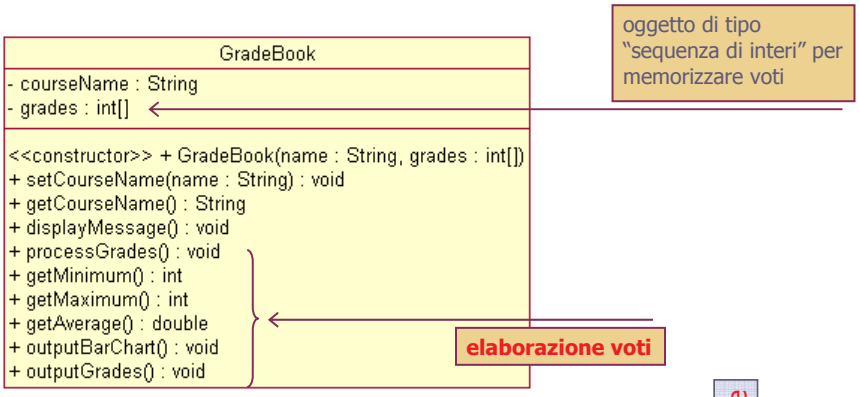
Istruzioni di controllo :
Parte I

Per scrivere un programma che risolve un problema è importante avere una comprensione adeguata del problema e adottare principi di programmazione efficaci per la sua soluzione. Impareremo ora i principi della programmazione strutturata che sono cruciali per la creazione e la manipolazione di classi e oggetti

Sommario

- Definizione di algoritmo
- La nozione di controllo
- Strutture di controllo
  - Struttura sequenziale
  - Strutture di selezione (if, if-else)
  - Strutture di iterazione (while)

Progetto classe "GradeBook"



→ Progetto di algoritmi
→ Costrutti del linguaggio java (cicli e decisioni)

questa lezione

La nozione di *Algoritmo* (1)

- Definizione di informatica del ACM (Association for Computing Machinery)

  L'informatica è la scienza che studia gli algoritmi per la rappresentazione e l'elaborazione dell'informazione

- L'enfasi è posta sulla nozione di algoritmo che è forse la più importante di tutta l'informatica
- Il computer è solo lo strumento informatico che grazie alle sue enormi potenzialità consente di trattare (rappresentare ed elaborare) grandi quantità di informazione

La nozione di *Algoritmo* (2)

- Problema → Algoritmo → Programma
- Ogni volta che dobbiamo risolvere un problema, dobbiamo individuare un metodo risolutivo o algoritmo
- Un algoritmo per la risoluzione di un problema è una sequenza finita di passi (azioni) che eseguiti ordinatamente a partire dai dati di input del problema consentono di ricavare i suoi dati di output

Definizione di *Algoritmo* (1)

- Definiamo la nozione di algoritmo in maniera intuitiva attraverso il seguente esempio Ricerca di un numero telefonico
  - Informazioni da rappresentare: *numeri telefonici di persone*
  - Due possibili fonti di informazione
    - *Elenco telefonico*: informazioni organizzate in ordine alfabetico prima per città e poi per cognome
    - *Rubrica personale*: informazioni organizzate per gruppi di lettere in base al nome o al cognome e, all'interno di ogni gruppo, in maniera casuale
  - Stesse informazioni organizzate in maniera diversa a seconda del supporto

Definizione di *Algoritmo* (2)

- Ricerca nell'elenco telefonico
  - Apri l'elenco in un punto *approssimativamente* vicino a quello in cui dovrebbe trovarsi la città
  - Procedi in avanti oppure all'indietro *per gruppi di pagine* fino a trovare la città d'interesse
  - Procedi in avanti oppure all'indietro *per gruppi di pagine* fino a trovare la pagina contenente il cognome cercato
  - Nella pagine individuata cerca il numero di telefono in maniera sequenziale

Definizione di *Algoritmo* (3)

- Ricerca nella rubrica personale
  - Seleziona la pagina associata al gruppo di lettere contenente l'iniziale del nome o del cognome della persona da cercare
  - Nella pagina selezionata cerca il numero di telefono in maniera sequenziale
  - Se la ricerca non ha esito positivo allora il numero non è presente nella rubrica

Definizione di *Algoritmo* (4)

- Osservazioni
  - Il metodo risolutivo per la ricerca di un numero telefonico è influenzato dal modo in cui i dati (numeri telefonici) sono organizzati
  - L'algoritmo deve essere espresso in un linguaggio comprensibile al suo esecutore
  - Due possibili esecutori di un algoritmo
    - Esecutore umano
    - Esecutore elettronico

Definizione di *Algoritmo* (5)

- Esecutore umano
  - Algoritmo espresso in linguaggio naturale
  - Entrambe i metodi risolutivi per la ricerca di un numero telefonico possono essere considerati algoritmi
  - L'esecutore umano è dotato di *buon senso* e *intuizione* che gli consentono di eseguire un algoritmo anche in presenza di situazioni ambigue come ad esempio:
    - Apri l'elenco in un punto *approssimativamente* vicino a quello cercato
    - Procedi in avanti oppure all'indietro *per gruppi di pagine* fino a trovare la città d'interesse

Definizione di *Algoritmo* (6)

- Esecutore elettronico
  - È uno strumento in grado di eseguire azioni (elaborazioni) elementari su oggetti (dati) elementari
  - Un algoritmo per un elaboratore elettronico deve essere:
    - Non ambiguo – le operazioni devono essere univocamente interpretabili
    - Eseguibile – ogni operazione deve poter essere eseguita con le risorse a disposizione
    - Finito – l'esecuzione dell'algoritmo deve terminare in un tempo finito per ogni possibile configurazione dei dati di input

Definizione di *Algoritmo* (7)

- Il metodo risolutivo per la ricerca di un numero telefonico in una rubrica è un algoritmo se l'esecutore è un elaboratore elettronico
- Il metodo risolutivo per la ricerca di un numero telefonico nell'elenco telefonico non è un algoritmo per l'esistenza delle seguenti operazioni ambigue:
  - Apri l'elenco in un punto *approssimativamente* vicino a quello cercato (quanto vicino?)
  - Procedi in avanti oppure all'indietro *per gruppi di pagine* fino a trovare la città d'interesse (quante?)

Considerazioni

- Da questo momento in poi quando parleremo di algoritmi ci riferiremo a metodi risolutivi per problemi computazionali ossia problemi risolvibili con un elaboratore elettronico
- Compito dell'informatico è produrre algoritmi, cioè capire la sequenza di passi che porta alla soluzione del problema, e codificarli in programmi (metodi), cioè renderli comprensibili all'elaboratore
- Perché un algoritmo sia ben progettato, ognuno dei metodi che lo compongono deve essere composto da poche linee di codice

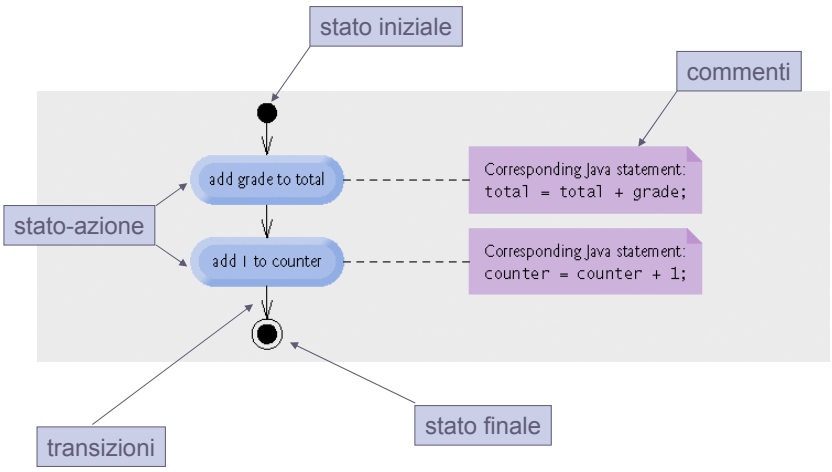
La nozione di Controllo

- Riassumendo un algoritmo è una procedura che risolve un problema in termini di:
  - Azioni da eseguire
  - Ordine in cui queste azioni vengono eseguite
- L'ordine di esecuzione delle azioni di un programma è chiamato controllo del programma
- Nel seguito esamineremo le strutture di controllo disponibili in Java

Strutture di controllo

- Esecuzione sequenziale: le istruzioni di un programma vengono eseguite una dopo l'altra nell'ordine in cui sono state scritte
- Trasferimento del controllo: meccanismo che consente al programmatore di variare l'ordine sequenziale. Viene realizzato attraverso le seguenti strutture di controllo
  - Struttura sequenziale
  - Struttura di selezione
  - Struttura di iterazione

Struttura sequenziale



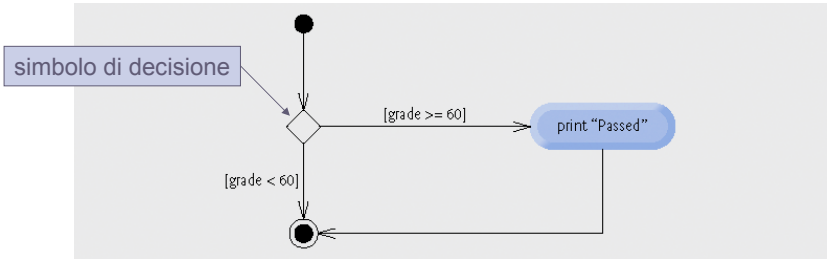
Strutture di selezione

- Servono a decidere quale azione intraprendere tra diverse alternative possibili
- Java prevede tre diversi tipi di strutture di selezione:
  - Selezione singola: istruzione **if**
  - Selezione doppia: istruzione **if … else**
  - Selezione multipla: istruzione **switch**

Non trattata

Selezione singola

Se il voto è maggiore o uguale a 60
Visualizza “Promosso”



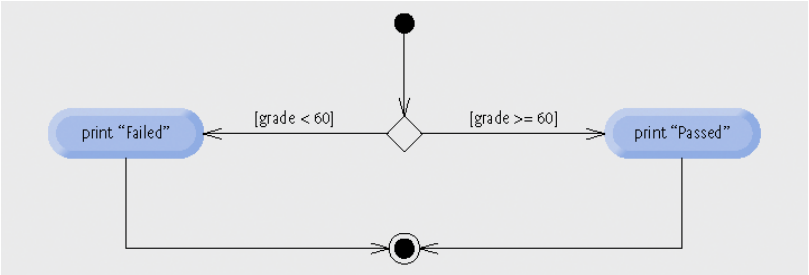
Codice Java

```
if ( grade >= 60 )
        System.out.println( “Passed” )
```

Struttura di selezione doppia

```
Se il voto è maggiore o uguale a 60
        Visualizza “Promosso”
Altrimenti
        Visualizza “Bocciato”
```

```
if ( grade >= 60 )
        System.out.println( “Passed” )
else
        System.out.println( “Failed” )
```



Istruzioni if…else nidificate (1)

```
Se voto è maggiore o uguale a 90
  Visualizza “A”
Altrimenti
  Se voto è maggiore o uguale a 80
      Visualizza “B”
  Altrimenti
      Se voto è maggiore o uguale a 70
            Visualizza “C”
      Altrimenti
            Visualizza “D”
```

```
if ( grade >= 90 )
  System.out.println( “A” );
else
  if ( grade >= 80 )
     System.out.println( “B” );
  else
     if ( grade >= 70 )
           System.out.println( “C” );
     else
           System.out.println( “D” );
```

```
if ( grade >= 90 )
  System.out.println( “A” );
else if ( grade >= 80 )
  System.out.println( “B” );
else if ( grade >= 70 )
  System.out.println( “C” );
else
  System.out.println( “D” );
```

Istruzioni if…else nidificate (2)

```
if ( x > 5 )
    if ( y > 5 )
        System.out.println( “x e y sono > di 5” );
else
    System.out.println( “x è <= a 5” );
```

Cosa visualizza se x è 4?

NIENTE!

Infatti, il compilatore java la interpreta come segue:

```
if ( x > 5 )
    if ( y > 5 )
        System.out.println( “x e y sono > di 5” );
    else
        System.out.println( “x è <= a 5” );
```

Per forzare la stampa di “x è <= a 5” quando x vale 4 bisogna scrivere il programma come segue

```
if ( x > 5 ) {
    if ( y > 5 )
        System.out.println( “x e y sono > di 5” );
}
else System.out.println( “x è <= a 5” );
```


Blocchi

L’istruzione **if** si aspetta solo una istruzione all’interno del proprio corpo. Per includere più istruzioni nel corpo dell’istruzione **if** bisogna usare le parentesi graffe come segue

```
if ( grade >= 60 )
    System.out.println( “Promosso” );
else {
    System.out.println( “Bocciato” );
    System.out.println( “Deve ripetere l’esame” );
}
```

Se grade è minore di 60 il programma visualizza:

Bocciato
Deve ripetere l’esame

Strutture di iterazione

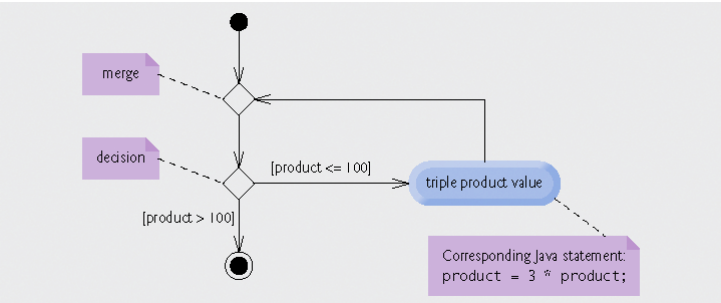
- Consentono di eseguire ripetutamente una azione fino a quando una data condizione rimane vera
- Java prevede tre diversi tipi di strutture di iterazione:
  - Istruzione **while**
  - Istruzione **for**
  - Istruzione **do … while**

Istruzione while (1)

Esempio: Trovare la prima potenza di 3 superiore a 100

Pseudocodice
Inizializza la variabile prodotto a 3
Finché prodotto è minore o uguale a 100
esegui prodotto = prodotto * 3

Codice Java
```
int prodotto = 3;
while ( prodotto <= 100 )
      prodotto = prodotto * 3;
```



Istruzione while (2)

- Usiamo l'istruzione while per formalizzare gli elementi richiesti per effettuare una iterazione controllata da un contatore (esempio: visualizzare i numeri da 1 a 10)
  1. Una variabile di controllo
  2. Il valore iniziale della variabile di controllo
  3. L'incremento o decremento che la variabile di controllo subisce ad ogni iterazione del ciclo
  4. La condizione che verifica se la variabile ha raggiunto il suo valore finale, cioè che determina la fine del ciclo

Istruzione while (3)

```
/* Iterazione controllata da contatore con l'uso dell'istruzione while */

public class WhileCounter {
   public static void main( String[] args ) {

      int counter = 1;
      while ( counter <= 10 ) {
         System.out.printf( "%d ", counter );
         counter++;
      }

   }
}  //----- fine classe WhileCounter
```

L'istruzione while può essere usata per implementare qualsiasi iterazione controllata da contatore

Esercizi proposti

- Scrivere una classe java che prende in input 10 numeri interi e visualizza sullo schermo quanti di essi sono pari e quanti sono dispari
  - Suggerimento: bisogna usare la struttura di selezione singola e la struttura di iterazione **while**