Singleton: le classi con una sola istanza

Con il termine **Singleton** si indica una modalità di dichiarazione delle classi che ha lo scopo di garantire la creazione di *una e una sola* istanza della classe. La creazione e l'accesso a questa istanza è gestito direttamente dalla classe dichiarata in modalità *Singleton*.
Gli elementi che caratterizzano un *Singleton* sono:
- il costruttore privato, per evitare la creazione di oggetti da classi esterne;
- un metodo statico, per accedere all'unica istanza dell'oggetto.

La struttura generica di una classe dichiarata in modalità *Singleton* è la seguente:

```
class ClasseSingleton
{
   private static ClasseSingleton istanza = null;

   private ClasseSingleton() {}

   public static synchronized ClasseSingleton getIstanza()
   {
      if (istanza == null)
      {
         istanza = new ClasseSingleton();
      }
      return istanza;
   }
}
```

La parola chiave **synchronized**, usata nella dichiarazione del metodo, serve per evitare esecuzioni concorrenti dello stesso metodo e per garantire che la creazione dell'istanza venga eseguita una e una sola volta. Il metodo *getIstanza* restituisce l'unica istanza disponibile. Alla prima esecuzione del metodo l'istanza sarà *null* e, in questo caso, verrà creata usando il costruttore privato.
La dichiarazione di una classe *Singleton* è utile per gestire e mantenere un insieme di informazioni condivise da tutta l'applicazione per l'intera esecuzione del programma, come per esempio le informazioni sulla connessione al database o sulla gestione di un file di log.

PROGETTO

Dichiarare una classe Singleton per la gestione condivisa di un file di log.

Le applicazioni, soprattutto quelle di grandi dimensioni, hanno la necessità di gestire un file di log in cui inserire i messaggi di informazione o di errore. Questi messaggi sono utili sia al programmatore, interessato alla ricerca di malfunzionamenti, che all'utente finale per comprendere lo stato di esecuzione.
La classe *Log* è dichiarata con la modalità *Singleton* in modo che la sua unica istanza possa essere richiesta da tutti i punti del programma.
Il metodo costruttore viene eseguito una sola volta durante tutta l'esecuzione del programma e ha lo scopo di azzerare l'attributo *numRighe*, un contatore per registrare il numero delle righe scritte, e di aprire il file di log.
Il metodo *scrivi* è un metodo di istanza e ha il compito di scrivere il messaggio nel file di log.
Il metodo *chiudi* è usato per chiudere il file.

Per la spiegazione delle classi che gestiscono la scrittura sui file di testo si rimanda al capitolo 5 del libro di testo.

Il codice completo della classe *Log* e un esempio di utilizzo sono riportati di seguito.

IMPLEMENTAZIONE DELLA CLASSE (*Log.java*)

```
import java.io.*;

class Log
{
   private static Log istanza = null;

   private int numRighe;
   private PrintWriter fOUT;

   private Log()
   {
      numRighe = 0;

      // Apre il file di log
      try
      {
         FileWriter f = new FileWriter("log_01.txt");
         fOUT = new PrintWriter(f);
      }
      catch (IOException e)
      {
         System.out.println("Errore nell'apertura del file.");
         System.exit(1);
      }
   }

   public static synchronized Log getIstanza()
   {
      if (istanza == null)
      {
         istanza = new Log();
      }
      return istanza;
   }

   public synchronized void scrivi(String msg)
   {
      // Aggiorna il numero di righe
      numRighe++;

      // Scrive sul file di log
      fOUT.println(msg);
   }
```

```
   public void chiudi()
   {
      fOUT.close();
   }
}
```

PROGRAMMA JAVA (*ProgLog.java*)

```
class ProgLog
{
   public static void main(String args[])
   {
      Log log = Log.getIstanza();

      log.scrivi("Inizio programma.");

      //. . .

      log.scrivi("Fine programma.");
      log.chiudi();
   }
}
```