

18. What is serialization?

Brief summary: Serialization is the process of converting an object into a format that can be stored in memory or transmitted over a network. For example, the object can be converted into a text file containing JSON or XML, or a binary file.

Serialization is the process of converting an object into a format that can be stored in memory or transmitted over a network. For example, the object can be converted into a text file containing JSON or XML, or a binary file.

Deserialization is the opposite process - using the content of a file to recreate objects.

Let's write a program that reads personal data from the console, and then serializes it as an XML file. If the user restarts the program, this data can be reconstructed using the XML file stored in the computer's memory. Also, we could transfer this file to another computer, where it could also be used to recreate the object containing personal data. Let's see this in the code:

```
var name = Read("name");
var lastName = Read("last name");
var residence = Read("place of residence");
var hobby = Read("hobby");

var person = new Person(name, lastName, residence, hobby);

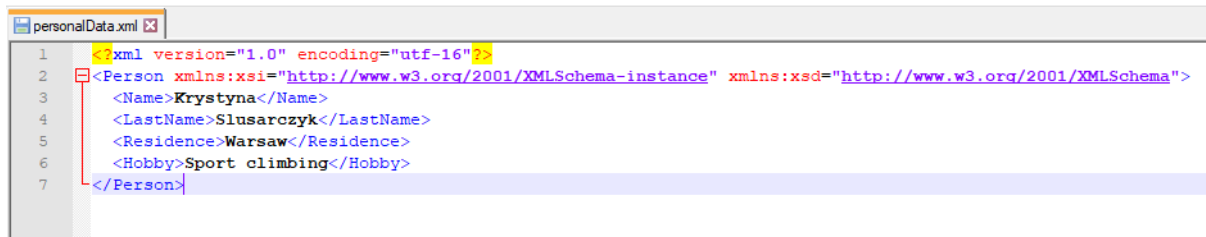
string xml = Serialize(person);
Console.WriteLine(
    $"The person object serialized to XML is:\n{xml}");

SaveXmlToFile("personalData.xml", xml);
```

```
static string Read(string what)
{
    Console.WriteLine($"Enter the {what}:");
    return Console.ReadLine();
}
```

For brevity, I skipped the code that does actual serialization and file writing. I used the built-in `XmlSerializer` class for serialization. Full code can be found in the solution published to Github.

I've run this code and entered some personal data. The file that was produced looks as follows:



```
personalData.xml
1 <?xml version="1.0" encoding="utf-16">
2 <Person xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
3   <Name>Krystyna</Name>
4   <LastName>Slusarczyk</LastName>
5   <Residence>Warsaw</Residence>
6   <Hobby>Sport climbing</Hobby>
7 </Person>
```

As you can see, all the information that was stored in the `Person` object is saved to the XML file. We should now be able to read it in the program. Let's change the code so if the "personalData.xml" file exists, it reads its content instead of asking the user to enter the data.

```
const string filePath = "personalData.xml";
if (File.Exists(filePath))
{
    using Stream reader = new FileStream(filePath, FileMode.Open);
    var xmlSerializer = new XmlSerializer(typeof(Person));

    StreamReader stream = new StreamReader(reader, Encoding.UTF8);
    var person = (Person?)xmlSerializer.Deserialize(
        new XmlTextReader(stream));
    Console.WriteLine($"Personal data read from xml:\n{person}");
}
else
{
    var name = Read("name");
    var lastName = Read("last name");
}
```

The code checks if the file already exists - if so, it reads its content and deserializes it to recreate the `Person` object.

Not only XML can be used as the format to store objects. One of the most common formats is JSON. The name comes from JavaScript Object Notation because it derives from JavaScript objects format. JSON is typically used for communication over a network. For example, when you fill a form on a website, most likely the data from the form is wrapped in JSON format and sent to the server, which then reads it and (in the case of the C# backend) translates it to C# objects.

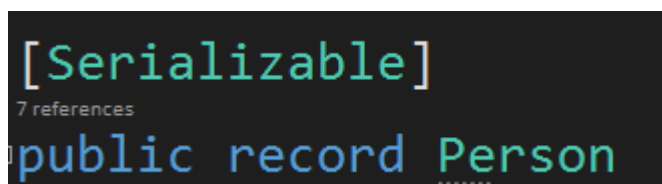
This is how the data showed previously as XML would look in JSON format:

```
1 {
2   "Name": "Krystyna",
3   "LastName": "Slusarczyk",
4   "Residence": "Warsaw",
5   "Hobby": "Sport climbing"
6 }
```

Probably the most popular library used for JSON serialization and deserialization is **JSON.Net** developed by **Newtonsoft**. It allows us to serialize and deserialize objects to/from JSON format very easily:

```
string json = JsonConvert.SerializeObject(person, Formatting.Indented);
Person recreatedPerson = JsonConvert.DeserializeObject<Person>(json);
```

One more thing that we should mention on this topic. The interviewer can ask you "What does the Serializable attribute do?".



This attribute indicates that instances of a class can be serialized with BinaryFormatter or SoapFormatter. It is not required for XML or JSON serialization. The BinaryFormatter serializes objects to a binary format (so, simply speaking, a chain of zeros and ones) and the SoapFormatter to the SOAP format, which is a little similar to XML. If you are curious, check out this article:

<https://pl.wikipedia.org/wiki/SOAP>

Let's summarize. Serialization is a process of translating objects and other data structures into a format that can be stored as a file or binary data, and potentially transmitted over a network. Serialized objects can later be reconstructed.

Bonus questions:

- **"What are the uses of serialization?"**

It can be used to send objects over a network, or to store objects in a file for later reconstruction, or even to store them in a database - for example to save a "snapshot" of an object every time a user makes some changes to it, so we can log the history of the changes.

- **"What does the Serializable attribute do?"**

This attribute indicates that instances of a class can be serialized with BinaryFormatter or SoapFormatter. It is not required for XML or JSON serialization.

- **"What is deserialization?"**

Deserialization is the opposite of serialization: it's using the content of a file to recreate objects.