Eksperymenty z serializacją i deserializacją

- Krzysztof Molenda (05.01.2021)
- Cel: wykonanie eksperymentów z serializacją i deserializacją obiektów w C#, stosując różne techniki i frameworki.
- Po wykonaniu ćwiczenia powinieneś rozumieć ideę serializacji/deserializacji oraz umieć zaimplementować ją w podstawowych wariantach.

Wykorzystując załączony kod, wykonaj serializację, a następnie deserializację obiektu typu `Person`, do pamięci (`MemoryStream`) oraz do pliku dyskowego (`FileStream`) stosując:

1. Serializację binarną
2. Serializację do formatu XML, wykorzystując `DataContract`
3. Serializację do formatu JSON (`System.Text.Json`)

Sprawdź działanie serializacji i deserializacji dla składników prywatnych oraz `readonly` (zmodyfikuj klasę `Person`).

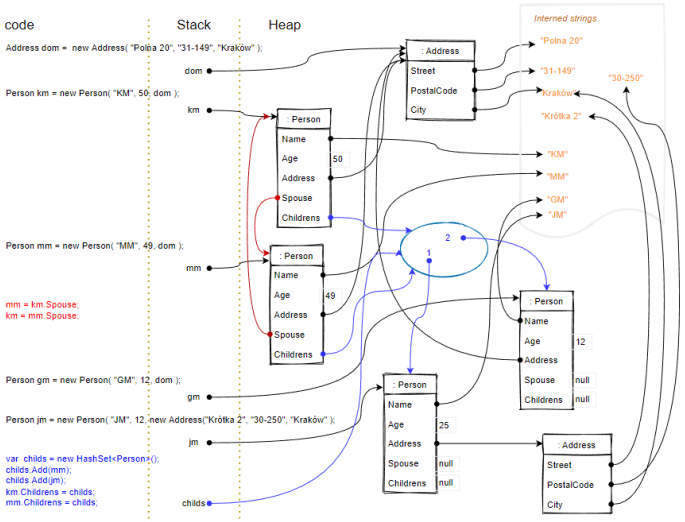
Sprawdź działanie serializacji i deserializacji włączając/wyłączając niektóre składniki.

Referencje:

- https://docs.microsoft.com/pl-pl/dotnet/standard/serialization/
- https://docs.microsoft.com/pl-pl/dotnet/framework/wcf/feature-details/serialization-and-deserialization
- https://docs.microsoft.com/pl-pl/dotnet/standard/serialization/system-text-json-overview

---

Poniższy kod startowy buduje graf obiektów:




```
using System;
using System.Collections.Generic;

namespace SerializationExamples
{
    class Program
    {

        static void Main(string[] args)
        {
            Console.WriteLine("Experiments with serialization");
            var dom = new Address("Polna 20", "31-149", "Kraków");
            var km = new Person("KM", 50, dom);
            var mm = new Person("MM", 40, dom);
            mm.Spouse = km;
            km.Spouse = mm;
            var gm = new Person("GM", 12, dom);
            var jm = new Person("JM", 25, new Address("Krótka 2", "30-250", "Kraków"));
            var childs = new HashSet<Person> { gm, jm };
            km.Childrens = mm.Childrens = childs;
        }

    }
}
```

```
using System;

namespace SerializationExamples
{
    public class Address : ICloneable
    {
        public string Street {get; set;}
        public string PostalCode {get; set;}
        public string City {get; set;}

        public Address(string street, string postalCode, string city)
        {
            Street = street; PostalCode = postalCode; City = city;
        }

        public override string ToString() => $"(HashCode: {this.GetHashCode()}: {Street}, {PostalCode} {City})";

        object ICloneable.Clone() => this.MemberwiseClone();
        public Address Clone() => (Address)this.MemberwiseClone();
    }
}
```

```
using System;
using System.Collections.Generic;

namespace SerializationExamples
{
    public class Person : ICloneable
```

```
    {
        public string Name {get; set;}
        public int Age {get; set;}
        public Address Address {get; set;}
        public Person Spouse {get; set;}
        public ICollection<Person> Childrens {get; set;} = null;

        public Person(string name, int age, Address address = null, Person spouse =  null)
        {
            Name = name;
            Age = age;
            Address = address;
            Spouse = spouse;
        }

        public Person(Person other): this(other.Name, other.Age, other.Address, other.Spouse)
        { }

        public override string ToString() => $"id: {GetHashCode()} -> name: {Name}, age: {Age}, address: {Addres

        object ICloneable.Clone() => this.MemberwiseClone();
        public Person Clone() => (Person)this.MemberwiseClone();

        public Person DeepClone()
        {
            Person clone = this.Clone();
            clone.Address = this.Address.Clone();
            clone.Spouse = null;
            clone.Childrens = null;
            return clone;
        }

    }
}
```