Lecture 4. File input-output. Serialization

Programming II

School of Business Informatics
Spring 2018

(: An optimist says: "The glass is half-full"
A pessimist says: "The glass is half-empty"
A programmer says: "The glass is twice as large as necessary" :)

Key points from last week

- Collections allow to store together related data
- They vary in their internal implementation, which in turn determines the efficiency of different operations
- The List is the most widely used dynamic container. It uses an array for data storage and supports dynamic resizing
- An associative container (Dictionary) stores key-value pairs and enabled quick lookup of values by their keys.

Permanent data storage

All the data that is created inside a program as objects, temporary variables, etc. is stored in volatile memory and is lost as soon as the program finishes or a power failure happens.

To save data between program executions it needs to be stored in **non-volatile** memory, e.g. hard drive, flash drive etc.

Files

- A file is a named sequence of bytes saved in non-volatile memory and is used for permanent data storage
- In all modern operating systems files are organized in a filesystem having hierarchical structure
- The file type determined by its extension, e.g. ".txt", ".jpg", ".exe"

File paths

Files/directories are addressed using absolute or relative paths:

- Absolute paths start with the drive letter, follow through all folders until the desired file is reached:
  C:/Users/Data/info.txt
- Relative paths are expressed in relation to the current directory (also called working directory):
  Data/info.txt
- Relative paths often contain special symbolic names:
  - . - references the current directory
  - .. - references the parent directory
- Both forward and backward slashes are allowed in the file path

Text files

- Text files store all information, including numbers, dates, etc. in some string representation
- Text files can be changed in any editor
- Application source code is also stored as a text file
- Encoding matters (today UTF-8 became the most commonly used encoding)

Binary files

- All information is stored in exactly the same form as it is stored in memory
- Faster IO - no need to convert to/from string
- Cannot be easily viewed or changed in other programs (editors)

.NET Standard classes: managing the file system

- **File** class - provides static methods for creating, copying, deleting, moving and opening files, reading files as text and binary
- **Directory** - provides static methods for creating, moving and enumerating through directories and subdirectories
- **Path** - provides methods and properties for processing directory strings

Path class

Use methods of the **Path** class instead of manual string manipulation for paths within the file system

Useful methods:

- Combine - combined several parts to form a single string path
- GetDirectoryName - extracts directory only
- GetFileNameWithoutExtension - extracts just the file name without directory info
- GetExtension - extracts the extension from a file path

Path methods do not check for actual presence of a file or directory on the hard drive. They only perform string manipulation.

.NET IO classes: reading/writing files

- **File** class - ReadAllXXX, WriteAllXXX methods to read/write the whole content of a file
- **FileStream** class - managing files as streams
- **BinaryReader/BinaryWriter** - reading and writing binary data. (used in combination with FileStream)
- **StreamReader/StreamWriter** - reading and writing textual data (can be used together with FileStream or on its own)

Why file streams?

- Representing a file as a stream allows to read data sequentially
- Reading a large file is never done in a single operation
- The idea of streams is widely used in computer architecture - network stream, memory stream, video stream etc.

File.ReadAllText or File.ReadAllBytes should only be used for relatively small files (dozen Mb at most)

Demo: file IO

See Lecture 4 supplement from Canvas

Data formats

Binary and text:

- Binary formats are opened by particular applications
- Text formats can be changed in any text editor

Human-readable and machine-readable:

- Human-readable formats contain both data and presentation details
- Machine-readable formats contain only data formatted in such a way that makes it is easy for automated machine processing

Open and proprietary:

- Proprietary formats don't have an open specification
- Open formats are well standardized

Demo: data formats

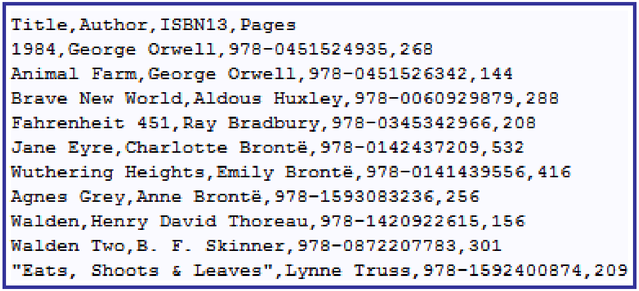
- Open the browser developer tab
- Open a doc or excel file in a standard notepad

Table of popular formats

	Binary	Text
Human-readable	DOC(X), XLS(X), PDF	HTML
Machine-readable	proprietary	CSV, XML, JSON

CSV format

- A CSV (comma-separated values) file stores tabular data in plain text
- A line in the text file represents a row of the table
- Columns in each line are separated by a delimiter (usually a comma)
- The format is not fully standardized



```
Title,Author,ISBN13,Pages
1984,George Orwell,978-0451524935,268
Animal Farm,George Orwell,978-0451526342,144
Brave New World,Aldous Huxley,978-0060929879,288
Fahrenheit 451,Ray Bradbury,978-0345342966,208
Jane Eyre,Charlotte Brontë,978-0142437209,532
Wuthering Heights,Emily Brontë,978-0141439556,416
Agnes Grey,Anne Brontë,978-1593083236,256
Walden,Henry David Thoreau,978-1420922615,156
Walden Two,B. F. Skinner,978-0872207783,301
"Eats, Shoots & Leaves",Lynne Truss,978-1592400874,209
```

XML

- Contains data within a special markup of tags and attributes
- Tags should be paired: opening (<book>) and closing (</book>)
- An opening tag can contain attributes (id="bk101")

```xml
<?xml version="1.0"?>
<catalog>
   <book id="bk101">
      <title>1984</title>
      <author>George Orwell</author>
      <isbn>978-0451524935</isbn>
      <pages>268</pages>
   </book>
   <book>
      <title>Animal Farm</title>
      ...
   </book>
</catalog>
```

JSON

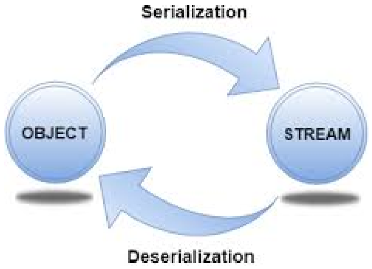
```
[
    {
        "id": "bk101",
        "title": "1984",
        "author": "George Orwell",
        "isbn": "978-0451524935",
        "pages": 268
    }

]
```

- Both XML and JSON are text formats that can be used to represent complex object-oriented data structures
- Compared to XML, JSON is a more compact format. It gained massive popularity in modern applications.

Serialization

Serialization is a process of converting an object state to a format that can be saved in a stream. It is meant to happen automatically based on the class definition.

Deserialization is an opposite process - restoring an object state from a stream.



.NET Framework: Serialization

- BinaryFormatter - binary serialization
- XmlSerializer - serialization to XML
- DataContractJsonSerializer or external NewtonsoftJson library - serialization to JSON

BinaryFormatter saves and restores the same object graph. With XML and JSON serialization we have to disjoin references in aggregated classes to remove unnecessary duplication (will be discussed more in detail at the seminar)