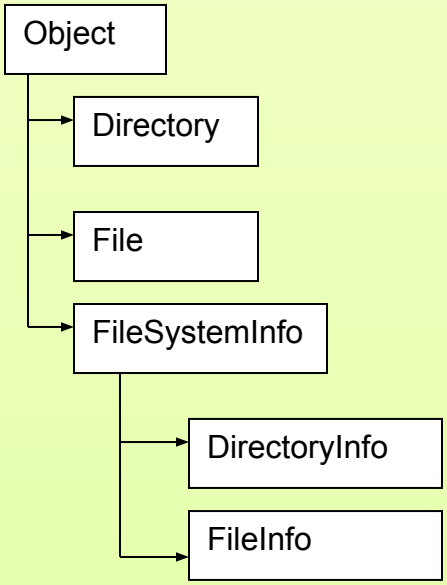
Моделі обміну даними

1. Простір імен System.IO
2. Класи, що надають потоки Stream
3. Класи System.IO для роботи з символьними даними
4. Serialization

System.IO

- Простір імен **System.IO** містить набір типів для виконання операцій з файлами і іншими об’єктами введення-виведення.
- Всі типи **System.IO** знаходяться в бібліотеці **mscorlib.dll.**
- Більшість класів призначені для роботи з каталогами і файлами на диску, а також з буфером в оперативній пам’яті чи областями оперативної пам’яті



System.IO Namespace -1

Class	Description
BinaryReader	Reads primitive data types as binary values in a specific encoding.
BinaryWriter	Writes primitive types in binary to a stream and supports writing strings in a specific encoding.
BufferedStream	Надає буфер для зчитування-запису з іншого потоку.
Directory	Надають статичні члени для створення, переміщення, видалення, отримання інформації про каталоги, підкаталоги в файловій системі.
DirectoryInfo	призначені для створення, переміщення, видалення, отримання інформації про каталоги, підкаталоги в файловій системі.
File	Provides static methods for the creation, copying, deletion, moving, and opening of files, and aids in the creation of FileStream objects.
FileInfo	дозволяє отримати інформацію про файл, а також здійснювати різні операції, наприклад по створенню і видаленню
FileStream	Exposes a Stream around a file, supporting both synchronous and asynchronous read and write operations.

System.IO Namespace-2

FileSystemInfo	призначені для роботи із загальними характеристиками файлу FileInfo чи каталогу DirectoryInfo.
IOException	The exception that is thrown when an I/O error occurs.
MemoryStream	Надає потік для запису в пам’ять.
Stream	Provides a generic view of a sequence of bytes.
StreamReader	Implements a TextReader that reads characters from a byte stream in a particular encoding.
StreamWriter	Implements a TextWriter for writing characters to a stream in a particular encoding.
StringReader	Implements a TextReader that reads from a string.
StringWriter	Implements a TextWriter for writing information to a string. The information is stored in an underlying StringBuilder.
TextReader	Represents a reader that can read a sequential series of characters.
TextWriter	Represents a writer that can write a sequential series of characters. This class is abstract.

The Abstract FileSystemInfo Base Class

Property	Meaning in Life
`Attributes`	Gets or sets the attributes associated with the current file that are represented by the `FileAttributes` enumeration (e.g., is the file or directory read-only, encrypted, hidden, or compressed?).
`CreationTime`	Gets or sets the time of creation for the current file or directory.
`Exists`	You can use this to determine whether a given file or directory exists.
`Extension`	Retrieves a file's extension.
`FullName`	Gets the full path of the directory or file.
`LastAccessTime`	Gets or sets the time the current file or directory was last accessed.
`LastWriteTime`	Gets or sets the time when the current file or directory was last written to.
`Name`	Obtains the name of the current file or directory.

DirectoryInfo Type

Member	Meaning in Life
`Create()` `CreateSubdirectory()`	Create a directory (or set of subdirectories) when given a path name.
`Delete()`	Deletes a directory and all its contents.
`GetDirectories()`	Returns an array of `DirectoryInfo` objects that represent all subdirectories in the current directory.
`GetFiles()`	Retrieves an array of `FileInfo` objects that represent a set of files in the given directory.
`MoveTo()`	Moves a directory and its contents to a new path.
`Parent`	Retrieves the parent directory of this directory.
`Root`	Gets the root portion of a path.

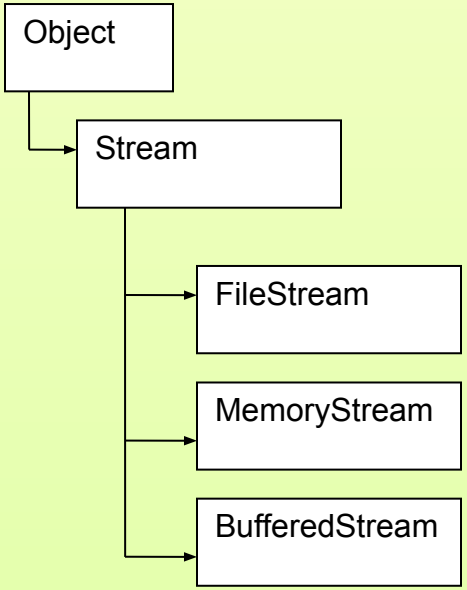
FileInfo Class

Member	Meaning in Life
`AppendText()`	Creates a `StreamWriter` object (described later) that appends text to a file.
`CopyTo()`	Copies an existing file to a new file.
`Create()`	Creates a new file and returns a `FileStream` object (described later) to interact with the newly created file.
`CreateText()`	Creates a `StreamWriter` object that writes a new text file.
`Delete()`	Deletes the file to which a `FileInfo` instance is bound.
`Directory`	Gets an instance of the parent directory.
`DirectoryName`	Gets the full path to the parent directory.
`Length`	Gets the size of the current file.
`MoveTo()`	Moves a specified file to a new location, providing the option to specify a new file name.
`Name`	Gets the name of the file.

class Stream

stream (потік) – сутність, що використовується для роботи з блоками даних.

- **Stream** – абстрактний клас для всіх потоків.
- потік – це абстракція послідовності байтів, таких як файл, пристрій введення/виведення, область в оперативній пам'яті
- **клас Stream** забезпечує як синхронну так і асинхронну взаємодію з середовищем зберігання даних (файлом на диску чи областю в оперативній пам'яті).
- класи, похідні від **Stream** призначені для роботи з блоками двійкових даних, підтримують пошук в потоці даних.



Основні операції

- **Читання з потоку** – переміщення даних з потоку в структуру даних, таку як масив бітів (**Read**(), **ReadByte**()).
- **Запис в потік** – переміщення даних зі структури даних в потік (**Write**(), **WriteByte**()).
- **Пошук** – запит і модифікація поточної позиції в потоці (**Seek**()).

Public Properties

Name	Description
CanRead	When overridden in a derived class, gets a value indicating whether the current stream supports reading.
CanSeek	When overridden in a derived class, gets a value indicating whether the current stream supports seeking.
CanTimeout	Gets a value that determines whether the current stream can time out.
CanWrite	Повертає величину, яка вказує чи даний потік може бути використаний на запис.
Length	Довжина потоку в байтах
Position	Повертає або встановлює позицію в потоці.
ReadTimeout	Gets or sets a value that determines how long the stream will attempt to read before timing out.
WriteTimeout	Gets or sets a value that determines how long the stream will attempt to write before timing out.

Public Methods -1

Name	Description
BeginRead	Begins an asynchronous read operation.
BeginWrite	Begins an asynchronous write operation.
Close	Закриває потік і звільняє всі ресурси, пов’язані з цим потоком.
CreateObjRef	Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
Dispose	Overloaded. Releases all resources used by the Stream object.
EndRead	Waits for the pending asynchronous read to complete.
EndWrite	Ends an asynchronous write operation.
Flush	Очищає всі буфери для даного потоку і записує всі дані буфера у вибраний пристрій.

Public Methods -2

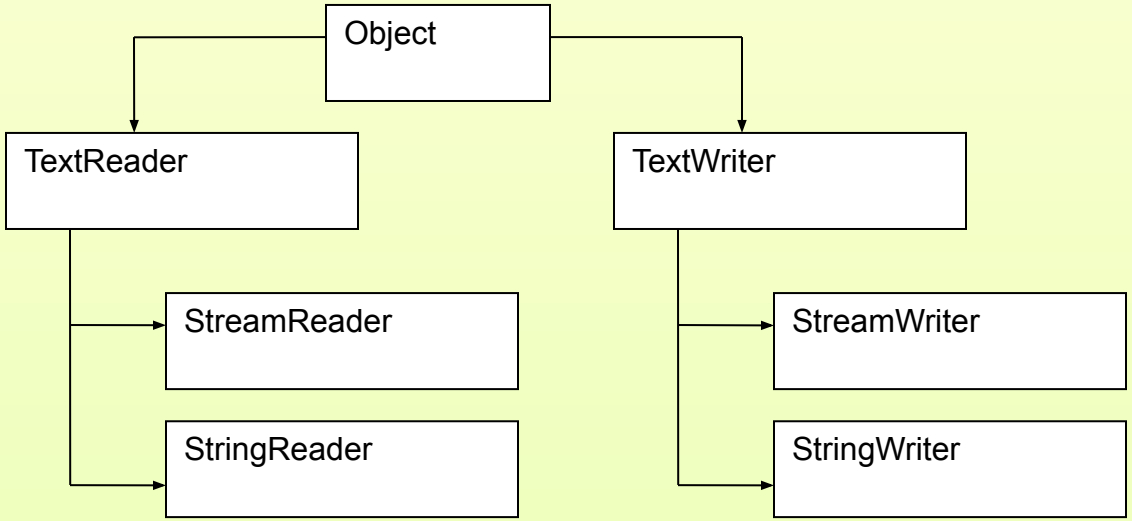
GetLifetimeService	Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
InitializeLifetimeService	Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
Read	When overridden in a derived class, reads a sequence of bytes from the current stream and advances the position within the stream by the number of bytes read.
ReadByte	Reads a byte from the stream and advances the position within the stream by one byte, or returns -1 if at the end of the stream.
Seek	When overridden in a derived class, sets the position within the current stream.
Synchronized	Creates a thread-safe (synchronized) wrapper around the specified **Stream** object.
Write	When overridden in a derived class, writes a sequence of bytes to the current stream and advances the current position within this stream by the number of bytes written.
WriteByte	Writes a byte to the current position in the stream and advances the position within the stream by one byte.

class FileStream

- Клас **FileStream** забезпечує реалізацію абстрактних членів класу **Stream** для роботи з файлами на диску.
- дозволяє відкрити існуючі файли і створити нові, використовуються значення з перелічуваних типів **FileMode, FileAccess, FileShare.**

```
FileStream mys=new FileStream("test.dat",
                FileMode.OpenOrCreate,FileAccess.ReadWrite);
mys.WriteByte((byte)6);
mys.Position=0;
Console.Write(mys.ReadByte());
mys.Close();
```

Класи System.IO для роботи з символьними даними.



Абстрактні класи **TextReader** та **TextWriter** забезпечують похідним класам набір можливостей по зчитуванню та запису символьних даних. За замовчуванням ці типи працюють з кодуванням Unicode.

Класи **StreamReader** та **StreamWriter** дають змогу зчитувати та записувати символьні дані в потік.

Класи **StringWriter і StringReader** дозволяють звертатися до текстової інформації як до потоку в оперативній пам’яті.

```
FileInfo f= new FileInfo(“temp.txt”);
//отримуємо об’єкт StreamWriter і записуємо в файл декілька рядків тексту
StreamWriter w=f.CreateText();
w.WriteLine(“First line”);
w.WriteLine(“Second line”);
for (int i=0;i<10;i++0;
{
    w.Write(i+” “);
}
// вставляємо символ нового рядка
w.Write(w.NewLine);
w.Close();


//для зчитування інформації з файлу:
StreamReader r=File.OpenText(“temp.txt”);
string inp=r.ReadToEnd();
r.Close();
```

What is Serialization?

Serialization (**Серіалізація) –** процес перетворення стану об’єкта (чи набору взаємопов’язаних об’єктів) в спеціальне представлення (наприклад, в форматі XML чи двійковому форматі), яке може бути розміщено в потік і далі відновлено з нього.
Deserialization (Десеріалізація) – зворотній процес відновлення об’єкта в його оригінальному стані з потоку байтів.

```
[Serializable]
 public class Person
 {
 …
 }
```

```
Person st1 = new Person();
 st1.FirstName = “Iryna";
 st1.LastName = “Koval";
 st1.BirthDate = new DateTime(1981, 8, 17);
```

Deserialization

Serialization

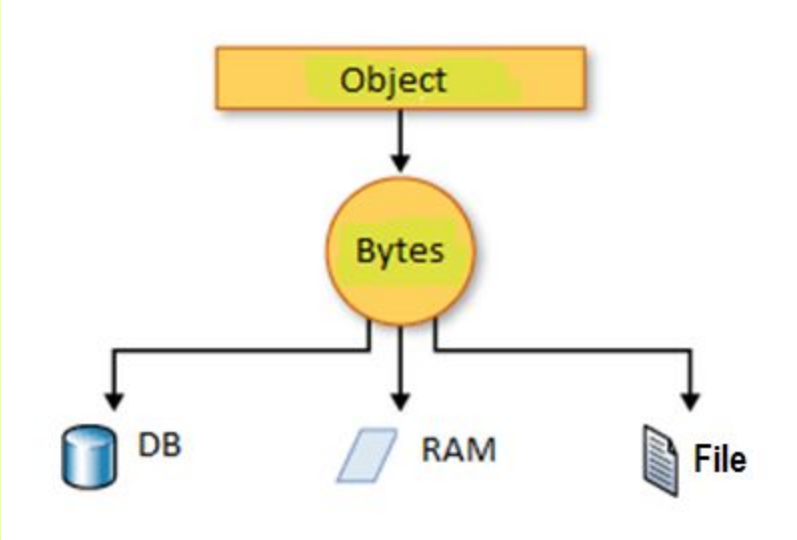
```
AC ED 00 05 73 72 00 0A 53 65 72 69 61 6C 54 65
73 74 A0 0C 34 00 FE B1 DD F9 02 00 02 42 00 05
63 6F 75 6E 74 42 00 07 76 65 72 73 69 6F 6E 78
70 00 64
```

What is Serialization?

Serialization/Deserialization найчастіше використовується для транспортування об’єктів (віддалена взаємодія) або зберігання об’єктів (в файлі чи базі даних).

Серіалізація містить дані про стан об’єкта та його зв’язки з іншими класами, дані про збірку і ін.

Набір взаємопов’язаних об’єктів, сериалізованих в потік, називається ***графом об’єктів***. Графи дозволяють фіксувати відношення між об’єктами



Серіалізація об’єктів в .NET

В бібліотеці класів .NET є два окремі механізми серіалізації:

- XmlSerializer (використовується для Web Services)
- SoapFormatter/BinaryFormatter (для зберігання даних)

  * **BinaryFormatter** сериалізує об’єктний граф в компактному потоці двійкового формату

  ***SoapFormatter** представляє граф як повідомлення протоколу **SOAP** (Simple Object Access Protocol – простий протокол доступу до об’єктів) в форматі **XML**.
- Можна здійснювати серіалізацію у власному форматі. Для цього клас повинен реалізувати інтерфейси **IFormatter** і **ISerializable** з простору імен **System.Runtime.Serialization**

XmlSerializer, SoapFormatter чи BinaryFormatter

XmlSerializer

обмеження (для класу необхідність конструктора без параметрів, тільки public read/write властивості та поля можуть бути серіалізовані, вимагається додавання атрибуту [Serializable] до класу).

переваги (добра підтримка для налаштування XML документів)

XmlSerializer найбільш зручна для міжплатформової роботи або для конструювання об’єктів з існуючих XML документів.

SoapFormatter і BinaryFormatter

обмеження (вимагається додавання атрибуту [Serializable] до класу)

переваги (можуть серіалізувати приватні поля)

BinaryFormatter найбільш ефективний спосіб при здійсненні серіалізації-десеріалізації на платформі .NET.

class XmlSerializer

- серіалізує і десеріалізує лише поля і властивості, оголошені `public`, ігнорує інші
- клас для серіалізації – `public`, з конструктором без аргументів, використовується відображення
- поля складних типів серіалізуються як вкладені, ті ж вимоги до класу
- для ігнорування відкритого поля атрибут `[XmlIgnore]`
- **XmlSerializer** не застосовна для класів, що реалізують інтерфейс IDictionary, наприклад Hashtable.

public class XmlSerializer

Namespace: System.Xml.Serialization

Name (9 overload)	**Description**
XmlSerializer ()	за замовчуванням
XmlSerializer (Type)	Новий об’єкт **XmlSerializer** може серіалізувати об’єкти вказаного типу в документи XML і десеріалізувати документи XML documents в об’єкти вказаного типу.
XmlSerializer (XmlTypeMapping)	використовуючи об’єкт, що відображає один тип в інший
XmlSerializer (Type, String)	Зазначає вказаний простір імен для всіх XML елементів.

Public Method `Serialize`

Серіалізує об’єкт в документ XML.

Name (9 overload)	Description
Serialize (Object, XmlSerializationWriter)	Серіалізує вказаний об’єкт і записує XML документ використовуючи вказаний XmlSerializationWriter.
Serialize (Stream, Object)	Серіалізує вказаний об’єкт і записує XML документ у файл Stream.
Serialize (TextWriter, Object)	Серіалізує вказаний об’єкт і записує XML документ у файл, використовуючи вказаний TextWriter.
Serialize (XmlWriter, Object)	Серіалізує вказаний об’єкт і записує XML документ у файл, використовуючи вказаний XmlWriter.

Public Method `Deserialize`

Десеріалізація XML документа.

Name (7 overload)	Description
`Object Deserialize (Stream)`	Десеріалізація XML документу з потоку Stream.
`Object Deserialize (TextReader)`	Deserializes the XML document contained by the specified TextReader.
`Object Deserialize (XmlReader)`	Deserializes the XML document contained by the specified XmlReader.
`protected virtual Object Deserialize (XmlSerializationReader)`	Deserializes the XML document contained by the specified XmlSerializationReader.
`Object Deserialize (XmlReader, String)`	Deserializes the XML document contained by the specified XmlReader and encoding style.

```
const string strFilter = "Person XML files(*.PersonXml)|" +
                         "*.PersonXml|All files (*.*)|*.*";
XmlSerializer xmlser = new XmlSerializer(typeof(Person));
```

```
void FileSaveAsOnClick(object objSrc, EventArgs args)
{
  SaveFileDialog dlg = new SaveFileDialog();
  dlg.Filter = strFilter;

  if (dlg.ShowDialog() == DialogResult.OK)
  {
     StreamWriter sw = new StreamWriter(dlg.FileName);
     xmlser.Serialize(sw, personpnl.Person);
     sw.Close();
  }
}
```

BinaryFormatter

- визначений в просторі імен **System.Runtime.Serialization.Formatters.Binary**
- серіалізує об’єктний граф в компактному потоці двійкового формату
- Для маркування членів, що не повинні серіалізуватися – атрибут [NonSerialized]
- Конструктор не викликається при десеріалізації

BinaryFormatter

```
[Serializable]
public class MyObject {
  public int n1 = 0;
  public int n2 = 0;
  public String str = null;
}
```

```
MyObject obj = new MyObject();
obj.n1 = 1;
obj.n2 = 24;
obj.str = "Some String";
IFormatter formatter = new BinaryFormatter();
Stream stream = new FileStream("MyFile.bin", FileMode.Create,
FileAccess.Write, FileShare.None);
formatter.Serialize(stream, obj);
stream.Close();
```

```
IFormatter formatter = new BinaryFormatter();
Stream stream = new FileStream("MyFile.bin", FileMode.Open,
FileAccess.Read, FileShare.Read);
MyObject obj = (MyObject) formatter.Deserialize(stream);
stream.Close();

Console.WriteLine("n1: {0}", obj.n1);
Console.WriteLine("n2: {0}", obj.n2);
Console.WriteLine("str: {0}", obj.str);
```

SoapFormatter

```
<SOAP-ENV:Envelope
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:SOAP- ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP- ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle=
  "http://schemas.microsoft.com/soap/encoding/clr/1.0"
  "http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:a1="http://schemas.microsoft.com/clr/assem/ToFile">

  <SOAP-ENV:Body>
    <a1:MyObject id="ref-1">
      <n1>1</n1>
      <n2>24</n2>
      <str id="ref-3">Some String</str>
    </a1:MyObject>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```