**1. Классы в System.IO:**

- File - предоставляет статические методы для работы с файлами.

- FileInfo - представляет информацию о файле и предоставляет методы для работы с файлами.

- Directory - предоставляет статические методы для работы с каталогами.

- DirectoryInfo - представляет информацию о каталоге и предоставляет методы для работы с каталогами.

- Path - предоставляет методы для работы с путями к файлам и каталогам.

- FileStream - предоставляет методы для чтения и записи байтов в файл.

**2. Классы Directory и DirectoryInfo:**

- Directory - предоставляет статические методы для работы с каталогами (например, создание, удаление).

- DirectoryInfo - предоставляет объектно-ориентированный интерфейс для работы с информацией о каталоге.

**3. Классы File и FileInfo:**

- File - предоставляет статические методы для работы с файлами (например, чтение, запись).

- FileInfo - предоставляет методы экземпляра для работы с информацией о файле.

**4. Классы StreamReader и StreamWriter:**

- StreamReader - читает символы из потока байтов в определенной кодировке.

- StreamWriter - записывает символы в поток байтов в определенной кодировке.

**5. Классы BinaryWriter и BinaryReader:**

- BinaryWriter - записывает примитивные типы данных в поток в двоичной форме.

- BinaryReader - читает примитивные типы данных из потока в двоичной форме.

**6. Сжатие и восстановление файлов:**

- Для сжатия и восстановления файлов можно использовать классы GZipStream в .NET.

**7. Алгоритм сжатия GZip:**

- GZip использует алгоритм DEFLATE, который является комбинацией LZ77 и Huffman Coding.

**8. Класс Path:**

- Path - предоставляет методы для работы с путями к файлам и каталогам, такие как объединение путей и извлечение информации о пути.

**9. Текстовый и бинарный файл:**

- Текстовый файл содержит читаемые символы (например, текст).

- Бинарный файл содержит данные в бинарном формате (например, изображения, исполняемый код).

**10. Произвольный доступ к файлу:**

- Произвольный доступ к файлу (random access) обозначает возможность чтения/записи данных в любом месте файла, не последовательно. Класс FileStream предоставляет такую функциональность.

**11. Конструкция using с файловыми потоками:**

- Конструкция using используется для автоматического освобождения ресурсов (например, файловых потоков), когда они больше не нужны. Например:

using (FileStream fs = new FileStream("example.txt", FileMode.Open))

{

// Ваш код работы с файлом

} // Здесь файловый поток автоматически закрывается и освобождается

Это гарантирует, что ресурсы будут правильно освобождены, даже если возникнет исключение внутри блока using.