

# C# Primer Errata - February, 2002

## Chapter 1

pg. 14, line 3

```
bool traceOn = false,
```

should read

```
bool traceOn = false;
```

pg. 33, 9 lines from bottom

```
ArrayList m_text = new ArrayList();
```

should read

```
ArrayList text = new ArrayList();
```

pg. 33, 2 lines from bottom

```
m_text.Add( text_line );
```

should read

```
text.Add( text_line );
```

pg. 44, line 9

```
if ( ! File.Exists( file_name ) )
```

should read

```
if ( ! File.Exists( file_name ) )
```

**pg. 50**, 7 lines from the bottom

```
long Unsigned 64-bit int ...
```

**should read**

```
ulong Unsigned 64-bit int ...
```

**pg. 55**, line 13

```
Logical AND (&& and &)
```

```
Logical OR (|| and |)
```

**should read**

```
Logical AND (bool) ( & )
```

```
Logical OR (bool) ( | )
```

```
Logical AND (short Circuit) ( && )
```

```
Logical OR (short Circuit) ( || )
```

## Chapter 2

**pg. 72**, line 4

```
for ( int iy = 0; iy < mat.cols; ++iy );
```

should read

```
for ( int iy = 0; iy < mat.cols; ++iy )
```

**pg. 75**, 14 lines from the bottom

```
Point3D x_offset = new Point3D(1.0) // ...
```

should read

```
Point3D x_offset = new Point3D(1.0); // ...
```

**pg. 78**, line 10

```
StringNode node = new Node( "a node" );
```

should read

```
StringNode node = new StringNode( "a node" );
```

**pg. 96**, line 4

```
Matrix result = new matrix( mat.rows, mat.cols );
```

should read

```
Matrix result = new Matrix( mat.rows, mat.cols );
```

**pg. 110**, line 3, line 8, |= should read !=

**pg. 112**, spanning most of the page

```
BitVec ==> BitVector
```

## Chapter 3

pg. 119, line 16

```
AudioCD ==> AudioBook
```

pg. 119, 4 lines from bottom

```
mat.check_in();
```

the instance of `check-in()` to be executed is determined at compile time according to `mat`'s class type. Because the function to invoke is resolved before the program begins running, this is called *static binding*.

should read

The key mechanism that makes polymorphism work is *dynamic binding*. In non-object-oriented programs, when we write

```
mat.check_in();
```

the instance of `check-in()` to be executed is determined at compile time according to `mat`'s class type. Because the function to invoke is resolved before the program begins running, this is called *static binding*.

pg. 145, line 10

```
Cival ==> Civil
```

## Chapter 4

pg. 160, 6 lines down from Section 4.1

There are three, not two, Sort() instances associated with ArrayList. The third instance supports sorting a subset of contiguous elements. The changes in the treatment are: pair ==> set; two is deleted; either ==> any.

pg. 166, the else clause of insert\_value()

```
m_rchild = new BTreeNode( val );
```

should read

```
m_rchild = new TreeNode( val );
```

pg. 167, INumericSequence definition

```
public interface INumericSequence : IEnumerable
{
    bool generate_sequence( int position );
    void display();
    void display( int first );
    void display( int first, int last );
}
```

should read

```
public interface INumericSequence : IEnumerable
{
    bool GenerateSequence( int position );
    void Display();
    void Display( int first );
    void Display( int first, int last );
}
```

pg. 169, Fibonacci definition

pg. 170, 171, 183

The changes to the INumericSequence methods to capitalization also occurs on these pages.

**pg. 184**, -1 lines from bottom

```
public void doit(){ doSomething(); }
```

**should read**

```
public void doit(){ doSomething( myObj ); }
```

## Chapter 5

**pg. 208**, line 15

```
lessMinCnt;
```

**should read**

```
lessMinCnt++;
```

**pg. 214**, last code line on page

```
Console.WriteLine("Directory full name ", dir.FullName);
```

**should read**

```
Console.WriteLine("Directory full name {0}", dir.FullName);
```

**pg. 226**, last paragraph

Table 5.1

**should read**

Table 5.2

**pg. 229**, line 3

(a to z, A to Z, 0 to 9)

**should read**

(a to z, A to Z, \_, 0 to 9)

**pg. 230**, line 16

```
5abc2001
```

**should read**

```
527ar2001
```

**pg. 238**, line 3

```
Thread ping = new Thread( new ThreadStart( p1.play ))
```

**should read**

```
Thread ping = new Thread( new ThreadStart( p1.play ));
```

## Chapter 6

None

## **Chapter 7**

**pg. 315**, line 3

using a editor

**should read**

using an editor

## Chapter 8

**pg. 350**, paragraph just before the code.

The sentence, “An app domain acts as a container for a process” should be changed to “An app domain acts as a container for an assembly.”

**pg. 358**, 1 line from bottom

```
PropertyInfo[] parray = t.GetProperties( myflags );
```

should read

```
PropertyInfo[] parray = t.GetProperties( f );
```

**pg. 367**, code at top of page

```
if( tt.isClass )
```

should read

```
if( tt.IsClass )
```