Istruzioni di selezione in Java

Fondamenti di architettura e programmazione
Corso di laurea in Comunicazione digitale

Ordine di esecuzione

- Nei metodi l'ordine di esecuzione delle istruzioni è sequenziale se non specificato altrimenti
- Alcune istruzioni consentono di alterare l'ordine sequenziale:
  - Decidere se eseguire o meno un'istruzione
  - Eseguire un'istruzione ripetutamente
- L'ordine di esecuzione delle istruzioni si chiama *flusso di controllo*

Flusso di controllo

- Il controllo dell'esecuzione del programma avviene mediante:
  - Istruzioni condizionali o di selezione
  - Istruzioni di ripetizione
- Entrambe i tipi di istruzioni si servono di espressioni condizionali e operatori

Istruzione condizionale

- Consente di stabilire qual'è la prossima istruzione eseguire
- Detta anche ***istruzione di selezione*** perchè consente di scegliere e prendere decisioni
- Le istruzioni condizionali Java
  - *if statement*
  - *if-else statement*
  - *switch statement*

Istruzione `if`

- Sintassi dell'istruzione `if`

La condizione deve essere *espressione booleana*.
Deve essere valutata vero o falso.

`if` una parola riservata Java

```
if ( condizione )
   istruzione;
```

Se la condizione è vera, viene eseguita l'istruzione.
Se è falsa, l'istruzione è tralasciata.

Esempio d'istruzione `if`

```
if (somma > MAX)
   somma = somma - MAX;
   System.out.println ("La somma è " + somma);
```

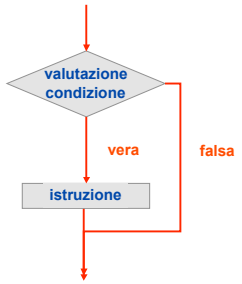
Prima si valuta la condizione: l'espressione `(somma > MAX)`

Se la condizione è vera, viene eseguita **l'istruzione di assegnamento,** altrimenti questa viene saltata.

In ogni caso viene eseguita l'istruzione `println`.
Non farsi ingannare dall'indentazione

- Esempio Age.java

Semantica dell'istruzione `if`



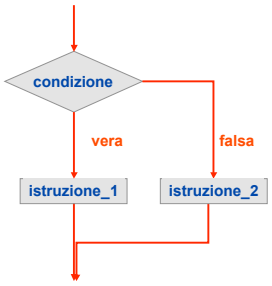
Istruzione `if-else`

- La clausola *else* può essere aggiunta all'istruzione *if*

```
if ( condizione )
   istruzione_1;
else
   istruzione_2;
```

- `condizione` vera viene eseguita `istruzione_1`; se è falsa viene eseguita `istruzione_2`
- Viene eseguita una sola istruzione in alternativa e mai entrambe
- Esempio Wages.java

Semantica dell'istruzione `if-else`



Istruzione blocco

- Più istruzioni possono essere raggruppate in un *blocco*
  - Il blocco è delimitato dalle parentesi graffe `{ … }`
- Un blocco può essere usato là dove la sintassi di Java vuole un istruzione
- Esempio: in un'istruzione `if-else`, la porzione `if`, o la porzione `else` o entrambe, possono essere blocchi
- See Guessing.java

Istruzioni `if` innestati

- L'istruzione da eseguire come risultato della valutazione di una condizione potrebbe essere a sua volta un'istruzione `if-else`
- Queste istruzioni sono dette *istruzioni* `if` *innestate*
- esempio MinOfThree.java
- La porzione `else` è associata all'ultima istruzione `if` (non fatevi ingannare dall'indentazione)

Confronti tra caratteri

- Gli operatori di relazione possono essere usati sui dati di tipo carattere
- Il risultato dipende dalla posizione nella tabella Unicode

```
if ('+' < 'J')
   System.out.println ("+ è minore di J");
```

- La condizione è vera perché il car `'+'` viene prima del car `'J'` in Unicode:
- Le maiuscole (A-Z) e le minuscole (a-z) sono in ordine alfabetico nella tabella Unicode

Confronti tra stringhe

- Una stringa in Java è un oggetto
- Non possiamo usare gli operatori relazionali `==`, `<`, `>`
- Uguaglianza. Usare il metodo `equals` per confrontare il contenuto di due oggetti stringa
  - `stringa_1.equals(stringa_2)`

Confronti tra stringhe

- Confronti. Usare il metodo `compareTo` per confrontare stringhe
  - Secondo l'ordine dei caratteri nella tabella Unicode
  - `stringa_1.compareTo(stringa_2)`
    - `< 0` se `stringa_1` precede `stringa_2`
    - `> 0` se `stringa_1` segue `stringa_2`
    - `== 0` se sono uguali

Errore comune

- Utilizzare `==` anziché `equals`
- L'operatore `==`
  - Verifica che le due stringhe si riferiscono allo stesso oggetto e non se si riferiscono a oggetti con lo stesso contenuto

```
String nome = "carlo";
if (nome == "carlo") … vero

String soprannome = "carlo magno";
nome = soprannome.substring (0, 5);
if (nome == "carlo") … falso
```

- UguaglianzaStringhe.java

Confronti tra oggetti

```
Rectangle scatola = new Rectangle (5, 10, 20, 30);
Rectangle r = scatola;
Rectangle confezione = new Rectangle (5, 10, 20, 30);
```

`scatola == r` … vero
- Perché le due variabili si riferiscono allo stesso oggetto

`scatola == confezione` … falso
- Perché si riferiscono a due oggetti diversi, anche se hanno contenuti identici

`scatola.equals(confezione)` …vero
- Verifica la corrispondenza dei campi

Confronto con `null`

- Il riferimento a un oggetto può avere il valore speciale `null` quando l'oggetto non è stato ancora creato
- Per verificare se il riferimento è `null` si usa l'operatore di relazione `==`
  - `if (conto == null)` ….
    - Per domandarsi se l'oggetto a cui si riferisce la variabile `conto` è stato instanziato
- Alcuni metodi restituiscono `null` quando non sono in grado di restituire un oggetto valido
- Il riferimento `null` è diverso dalla stringa vuota
- La stringa vuota è una stringa a tutti gli effetti una stringa che non contiene nulla, di lunghezza `0`

Confronti tra valori in virgola mobile

- Attenzione va posta anche per il confronto di uguaglianza tra valori in virgola mobile (`float` o `double`)
- Raramente si usa l'operatore di uguaglianza (`==`) per confrontare due numeri di tipo `float`
- È meglio considerare se i due valori sono sufficientemente vicini, anche se non identici, a causa delle approssimazioni nella rappresentazione
- Es: Precisione.java

```
final double EPSILON = 1E-14;
if (Math.abs (f1 - f2) < EPSILON)
   System.out.println ("Praticamente uguali.");
```

Istruzione `switch`

- L'istruzione **`switch`** consente di decidere qual'è l'istruzione successiva da eseguire tra diverse
- L'istruzione **`switch`** valuta un'espressione, quindi ne confronta il risultato con i diversi *casi* elencati
- Ogni caso è caratterizzato da un valore e da una lista di istruzioni da eseguire
- Il flusso di controllo è trasferito alla lista associata al primo valore uguale all'espressione

Istruzione `switch` - 2

- La sintassi dell'istruzione *switch*:

```
switch ( espressione )
{
   case valore_1 :
      lista_istruzioni_1
   case valore_2 :
      lista_istruzioni_2
   case valore_3 :
      lista_istruzioni_3
   case ...

}
```

`switch` e **`case`** sono parole riservate

se *espressione* uguaglia *valore_2*, il controllo passa qui

Istruzione `switch` - 3

- Spesso si usa un'istruzione **`break`** come ultima istruzione in ogni lista
- L'istruzione **`break`** passa il controllo alla fine dell'istruzione **`switch`**
- Se non si usa l'istruzione **`break`**, il flusso di controllo continua ai casi successivi senza controllare l'espressione ulteriormente
- Qualche volta questo è utile, ma di solito i valori dei diversi casi sono mutualmente esclusivi e al più un caso corrisponde al valore dell'espressione

Istruzione `switch` - 4

- Un'istruzione **`switch`** può avere un caso di default
- Il caso di default non ha un valore associato ma usa semplicemente la parola riservata **`default`**
- Se è presente il caso di default, il controllo è trasferito all'istruzione associata
- Solitamente il caso di default è messo alla fine dell'istruzione **`switch`**
- Se non è presente il caso di **`default`** e nessun valore corrisponde, il controllo passa all'istruzione successiva all'istruzione **`switch`**

Istruzione `switch` – 5

- L'espressione di un'istruzione **`switch`** deve produrre un valore di tipo *intero*, cioè un **`int`** o **`char`**; non può essere un valore in virgola mobile
- Notare che la condizione implicita nell'istruzione switch è l'uguaglianza – si cerca di uguagliare il valore di un caso
- Non si possono eseguire condizioni di relazione

Istruzione `switch` - 6

```
int digit;   …
switch (digit) {
  case 1: System.out.println ("uno");
  break;
  case 2: System.out.println ("due");
  break;
  case 3: System.out.println ("tre");
  break;
  default: System.out.println ("errore");
}
```

Istruzione `switch` - 7

Ma non è possibile:

```
String parola;   …
switch (parola) {
  case "uno": System.out.println
  ("1"); break;
  case "due": System.out.println
  ("2"); break;
  …
}
```


Istruzione `switch` – 8

L'enunciato `switch` è eqivalente a diversi `if` innestati

```
int digit;   …
if (digit == 1)
  System.out.println ("uno");
else if (digit == 2)
  System.out.println ("due");
else if (digit == 3)
  System.out.println ("tre");
else System.out.println ("errore");
```


Esempi con istruzione switch

- Enuciato_switch.java
- GradeReport.java
  - Esempi con uso dell'istruzione `break` in ogni caso
- Conta_giorni.java
  - Esempio senza l'uso dell'istruzione `break` in ciascun caso