Overload the FailSoftArray indexer

```
/*
C#: The Complete Reference
by Herbert Schildt

Publisher: Osborne/McGraw-Hill (March 8, 2002)
ISBN: 0072134852
*/

// Overload the FailSoftArray indexer.

using System;

class FailSoftArray {
int[] a; // reference to underlying array

public int Length; // Length is public

public bool errflag; // indicates outcome of last operation

// Construct array given its size.
public FailSoftArray(int size) {
a = new int[size];
Length = size;
}

// This is the int indexer for FailSoftArray.
public int this[int index] {
// This is the get accessor.
get {
if(ok(index)) {
errflag = false;
return a[index];
} else {
errflag = true;
```

```
return 0;
}
}

// This is the set accessor
set {
if(ok(index)) {
a[index] = value;
errflag = false;
}
else errflag = true;
}
}

/* This is another indexer for FailSoftArray.
This index takes a double argument. It then
rounds that argument to the nearest integer
index. */
public int this[double idx] {
// This is the get accessor.
get {
int index;

// round to nearest int
if( (idx – (int) idx) < 0.5) index = (int) idx; else index =
(int) idx + 1; if(ok(index)) { errflag = false; return
a[index]; } else { errflag = true; return 0; } } // This is
the set accessor set { int index; // round to nearest int if(
(idx - (int) idx) < 0.5) index = (int) idx; else index = (int)
idx + 1; if(ok(index)) { a[index] = value; errflag = false; }
else errflag = true; } } // Return true if index is within
bounds. private bool ok(int index) { if(index >= 0 & index <
Length) return true; return false; } } // Demonstrate the
fail-soft array. public class FSDemo1 { public static void
Main() { FailSoftArray fs = new FailSoftArray(5); // put some
values in fs for(int i=0; i < fs.Length; i++) fs[i] = i; //
now index with ints and doubles Console.WriteLine("fs[1]: " +
fs[1]);   Console.WriteLine("fs[2]:   "   +   fs[2]);
```

```
Console.WriteLine("fs[1.1]:     "     +     fs[1.1]);
Console.WriteLine("fs[1.6]: " + fs[1.6]); } } [/csharp]
```