**1. Какие классы содержатся в System.IO?**

В пространстве имен System.IO содержится множество классов для работы с файлами, каталогами, потоками данных и файловыми путями. Вот основные классы:

* **File** и **FileInfo** — для работы с файлами.
* **Directory** и **DirectoryInfo** — для работы с каталогами.
* **Path** — для работы с путями к файлам и каталогам.
* **FileStream** — для чтения и записи файлов.
* **StreamReader** и **StreamWriter** — для работы с текстовыми файлами.
* **BinaryReader** и **BinaryWriter** — для работы с бинарными файлами.
* **MemoryStream**, **BufferedStream** — для потоковой работы с данными.
* **GZipStream**, **DeflateStream** — для сжатия данных.

**2. Для чего используются классы Directory и DirectoryInfo? В чем отличие?**

* **Directory**: предоставляет статические методы для выполнения операций с каталогами, таких как создание, удаление, перемещение и получение списка файлов и подкаталогов.
* **DirectoryInfo**: предоставляет аналогичный функционал, но в виде экземплярного класса. Он сохраняет информацию о каталоге и может быть полезен для работы с объектами, представляющими каталоги.

**Отличие**:

* Directory — статический класс, не требует создания экземпляра.
* DirectoryInfo — не статический, позволяет сохранять состояние и информацию об объекте.

Пример:

// Использование Directory

var files = Directory.GetFiles("path/to/directory");

// Использование DirectoryInfo

var dirInfo = new DirectoryInfo("path/to/directory");

var filesInfo = dirInfo.GetFiles();

**3. Для чего используются классы File и FileInfo? Какие методы они содержат?**

* **File**: предоставляет статические методы для работы с файлами, такие как создание, копирование, удаление, проверка существования и чтение/запись.
* **FileInfo**: предоставляет аналогичный функционал, но через экземпляры, что полезно при необходимости сохранить состояние файла.

**Основные методы**:

* File.Create, File.Delete, File.Copy, File.Exists, File.ReadAllText, File.WriteAllText.
* FileInfo.Create, FileInfo.Delete, FileInfo.CopyTo, FileInfo.Exists, FileInfo.Open.

Пример:

// File

File.WriteAllText("example.txt", "Hello, world!");

// FileInfo

var fileInfo = new FileInfo("example.txt");

fileInfo.MoveTo("new\_location.txt");

**4. Для чего используются классы StreamReader и StreamWriter?**

* **StreamReader**: используется для чтения текста из потоков, таких как файлы или сетевые потоки.
* **StreamWriter**: используется для записи текста в потоки.

Пример:

// StreamReader

using (StreamReader reader = new StreamReader("example.txt"))

{

string content = reader.ReadToEnd();

}

// StreamWriter

using (StreamWriter writer = new StreamWriter("example.txt"))

{

writer.WriteLine("Hello, world!");

}

**5. Для чего используются классы BinaryWriter и BinaryReader?**

* **BinaryWriter**: для записи данных в бинарном формате.
* **BinaryReader**: для чтения данных из бинарного формата.

Пример:

using (var stream = new FileStream("binary.dat", FileMode.Create))

using (var writer = new BinaryWriter(stream))

{

writer.Write(123); // Запись числа

writer.Write("Hello"); // Запись строки

}

using (var stream = new FileStream("binary.dat", FileMode.Open))

using (var reader = new BinaryReader(stream))

{

int number = reader.ReadInt32();

string text = reader.ReadString();

}

**6. Как можно сжимать и восстанавливать файлы?**

Для сжатия и восстановления файлов используются классы:

* **GZipStream**: работает с форматом GZip.
* **DeflateStream**: работает с алгоритмом Deflate.

Пример с GZipStream:

// Сжатие

using (FileStream sourceStream = new FileStream("file.txt", FileMode.Open))

using (FileStream compressedStream = new FileStream("file.gz", FileMode.Create))

using (GZipStream compressionStream = new GZipStream(compressedStream, CompressionMode.Compress))

{

sourceStream.CopyTo(compressionStream);

}

// Восстановление

using (FileStream compressedStream = new FileStream("file.gz", FileMode.Open))

using (FileStream decompressedStream = new FileStream("file\_decompressed.txt", FileMode.Create))

using (GZipStream decompressionStream = new GZipStream(compressedStream, CompressionMode.Decompress))

{

decompressionStream.CopyTo(decompressedStream);

}

**7. Расскажите алгоритм сжатия GZip.**

GZip использует комбинацию:

1. Алгоритма **Deflate** для сжатия данных (основан на LZ77 и кодировании Хаффмана).
2. Заголовка GZip для хранения метаданных (размер, CRC и т. д.).
3. Алгоритм разбивает поток данных на блоки, сжимает их и записывает в результирующий поток.

**8. Для чего служит класс Path?**

Класс Path используется для работы с путями к файлам и каталогам. Он содержит методы для:

* Получения имени файла или расширения (Path.GetFileName, Path.GetExtension).
* Получения каталога (Path.GetDirectoryName).
* Объединения путей (Path.Combine).
* Создания временных файлов и директорий (Path.GetTempPath).

Пример:

string fullPath = Path.Combine("C:\\Temp", "file.txt");

string fileName = Path.GetFileName(fullPath);

**9. Что такое текстовый и бинарный файл. Как выполнить чтение и запись в эти файлы?**

* **Текстовый файл**: хранит данные в формате, понятном для чтения человеком (ASCII, UTF-8).
  + Чтение: StreamReader.
  + Запись: StreamWriter.
* **Бинарный файл**: хранит данные в машинно-ориентированном формате.
  + Чтение: BinaryReader.
  + Запись: BinaryWriter.

**10. Что такое произвольный доступ к файлу? Приведите пример.**

Произвольный доступ позволяет перемещаться по файлу, читать и записывать данные в любом месте. Используется FileStream.

Пример:

using (FileStream stream = new FileStream("file.bin", FileMode.OpenOrCreate))

{

stream.Seek(10, SeekOrigin.Begin); // Перемещение на 10-й байт

stream.WriteByte(255); // Запись байта

}

**11. Как применяется конструкция using (не директива) при работе с файловыми потоками?**

using автоматически закрывает и освобождает ресурсы потока после завершения работы.

Пример:

using (FileStream stream = new FileStream("file.txt", FileMode.Open))

using (StreamReader reader = new StreamReader(stream))

{

string content = reader.ReadToEnd();

}