1. **Какие классы содержаться с System.IO?**

***BinaryReader, BinaryWriter***

Эти классы позволяют сохранять и извлекать элементарные типы данных (целочисленные, булевские, строковые и т.п.) в двоичном виде

***BufferedStream***

Этот класс предоставляет временное хранилище для потока байтов, которые могут затем быть перенесены в постоянные хранилища

***Directory, DirectoryInfo***

Эти классы используются для манипуляций структурой каталогов машины. Тип Directory представляет функциональность, используя статические члены. Тип DirectoryInfo обеспечивает аналогичную функциональность через действительную объектную ссылку

***DriveInfo***

Этот класс предоставляет детальную информацию относительно дисковых устройств, используемых данной машиной

***File, FileInfo***

Эти классы служат для манипуляций множеством файлов данной машины. Тип File представляет функциональность через статические члены. Тип FileInfo обеспечивает аналогичную функциональность через действительную объектную ссылку

***FileStream***

Этот класс обеспечивает произвольный доступ к файлу (т.е. возможности поиска) с данными, представленными в виде потока байт

***FileSystemWatcher***

Этот класс позволяет отслеживать модификации внешних файлов в определенном каталоге

***MemoryStream***

Этот класс обеспечивает произвольный доступ к данным, хранящимся в памяти, а не в физическом файле

***Path***

Этот класс выполняет операции над типами System.String, содержащими информацию о пути к файлу или каталогу в независимой от платформы манере

***StreamWriter, StreamReader***

Эти классы используются для хранения (и извлечения) текстовой информации из файла. Эти классы не поддерживают произвольного доступа к файлу

***StringWriter, StringReader***

Подобно классам StreamWriter/StreamReader, эти классы также работают с текстовой информацией. Однако лежащим в основе хранилищем является строковый буфер, а не физический файл

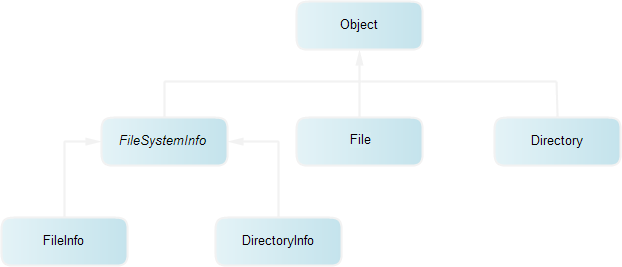
1. **Для чего используются классы Directory и DirectoryInfo? В чем отличие?**

Для работы с каталогами в пространстве имен System.IO предназначены сразу два класса: **Directory** и **DirectoryInfo**.

* Классы Directory и File содержат только статические методы, а их экземпляры никогда не создаются. Для их использования нужно просто предоставить путь к интересующему объекту файловой системы при каждом вызове метода-члена. Если требуется выполнить только одну операцию в отношении папки или файла, то применение этих классов является наиболее эффективным подходом, поскольку позволяет избегать накладных расходов, связанных с созданием экземпляров.
* Классы DirectoryInfo и FileInfo реализуют практически те же общедоступные методы, что и Directory и File, а также некоторые общедоступные свойства и конструкторы, но поддерживают состояние, а их члены не являются статическими.

Сначала понадобится создать их экземпляр, который затем ассоциировать с определенной папкой или файлом. Это означает, что применение этих классов будет более эффективным подходом, если необходимо выполнять множество операций в отношении одного и того же объекта. Дело в том, что во время создания они считывают информацию об аутентификации и прочие сведения о соответствующем объекте файловой системы и больше не нуждаются в ее повторном чтении, сколько бы методов не вызывалось для данного объекта (экземпляра класса). В отличие от этого, классы, не поддерживающие состояние, нуждаются в проверке деталей, связанных с файлом или папкой, при каждом вызове какого-либо метода.

Обратите внимание на рисунке, что классы Directory и File непосредственно расширяют System.Object, в то время как DirectoryInfo и FileInfo наследуются от абстрактного класса **FileSystemInfo**:



Классы DirectoryInfo и FileInfo унаследовали значительную часть своего поведения от абстрактного базового класса FileSystemInfo. По большей части члены класса FileSystemInfo используются для получения общих характеристик (таких как время создания, различные атрибуты и т.д.) определенного файла или каталога. В таблице ниже перечислены некоторые основные свойства, представляющие интерес:

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

Подобно паре Directory/DirectoryInfo для работы с файлами предназначена пара классов **File** и **FileInfo**. С их помощью мы можем создавать, удалять, перемещать файлы, получать их свойства и многое другое.

Некоторые полезные методы и свойства класса FileInfo:

* **CopyTo(path)**: копирует файл в новое место по указанному пути path
* **Create()**: создает файл
* **Delete()**: удаляет файл
* **MoveTo(destFileName)**: перемещает файл в новое место
* Свойство **Directory**: получает родительский каталог в виде объекта DirectoryInfo
* Свойство **DirectoryName**: получает полный путь к родительскому каталогу
* Свойство **Exists**: указывает, существует ли файл
* Свойство **Length**: получает размер файла
* Свойство **Extension**: получает расширение файла
* Свойство **Name**: получает имя файла
* Свойство **FullName**: получает полное имя файла

Класс File реализует похожую функциональность с помощью статических методов:

* **Copy()**: копирует файл в новое место
* **Create()**: создает файл
* **Delete()**: удаляет файл
* **Move**: перемещает файл в новое место
* **Exists(file)**: определяет, существует ли файл

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

Эти классы используются для хранения (и извлечения) текстовой информации из файла. Эти классы не поддерживают произвольного доступа к файлу

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

Подобно классам StreamWriter/StreamReader, эти классы также работают с текстовой информацией. Однако лежащим в основе хранилищем является строковый буфер, а не физический файл

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

Кроме классов чтения-записи .NET предоставляет классы, которые позволяют сжимать файлы, а также затем восстанавливать их в исходное состояние.

Это классы **DeflateStream** и **GZipStream**, которые находятся в пространстве имен **System.IO.Compression** и представляют реализацию одного из алгоритмов сжатия Deflate или GZip.

1. **Расскажите алгоритм сжатия GZip.**

GZIP обеспечивает сжатие без потерь, иными словами, исходные данные можно полностью восстановить при распаковке. Он основан на алгоритме DEFLATE, который использует комбинацию алгоритма LZ77 и алгоритма Хаффмана

1. **Что такое тестовый и бинарный файл. Как можно выполнить чтение и запись в эти файлы.**
2. **Что такое произвольный доступ к файлу? Приведите пример.**

В предыдущих примерах использовались *последовательные файлы*, т.е. файлы со строго линейным доступом, байт за байтом. Но доступ к содержимому файла может быть и произвольным. Для этого служит, в частности, методSeek()

**10.Как применяется конструкция using (не директива) при работе с файловыми потоками? Для чего ее используют.**

Ключевое слово Using упрощает работу с объектами которые реализуют интерфейс IDisposable.  
  
Интерфейс IDisposable содержит один метод .Dispose(), который используется для освобождения ресурсов, которые захватил объект. При использовании Using не обязательно явно вызывать .Dispose() для объекта.

**using** (SqlConnection conn = **new** SqlConnection()) {   
// какая-нибудь SQL операция   
}

При этом компилятор генерирует следующий код:

SqlConnection conn = **new** SqlConnection();   
**try** {   
   
} **finally** {   
    // здесь для conn вызывается .Dispose()   
}

Using блоки делают код более читабельным и компактным.