Code refactor - Demo

Un *code refactor* rappresenta un frammento di codice (da una singola istruzione a un intero programma) che implementa una determinata funzionalità; il codice funziona correttamente, ma può essere sottoposto a *refactoring* per migliorarlo sotto il profilo della chiarezza, della leggibilità e manutenibilità, delle perfomance. Il *refactoring* va dalla semplice modifica del nome di una variabile, alla completa riscrittura di un procedimento.

I *code refactor* devono essere modificati al computer, sfruttando le funzionalità offerte da Visual Studio.

Dimostrazione

Di seguito sono mostrati alcuni *code refactor*; per ognuno viene proposta la versione modificata. Ogni *code refactor*:

- Può essere preceduto da una breve spiegazione sulla sua finalità.
- Può essere seguito da un suggerimento sulle modifiche da effettuare.

Una puntualizzazione: non esiste "il modo giusto" di fare *refactoring*; le soluzioni presentate di seguito rappresentano "un modo" per migliorare il codice.

1 Inserimento nominativi

Il codice implementa l'inserimento dell'elenco dei nominativi degli alunni di una classe.

```
Console.Write("Inserisci numero alunni");
int n = int.Parse(Console.ReadLine());
string[] vettore = new string[n];
for (int i = 0; i < vettore.Length; i++)
{
    Console.Write("Nominativo alunno: ");
    string tmp = Console.ReadLine();
    vettore[i] = tmp;
}
```

Le variabili `n` e `vettore` hanno un nome inadeguato. L'assegnazione dell'input alla variabile `tmp` è superflua.

```
Console.Write("Inserisci numero alunni");
int numAlunni = int.Parse(Console.ReadLine());
string[] nomiAlunni = new string[numAlunni];
for (int i = 0; i < nomiAlunni.Length; i++)
{
    Console.Write("Nominativo alunno: ");
    nomiAlunni[i] = Console.ReadLine();
}
```

2 Ricerca nominativo

Il codice implementa un procedimento di ricerca di un alunno all'interno di una classe.

```
static bool IndiceAlunno(string[] classe, string s)
{
    bool ok = false;
    for (int i = 0; i < classe.Length; i++)
    {
        if (classe[i] == s)
            ok = true;  //!il procedimento dovrebbe terminare!
    }
    return ok;
}
```

Il nome del metodo è completamente fuorviante; i nomi del parametro `s` e della variabile `ok` sono inadeguati. Se il nominativo viene trovato, il procedimento dovrebbe terminare immediatamente. La variabile booleana è superflua.

```
static bool ClasseContiene(string[] classe, string nome)
{
    for (int i = 0; i < classe.Length; i++)
    {
        if (classe[i] == nome)
            return true;
    }
    return false;
}
```

Utilizzando il `foreach` si può implementare una versione ancora più semplice e leggibile:

```
static bool ClasseContiene(string[] classe, string nome)
{
    foreach (string nomeAlunno in classe)
    {
        if (nomeAlunno == nome)
            return true;
    }
    return false;
}
```