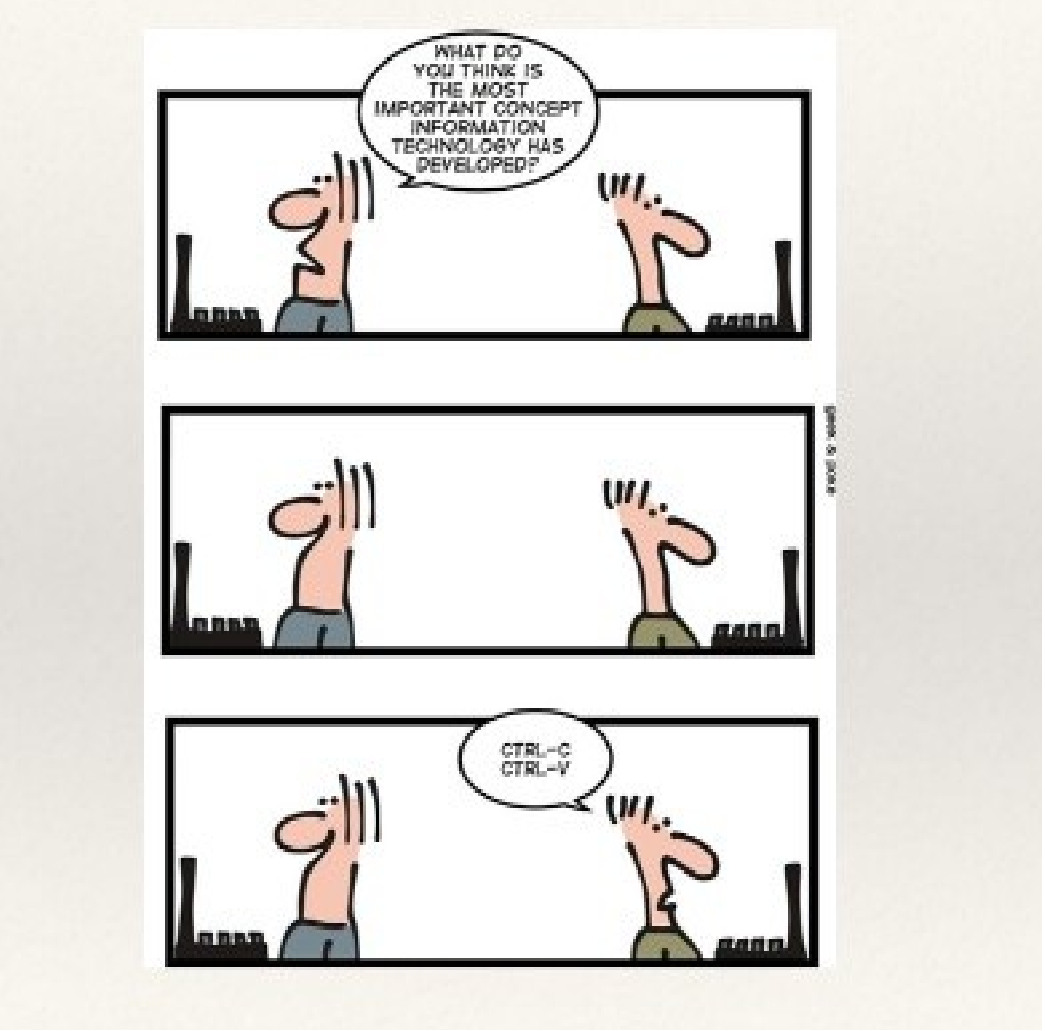
Static e Sigleton

Prof. Viglietti Francesco
www.in4matika.altervista.org



Dichiarazione e uso del metodo static

I metodi dichiarati nelle classi si possono distinguere in:
I **metodi di istanza** sono utilizzabili dalle istanze della classe. In pratica dalla classe si crea un oggetto e, utilizzando l'oggetto, si invoca l'esecuzione dei metodi di istanza. Questi possono fare riferimento agli attributi dell'oggetto, leggendo o modificando il loro valore.
I **metodi di classe** esistono e possono essere utilizzati senza che venga creato un oggetto della classe che li contiene. Questi metodi non possono fare riferimento agli attributi definiti nella classe, a meno che anche questi siano statici. Essi devono essere dichiarati inserendo la parola chiave **static** nell'intestazione del metodo.
L'esecuzione di un metodo statico da altre classi si attiva indicando il nome della classe seguito dal nome del metodo. Per esempio, il calcolo della radice quadrata di un numero, si può eseguire scrivendo il comando Math.sqrt(numero).

esempio

Per apprezzare meglio le differenze tra i metodi di istanza e di classe definiamo la classe Operazione che contiene un metodo somma.

```
class Operazione
{
   private int a, b;

   public Operazione(int a, int b)
   {
      this.a = a;
      this.b = b;
   }

   public int somma()
   {
      return a + b;
   }
}
```

```
class Operazione
{
   public static int somma(int a, int b)
   {
      return a + b;
   }
}
```

Per eseguire la somma di due numeri, si deve usare la classe *Operazione* nel seguente modo.

```
int risultato;

Operazione op = new Operazione(2,5);
risultato = op.somma();
```

```
int risultato;

risultato = Operazione.somma(2,5);
```

Esercizio: *Dichiarare un metodo statico che confronta due array di numeri interi e restituisce il valore true se tutti gli elementi degli array sono uguali.*

Campi statici.

Oltre ai metodi, possono essere creati anche campi statici. Quindi essi esistono all'interno della classe e non devono essere istanziati per poterli usare. I campi statici possono essere inizializzati mediante il blocco static ad esempio:

```
class Dadi {
  static int primo;
  static int secondo;
  static {
    primo = 1;
    secondo = 2;
  }
}
```

In generale non si deve abusare dei metodi e dei campi static, bisogna usarli meno possibile.

Singleton

Rappresenta un tipo particolare di classe che garantisce che soltanto un'unica istanza della classe stessa possa essere creata all'interno di un programma. Per ottenere questo tipo di comportamento è necessario avvalersi del modificatore di accesso "***private***" anche per il costruttore della classe (cosa che generalmente non viene mai praticata in una classe "standard") ed utilizzare un metodo statico che consenta di accedere all'unica istanza della classe.

Esempio Singleton

```
public class Singleton{
  private static Singleton istanza=null;
  private Singleton()  {  }
  public static Singleton getInstance(){
    if (istanza == null)
        istanza = new Singleton();
    return istanza;
  }
  public void helloWorld(){
    System.out.println("Hello World");
  }
}
.........
.........
public class usaSingleton{
  public static void main(String args[]) {
    Singleton.getInstance().helloWorld();
  }
}
```

Come si nota dal codice, il costruttore della classe Singleton è stato definito private e, in tal modo, l'unico punto di accesso alla classe per il mondo esterno viene fornito attraverso il metodo statico getInstance() che si occupa di restituire (creandola prima se non è mai stata creata) l'unica istanza della classe.

Esempicon synchronized

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

   private ClasseSingleton() {}

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

Ancora Singleton

La parola chiave **synchronized**, 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.

In definitiva

Quando può rivelarsi utile avvalersi dei singleton?
In generale, tale scelta viene effettuata in tutti quei casi in cui è necessario che venga utilizzata una sola istanza di una classe.
Ciò consente di:
- avere un accesso controllato all'unica istanza della classe
- avere uno spazio di nomi ridotto
- evitare la dichiarazione di variabili globali
- assicurarsi di avere un basso numero di oggetti utilizzati in condivisione grazie al fatto che viene impedita la creazione di nuove istanze ogni volta che si voglia utilizzare la stessa classe.