Fondamenti di Informatica

Lezione 3

Danilo Amendola - daniloamendola.github.io

15 marzo 2018

University of Trieste

Riepilogo: Il primo programma di sempre

HelloWorld.java:

```
import csfpackages.hellopackage;
//Hello World in Java
class HelloWorld {
  public static void main( String args[] ) {
    System.out.println( "Hello World!" );
  } /* main closed */   } /* class closed */
```

Per eseguire il programma HelloWorld.java:

- edit HelloWorld.java
- javac HelloWorld.java
- java HelloWorld

NOTA: **import csfpackages.hellopackage;** $\rightarrow$ indica il package in cui posizioniamo la classe.

Curiosità: su sito helloworldcollection trovate il programma HelloWorld scitto in oltre 556 (ad oggi) linguaggi diversi.

Blocco di comandi, dichiarazioni e visibilità

Spesso abbiamo l'esigenza di eseguire un blocco di comandi che possono essere raggruppati logicamente. Per raggruppare le istruzioni in blocco le racchiudiamo tra $\{istruzioni\}$

```
int a = 2, b =4, c;
{c = a + b;   b = c - a;}
```

Se un blocco contiene anche delle dichiarazini, la visibilità delle variabili dichiarate è propagata solo ai blocchi più interni, ma non viceversa;

- un blocco figlio vede tutte le varibili del blocco padre,
- un blocco padre non vede le varibili dichiarate nel blocco figlio.

```
{
  int a = 2, b =4;
  {
    c = 3, d = 5;
    c = a + d; //permesso!
  }

  a = b; //permesso;
  c = a; //non permesso! errore!
}
```

Errori

Scrivendo un programma si possono introdurre errori che impediscono la successiva fase di interpretazione del calcolatore. Gli errori possono essere:

- **Sintattici**. Sono errori che riguardano l'ortografia del testo, es. la mancanza di un ;, l'uso errato di costrutti, una parola chiave scritta male;
- **Errori a run time**. Sono gli errori che si presentano in fase di esecuzione del programma (non si notano in fase di compilazione) e che sono legati ai dati forniti (o mancanti) dall'utilizzatore del programma. Sono errori insidiosi perchè possono presentarsi saltuariamente e apparentemente senza una logica.
- **Errori logici**. Sono gli errori più difficili da scoprire perchè presentano una apparente dicotomia tra la sintassi (quello che ci sembra di aver scritto) e la semantica (quello che vorremo che il programma facesse). Questi errori richiedono per essere scoperti una conoscenza approfondita del linguaggio di programmazione.

I primi esempi: stampa a schermo (print)

Scrivi un programma che stampa la prima lettera del tuo nome compendola utilizzando caratteri disposti nel modo appropriato.

- System.out.print(): stampa il messaggio;
- System.out.println(); stampa il messaggio e va su una nuova linea.

```
public class Dcentrata{
public static void main(String[] args)
{
System.out.println("DDDD");
System.out.println("DDDDDD");
System.out.println("DD  DDD");
System.out.println("DD   DDD");
System.out.println("DD   DDD");
System.out.println("DD   DDD");
System.out.println("DD   DDD");
System.out.println("DD  DDD");
System.out.println("DDDDDD");
System.out.println("DDDD");
}
}
```

Variabili / Costanti

Utilizzando la parola chiave **final** possiamo dichiarare una variabile costante. Le varibili costanti non possono essere poi modificate durante l'esecuzione del programma.

Programma esempio:

```
class Costanti {
  public static void main(String args []) {
    double celsius = 18.0, fahrenheit; //inizializzare sempre!
    final int costante = 32;
    fahrenheit = (celsius * 9/5.0) + costante; //divisione reale
    System.out.println(fahrenheit);
  }
}
```

Descrizione del programma:

- Dichiarazione e inizializzazione di una variabile reale e dichiarazione di una variabile reale.
- Dichiarazione di una costante intera.
- Assegnamento ad una variabile reale del valore di una espressione reale.
- Stampa del valore di una variabile reale.

Esempio: Area del Cerchio

```
class AreaCerchio {
  public static void main(String args []) {
    double raggio = 3.5;
    double area = raggio * raggio * 3.14; // 3.14 e' una costante

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

Descrizione del programma *AreaCerchio*:

- Dichiarazione di una variabile reale con inizializzaizone ad un valore costante;
- Dichiarazione ed assegnamento ad una variabile per calcolare il valore di una espressione reale;
- Stampa di una variabile a schermo.

Esempio: Media tra numeri

```
class Media {
  public static void main(String args []) {
    int a = 1, b = 3, c = 9, d = 20;
    int media = 0; /* Inizialmente media viene posta a 0 */
    media = (a + b + c + d) / 4;

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

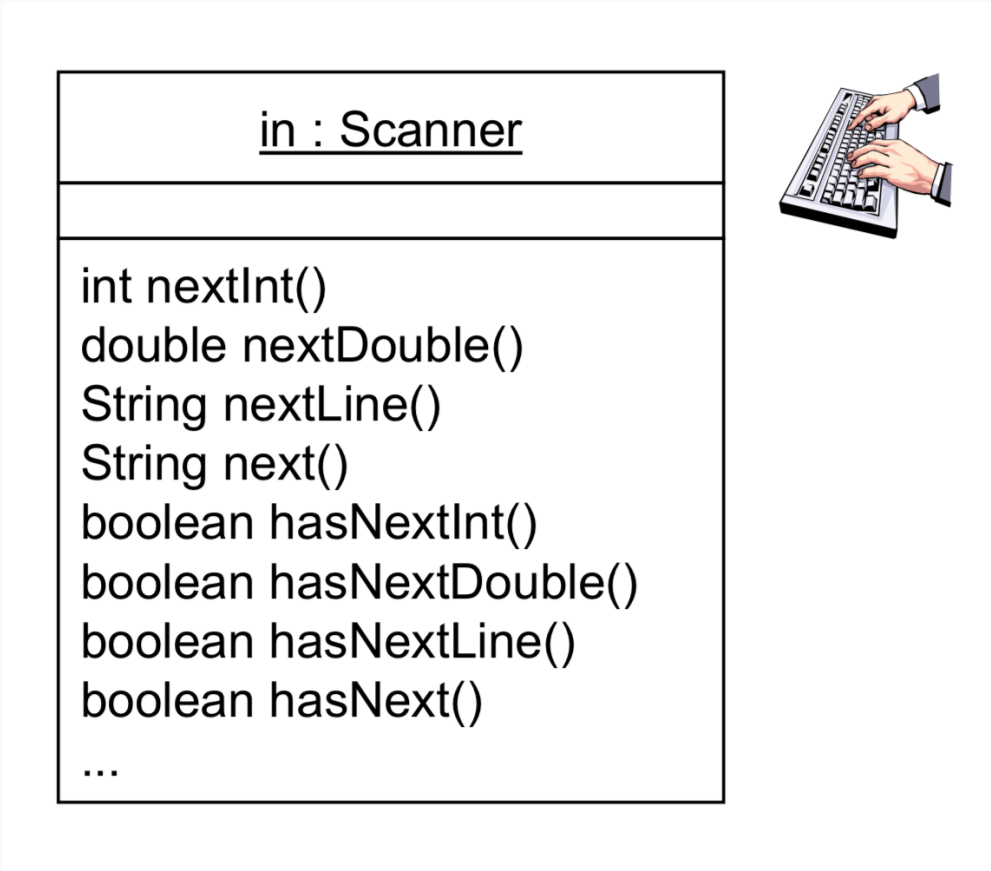
Descrizione del programma *Media*:

- Dichiarazione di quattro variabile con inizializzaizone ad un valore costante;
- Dichiarazione ed assegnamento ad una variabile per calcolare il valore della media;
- Stampa della variabile a schermo.

Un oggetto per la lettura da tastiera i

Java ha un oggetto nelle API () che rappresenta la tastiera: **System.in** [più complicato da usare].

Per semplificare la scrittura dei programmi è stato introdotto l'oggetto **Scanner** nel package java.util.



Un oggetto per la lettura da tastiera ii

Per usare l'oggetto Scanner bisogna importarlo usnado la clausola **import java.util.Scanner**, o con **import java.util.***

Quindi creare l'oggeto con **in = new Scanner(System.in);**

Ed ecco quindi alcune operazioni di Scanner:

- **int nextINt()**: legge un intero da tastiera e lo ritorna;
- **double nextDouble()**: legge un double da tastiera e lo ritorna;
- **String nextLine()**: lege una linea di testo e la restituisce;
- **String next()**: legge un toket, i.e. sequenza distinta da sepratori (e.g. _ -.,);
- **boolean hasNextInt() — booblean hasNextDouble()**:verifica se in input è disponibile una ulteriore intero—double;
- **boolean hasNextLine() — boolean hasNext()**: verifica se in input è disponibile un ulteriore linea—tocken.

Esempio: Lettura di valori da tastiera con Scanner

```
import java.util.*;
class Lettura{
public static void main(String args []) {

Scanner tastiera = new Scanner(System.in);

System.out.print("Nome: ");
String nome = tastiera.next();

System.out.print("Eta': ");
int eta = tastiera.nextInt();

System.out.print(nome);
System.out.println(eta');
}
}
```

Descrizione del programma *Lettura*:

- Creazione di un canale di lettura da tastiera;
- Scrittura di una stringa costante;
- Lettura da tastiera di una stringa e di un intero;
- Scrittura della stringa e dell'elemento intero letto.

Esempio di Lettura da testiera con BufferReader (System.in)

Un altro modo per leggere da tastiera:

"In general, each read request made of a Reader causes a corresponding read request to be made of the underlying character or byte stream." [BufferReader in docs.oracle.com]

```
import java.io.*;
public class Input {
  public static void main(String[] args) throws IOException {

    BufferedReader in = new BufferedReader(new InputStreamReader(
   System.in));

    System.out.print("Inserisci a: ");
    String linea1 = in.readLine();

    System.out.print("Inserisci b: ");
    String linea2 = in.readLine();

    int a = Integer.parseInt(linea1);
    int b = Integer.parseInt(linea2);

    System.out.println( a * b );
  }
}
```

Istruzioni per il controllo del flusso

I costrutti della programmazione che utilizzeremo per modificare il flusso del programma sono:

- **if**(condizione) {espressione};
- **if**(condizione) {espressione vera}; **else** {espressione falsa};
- **while**(condizione) {espressione ripetuta se condizione vera};
- **do** {espressione ripetura se condizione vera} **while**(condizione);
- **for**(init;condizione;increm) {espressione ripetuta se condizione vera};
- **switch**(valore) {caso eseguito se verificato in valrore};
- **break**; $\rightarrow$ interrompe l'esecuzione ciclica di comandi in corso;
- **continue**; $\rightarrow$ salta i comandi che la seguono in un ciclo.

Esempio: Confronti tra numeri

```
class Confronti {
  public static void main(String args []) {
    int a = -20, b = 30, c = 0, d = 17;
    boolean primo = a > b;
    boolean secondo = c < d;

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

Descrizione del programma *Confronti*:

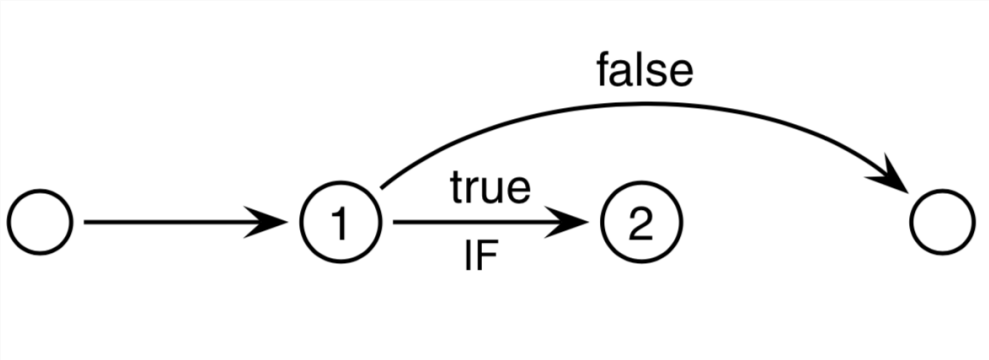
- Dichiarazione di quattro variabile con inizializzaizone ad un valore costante;
- Dichiarazione ed assegnamento ad una variabile boolean per calcolare l'espressione "a $>$ b";
- Dichiarazione ed assegnamento ad una variabile boolean per calcolare l'espressione "c $<$ d";
- Stampa delle variabile risultanti dai confronti a schermo.

L'istruzione IF i

L'istruzione **if** ha la sintassi:

```
if ( condizione ) {
  // istruzioni da eseguire se la condizione e' vera
}
```

Istruzione IF:



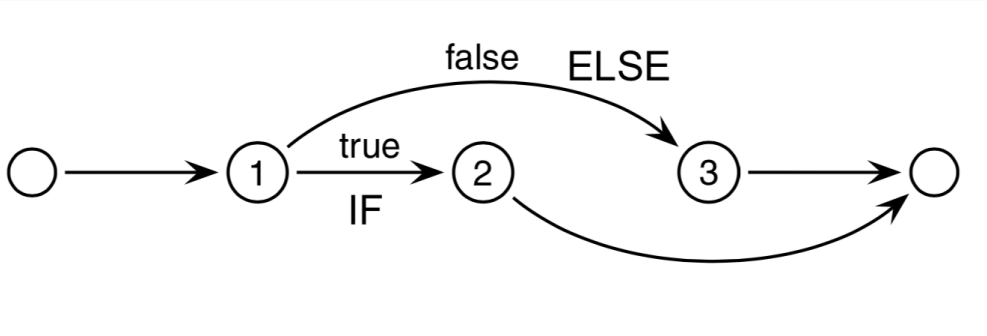
L'istruzione IF ii

L'istruzione IF..ELSE:

```
if ( condizione ) {
  // istruzioni da eseguire se la condizione e' vera
} else {
  // istruzioni da eseguire se la condizione e' false
}
```

n.b. la parte "**else** { **...** }" è opzionale

Istruzione IF..ELSE:



L'istruzione IF iii

Un esempio di istruzione IF:

```
if ( k <= max ) {
  System.out.println( k + " e' minore o uguale a " +  max);
} else {
  System.out.println( k + " e' maggiore di " + max);
}
```

Blocco IF..ELSE che confronta k con max e stampa un messaggio opportuno se $k \leq max$, minore o uguale a max o in alternativa se k è maggiore di max.

If..else annidati

Nell'esempio vediamo due if..else annidati, notare l'indentazione:

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

    int a = 4;
    final int b = 7;

    if (a > b)
      System.out.println("a > b");
    else
      if (a < b)
        System.out.println("a < b");
      else
        System.out.println("a = b");
  }
}
```

Possiamo avere if annidati sia sul primo che sul secondo ramo dell'If..Else.

Esercizi sugli IF

Provate a svolgere questi esercizi. La soluzione vi verrà fornita sul sito.

Esercizio 1) Scrivere un programma che dati in input due numeri interi determina quale dei due ha il valore più alto.

Esercizio 2) Scrivere un programma che preso in input un numero, ne calcola il resto della divisione col numero 4:

- se il resto è 0 scrive "il resto è zero";
- se il resto è 1 scrive "il resto è uno";
- se il resto è 2 scrive "il resto è due";
- se il resto è 3 scrive "il resto è tre".

L'istruzione WHILE

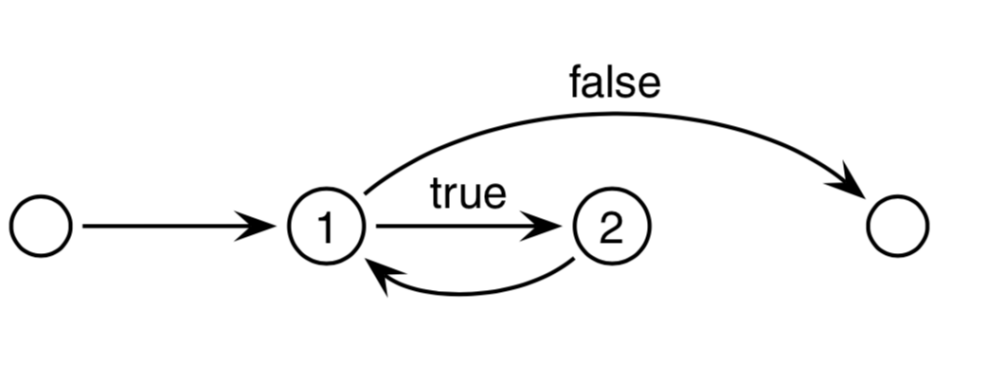
Sintassi del WHILE

```
while (condizione) {
// { blocco di istruzioni da eseguire ciclicamente }
}
```

Un esempio di ciclo WHILE:

```
int i = 0;
while (i < 10) {
System.out.println(i);
i = i + 1;
}
```

Istruzione WHILE:



L'istruzione DO WHILE

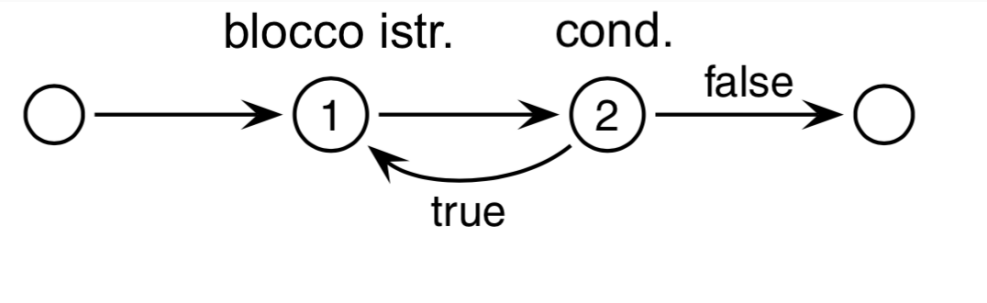
Sintassi del DO..WHILE

```
do {
  // { blocco di istruzioni da eseguire ciclicamente (almeno 1 volta
     ) }
} while (condizione);
```

Un esempio di ciclo DO..WHILE:

```
int i = 0;
do {
  System.out.println(i);
  i = i + 1;
} while (i < 10);
```

Istruzione DO..WHILE:



DO..WHILE vs WHILE

In cosa differiscono quindi DO..WHILE e WHILE:

```
public class test{
  public static void main(String args[]){
    System.out.println("DO..WHILE");
    int i = 0;
    do{                              // Proviamo con il DO..WHILE
      System.out.println( + i);
      i = i + 1;
    } while (i < 0);

    System.out.println("WHILE");
    i = 0;
    while (i < 0) {                  // Ora proviamo con il WHILE
      System.out.println(i);
      i = i + 1;
    } } }
```

Risultato:

```
DO..WHILE
0
WHILE
-
```

ATTENZIONE: Il DO..WHILE esegue sempre il blocco istruzioni al primo ciclo!

Esempio di ciclo WHILE

Estendiamo il programma *Media*, in *MediaWhile*, il programma chiederà prima quante volte vogliamo calcolare la media tra 4 numeri, quindi leggerà da tastiera ciclicamente i numeri e stamperà a schermo la media risultante.

```
import java.util.*;
class MediaWhile {
  public static void main(String args []) {
    Scanner tastiera = new Scanner(System.in);
    System.out.print("Quante medie tra 4 numeri vuoi calcolare? ");
    int num_medie = tastiera.nextInt();
    int i = 0;
    int a, b, c, d;
    while( i < num_medie){
      System.out.print("Dammi 4 numeri interi: ");
      a = tastiera.nextInt();
      b = tastiera.nextInt();
      c = tastiera.nextInt();
      d = tastiera.nextInt();
      int media = 0; /* Inizialmente media e' posta a 0 */
      media = (a + b + c + d) / 4;
      System.out.println("La media e' " + media);
      i++;
    } } }
```

Esercizi proposti su WHILE e DO..WHILE

1) Scrivere un programma che prende in ingresso un numero e lo stampa; quindi ricomincia; il programma termina quando riceve in ingresso -1.

2) Scrivere un programma che stampa un triangolo isoscele formato da lettere "T"; il triandolo deve essere formato da 5 righe.

3) Scrivere un programma che calcola i primi 10 numeri di Fibonacci.

L'istruzione FOR i

L'istruzione for:

```
for (inizializzazione; controllo; incremento) {
  // Esegui un blocco di codice per un certo numero di volte
}
```

Una classe che usa un for per contare fino ad un certo numero:

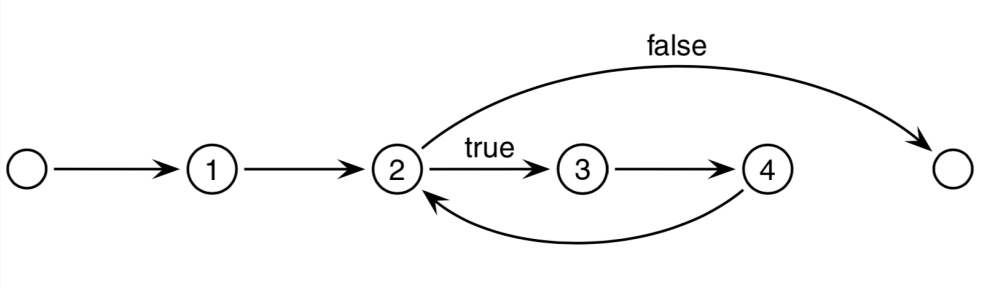
```
class For{
  public static void main(String args []) {
    final int limite = 5;
    for (int conta = 1; conta < limite; conta++)
      System.out.println(conta);
  }
}
```

L'istruzione FOR ii

Il costrutto for può essere sintetizzato così:

```
for ( int i = 0 ; i < 10 ; i++ )
  System.out.println(i);
```

Istruzione FOR:



Esempio con cicli FOR annidati

```
for (int i = 1; i <= 4 ; i++) {
  for (int j = 1; j <= 6; j++) {
    System.out.print("*");
  }
  System.out.println();
}
```

Cosa produce in uscita questo blocco di codice?

Esempi cicli WHILE vs FOR

I due esempi rappresentano due programmi che eseguono un ciclo che stampa a schermo i numeri da 0 a 10, riga per riga.

Esempio ciclo WHILE:

```
int i = 0;
while (i < 10) {
  System.out.println(i);
  i++;
}
```

Esempio cilo FOR:

```
for (int i = 0; i < 10; i++) {
  System.out.println(i);
}
```

Esercizi proposti sul ciclo FOR

1) Convertire i programmi scritti con il while utilizzando il for.

2) Scrivere un programma che visualizzi i numeri multipli di 5 da 25 a 155 (in ordine crescente e decrescente).

3) Dati in ingresso 10 numeri da tastiera, restituisce il valore più grande e il più piccolo.

4) Scrivere un programma che dato un numero letto da tastiera, determina se è primo o no (un numero si dice primo se è divisibile solo per 1 e per se stesso)

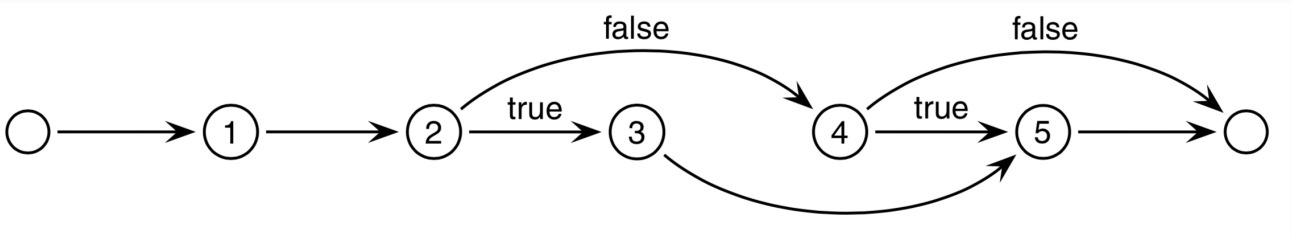
L'istruzione SWITCH

L'istruzione Switch consente di specificare un certo numero di casi, o il caso di *default* quando nessuno dei casi specificati si verifichi.

Sintassi dello Switch:

```
switch (a) {
case 1:
// fai qualcosa
break;
case 2:
// fai qualcosa
break;
default:
// fai qualcosa
break;
}
```

Istruzione SWITCH:



Esercizio sullo SWITCH

Provate a svolgere questi esercizi. La soluzione vi verrà fornita sul sito.

Esercizio 1) Scrivere un programma (usando SWITCH) che dati in input due numeri interi determina quale dei due ha il valore più alto.

Esercizio 2) Scrivere un programma (usando lo SWITCH) che preso in input un numero, ne calcola il resto della divisione col numero 4:

- se il resto è 0 scrive "il resto è zero";
- se il resto è 1 scrive "il resto è uno";
- se il resto è 2 scrive "il resto è due";
- se il resto è 3 scrive "il resto è tre".

Esempio sul: break

Costruttori di branching: Break e Continue.

Break: termina l'esecuzione di uno o più blocchi di codice.

L'istruzione break è cruciale in combinazione con lo quella di switch, consentendo di partizionare i casi che condivino il blocco i istruzioni:

```
class Break {
  public static void main(String args []) {
    int m = 3, g = 0;
    switch(m) {
      case 1: case 3: case 5: case 7: case 8: case 10: case 12:
        System.out.println(31);
        break;
      case 2: System.out.println(28);
      break;
      case 4: case 6: case 9: case 11:
        System.out.println(30);
        break;
      default:
        System.out.println("Errore");
    }
  }
}
```

Esempio sul comando: continue

Costruttori di branching: Break e Continue.

Continue: salta all'istruzione di controllo della condizione, quindi salta solo l'iterazione corrente.

Non si usa con lo switch ma con le istruzioni For e While.

```
class Continue{
  public static void main(String args []) {
    final int limite = 10;
    for (int conta = 1; conta < limite; conta++) {
      if (conta == 5)
        continue;
      System.out.println(conta);
    }
  }
}
```

Cicli annidati: Tabella pitagorica

Il programma stampa a schermo la Tavola Pitagorica:

```
class TabellaPitagorica{
  public static void main (String args[]){
    System.out.prinln("Tavola Pitagorica:");
    int prod = 0;
    //Il programma cicla con due for innestati
    for (int riga=1; riga<=10; riga++){
      for (int colonna=1; colonna<=10; colonna++){
        prod = riga * colonna;
        System.out.print(prod + "\t");
      }
      System.out.print("\n"); // avremmmo potuto usare println
     semplicemente
    }
  }
}
```

FINE

Riferimenti

Riferimenti:

- Laboratorio di Programmazione - Cristian del Fabbro;
- Elementi di programmazione in Java, Rosatio Culmoneù;
- Fondamenti di Informatica, Luca Cabibbo;
- Elementi di programmazione in Java, R.Culmone;