8. Übungsblatt Thema: Interfaces

Einführung:

In diesem Blatt sollen Interfaces anhand von Serialisierung und Deserialisierung eines Objekts geübt werden. Serialisierung verwandelt ein C# Objekt in einen String den man zum Beispiel in ein Text File abspeichern kann. Deserialisierung verwandelt einen String im richtigen Format wieder zurück in das entsprechende C# Objekt. Auf diese Weise kann Objekte eines Programms dauerhaft speichern und wieder laden.

Implementierung von Serialisierung eines Objekts vom Typ GameEntity

Beschreibung:

Die Klasse GameEntity hat folgende Struktur

```
{
  public class GameEntity
  {
    public int HP { get; set; }

    public int X { get; set; }

    public int Y { get; set; }

    public int Level { get; set; }

    public int Damage { get; set; }

    public int Defense { get; set; }

    public GameEntity(int hp, int x, int y, int level,int Damage, int defense)
    {
        // Some Code ...
    }

    public GameEntity() : this(0, 0, 0, 0, 0, 0) { }
}
```

Dabei implementiert diese Klasse ein Interface namens ISaveable

Interface ISaveable ist folgender maßen aufgebaut

```
public interface ISaveable
{
    string Save();

    void Load(string save);
}
```

Zuerst soll die Funktionalität für die Methode Save implementiert werden.

Format der Serialisierung:

Diese Methode gibt einen string zurück. Dieser String besteht aus mehren Zeilen. Jeder Zeile ist ein Paar aus dem Namen und Wert eines öffentlichen Property in der Klasse GameEntity. Name und Wert sind durch ein "=" getrennt. Die Paare werden mit Zeilen voneinander getrennt.

Test Program:

```
var entity = new GameEntity(100, 10, 0, 5, 12, 6);
Console.WriteLine(entity.Save());
Console.WriteLine($"{new string('=', 50)}");
entity = new GameEntity(200, -45, 45, 10, 2, 0);
Console.WriteLine(entity.Save());
```

Ausgabe sollte sein:

HP=100
X=10
Y=0
Level=5
Damage=12
Defense=6
==
HP=200
X=-45
Y=45
Level=10
Damage=2
Defense=0

---

2. Implementierung von Deserialisierung eines Objekts vom Typ GameEntity

Beschreibung:

Nun soll die methode Load des Interfaces implementiert werden. Diese Funktion soll einen String als Parameter bekommen. Es wird davon ausgegangen , dass der String im geeigneten Format vorliegt. Ist also ein Rückgabewert der Methode Save.

Nach der Ausführung soll die Instance die gleichen Property Werte haben wie die Werte aus dem Sting.

Test Program:

```
var entity = new GameEntity();
string savedFormat = "HP=100\n" +
    "X=0\n" +
    "Y=0\n" +
    "Level=2\n" +
    "Damage=5\n" +
    "Defense=8";
entity.Load(savedFormat);
Console.WriteLine(entity.ToString());
entity = new GameEntity(50, 0, 0, 10, 10, 5);
Console.WriteLine(entity.ToString());
savedFormat = "HP=10\n" +
    "X=5\n" +
    "Y=89\n" +
    "Level=4\n" +
    "Damage=2\n" +
    "Defense=1";
entity.Load(savedFormat);
Console.WriteLine(entity.ToString());
```

Ausgabe sollte sein:

HP: 100, X: 0, Y: 0, Level: 2, Damage: 5, Defense: 8
HP: 50, X: 0, Y: 0, Level: 10, Damage: 10, Defense: 5
HP: 10, X: 5, Y: 89, Level: 4, Damage: 2, Defense: 1

---

3. Implementierung von Exceptions im Falle von fehlerhaften Format

Beschreibung:

Es kann vorkommen, dass der String als Parameter für Load nicht richtig ist. Wir wollen bei diesen fehlerhaften Formaten eine Exception werfen. Dabei soll diese Exception uns genauere Informationen liefern was schief gelaufen ist.

Die geworfene Exception soll vom Type NoValidPropertyEntryException sein. Dieser Type soll von Exception erben.

In folgenden Fällen soll eine Exception geworfen werden.

- In einer Zeile kam kein = vor zu Trennung von Name und Wert. Die Exception soll folgende Fehlermeldung haben.
  "Entry for property is not in the following format (name=value)"
- In einer der Zeilen befindet sich auf der linken Seite ein Name für Property was nicht in der Klasse definiert ist. Die Exception soll folgende Fehlermeldung habenwerden
  "{propertyName} as property name is not valid".
  propertyName ist dabei der falscher Name der auf der linken Seite von = gefunden wurde. Außerdem soll die Exception unter dem Property namens PropertyName den Wert von "propertyName" haben.
- In einer der Zeilen kann der Wert auf der rechten Seite nicht in einen Integer umgewandelt werden. Die Exception soll folgende Fehlermeldung haben.
  "For Property {propertyName} the value is not convertible to an integer"
  Für propertyName gilt das gleiche wie beim oberen Punkt. Außerdem soll die Exception ein weiteres Property namens PropertyValue haben. Dieses Property hat den Wert, der nicht in einen Integer umgewandelt werden konnte.

Test Program:

```
var entity = new GameEntity();
string wrongFormat = "HP=100\n" +
    "X0\n" + // Error
    "Y=0\n" +
    "Level=2\n" +
    "Damage=5\n" +
    "Defense=8";
try
{
    entity.Load(wrongFormat);
}
catch (NoValidPropertyEntryException e)
{
    Console.WriteLine(e.Message);
}
wrongFormat = "MP=100\n" + // Error
    "X=0\n" +
    "Y=0\n" +
    "Level=2\n" +
    "Damage=5\n" +
    "Defense=8";
try
{
    entity.Load(wrongFormat);
}
catch (NoValidPropertyEntryException e)
{
    Console.WriteLine(e.Message);
    Console.WriteLine($"e.PropertyName = {e.PropertyName}");
}
wrongFormat = "HP=100\n" +
    "X=0\n" +
    "Y=0\n" +
    "Level=a\n" + // Error
    "Damage=5\n" +
    "Defense=8";
try
{
    entity.Load(wrongFormat);
}
catch (NoValidPropertyEntryException e)
{
    Console.WriteLine(e.Message);
    Console.WriteLine($"e.PropertyName = {e.PropertyName}");
    Console.WriteLine($"e.PropertyValue = {e.PropertyValue}");
}
```

Ausgabe sollte sein:
Entry for property does not consist of (name=value)
MP as property name is not valid

e.PropertyName = MP
For Property Level the value is not convertible to an integer
e.PropertyName = Level
e.PropertyValue = a

---

4. Implementierung von IEquatable in GameEntity

Beschreibung:

Die Klasse GameEntity soll nun auch das generische Interface IEquatable implementieren vom namespace System.

Ihr ist der Link zur Dokumentation des Interfaces https://docs.microsoft.com/en-us/dotnet/api/system.iequatable-1?view=net-5.0

Ein Beispiel Code wie Equals(T) für das Interface implementiert wird. https://docs.microsoft.com/en-us/dotnet/api/system.iequatable-1.equals?view=net-5.0

Tip: Hier muss nicht der Empfehlung nachkommen werden auch noch Equals(object) und GetHashCode zu implementieren laut der Dokumentation des Interfaces.

Test Program:

```
var entity = new GameEntity();
var otherEntity = new GameEntity(2, 24, 278, 2, 2, 10);
Console.WriteLine($"entity.Equals(otherEntity) = {entity.Equals(otherEntity)}");
otherEntity = new GameEntity();
Console.WriteLine($"entity.Equals(otherEntity) = {entity.Equals(otherEntity)}");
entity = new GameEntity(2, 24, 278, -2, 2, 10);
otherEntity = new GameEntity(2, 24, 278, 2, 2, 10);
Console.WriteLine($"entity.Equals(otherEntity) = {entity.Equals(otherEntity)}");
```

Ausgabe sollte sein: entity.Equals(otherEntity) = False
entity.Equals(otherEntity) = True
entity.Equals(otherEntity) = False
