Tecniche iterative

Walter Didimo

Tecniche iterative

In questa lezione svolgeremo alcuni esercizi di definizione di classi con metodi che usano

- istruzioni condizionali
- istruzioni iterative

Gli esercizi svolti serviranno tra l'altro ad illustrare alcune classiche tecniche iterative

La classe NumeroNaturale

Vogliamo realizzare una classe *NumeroNaturale*, in base alle seguenti specifiche:

- un oggetto *NumeroNaturale* rappresenta un numero naturale
- la classe *NumeroNaturale* ha un costruttore che consente di creare un oggetto che rappresenta un numero naturale specificato
- la classe *NumeroNaturale* avrà un certo insieme di metodi di istanza che serviranno a verificare se il numero rappresentato ha certe proprietà

Classe NumeroNaturale: lo scheletro

```
class NumeroNaturale {

        private int numero; // il numero rappresentato

        /* Costruttore: crea un NumeroNaturale che
           rappresenta il numero specificato */
        public NumeroNaturale (int num) {…}

        /* Restituisce true se il numero è primo */
        public boolean isPrimo () {…}

        /* Restituisce true se il numero è perfetto */
        public boolean isPerfetto () {…}

        /* Visualizza tutti i divisori interi
         sulla finestra di output specificata */
        public void stampaDivisori (OutputWindow out) {…}
}
```

Classe NumeroNaturale: costruttore

```
class NumeroNaturale{

        private int numero; // il numero rappresentato

        /* Costruttore: crea un NumeroNaturale che
           rappresenta il numero specificato */
        public NumeroNaturale (int num) {
                this.numero = num;
        }
        ……
}
```

Classe NumeroNaturale: isPrimo

Prima soluzione – uso di un ciclo *for*

```
public boolean isPrimo (){
        boolean primo;
        if (this.numero > 1){
                primo = true;
                for (int i=2; i<this.numero; i++)
                        if (this.numero%i == 0)
                                primo = false;
        }
        else
                primo = false;
        return primo;
}
```

Scandisce tutti i potenziali divisori fino al numero stesso (escluso)

Il ciclo non si arresta se trova un divisore

Classe NumeroNaturale: isPrimo

Seconda soluzione – uso di un ciclo *for* "più breve"

```
public boolean isPrimo (){
        boolean primo;
        if (this.numero > 1){
                primo = true;
                int radice = (int)Math.sqrt(this.numero);
                for (int i=2; i<=radice; i++)
                        if (this.numero%i == 0)
                                primo = false;
        }
        else
                primo = false;
        return primo;
}
```

Scandisce i potenziali divisori fino alla radice quadrata del numero

Il ciclo non si arresta se trova un divisore

Classe NumeroNaturale: isPrimo

Terza soluzione – uso di un ciclo *while*

```
public boolean isPrimo (){
        boolean primo;
        if (this.numero > 1){
                primo = true;
                int radice = (int)Math.sqrt(this.numero);
                int i = 2;
                while (primo && i<=radice){
                        if (this.numero%i == 0)
                                primo = false;
                        i++;
                }
        }
        else
                primo = false;
        return primo;
}
```

Scandisce i potenziali divisori fino alla radice quadrata del numero

Il ciclo si arresta appena trova un divisore

Classe NumeroNaturale: isPerfetto

Un numero è perfetto se è pari alla somma dei suoi divisori interi (escluso ovviamente il numero stesso)

I primi numeri perfetti sono: 6, 28, 496, …

Classe NumeroNaturale: isPerfetto

Prima soluzione – uso di un ciclo *for*

```
public boolean isPerfetto (){
        boolean perfetto;
        int somma = 0;

        for (int i=1; i<=this.numero/2; i++)
                if (this.numero%i == 0)
                        somma += i;

        if (somma == this.numero)
                perfetto = true;
        else
                perfetto = false;
        return perfetto;
}
```

Scandisce tutti i potenziali divisori interi e ne calcola la somma

Classe NumeroNaturale: isPerfetto

Seconda soluzione – uso di un ciclo *while*

```
public boolean isPerfetto (){
        boolean perfetto;
        int somma = 0;
        int i=1;
        while (somma<=this.numero && i<=this.numero/2){
                if (this.numero%i == 0)
                        somma += i;
                i++;
        }

        if (somma == this.numero)
                perfetto = true;
        else
                perfetto = false;
        return perfetto;
}
```

Scandisce tutti i potenziali divisori interi e ne calcola la somma

Il ciclo di arresta se la somma supera il numero

Classe NumeroNaturale: stampaDivisori

In questo caso è necessario guardare tutti i potenziali divisori, senza ulteriori condizioni di arresto

```
public void stampaDivisori (OutputWindow out){
       for (int i=1; i<=this.numero/2; i++)
              if (this.numero%i==0)
                     out.writeln (i);
}
```

Uso della classe NumeroNaturale

Vediamo ora una classe, *ProvaNumeroNaturale*, che usa le funzionalità della classe *NumeroNaturale*

Il metodo *main* della classe *ProvaNumeroNaturale* svolge queste azioni:

- fa inserire all'utente un numero naturale *n*
- dice all'utente se *n* è primo o meno
- visualizza all'utente tutti i numeri perfetti fino ad *n*
- visualizza all'utente tutti i divisori interi di *n*

La classe ProvaNumeroNaturale

```
class ProvaNumeroNaturale{

   public static void main (String[] args){

      /* Acquisice un numero naturale e crea un
         oggetto NumeroNaturale */

      InputWindow in = new InputWindow ();
      int num = in.readInt("Inserisci un naturale");
      NumeroNaturale naturale = new NumeroNaturale(num);


      /* Dice se il numero è primo */

      OutputWindow out = new OutputWindow ();
      if (naturale.isPrimo ())
          out.writeln ("Il numero è primo");
      else
          out.writeln ("Il numero non è primo");
```

…… *(continua)*

La classe ProvaNumeroNaturale

```
        /* Visualizza i numeri perfetti fino a num */
        out.writeln ("Numeri perfetti fino a " + num + ":");
        for (int i=1; i<=num; i++){
                NumeroNaturale nat = new NumeroNaturale (i);
                if (nat.isPerfetto ())
                        out.write (i + " ");
         }
         /* Visualizza i divisori interi di num */
         out.writeln();
         out.writeln ("Divisori interi di "+ num ":");
                naturale.stampaDivisori (out);
   }// fine main
}// fine classe
```

La classe AnalizzatoreDiStringa

Come ulteriore esercizio, vogliamo ora realizzare una classe *AnalizzatoreDiStringa* in base alle seguenti specifiche

- un oggetto *AnalizzatoreDiStringa* rappresenta una stringa, che gli viene passata all'atto della sua creazione
- un oggetto *AnalizzatoreDiStringa* avrà dei metodi che permettono di ottenere alcune utili informazioni sulla stringa che esso rappresenta

AnalizzatoreDiStringa: scheletro

```
class AnalizzatoreDiStringa{

        /* la stringa rappresentata */
        private String stringa;

        /* costruttore */
        public AnalizzatoreDiStringa (String s){…}

        /* restituisce la frequenza del carattere c */
        public int frequenza (char c){…}

        /* restituisce il numero di vocali nella stringa */
        public int numeroVocali (){…}

        /* restituisce il numero di consonanti nella stringa */
        public int numeroConsonanti (){…}

        /* visualizza le frequenze dei caratteri 'a'--'z',
           nella finestra specificata*/
        public void stampaFrequenzeAZ (OutputWindow out)
}
```

AnalizzatoreDiStringa: costruttore

```
/* costruttore */
public AnalizzatoreDiStringa (String s){
      this.stringa = s;
}
```

Il costruttore deve solo copiare il riferimento alla stringa passatagli come parametro nella variabile di istanza dell'oggetto

AnalizzatoreDiStringa: frequenze

```
/* restituisce la frequenza di un carattere */
public int frequenza (char c){
      int f = 0;
      for (int i=0; i<this.stringa.length(); i++){
            char ch = this.stringa.charAt(i);
            if (ch==c)
                  f++;
      }
      return f;
}
```

Nel ciclo viene calcolata la frequenza del carattere c, e viene memorizzata nella variabile f

AnalizzatoreDiStringa: numeroVocali

```
/* restituisce il numero di vocali nella stringa */
public int numeroVocali (){
       int vocali = 0;
       String temp = this.stringa.toUpperCase();
       for (int i=0; i<temp.length(); i++){
              char ch = temp.charAt(i);
              if ((ch == 'A') ||
                  (ch == 'E') ||
                  (ch == 'I') ||
                  (ch == 'O') ||
                  (ch == 'U'))
                    vocali++;
       }
       return vocali;
}
```

Converte tutti i caratteri in maiuscolo, per non dover distinguere tra minuscole e maiuscole

AnalizzatoreDiStringa: numeroConsonanti

```
/* restituisce il numero di consonanti nella stringa */
public int numeroConsonanti(){
       int consonanti = 0;
       String temp = this.stringa.toUpperCase();
       for (int i=0; i<temp.length(); i++){
              char ch = temp.charAt(i);
              if ((ch>=66 && ch<=68) ||
                  (ch>=70 && ch<=72) ||
                  (ch>=74 && ch<=78) ||
                  (ch>=80 && ch<=84) ||
                  (ch>=86 && ch<=90))
                     consonanti++;
       }
       return consonanti;
}
```

Il confronto è fatto sulle posizioni occupate dalle consonanti nell'alfabeto Unicode 2.0

AnalizzatoreDiStringa: stampaFrequenzeAZ

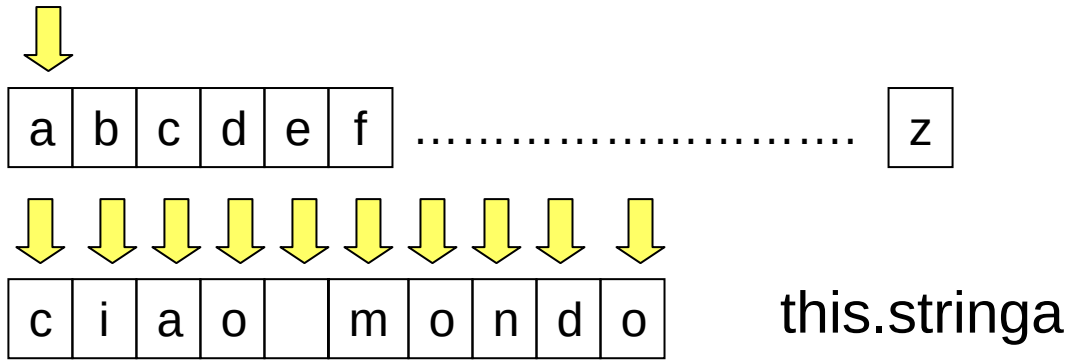
```
/* visualizza le frequenze dei caratteri 'a'--'z', nella
finestra specificata*/

public void stampaFrequenzeAZ (OutputWindow out){
      for (char c='a'; c<='z'; c++){
            int f = 0;   // frequenza del carattere c
            for (int i=0; i<this.stringa.length(); i++){
                  char ch = this.stringa.charAt(i);
                  if (ch==c)
                        f++;
            }
            out.writeln ("frequenza di " + c + " = " + f);
      }
}
```

for annidati: il più esterno scandisce i caratteri da ‘a’ a ‘z’; il più interno scandisce i caratteri della stringa (il *for* più interno è eseguito ad ogni nuovo ciclo del *for* esterno)

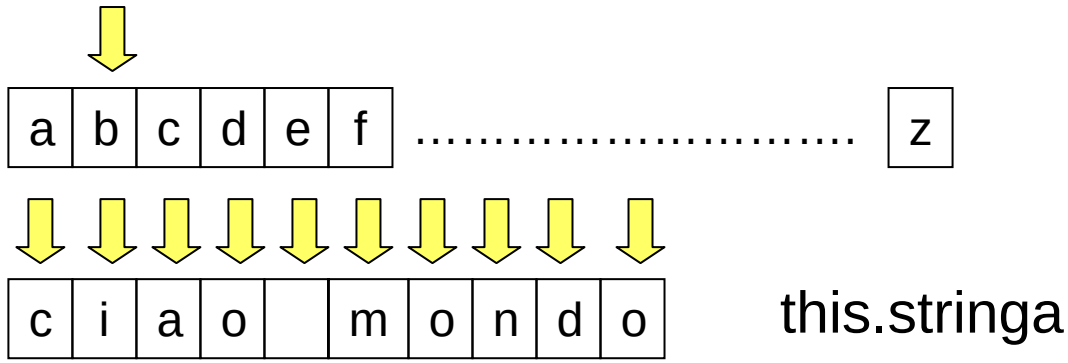
Cicli for annidati: un esempio

```
for (char c='a'; c<='z'; c++){
        int f = 0;    // frequenza del carattere c
        for (int i=0; i<this.stringa.length(); i++){
              char ch = this.stringa.charAt(i);
              if (ch==c)
                    f++;
        }
}
```



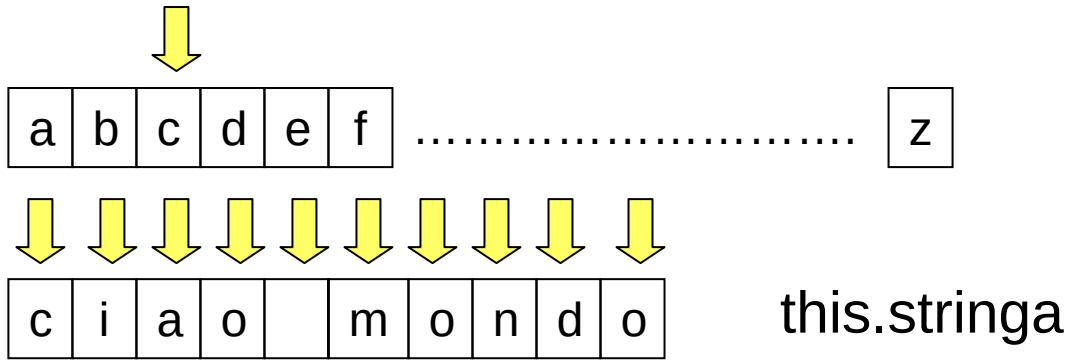
Cicli for annidati: un esempio

```
for (char c='a'; c<='z'; c++){
        int f = 0;    // frequenza del carattere c
        for (int i=0; i<this.stringa.length(); i++){
              char ch = this.stringa.charAt(i);
              if (ch==c)
                    f++;
        }
}
```



Cicli for annidati: un esempio

```
for (char c='a'; c<='z'; c++){
        int f = 0;    // frequenza del carattere c
        for (int i=0; i<this.stringa.length(); i++){
              char ch = this.stringa.charAt(i);
              if (ch==c)
                    f++;
        }
}
```



AnalizzatoreDiStringa: stampaFrequenzeAZ

```
/* visualizza le frequenze dei caratteri 'a'--'z', nella
finestra specificata – soluzione più elegante */

public void stampaFrequenzeAZ (OutputWindow out){
       for (char c='a'; c<='z'; c++){
         out.writeln ("frequenza di " + c + " = " +
                        this.frequenza(c));
       }
}
```

Viene riusato il metodo frequenze scritto in precedenza (il ciclo del calcolo della frequenza di un carattere è incapsulato nel metodo)

Glossario dei termini principali

Termine	Significato
Numero perfetto	Un numero intero la cui somma dei divisori è uguale al numero stesso
Frequenza di un carattere	Numero di occorrenze di un carattere in una stringa