
                <blocco>
                } else {
                <blocco>
                }
```

Istruzioni iterative:

il ciclo while:

```
while (<condizione>) {
        <blocco>
}
```

il repeat:

```
do {
    <blocco>
  } while (<condizione>)
```

il ciclo for:

```
for (tipo i=v_iniz; i<v_fin; i=i+step) {
    <blocco>
    }
```

N.B. Si osservi la dichiarazione della variabile **i**. Una variabile può essere definita in qualsiasi punto del programma; se è definita in un blocco la sua definizione è locale al blocco, cioè non è definita fuori dal blocco stesso.

3.5 Operatori aritmetici e logici

Operatori aritmetici

Operatore	**Significato**	**Esempio**
+	somma	3 + 5
-	differenza	
*	moltiplicazione	
/	divisione	
%	modulo (resto della div)	resto= a % b;
++	incremento	i++; equivale a i=i+1;
--	decremento	i--; equivale a i=i-1;

Operatori di confronto e logici

Operatore	**Significato**	**Esempio**
!	negazione	if (!a==true)
==	uguale a	if (a==b)
!=	diverso da	if (a !=b)
<	minore di	(a < b)
>	maggiore di	
<=	minore o uguale a	
>=	maggiore o uguale a	
&&	AND Logico	if (a && b)
||	OR Logico	if (a || b)

3.6 Esempi svolti

Esempio 1
Scrivere il codice del metodo **private static** int mcd(int a, int b) che consenta di determinare il massimo comune divisore tra due naturali.

```
private static int mcd(int a, int b) {
        if ((a > 0) && (b>0)) {                 // notare sintassi and logico
                int r=1;
                while (r !=0 ) {
                        r = a % b;
                        a = b;
                        b = r;
                        }
                return a;
                }
        else return 1;                          // se i numeri non sono > 0
        }                                       // il metodo restituisce 1 !
```

Esempio 2
Costruire un array di 15 componenti intere con valori casuali compresi tra 100 e 200 e stamparlo. Si utilizzi il metodo random() che genera numeri di tipo double compresi tra 0.0 e 0.99999

```
class cla_02 {
        public static void main(String args[]) {
                long v[ ]=new long[15];
                for (int i=0; i<=14; i++) {
                        v[i]=(int)Math.round(100*Math.random()+ 100);
                }
                for (int i=0; i<=14; i++) {
                        System.out.println("v["+i+"]="+v[i]);
                }
        }
}
```

Nell'esempio, l'array è stato definito in modo dinamico con l'operatore new() che alloca la memoria. Si tratta sempre di una struttura dati dinamica o è statica come gli array in Pascal ?

Si è No.
E' dinamica l'allocazione dello spazio occupato dall'array e quindi la dimensione di un array può essere definita durante l'esecuzione. L'array può addirittura essere ridefinito con una diversa dimensione all'interno dello stesso programma, ma in questo caso si perdono tutte le componenti precedenti in quanto è allocato un nuovo array. Non è dinamica se per dinamico s'intende la possibilità di far aumentare o diminuire la sua dimensione conservando i precedenti dati.

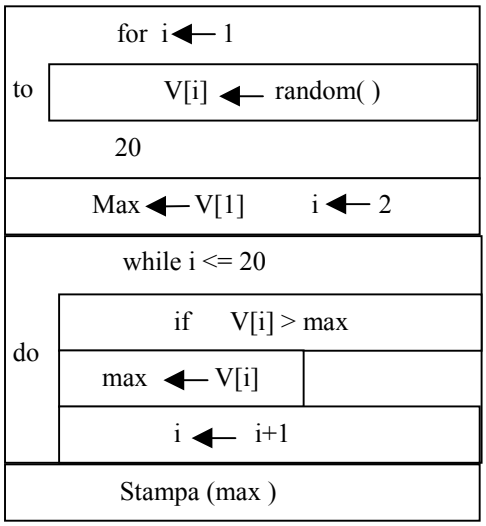
Esempio 3
Costruire un array di quattro componenti assegnando alla componente 0 il valore 1 e alla componente 3 il valore 4. Stampare l'array ottenuto e quindi ridefinirlo di dimensione 8 e ristampare le componenti del nuovo array.

```
class cla_03 {
    public static void main(String args[]) {
            int n=4;
           int[ ] v = new int[n];
            v[0]=1;        v[3]=4;
            for (int i=0; i<=3; i++) {        System.out.print(v[i]+"  “);              }
            System.out.println(" array ridefinito");
            n=8;
           v = new int[n];
            for (int i=0; i<=7; i++) {          System.out.print(v[i]+"  “);            }
      }
 }
```

Si nota che la stampa del primo array evidenzia l'inizializzazione a zero delle componenti alle quali non è stato assegnato un valore. La ridefinizione dell'array azzera le componenti. La stampa sarà:
v[0]=**1** v[1]=0 v[2]=0 v[3]=**4** array ridefinito
v[0]=0 v[1]=0 v[2]=0 v[3]=0 v[5]=0 v[6]=0 v[7]=0.

3.E - Esercizi

3.1 Il seguente struttogramma determina il massimo di un array di 20 valori decimali compresi tra 0.00000 e 0.99999, generati con la funzione random, e stampa il risultato. Tradurre il codice in Java.
Nota: la funzione random, che genera un double Java può essere invocata con:
risultato ← Math.random();
Attenzione: la prima componente di un array java ha indice 0 !



3.2 Il seguente programma Pascal determina la somma degli elementi di un array di 20 interi compresi tra 0 e 99, generati con la funzione random, e stampa il risultato. Tradurre il codice Pascal in Java.
NOTA: la funzione random, che genera un intero, in Java ha la forma:
risultato ← (int) Math.round(100*Math.random());
Attenzione: la prima componente di un array java ha indice 0 !

```
program es_3_2;
var  i, tot:integer;
     V : array [1..20] of integer;
begin
   for i :=1 to 20 do
     V[i] := randon(100);
   tot :=0;
   for i:=1 to 20 do
     tot := tot + V[i];
   writeln(tot);
end.
```

3.3 Il seguente programma Pascal orla una matrice 5x5 di numeri decimali inserendo nella sesta riga la somma degli elementi di ogni colonna e nella sesta colonna la somma degli elementi di ogni riga e stampa solo l'orlatura.
Tradurre in Java il codice Pascal riportato.
Attenzione: la prima componente di un array java ha indice 0 !

```
program es_3_3;
   var i, j:integer;
          M : array [1..6,1..6] of real;
```

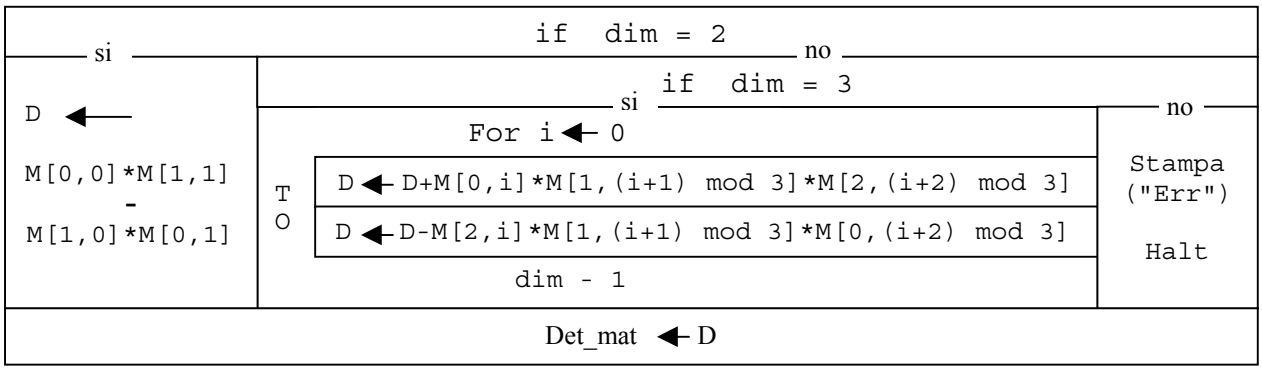
```
begin
   for i:=1 to 5 do
      for j:= 1 to 5 do
         M[i,j]:=random();
   for i:=1 to 5 do
      for j:= 1 to 5 do
         begin
          M[i,6]:=M[i,6]+M[i,j];
          M[6,j]:=M[6,j]+M[i,j];
         end;
   writeln ('ultima riga');
   for i:=1 to 5 do writeln (' M[6,',i,' ]=',M[6,i]);
   writeln ('ultima colonna');
   for i:=1 to 5 do writeln (' M[',i,',6 ]=',M[i,6]);
end.
```

3.4 la seguente function Pascal, dopo aver effettuato gli opportuni controlli sui dati ricevuti, calcola la potenza di esponente *e*, intero positivo e minore di 10, di un numero decimale *b* e restituisce il risultato. Tradurre il codice Pascal in Java inclusa l'intestazione o interfaccia.
Nota: fare riferimento all'esempio 1 del capitolo 3.

```
Function pot(b: real; e: integer): real;
     var i: integer; p:real;
begin
  if (e>0) and (e<10)
     then begin
          p := 1;
          for i=1 to e do
               p := p*b;
          end
     else p := 0; {valore privo di significato}
  pot := p;
end;
```

3.5 La seguente function calcola il determinate di una matrice quadrata 2x2 o 3x3 a seconda della valore *dim* passato. Tradurre sia l'intestazione Pascal sia lo struttogramma in linguaggio Java.
Nota: fare riferimento all'esempio 1 del capitolo 3.

```
Type matrice = array [0..2, 0..2] of real;
Function Det_mat(M : matrice; dim : integer):real;
   Var i : integer; D : real;
```



3.6 Si desidera scrivere l'intestazione di una Function che riceve un array di reali e la sua dimensione e restituisce la posizione del massimo.
Richieste: Scrivere l'intestazione Java della function.
Nota: fare riferimento all'esempio 1 del capitolo 3.

3.7 la seguente procedure Pascal riceve un array di numeri decimali (real) di dimensione effettiva *dim* e restituisce il massimo e la sua posizione relativa nell'array. Completare l'intestazione e il codice Java.
NOTA: la classe **ris** è un dato strutturato necessario per restituire in output l'equivalente della "procedure" in Java.
Attenzione: la prima componente di un array Java ha indice 0 !

```
Type vetto = array [1..100] of real;
```

```
procedure pos_max (v :vetto; dim :integer;
                   var max :real; var pos :integer);
Var i : integer;
Begin
    Max := v[1]; i := 2; pos := 1;
    while i<=dim do
        begin
            if  v[i] > max then
                begin
                    max := v[i]; pos := i;
                end,
            i := i+1;
        end;
end;
```

```
class ris {     public double max;
                public int pos;
          }
```

```
public static …… pos_max(double v[], …. ) {

   ris r = new ris();
   int i = ….;  r.pos:=….;
   r.max  = v[…];
   while (……..)  {
      if (……..) {
            ……..
            ……..
        }
   ……..
   }
   return r;
}
```

3.8 Realizzare una "procedure" o "metodo statico" che riceve un array di numeri decimali (double) di dimensione effettiva dim e restituisce il massimo, il minimo e le loro posizioni relative nell'array.
Richieste:
a) scrivere la *classe ris* necessaria per restituire i risultati richiesti, desumendolo per analogia dal codice della *classe ris* dell'esercizio precedente.
b) Scrivere il metodo richiesto in Java e la sua corretta intestazione.

3.9 Realizzare il main Java che invochi il "metodo" *pot()*, realizzato nell'esercizio 3.4, e stampi il risultato ottenuto.

3.10 Realizzare il main() Java che invochi il "metodo" realizzato nell'esercizio 3.8 e stampi i risultati ottenuti.