Metodologie di Programmazione

Lezione 10: Strutture di controllo e costrutti iterativi

Lezione 10: Sommario

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Strutture di controllo:
  - if
  - else
  - Il problema dell'else sospeso
  - Operatore di selezione ?
  - Istruzione switch
- Costrutti iterativi:
  - while
  - do…while
  - for

Strutture di controllo


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

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

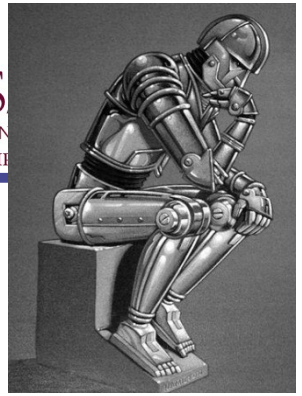
```
public static void main(String[] args)
{
      int x = 5;
      int y = 7;

      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 strutture di controllo **condizionali**
  - Alcune istruzioni possono essere o non essere eseguite sulla base di certe condizioni
- Mediante le strutture 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

```
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

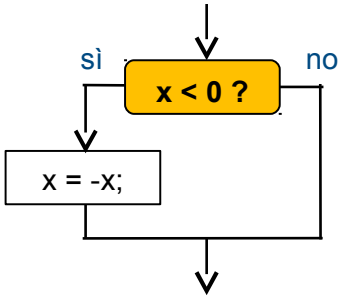
- Esempio:

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

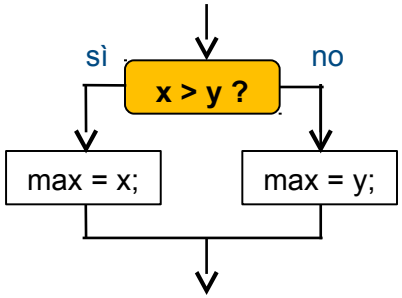
Enunciati if ed else con diagrammi di flusso

- 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;
```



Alcuni esempi di utilizzo di if ed else

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;`
controlla l'errore 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: estrazione del carattere centrale

- 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: estrazione del carattere centrale

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

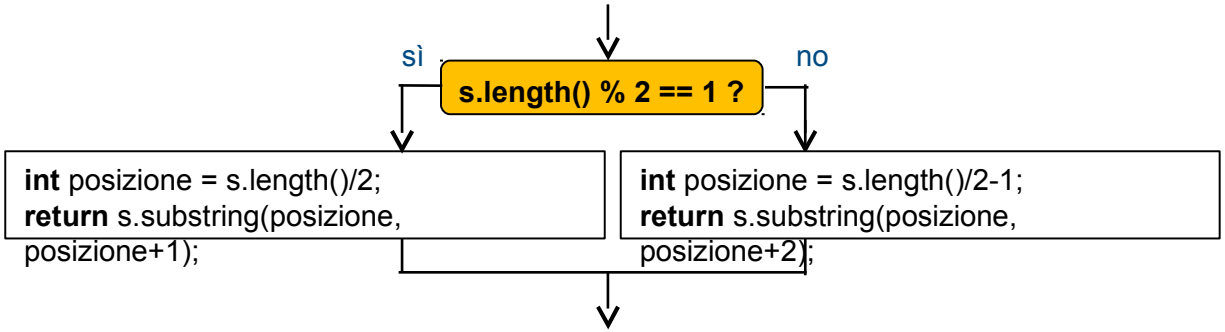
```
int posizione = s.length()/2;
return s.substring(posizione, posizione+1);
```

- 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);
```

Esercizio: estrazione del carattere centrale

- Progetta il flusso di controllo:



- 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: estrazione del carattere centrale

- Fase 5: 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

```
/**
 * Descrive gli effetti di un terremoto
 * @author navigli
 */
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 (tutti gli altri valori)

Altro modo di generare numeri casuali

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;
        }
    }
}
```

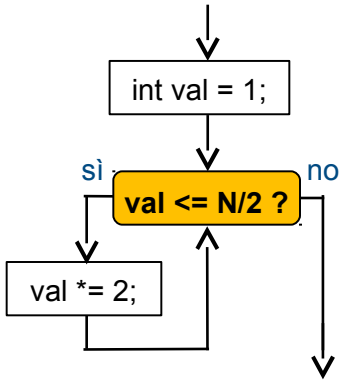
Nota che l'intero N è definito costante (final)

- 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!

Alcuni esempi di utilizzo dell'istruzione for

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 $H_N = 1+1/2+…+1/N$	`double somma = 0.0;` `for (int k = 1; k <= N; k++) somma += 1.0/k;`

Altri esempi di utilizzo dell'istruzione for

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, 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 uno schema a 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();
}
```

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: conta parola

- 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

```
public int contaParola(String testo, String w)
{
    int cont = 0;
    String[] tokens = testo.split(" ");

    for (int k = 0; k < tokens.length; k++)
        if (tokens[k].equals(w)) cont++;

    return cont;
}
```

Esercizio: stringa verticale

- Scrivere un metodo che legge una stringa da console e la stampa in verticale un carattere per linea
- Ad esempio, dato in input "ciao", viene stampato:

```
c
i
a
o
```

Esercizio: stringhe verticali

- 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: conta vocali

- 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: somma dei due numeri precedenti

- 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: conversione dei numeri

- 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: frase palindroma

- 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: la classe Cornice

- Progettare una classe Cornice che rappresenti una cornice NxN, con eventuale stringa contenuta al suo interno
- La cornice deve disporre del metodo toString() che ne restituisce la rappresentazione in formato stringa
- Ad esempio: new Cornice(6, "Cornici in Java").toString() restituisce la seguente stringa

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

Esercizio: terne pitagoriche

- 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: da cifre a lettere e viceversa

- 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"

Esercizio: stampa triangoli

- Scrivere un metodo che, dato un intero positivo dispari N, stampi un triangolo isoscele la cui base è costituita da N caratteri
- 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

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

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);
}
```

Da evitare: 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 (usa l'operatore +=)

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

Da evitare: spreco di 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: creare 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: varibili 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!

UNITELMA SAPIENZA

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

```
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];

}
```

```
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: usa double invece di float

```
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!!!

```java
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 void main(){
        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

Da evitare: chiamate identiche e multiple a un metodo costoso

```
String s = "frase da dividere in parole";
for (int k = 0; k < s.split(" ").length; k++)
    System.out.println(s.split(" ")[k]);
```

Ripete la stessa operazione di split tante volte quante sono le parole nella stringa!!!

E la ripete di nuovo per ogni iterazione!

```
String s = "frase da dividere in parole";
String[] a = s.split(" ");
for (int k = 0; k < a.length; k++)
    System.out.println(a[k]);
```