뇌를 자극하는 C# 4.0 프로그래밍

18. 파일 다루기

01. 파일 정보와 디렉토리 정보 다루기(1/4)

- 파일(File)은 컴퓨터 저장매체에 기록되는 데이터의 묶음
- 디렉토리(Directory)는 파일이 위치하는 주소
  - 파일(서류)를 담는다는 의미에서 폴더(Folder:서류철) 라고 부르기도 함

- .NET 프레임워크는 파일과 디렉토리 정보를 손쉽게 다룰 수 있도록 System.IO 네임스페이스에 다음과 같은 클래스 제공

클래스	설명
File	파일의 생성, 복사, 삭제, 이동, 조회를 처리하는 정적 메소드를 제공합니다.
FileInfo	File 클래스와 하는 일은 동일하지만 정적 메소드 대신 인스턴스 메소드를 제공합니다.
Directory	디렉토리의 생성, 삭제, 이동, 조회를 처리하는 정적 메소드를 제공합니다.
DirectoryInfo	Directory 클래스와 하는 일은 동일하지만 정적 메소드 대신 인스턴스 메소드를 제공합니다.

01. 파일 정보와 디렉토리 정보 다루기(2/4)

- File 클래스와 FileInfo 클래스는 같은 기능을 제공
  - File클래스는 같은 기능을 정적 메소드를 통해 제공하고 FileInfo 클래스는 인스턴스 메소드를 통해 제공
- Directory 클래스와 Directory 클래스도 같은 기능을 제공
  - 디렉토리에 대해 한 두가지 작업을 할 때에는 Directory 클래스, 여러 가지 작업을 해야 할 때는 DirectoryInfo 클래스를 이용

기능	File	FileInfo	Directory	DirectoryInfo
생성	Create()	Create()	CreateDirectory()	Create()
복사	Copy()	CopyTo()	-	-
삭제	Delete()	Delete()	Delete()	Delete()
이동	Move()	MoveTo()	Move()	MoveTo()
존재여부 확인	Exists()	Exists	Exists()	Exists
속성 조회	GetAttributes()	Attributes	GetAttributes()	Attributes
하위 디렉토리 조회	-	-	GetDirectories()	GetDirectories()
하위 파일 조회	-	-	GetFiles()	GetFiles()

01. 파일 정보와 디렉토리 정보 다루기(3/4)

- File 클래스와 FileInfo 클래스의 사용 예제

기능	File	FileInfo
생성	FileStream fs = File.Create(“a.dat”);	FileInfo file = new FileInfo(“a.dat”); FileStream fs = file.Create()
복사	File.Copy(“a.dat”,“b.dat”);	FileInfo src = new FileInfo("a.dat"); FileInfo dst = src.CopyTo("b.dat");
삭제	File.Delete("a.dat");	FileInfo file = new FileInfo("a.dat"); file.Delete();
이동	File.Move(“a.dat”,“b.dat”);	FileInfo file = new FileInfo("a.dat"); file.MoveTo("b.dat");
존재여부 확인	if (File.Exists("a.dat")) // …	FileInfo file = new FileInfo("a.dat"); if (file.Exists) // ...
속성 조회	Console.WriteLine(File.GetAttributes(“a.dat”));	FileInfo file = new FileInfo("a.dat"); Console.WriteLine(file.Attributes);

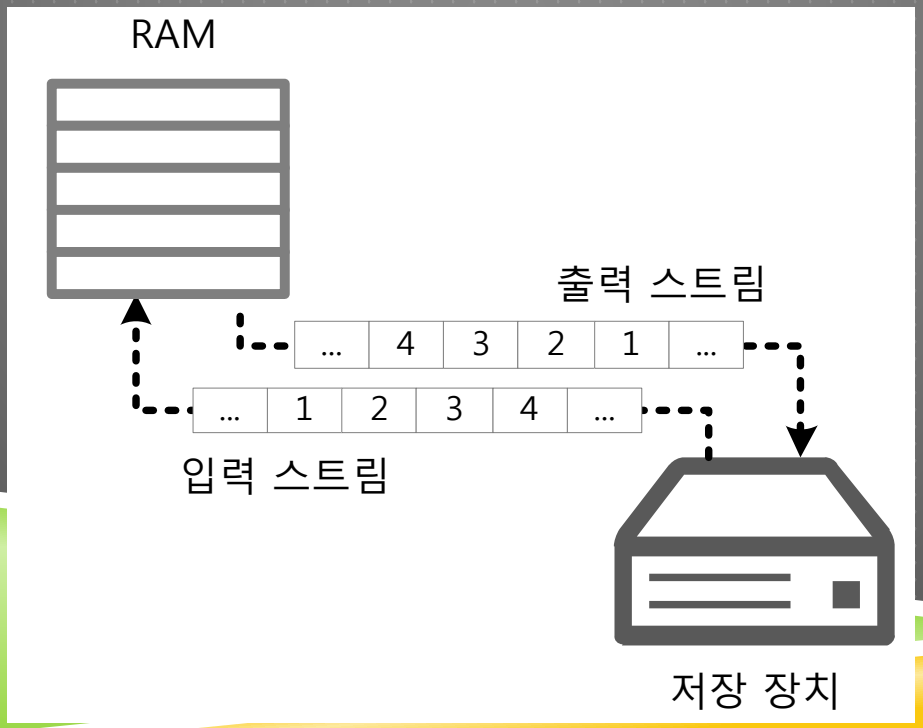
01. 파일 정보와 디렉토리 정보 다루기(4/4)

- Directory 클래스와 DirectoryInfo 클래스의 사용 예제

기능	File	FileInfo
생성	DirectoryInfo dir = Directory.CreateDirectory("a");	DirectoryInfo dir = new DirectoryInfo("a"); dir.Create();
삭제	Directory.Delete("a");	DirectoryInfo dir = new DirectoryInfo("a"); dir.Delete();
이동	Directory.Move("a", "b");	DirectoryInfo dir = new DirectoryInfo("a"); dir.MoveTo("b");
존재여부 확인	if (Directory.Exists("a.dat")) // …	DirectoryInfo dir = new DirectoryInfo("a"); if (dir.Exists) // ...
속성 조회	Console.WriteLine(Directory.GetAttributes("a")) ;	DirectoryInfo dir = new DirectoryInfo("a"); Console.WriteLine(dir.Attributes);
하위 디렉토리 조회	string[] dirs = Directory.GetDirectories("a");	DirectoryInfo dir = new DirectoryInfo("a"); DirectoryInfo[] dirs = dir.GetDirectories();
하위 파일 조회	string[] files = Directory.GetFiles ("a");	DirectoryInfo dir = new DirectoryInfo("a"); FileInfo[] files =dir.GetFiles();

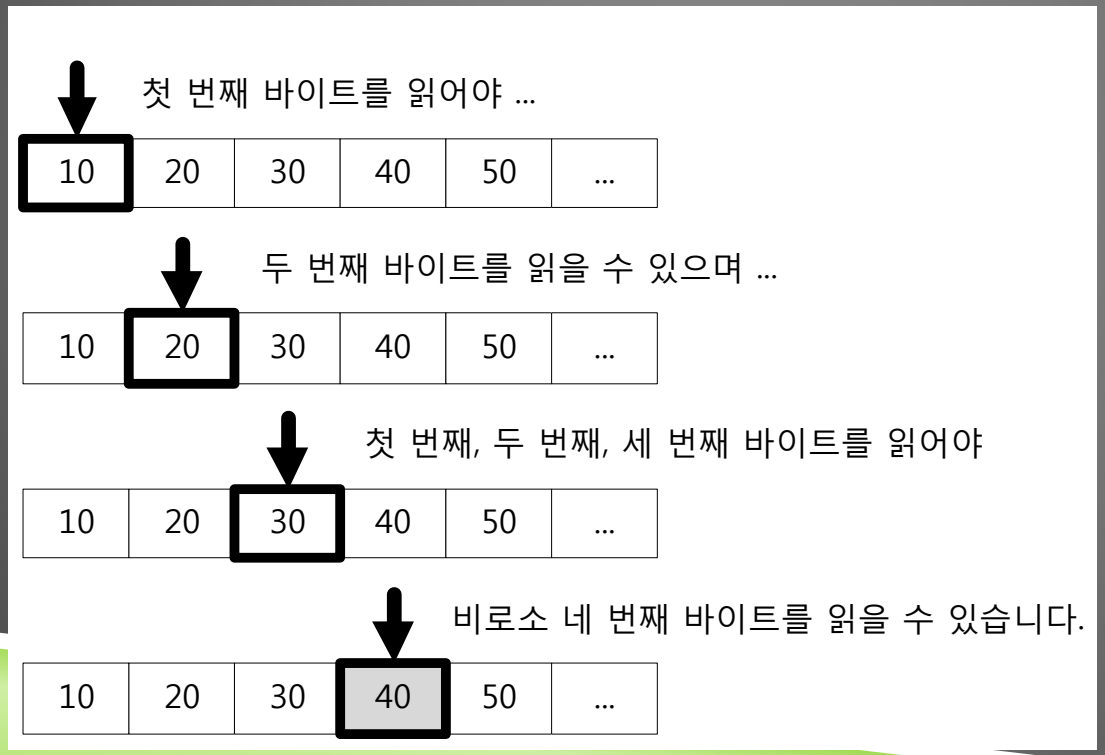
02. 파일을 읽고 쓰기 위해 알아야 할 것들(1/7)

- 스트림(Stream)
  - 영어로 시내, 강, 또는 도로의 차선을 뜻함
  - 파일을 다룰 때의 스트림은 "데이터가 흐르는 통로"를 뜻함
  - 메모리에서 하드디스크로 데이터를 옮길때, 스트림을 만들어 둘 사이를 연결한 뒤에 메모리의 데이터를 바이트 단위로 하드 디스크로 옮김.
    - 그 반대의 경우도 마찬가지.



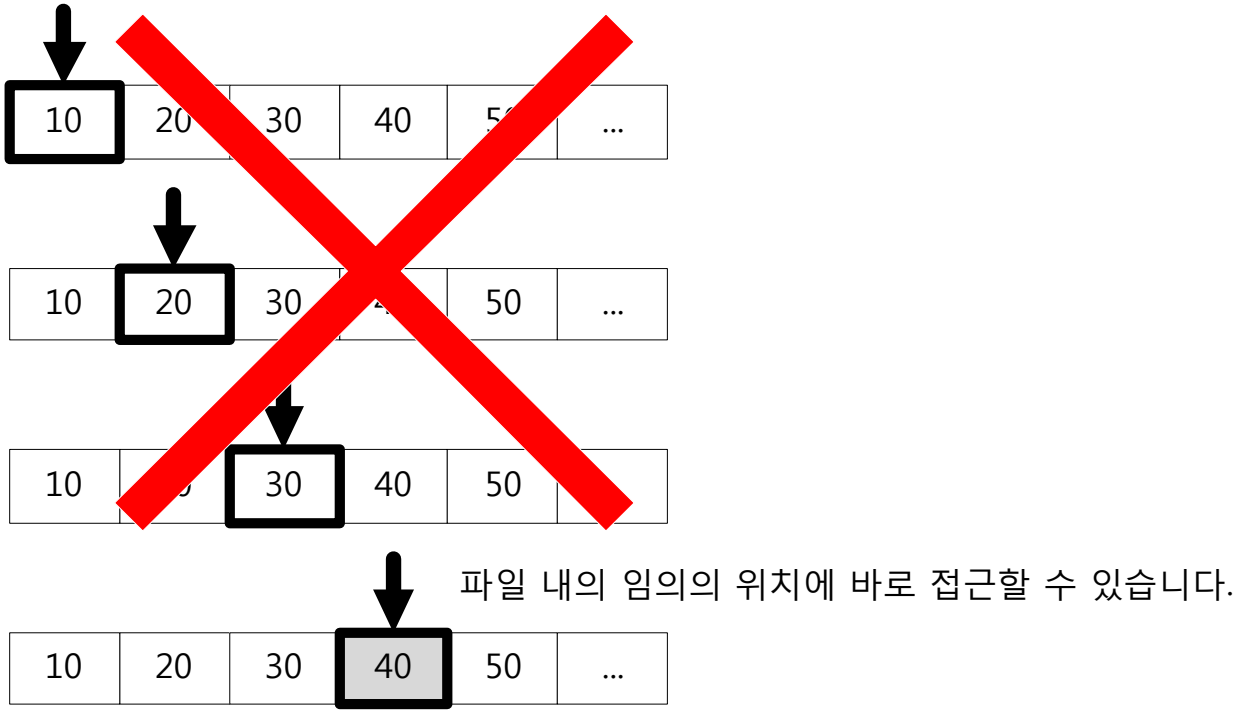
02. 파일을 읽고 쓰기 위해 알아야 할 것들(2/7)

- 순차접근(Sequential Access)
  - 처음부터 끝까지 순서대로 읽고 쓰는 접근 방식.
  - 스트림이 데이터의 "흐름"임을 고려하면 자연스러운 방식임.



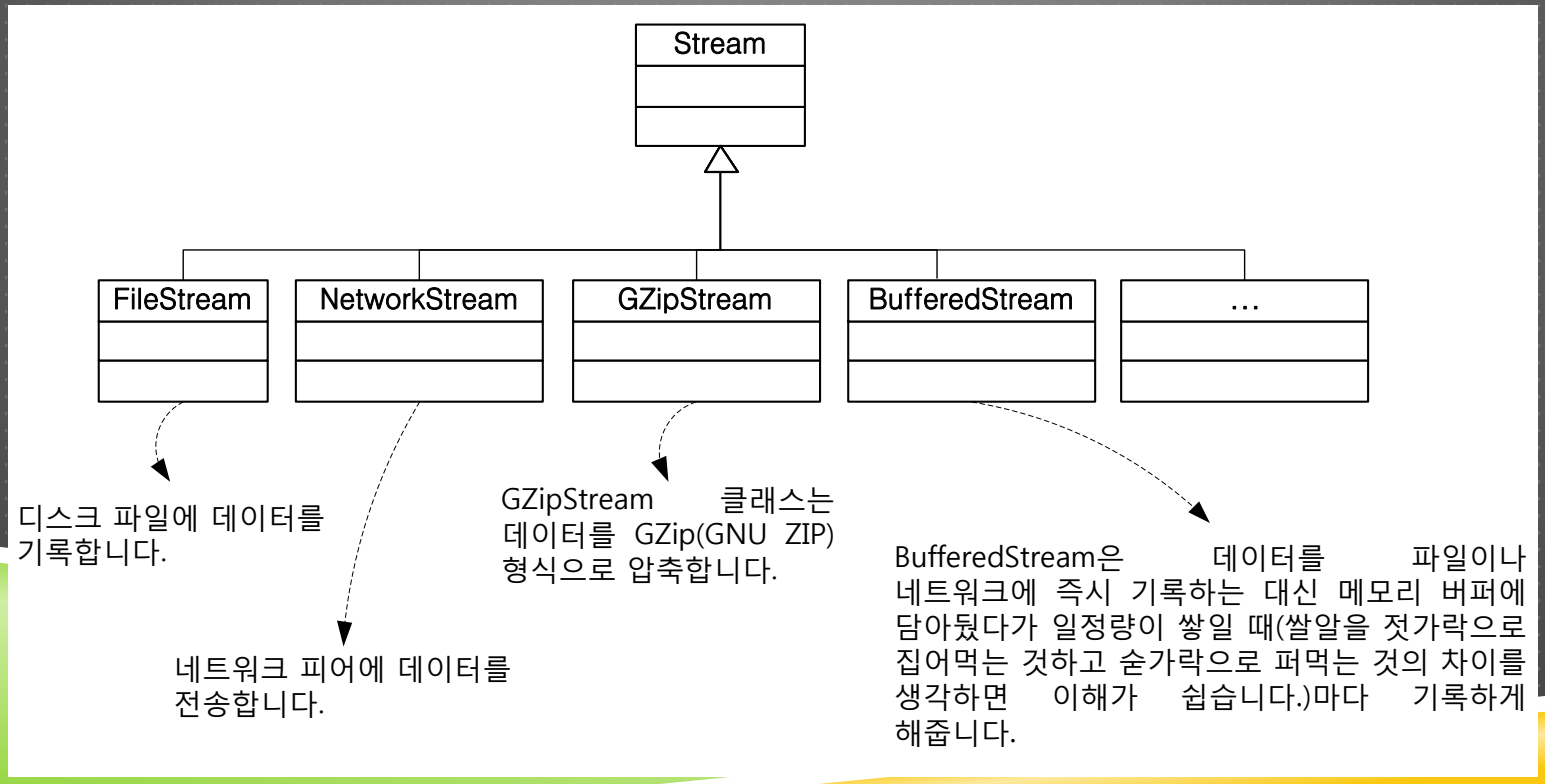
02. 파일을 읽고 쓰기 위해 알아야 할 것들(3/7)

- 임의접근(Random Access)
  - 파일 내의 임의의 위치에 있는 데이터에 즉시 접근하는 방식
  - 하드디스크와 같이 암과 헤드를 움직여 디스크의 어떤 위치에 기록된 데이터라도 즉시 접근할 수 있는 경우에 적합



02. 파일을 읽고 쓰기 위해 알아야 할 것들(4/7)

- System.IO.Stream 클래스
  - 입력 스트림, 출력 스트림의 역할을 모두 수행
  - 순차접근방식과 임의 접근 방식 모두 지원
  - 단, 추상 클래스이기 때문에 이 클래스의 파생 클래스를 이용해야 함.



02. 파일을 읽고 쓰기 위해 알아야 할 것들(5/7)

- FileStream 클래스의 사용 예(1/2)

a.dat 파일에 long 형식의 0x123456789ABCDEF0 쓰기

```
long someValue = 0x123456789ABCDEF0;

// 1) 파일 스트림 생성
Stream outStream =
    new FileStream("a.dat", FileMode.Create);

// 2) someValue(long 형식)을 byte 배열로 변환
byte[] wBytes =
    BitConverter.GetBytes(someValue);

// 3) 변환한 byte 배열을 파일 스트림을
       통해 파일에 기록
outStream.Write(wBytes, 0, wBytes.Length);

// 4) 파일 스트림 닫기
outStream.Close();
```

a.dat 파일로부터 long 형식에 0x123456789ABCDEF0 읽어들이기

```
byte[] rBytes = new byte[8];

// 1) 파일 스트림 생성
Stream inStream =
    new FileStream("a.dat", FileMode.Open);

// 2) rBytes의 길이만큼(8바이트) 데이터를
      읽어 rBytes에 저장
inStream.Read(rBytes, 0, rBytes.Length);

// 3) BitConverter를 이용하여 rBytes에
      담겨있는 값을 long 형식으로 변환
long readValue =
    BitConverter.ToInt64(rbytes, 0);

// 4) 파일 스트림 닫기
inStream.Close();
```

02. 파일을 읽고 쓰기 위해 알아야 할 것들(6/7)

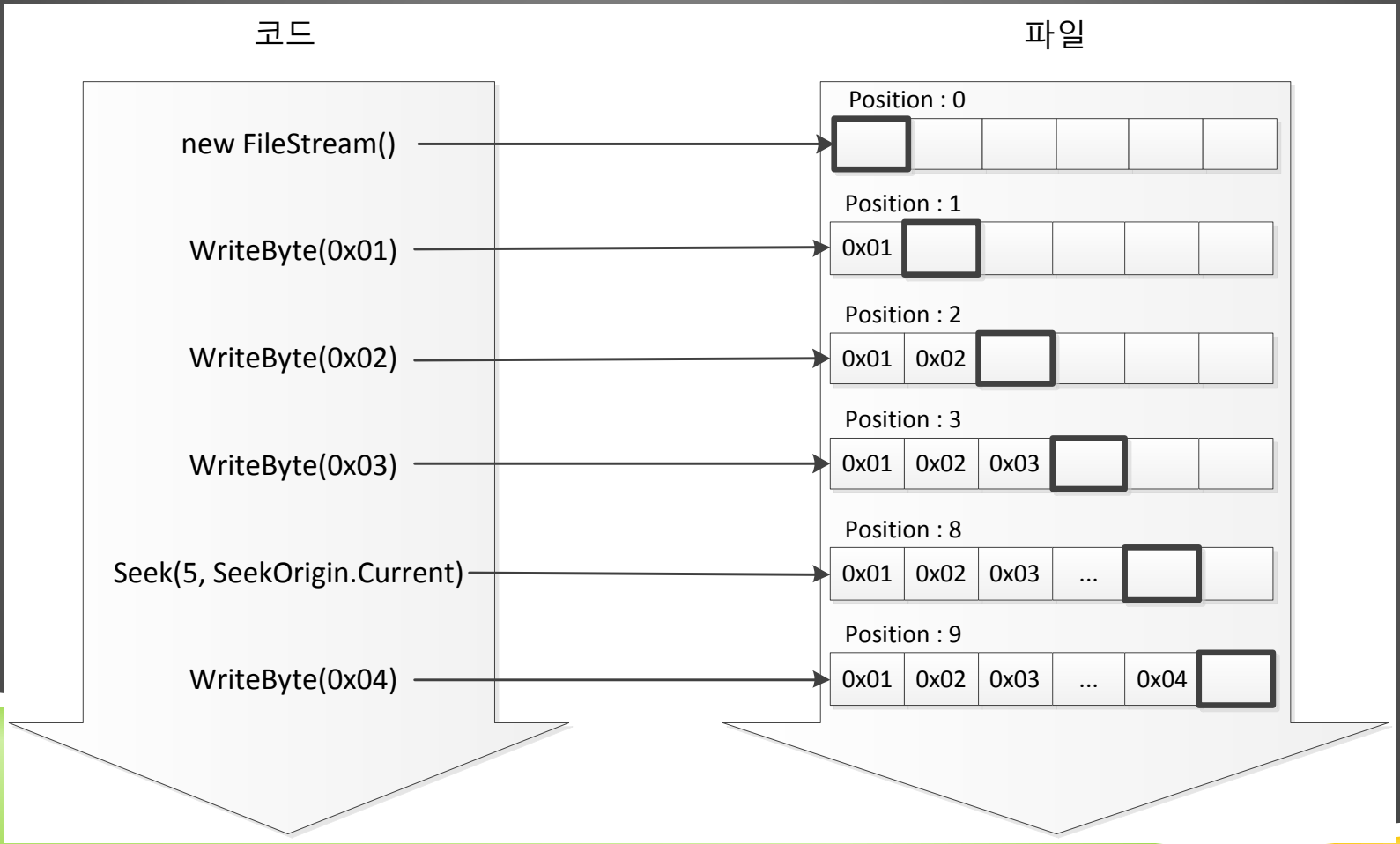
- FileStream 클래스의 사용 예(2/2)
  - 16진수 123456789ABCDEF0 를 바이트 단위로 쪼개면 12, 34, 56, 78, 9A, BC, DE, F0
  - 그런데 바이너리 에디터로 a.dat를 열어보면 다음과 같이 거꾸로 기록되어 있음.

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000   F0 DE BC 9A 78 56 34 12
```

  - 이것은 오류가 아닌, 컴퓨터 아키텍쳐가 데이터의 낮은 주소부터 기록하는 리틀 엔디안(Little Endian)방식이기 때문에 나타난 현상.
  - 타 시스템과 이진 파일을 교환하는 소프트웨어를 작성하는 경우에는 이러한 바이트 오더 관계를 반드시 고려해야 함.

02. 파일을 읽고 쓰기 위해 알아야 할 것들(7/7)

- 파일에 여러 개의 데이터를 기록하는 과정



03. 이진 데이터 처리를 위한 BINARYWRITER/BINARYREADER(1/3)

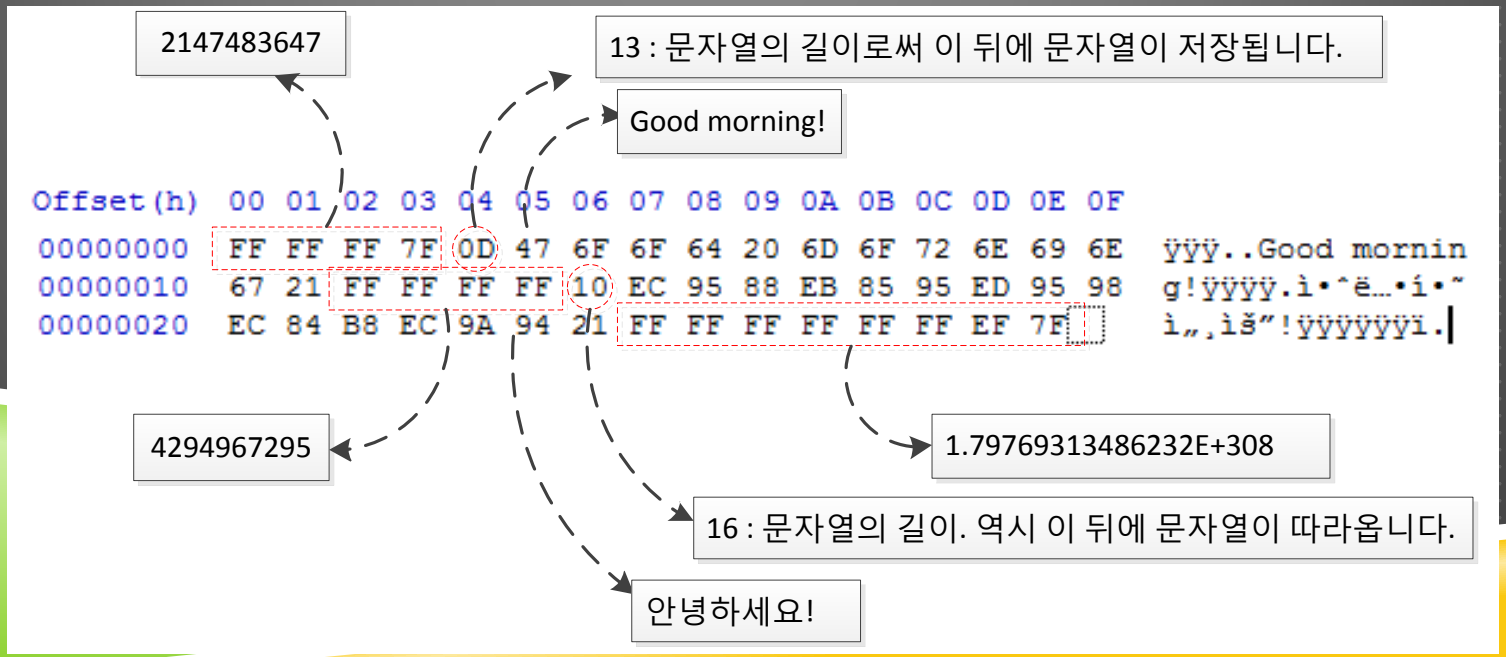
- FileStream 클래스는 파일처리를 위한 모든 것을 제공하지만, 사용하기에는 불편
- 특히 데이터를 저장할 때 반드시 byte 또는 byte 배열 형식으로 변환해야 하는 문제가 있음.
  - 이것은 데이터를 읽을 때도 마찬가지.
- .NET 프레임워크는 이런 불편함을 해소하기 위해 도우미 클래스를 제공
  - BinaryWriter와 BinaryReader가 그 예임.
  - BinaryWriter는 이진 데이터를 기록하기 위해,
  - BinaryReader는 이진 데이터를 읽기 위해 만들어진 클래스.
  - 이 클래스들은 "도우미"이기 때문에 FileStream과 같은 Stream의 파생 클래스와 함께 사용해야 함.

03. 이진 데이터 처리를 위한 BINARYWRITER/BINARYREADER(2/3)

- FileStream 과 BinaryWriter를 함께 사용하는 예

```
BinaryWriter bw = new BinaryWriter( new FileStream("a.dat", FileMode.Create) );
 bw.Write(int.MaxValue);
bw.Write("Good Morning!");
bw.Write(uint.MaxValue);
bw.Write("안녕하세요!");
bw.Write(double.MaxValue);
bw.Close();
```

FileStream처럼 BitConverter를 이용할 필요 없음!



03. 이진 데이터 처리를 위한 BINARYWRITER/BINARYREADER(3/3)

- FileStream과 BinaryReader를 함께 사용하는 예

```
BinaryReader br = new BinaryReader( new FileStream("a.dat", FileMode.Open) );

int    a = br.ReadInt32();
string b = br.ReadString();
double c = br.ReadDouble ();

br.Close();
```

04. 텍스트 파일 처리를 위한 STREAMWRITER/STREAMREADER(1/3)

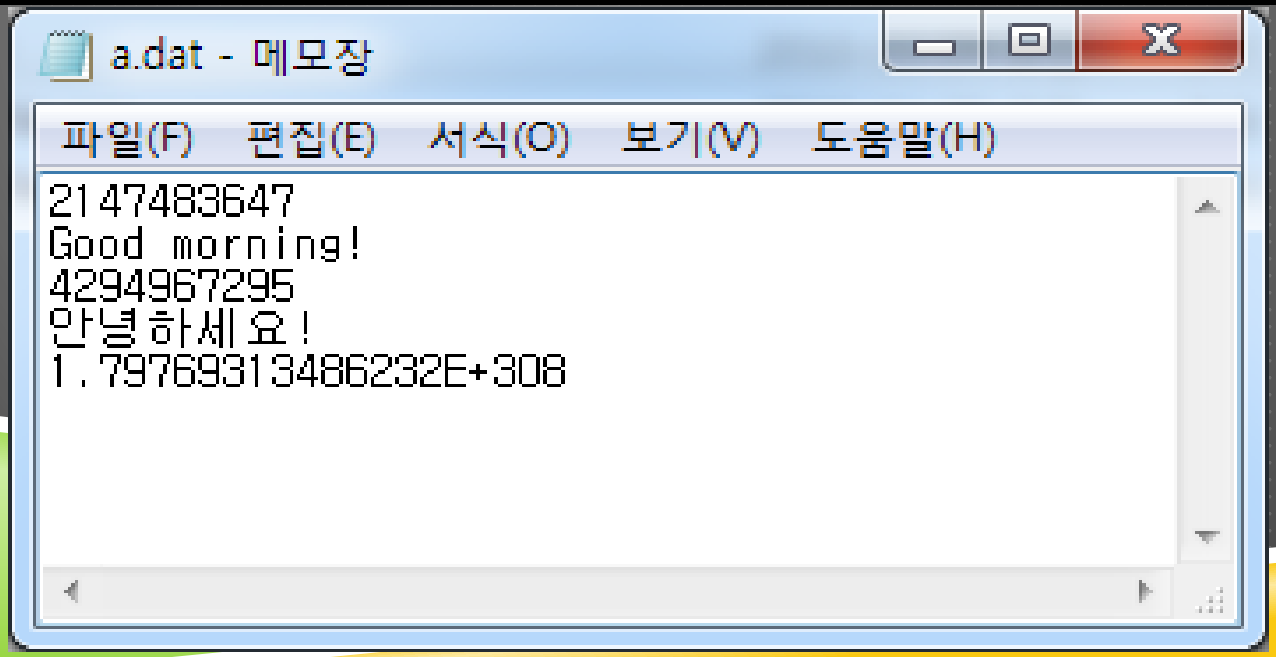
- 텍스트 파일은 구조가 간단하면서도 활용도가 높음
- ASCII 인코딩에서는 각 바이트가 문자 하나를 나타내기 때문에 바이트 오더의 문제에서 벗어날 수 있음.
- 프로그램이 생성한 파일의 내용을 사람이 바로 읽을 수 있음.

- .NET 프레임워크는 텍스트 파일을 읽고 쓸 수 있는 도우미 클래스인 StreamWriter와 StreamReader를 제공

04. 텍스트 파일 처리를 위한 STREAMWRITER/STREAMREADER(2/3)

- FileStream 과 StreamWriter를 함께 사용하는 예

```
StreamWriter sw = new StreamWriter( new FileStream("a.dat", FileMode.Create) );
sw.Write(int.MaxValue);
sw.Write("Good Morning!");
sw.Write(uint.MaxValue);
sw.Write("안녕하세요!");
sw.Write(double.MaxValue);
sw.Close();
```



04. 텍스트 파일 처리를 위한 STREAMWRITER/STREAMREADER(3/3)

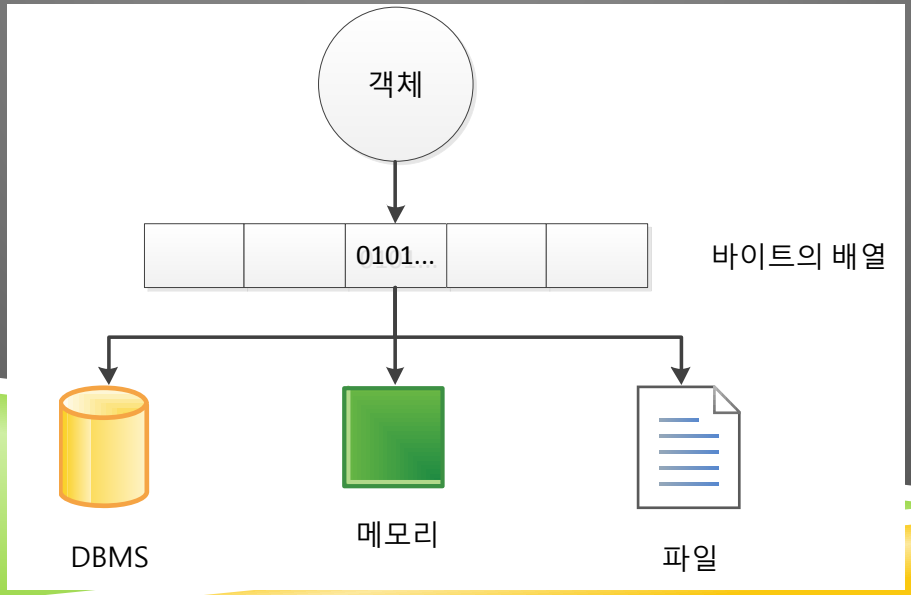
- FileStream 과 StreamReader를 함께 사용하는 예

```
StreamReader sr = new StreamReader( new FileStream("a.dat", FileMode.Open) );

while ( sr.EndOfStream == false )
{
    Console.WriteLine(sr.ReadLine());
}

sr.Close();
```

EndOfStream 프로퍼티는 스트림의 끝에 도달했는지를 알려줍니다.

05. 객체 직렬화하기(1/2)

- 클래스나 구조체 같은 복합 데이터 형식을 파일에 쓰고 읽으려면 어떻게 해야 할까?
  - 각 필드를 일일이 저장하는 코드를 작성해야 하나?
- 이 문제를 위해 직렬화(Serialization)라는 메커니즘을 제공
  - 직렬화란, 객체의 상태(여기에서 말하는 객체의 상태는 객체의 필드에 저장된 값들을 의미합니다.)를 메모리나 영구 저장 장치에 저장이 가능한 0과 1의 순서로 바꾸는 것을 말함



05. 객체 직렬화하기(2/2)

- 직렬화가 가능한 클래스를 만들기 위해선 다음과 같이 [Serialization] 애트리뷰트를 클래스 선언부 앞에 붙이면 됨.

```
[Serialization]
class MyClass
{
    // ..
}
```

객체 직렬화

```
Stream ws =
    new FileStream("a.dat", FileMode.Create);
BinaryFormatter serializer =
    new BinaryFormatter();

MyClass obj = new MyClass();
// obj의 필드에 값 저장…

serializer.Serialize(ws, obj);
ws.Close();
```

객체 역직렬화

```
Stream rs =
    new FileStream("a.dat", FileMode.Open);
BinaryFormatter deserializer =
    new BinaryFormatter();

MyClass obj =
    (MyClass)deserializer.Deserialize(rs);
rs.Close();
```