Table of Contents

Einleitung

In diesem Abschnitt befinden sich Anleitungsartikel zur EMS.

Was sind Faktoren?

Faktoren sind Objekte, die durch Namen und mögliche Wertausprägungen beschrieben werden. Bei den Wertausprägungen wird zunächst zwischen komplexen und atomaren Faktoren unterschieden.

Komplexe Faktoren besitzen als mögliche Wertausprägungen Subfaktoren, diese können wiederum parallel (also zeitgleich) oder alternativ (ein Subfaktor pro Experiment) betrachtet werden.

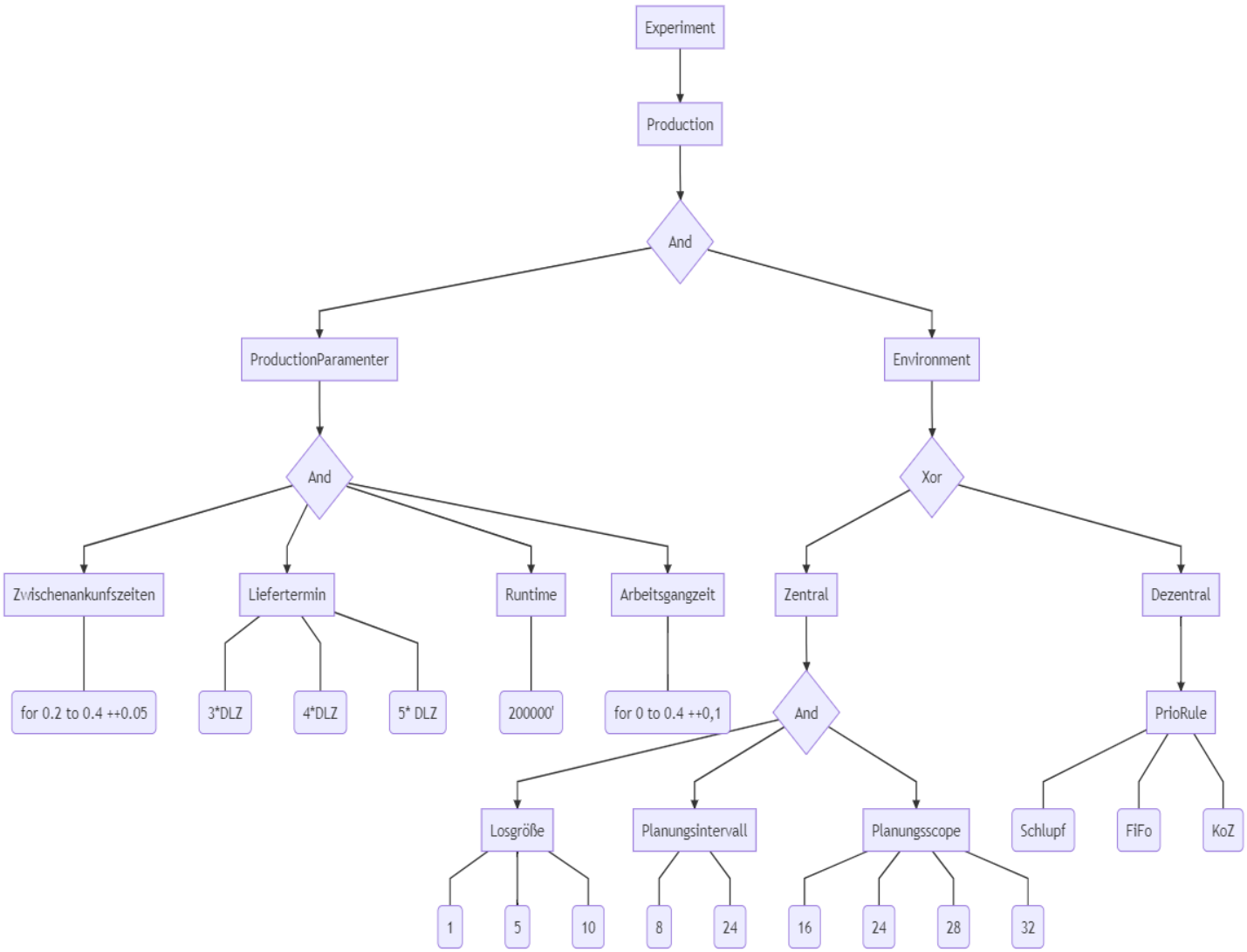
Atomare Faktoren können ihre Wertausprägungen aus einem diskreten oder kontinuielrichen Wertebereich beziehen.

Außerdem kann ein Faktor aktiviert oder deaktiviert sein. Deaktivierte Faktoren werden für ein Experiment nicht betrachtet.

Was ist ein Faktorbaum?

Ein Faktorbaum bildet Faktoren in einer Teil-Ganzes-Hierarchie ab, in dieser Struktur stehen die komplexen Faktoren für alle Knoten bis auf die Blattknoten. Blattknoten sind in einem Faktorbaum die atomaren Faktoren mit ihren Wertebereichen.

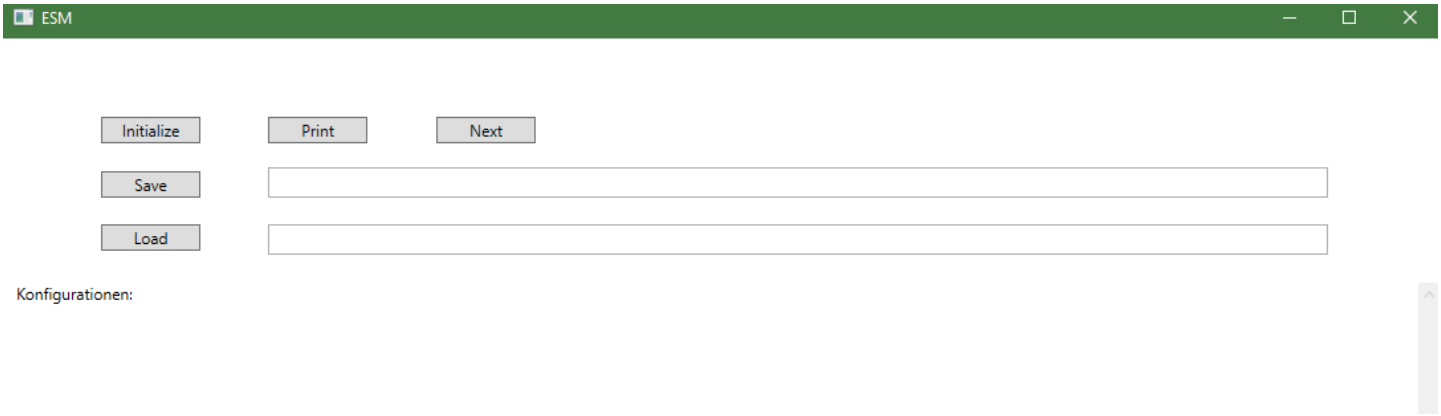
Ein Faktorbaum hat folgendes Erscheingungsbild:



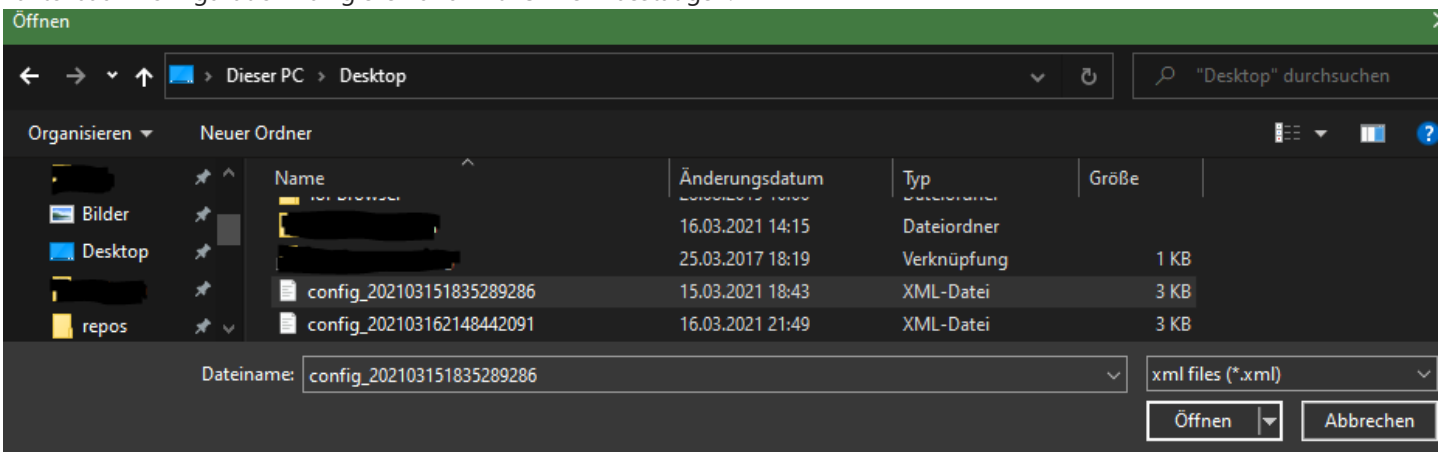
Anleitung

1. Faktorbaumkonfiguration laden

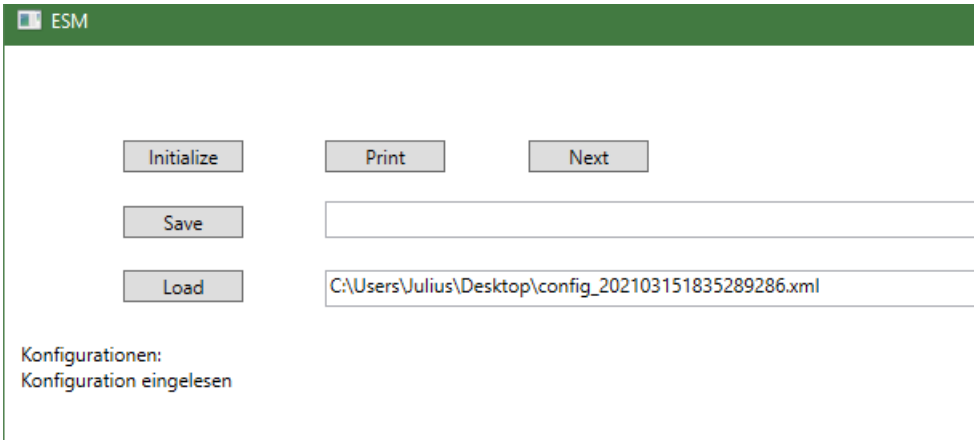
Nach Programmstart sieht man folgende Benutzeroberfläche. Hier haben Sie nun die Auswahl aus 5 Schaltflächen. Clicken Sie nun auf die Schaltfläche "Load" um einen Faktorbaum aus einer Konfiguration einzulesen.



Es öffnet sich nun ein Windows-Dialog zum öffnen einer XML-Datei, hier können Sie nun zu ihrer gewünschten Faktorbaumkonfiguration navigieren und mit "Öffnen" bestätigen.

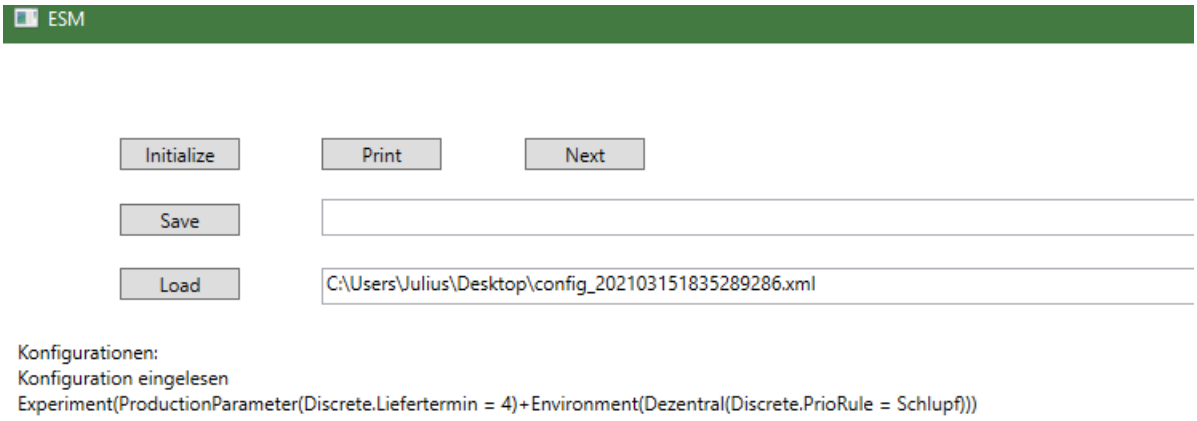


Nach dem öffnen der Datei erscheint nun der Dateipfad in dem Feld neben der "Load"-Schaltfläche und es erscheint eine Statusmeldung in dem Textblock der unteren Hälfte der Benutzeroberfläche.



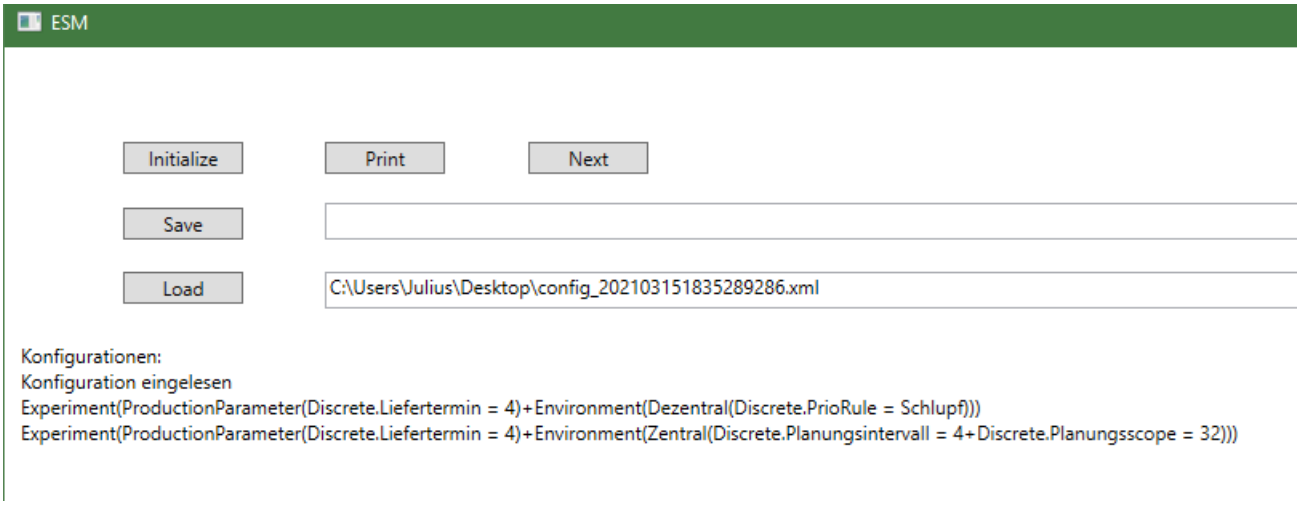
2. Faktorbaumkonfiguration ausgeben

Nachdem eine Konfiguration eingelesen wurde, clicken Sie auf die Schaltfläche "Print" um sich die aktuelle Konfiguration ausgeben zu lassen.

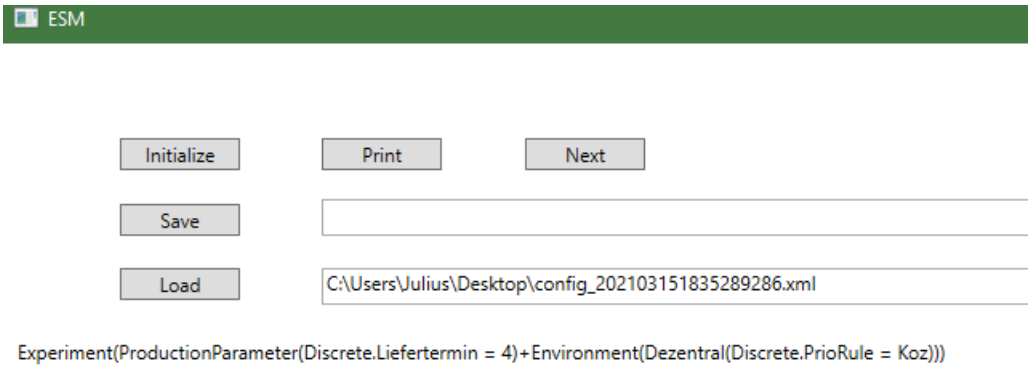


3. Faktorbaumkonfiguration weiterschalten und initialisieren

Wenn Sie eine neue Experimentkonfiguration basierend auf dem eingelesenen Faktorbaum erzeugen wollen, clicken Sie auf die Schaltfläche "Next", nun schaltet der Faktorbaum seine Wertausprägungen weiter.



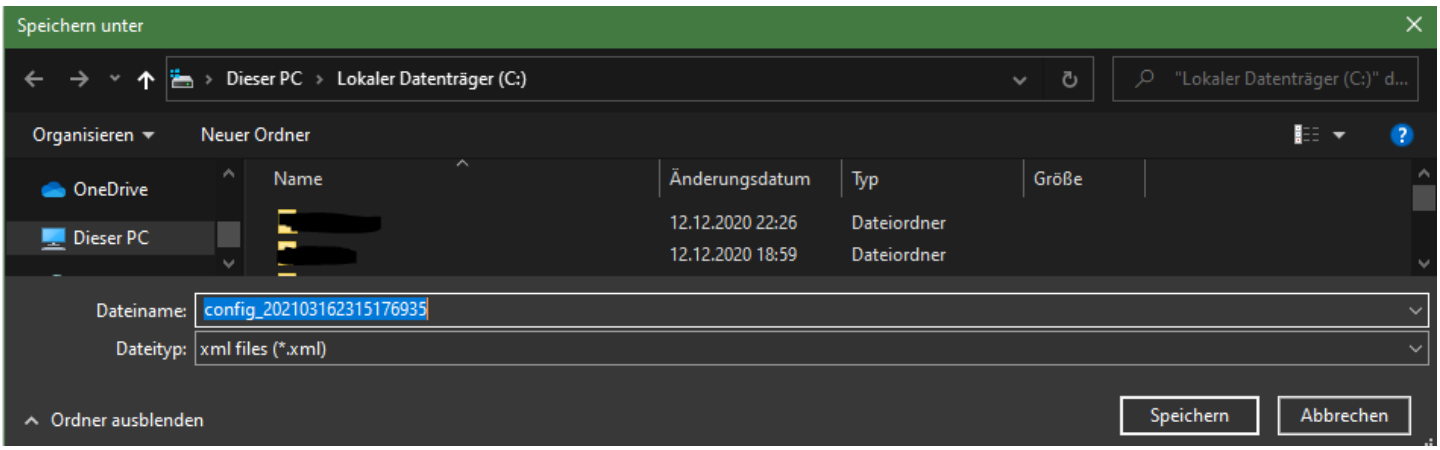
Wenn Sie merken, dass ihre eingelesene Konfiguration nicht dem Initialwert entspricht oder keine neuen Konfigurationen mehr möglich sind, können Sie den Faktorbaum über die Schaltfläche initialisieren.



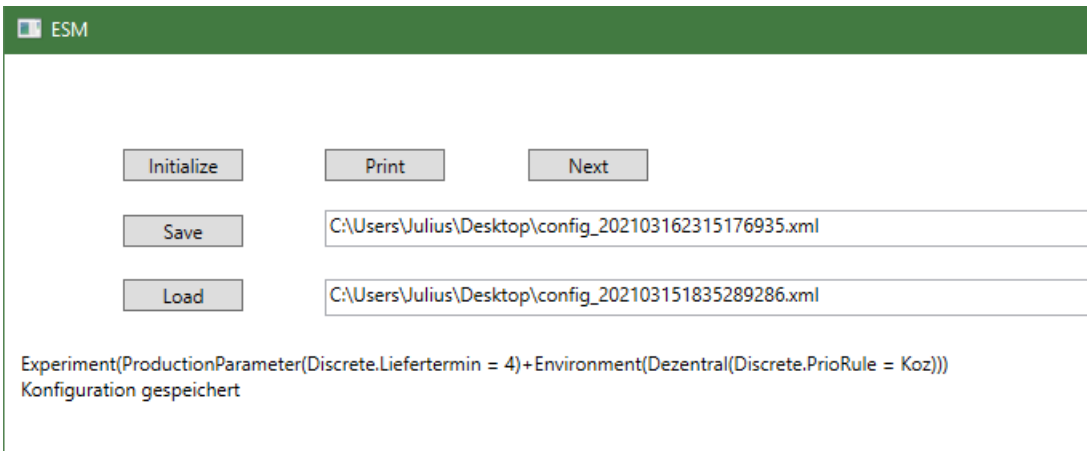
4. Faktorbaumkonfiguration speichern

Wenn Sie mit einer Konfiguration zufrieden sind, clicken Sie auf die Schaltfläche "Save". Sie sehen nun ein Dialogfeld zum Speichern der Datei, navigieren Sie zu ihrem gewünschten Verzeichnis und bestätigen Sie mit "Speichern". Die EMS benennt jede

zu speichernde Konfiguration automatisch mit einem Zeitstempel und Dopplungen auszuschließen, Sie können aber während des Speicherndialoges den Dateinamen ändern.



Nun erscheint eine Statusmeldung in dem Textblock und der Dateipfad erscheint neben der Schaltfläche "Save".



XML-Format eines Faktorbaums

Jede Experimentkonfiguration wird in einem serialisierbaren XML-Format gespeichert. Für Experimente sind die `<Factor>` Elemente mit ihren Attributen `FactorName` und `FactorValue` sowie `IsActive` relevant. Andere Elemente und Attribute dienen der EMS zum erzeugen des genauen Zustandes eines Faktorobjektes. Auf dieser Seite finden Sie eine Beispielkonfiguration die eingelesen werden kann, sowie die Schemadefinition der serialisierbaren XML-Dateien.

1 Beispiel Konfiguration eines Faktorbaums im XML Format

```
<?xml version="1.0" encoding="utf-8"?> <Factor xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xsi:type="FactorParallel" IsActive="true" FactorName="Experiment">
<Sub-Factors> <Factor xsi:type="FactorParallel" IsActive="true" FactorName="ProductionParameter"> <Sub-Factors>
<Factor xsi:type="Intervall" IsActive="true" FactorName="Zwischenankunftszeit" FactorValue="0,2">
<StartVal>0.2</StartVal> <EndVal>0.4</EndVal> <Increment>0.05</Increment> </Factor> </Sub-Factors>
<FactorIDX>0</FactorIDX> </Factor> <Factor xsi:type="FactorAlternative" IsActive="true"
FactorName="Environment"> <Sub-Factors> <Factor xsi:type="FactorParallel" IsActive="false"
FactorName="Zentral"> <Sub-Factors> <Factor xsi:type="ArrayValueOfInt32" IsActive="false"
FactorName="Planungsintervall" FactorValue="4"> <Values> <int>4</int> <int>8</int> </Values> <ValIDX>0</ValIDX>
</Factor> </Sub-Factors> <FactorIDX>0</FactorIDX> </Factor> <Factor xsi:type="FactorParallel" IsActive="true"
FactorName="Dezentral"> <Sub-Factors> <Factor xsi:type="ArrayValueOfString" IsActive="true"
FactorName="PrioRule" FactorValue="Koz"> <Values> <string>Koz</string> <string>FiFo</string>
<string>Schlupf</string> </Values> <ValIDX>0</ValIDX> </Factor> </Sub-Factors> <FactorIDX>0</FactorIDX>
</Factor> </Sub-Factors> <FactorIDX>1</FactorIDX> </Factor> </Sub-Factors> <FactorIDX>1</FactorIDX> </Factor>
```

2 Schemadefinition

Die Schemadefinition einer lesbaren Faktorbaumkonfiguration wird durch folgendes .xsd Formate beschrieben:

```
<?xml version="1.0" encoding="utf-8"?> <xs:schema xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xs="http://www.w3.org/2001/XMLSchema"
attributeFormDefault="unqualified" elementFormDefault="qualified"> <xsd:element name="Factor">
<xsd:complexType> <xsd:sequence> <xsd:element name="Sub-Factors"> <xsd:complexType> <xsd:sequence> <xsd:element
maxOccurs="unbounded" name="Factor"> <xsd:complexType> <xsd:sequence> <xsd:element name="Sub-Factors">
<xsd:complexType> <xsd:sequence> <xsd:element maxOccurs="unbounded" name="Factor"> <xsd:complexType>
<xsd:sequence> <xsd:element minOccurs="0" name="Sub-Factors"> <xsd:complexType> <xsd:sequence> <xsd:element
maxOccurs="unbounded" name="Factor"> <xsd:complexType> <xsd:sequence> <xsd:element name="Values">
<xsd:complexType> <xsd:sequence> <xsd:element minOccurs="0" maxOccurs="unbounded" name="string"
type="xsd:string" /> <xsd:element minOccurs="0" maxOccurs="unbounded" name="int" type="xsd:unsignedByte" />
</xsd:sequence> </xsd:complexType> </xsd:element> <xsd:element name="ValIDX" type="xsd:unsignedByte" />
</xsd:sequence> <xsd:attribute name="IsActive" type="xsd:boolean" use="required" /> <xsd:attribute
name="FactorName" type="xsd:string" use="required" /> <xsd:attribute name="FactorValue" type="xsd:string"
use="required" /> </xsd:complexType> </xsd:element> </xsd:sequence> </xsd:complexType> </xsd:element>
<xsd:element minOccurs="0" name="FactorIDX" type="xsd:unsignedByte" /> <xsd:element minOccurs="0"
name="Values"> <xsd:complexType> <xsd:sequence> <xsd:element maxOccurs="unbounded" name="int"
type="xsd:unsignedShort" /> </xsd:sequence> </xsd:complexType> </xsd:element> <xsd:element minOccurs="0"
name="ValIDX" type="xsd:unsignedByte" /> <xsd:element minOccurs="0" name="StartVal" type="xsd:decimal" />
<xsd:element minOccurs="0" name="EndVal" type="xsd:decimal" /> <xsd:element minOccurs="0" name="Increment"
type="xsd:decimal" /> </xsd:sequence> <xsd:attribute name="IsActive" type="xsd:boolean" use="required" />
<xsd:attribute name="FactorName" type="xsd:string" use="required" /> <xsd:attribute name="FactorValue"
type="xsd:string" use="optional" /> </xsd:complexType> </xsd:element> </xsd:sequence> </xsd:complexType>
</xsd:element> <xsd:element name="FactorIDX" type="xsd:unsignedByte" /> </xsd:sequence> <xsd:attribute
name="IsActive" type="xsd:boolean" use="required" /> <xsd:attribute name="FactorName" type="xsd:string"
use="required" /> </xsd:complexType> </xsd:element> </xsd:sequence> </xsd:complexType> </xsd:element>
<xsd:element name="FactorIDX" type="xsd:unsignedByte" /> </xsd:sequence> <xsd:attribute name="IsActive"
type="xsd:boolean" use="required" /> <xsd:attribute name="FactorName" type="xsd:string" use="required" />
</xsd:complexType> </xsd:element> </xs:schema>
```

Namespace EMS

Classes

App

Interaktionslogik für "App.xaml"

MainWindow

Interaktionslogik für MainWindow.xaml

Class App

Interaktionslogik für "App.xaml"

Inheritance

System.Object
System.Windows.Threading.DispatcherObject
System.Windows.Application
App

Implements

System.Windows.Markup.IQueryAmbient

Namespace: EMS

Assembly: EMS.dll

Syntax

```
public class App : Application, IHaveResources, IQueryAmbient
```

Implements

System.Windows.Markup.IQueryAmbient

Class MainWindow

Interaktionslogik für MainWindow.xaml

Inheritance

System.Object
System.Windows.Threading.DispatcherObject
System.Windows.DependencyObject
System.Windows.Media.Visual
System.Windows.UIElement
System.Windows.FrameworkElement
System.Windows.Controls.Control
System.Windows.Controls.ContentControl
System.Windows.Window
MainWindow

Implements

System.Windows.Media.Animation.IAnimatable
System.Windows.IFrameworkInputElement
System.Windows.IInputElement
System.ComponentModel.ISupportInitialize
System.Windows.Markup.IQueryAmbient
System.Windows.Markup.IAddChild

Namespace: EMS

Assembly: EMS.dll

Syntax

```
public class MainWindow : Window, DUCE.IResource, IAnimatable, IFrameworkInputElement, IInputElement,
ISupportInitialize, IHaveResources, IQueryAmbient, IAddChild, IWindowService
```

Constructors

MainWindow()

Konstruktor der Klasse.

Declaration

```
public MainWindow()
```

Implements

System.Windows.Media.Animation.IAnimatable
System.Windows.IFrameworkInputElement
System.Windows.IInputElement
System.ComponentModel.ISupportInitialize
System.Windows.Markup.IQueryAmbient
System.Windows.Markup.IAddChild

Namespace EMSFactorClient

Classes

Client

Klasse zum Verwalten eines Objektes vom Typ Factor.

Class Client

Klasse zum Verwalten eines Objektes vom Typ Factor.

Inheritance

System.Object
Client

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: EMSFactorClient

Assembly: EMS.dll

Syntax

```
public class Client
```

Constructors

Client()

Konstruktor der Klasse

Declaration

```
public Client()
```

Client(TextBlock)

Konstruktor der Klasse.

Declaration

```
public Client(TextBlock textBlock)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Windows.Controls.TextBlock	textBlock	Definiert die Eigenschaft StateBox

Methods

Build(Factor, Factor)

Baut eine Baumstruktur auf und Prüft ob rootNode kein Blatt ist

Declaration

```
public void Build(Factor rootNode, Factor nextNode)
```

Parameters

TYPE	NAME	DESCRIPTION
Factor	rootNode	Faktor der als Wurzel dient
Factor	nextNode	Faktor der als weiterer Knoten dient

Initialize()

Initialisiert einen Faktorbaum und gibt diesen in einem Textblock (GUI) aus

Declaration

```
public void Initialize()
```

LoadConfig(String)

Lädt einen Faktorbaum aus einer XML-Datei

Declaration

```
public void LoadConfig(string configPath)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	configPath	Dateipfad der XML-Datei

Next()

Prüft ob ein Faktorbaum weitergeschalten werden kann und schaltet diesen weiter falls möglich. Sollte ein weiterschalten nicht möglich sein wird eine Meldung ein TextBlock-Objekt geschrieben.

Declaration

```
public void Next()
```

PrintTree()

Gibt die Baumstruktur in der Konsole aus

Declaration

```
public void PrintTree()
```

PrintTreeGUI()

Gibt den Baum auf der GUI aus

Declaration

```
public void PrintTreeGUI()
```

WriteConfig(String)

Exportiert eine Experimentkonfiguration in eine XML-Datei.

Declaration

```
public void WriteConfig(string configPath)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	configPath	Dateipfad der XML-Datei

Namespace EMSFactorClasses

Classes

ArrayValue<T>

Beschreibt ein Objekt welches Werte aus einem diskrete Werte in einem Array vom Typ T hält.

Factor

Abstrakte Oberklasse, von der alle Faktorklassen geerbt haben.

FactorAlternative

Mit dieser Klasse werden alternative Faktoren beschrieben. Implementiert das alternative Verhalten von Faktoren.

FactorComplex

Basisklasse für komplexe Faktoren. Implementiert das parallele Verhalten von Faktoren.

FactorContinuous

Basisklasse für kontinuierliche atomare Faktoren.

FactorDiscrete

Basisklasse für diskrete atomare Faktoren.

FactorLeaf

Basisklasse für atomare Faktoren.

FactorParallel

Diese Klasse beschreibt ermöglicht es parallele Faktoren abzubilden.

Intervall

Beschreibt ein Objekt, welches einen kontinuierlichen Wertebereich über ein Intervall abbildet.

Class Factor

Abstrakte Oberklasse, von der alle Faktorklassen geerbt haben.

Inheritance

System.Object
Factor
FactorComplex
FactorLeaf

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: EMSFactorClasses

Assembly: EMS.dll

Syntax

```
public abstract class Factor
```

Fields

IsActive

Eigenschaft die beschreibt ob ein Faktor aktiv ist.

Declaration

```
public bool IsActive
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

Properties

Name

Eigenschaft die den Namen eines Faktors beschreibt.

Declaration

```
public string Name { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Methods

Activate()

Setzt die IsActive-Eigenschaft auf true.

Declaration

```
public virtual void Activate()
```

AddNode(Factor)

Fügt ein Objekt vom Typ Factor einer Liste hinzu.

Declaration

```
public virtual void AddNode(Factor factor)
```

Parameters

TYPE	NAME	DESCRIPTION
Factor	factor	Objekt vom Typ Factor welches einer Liste angehangen werden soll.

Deactivate()

Setzt die IsActive-Eigenschaft auf false.

Declaration

```
public virtual void Deactivate()
```

GetNext()

Schaltet ein Factor-Ojekt auf seine nächste Wertausprägung.

Declaration

```
public abstract void GetNext()
```

HasNext()

Prüft ob ein Factor-Objekt weitere Werte annehmen kann.

Declaration

```
public virtual bool HasNext()
```

Returns

TYPE	DESCRIPTION
System.Boolean	

IsComposite()

Gibt Auskunft darüber ob die Klasse ein Kompositum ist.

Declaration

```
public virtual bool IsComposite()
```

Returns

TYPE	DESCRIPTION
System.Boolean	"true" da Kompositum

PrintConfig()

NOT USED

Declaration

```
public virtual string PrintConfig()
```

Returns

TYPE	DESCRIPTION
System.String	

PrintNodes()

Gibt alle Elemente in der Liste nodes als Zeichenkette zurück.

Declaration

```
public abstract string PrintNodes()
```

Returns

TYPE	DESCRIPTION
System.String	

RemoveNode(Factor)

Entfernt ein Objekt vom Typ Faktor aus einer Liste.

Declaration

```
public virtual void RemoveNode(Factor factor)
```

Parameters

TYPE	NAME	DESCRIPTION
Factor	factor	Objekt vom Typ Factor welches aus einer Liste entfernt werden soll.

SetInitVal()

Initialisiert ein Factor-Object.

Declaration

```
public abstract void SetInitVal()
```

Class FactorComplex

Basisklasse für komplexe Faktoren. Implementiert das parallele Verhalten von Faktoren.

Inheritance

System.Object
Factor
FactorComplex
FactorAlternative
FactorParallel

Inherited Members

Factor.Name
Factor.IsActive
Factor.IsComposite()
Factor.PrintConfig()
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: EMSFactorClasses

Assembly: EMS.dll

Syntax

```
public class FactorComplex : Factor
```

Fields

nodes

Liste vom Typ Factor in der die Objekte der Subfaktoren abgelegt werden.

Declaration

```
public List<Factor> nodes
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.List<Factor>	

Properties

FactorIDX

Index für die Liste nodes, startet beim letzten Eintrag.

Declaration

```
public int FactorIDX { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

Activate()

Setzt den Wert der Eigenschaft this.IsActive und nodes.IsActive auf true.

Declaration

```
public override void Activate()
```

Overrides

Factor.Activate()

AddNode(Factor)

Fügt der Liste nodes ein Objekt vom Typ Factor hinzu.

Declaration

```
public override void AddNode(Factor factor)
```

Parameters

TYPE	NAME	DESCRIPTION
Factor	factor	Objekt vom Typ Factor, welches hinzugefügt werden soll.

Overrides

Factor.AddNode(Factor)

Deactivate()

Setzt den Wert der Eigenschaft this.IsActive und nodes.IsActive auf false.

Declaration

```
public override void Deactivate()
```

Overrides

Factor.Deactivate()

GetNext()

Prüft ob ein Faktor in nodes[i] weitergeschalten werden kann und zählt i so lange herunter bis ein weiterschalten möglich ist. Initialisiert gegebenfalls alle Elemente ab nodes[i + 1]. Ruft für das Element in nodes[i] GetNext() auf.

Declaration

```
public override void GetNext()
```

Overrides

Factor.GetNext()

HasNext()

Prüft ob ein Faktor weitergeschalten werden kann, indem geprüft wird ob alle Elemente in der Liste nodes nicht mehr

weitergeschalten werden können.

Declaration

```
public override bool HasNext()
```

Returns

TYPE	DESCRIPTION
System.Boolean	Einen booleschen Wert: true-kann hochzählen; false-kann nicht mehr hochzählen

Overrides

Factor.HasNext()

IsParallel()

Gibt an ob es sich um einen parallelen oder alternativen Faktor handelt.

Declaration

```
public virtual bool IsParallel()
```

Returns

TYPE	DESCRIPTION
System.Boolean	true = parallel; false = alternativ

PrintNodes()

Gibt die Eigenschaft Name und alle Elemente in nodes als Zeichenkette zurück.

Declaration

```
public override string PrintNodes()
```

Returns

TYPE	DESCRIPTION
System.String	Eine Zeichenkette mit der Eigenschaft Name und allen Elementen in nodes.

Overrides

Factor.PrintNodes()

RemoveNode(Factor)

Entfernt ein Objekt vom Typ Faktor aus der Liste nodes.

Declaration

```
public override void RemoveNode(Factor factor)
```

Parameters

TYPE	NAME	DESCRIPTION
Factor	factor	Objekt vom Typ Factor, welches entfernt werden soll.

Overrides

Factor.RemoveNode(Factor)

SetInitVal()

Initialisiert alle Elemente in nodes.

Declaration

```
public override void SetInitVal()
```

Overrides

Factor.SetInitVal()

Class FactorParallel

Diese Klasse beschreibt ermöglicht es parallele Faktoren abzubilden.

Inheritance

System.Object
Factor
FactorComplex
FactorParallel

Inherited Members

FactorComplex.nodes
FactorComplex.FactorIDX
FactorComplex.AddNode(Factor)
FactorComplex.RemoveNode(Factor)
FactorComplex.PrintNodes()
FactorComplex.SetInitVal()
FactorComplex.GetNext()
FactorComplex.HasNext()
FactorComplex.IsParallel()
FactorComplex.Deactivate()
FactorComplex.Activate()
Factor.Name
Factor.IsActive
Factor.IsComposite()
Factor.PrintConfig()
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: EMSFactorClasses

Assembly: EMS.dll

Syntax

```
public class FactorParallel : FactorComplex
```

Constructors

FactorParallel()

Konstruktor der Klasse. Benötigt für Deserialisierung.

Declaration

```
public FactorParallel()
```

FactorParallel(String)

Konstruktor der Klasse.

Declaration

```
public FactorParallel(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Definiert die Eigenschaft Name.

Class FactorAlternative

Mit dieser Klasse werden alternative Faktoren beschrieben. Implementiert das alternative Verhalten von Faktoren.

Inheritance

System.Object
Factor
FactorComplex
FactorAlternative

Inherited Members

FactorComplex.nodes
FactorComplex.FactorIDX
FactorComplex.AddNode(Factor)
FactorComplex.RemoveNode(Factor)
FactorComplex.HasNext()
FactorComplex.Deactivate()
FactorComplex.Activate()
Factor.Name
Factor.IsActive
Factor.IsComposite()
Factor.PrintConfig()
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: EMSFactorClasses

Assembly: EMS.dll

Syntax

```
public class FactorAlternative : FactorComplex
```

Constructors

FactorAlternative()

Konstruktor der Klasse. Benötigt für Deserialisierung.

Declaration

```
public FactorAlternative()
```

FactorAlternative(String)

Konstruktor der Klasse.

Declaration

```
public FactorAlternative(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Definiert die Eigenschaft Name.

Methods

GetNext()

Prüft ob das Ojekt an der Stelle nodes[FactorIDX] weitergeschalten werden kann und dekrementiert FactorIDX gegebenenfalls solange bis es möglich ist. Wenn zum nächsten Subfaktor weitergeschalten wird, wird dieser aktiviert und die Elemente ab nodes[FactorIDX + 1] werden deaktiviert.

Declaration

```
public override void GetNext()
```

Overrides

FactorComplex.GetNext()

IsParallel()

Gibt an ob es sich um einen parallelen oder alternativen Faktor handelt.

Declaration

```
public override bool IsParallel()
```

Returns

TYPE	DESCRIPTION
System.Boolean	true = parallel; false = alternativ

Overrides

FactorComplex.IsParallel()

PrintNodes()

Gibt die Eigenschaft Name und das Element in nodes[FactorIDX] als Zeichenkette zurück.

Declaration

```
public override string PrintNodes()
```

Returns

TYPE	DESCRIPTION
System.String	Eine Zeichenkette mit der Eigenschaft Name und dem Element in nodes[FactorIDX].

Overrides

FactorComplex.PrintNodes()

SetInitVal()

Ruft für alle Elemente in nodes SetInitVal() auf und deaktiviert alle Elemente außer nodes[nodes.Count - 1].

Declaration

```
public override void SetInitVal()
```

Overrides

FactorComplex.SetInitVal()

Class FactorLeaf

Basisklasse für atomare Faktoren.

Inheritance

System.Object
Factor
FactorLeaf
FactorContinuous
FactorDiscrete

Inherited Members

Factor.Name
Factor.IsActive
Factor.AddNode(Factor)
Factor.RemoveNode(Factor)
Factor.Deactivate()
Factor.Activate()
Factor.PrintConfig()
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: EMSFactorClasses

Assembly: EMS.dll

Syntax

```
public class FactorLeaf : Factor
```

Properties

OutVal

Eigenschaft die den Wert eines Faktors beschreibt.

Declaration

```
public string OutVal { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Methods

GetNext()

Schaltet zum nächsten Faktorwert

Declaration

```
public override void GetNext()
```

Overrides

Factor.GetNext()

HasNext()

Prüft ob es einen weiteren Faktorwert gibt.

Declaration

```
public override bool HasNext()
```

Returns

TYPE	DESCRIPTION
System.Boolean	

Overrides

Factor.HasNext()

IsComposite()

Gibt Auskunft darüber ob die Klasse ein Kompositum ist.

Declaration

```
public override bool IsComposite()
```

Returns

TYPE	DESCRIPTION
System.Boolean	"false" da Blatt

Overrides

Factor.IsComposite()

PrintNodes()

Gibt "Leaf" als Zeichenkette zurück.

Declaration

```
public override string PrintNodes()
```

Returns

TYPE	DESCRIPTION
System.String	Leaf

Overrides

Factor.PrintNodes()

SetInitVal()

Setzt die Eigenschaft OutVal auf den Initialwert.

Declaration

```
public override void SetInitVal()
```

Overrides

Factor.SetInitVal()

Class FactorDiscrete

Basisklasse für diskrete atomare Faktoren.

Inheritance

System.Object
Factor
FactorLeaf
FactorDiscrete
ArrayValue<T>

Inherited Members

FactorLeaf.OutVal
FactorLeaf.GetNext()
FactorLeaf.HasNext()
FactorLeaf.IsComposite()
FactorLeaf.SetInitVal()
Factor.Name
Factor.IsActive
Factor.AddNode(Factor)
Factor.RemoveNode(Factor)
Factor.Deactivate()
Factor.Activate()
Factor.PrintConfig()
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: EMSFactorClasses

Assembly: EMS.dll

Syntax

```
public class FactorDiscrete : FactorLeaf
```

Methods

PrintNodes()

Gibt eine Zeichenkette mit den Werten der Eigenschaften Name und OutVal sowie eine Information dazu ob es ein diskreter oder kontinuierlicher Faktor ist.

Declaration

```
public override string PrintNodes()
```

Returns

TYPE	DESCRIPTION
System.String	Eine Zeichenkette mit Name und OutVal

Overrides

FactorLeaf.PrintNodes()

Class ArrayValue<T>

Beschreibt ein Objekt welches Werte aus einem diskrete Werte in einem Array vom Typ T hält.

Inheritance

System.Object
Factor
FactorLeaf
FactorDiscrete
ArrayValue<T>

Inherited Members

FactorDiscrete.PrintNodes()
FactorLeaf.OutVal
FactorLeaf.IsComposite()
Factor.Name
Factor.IsActive
Factor.AddNode(Factor)
Factor.RemoveNode(Factor)
Factor.Deactivate()
Factor.Activate()
Factor.PrintConfig()
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: EMSFactorClasses

Assembly: EMS.dll

Syntax

```
public class ArrayValue<T> : FactorDiscrete
```

Type Parameters

NAME	DESCRIPTION
T	Platzhalter für den Datentyp

Constructors

ArrayValue()

Konstruktor der Klasse. Benötigt für Deserialisierung.

Declaration

```
public ArrayValue()
```

ArrayValue(String, T[])

Konstruktor der generischen Klasse. Unterstützte Datentypen für die Serialisierung: string, int, double Weitere Datentypen müssen in FactorDiscrete.cs ergänzt werden. Setzt die Eigenschaft OutVal auf den Wert in Values[0].

Declaration

```
public ArrayValue(string name, T[] vals)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Definiert die Eigenschaft Name
T[]	vals	Definiert die Eigenschaft Values

Properties

ValIDX

Index für den in OutVal genutzten Wert aus Values[ValIDX].

Declaration

```
public int ValIDX { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Values

Generisches Array in dem alle möglichen Werte des Faktors gespeichert werden.

Declaration

```
public T[] Values { get; set; }
```

Property Value

TYPE	DESCRIPTION
T[]	

Methods

GetNext()

Inkrementiert die Eigenschaft ValIDX um 1 und setzt die Eigenschaft OutVal auf Values[ValIDX].

Declaration

```
public override void GetNext()
```

Overrides

FactorLeaf.GetNext()

HasNext()

Prüft ob ValIDX gleich Values.Length - 1 ist.

Declaration

```
public override bool HasNext()
```

Returns

TYPE	DESCRIPTION
System.Boolean	einen booleschen Wert: true-weitere Werte vorhanden; false-alle Werte wurden ausgegeben

Overrides

FactorLeaf.HasNext()

SetInitVal()

Setzt die Eigenschaft ValIDX auf 0. Setzt die Eigenschaft OutVal auf Values[0].

Declaration

```
public override void SetInitVal()
```

Overrides

FactorLeaf.SetInitVal()

ValToString(Int32)

Führt für Elemente in Values an der Stelle i die ToString()-Methode aus und gibt das Ergebnis zurück

Declaration

```
public string ValToString(int i)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	Index für die Eigenschaft Values

Returns

TYPE	DESCRIPTION
System.String	Eine Zeichenkette mit dem Wert aus Values[i]

Class FactorContinuous

Basisklasse für kontinuierliche atomare Faktoren.

Inheritance

System.Object
Factor
FactorLeaf
FactorContinuous
Intervall

Inherited Members

FactorLeaf.OutVal
FactorLeaf.GetNext()
FactorLeaf.HasNext()
FactorLeaf.IsComposite()
FactorLeaf.SetInitVal()
Factor.Name
Factor.IsActive
Factor.AddNode(Factor)
Factor.RemoveNode(Factor)
Factor.Deactivate()
Factor.Activate()
Factor.PrintConfig()
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: EMSFactorClasses

Assembly: EMS.dll

Syntax

```
public class FactorContinuous : FactorLeaf
```

Methods

PrintNodes()

Gibt eine Zeichenkette mit den Werten der Eigenschaften Name und OutVal sowie eine Information dazu ob es ein diskreter oder kontinuierlicher Faktor ist.

Declaration

```
public override string PrintNodes()
```

Returns

TYPE	DESCRIPTION
System.String	Eine Zeichenkette mit Name und OutVal

Overrides

FactorLeaf.PrintNodes()

Class Intervall

Beschreibt ein Objekt, welches einen kontinuierlichen Wertebereich über ein Intervall abbildet.

Inheritance

System.Object
Factor
FactorLeaf
FactorContinuous
Intervall

Inherited Members

FactorContinuous.PrintNodes()
FactorLeaf.OutVal
FactorLeaf.IsComposite()
Factor.Name
Factor.IsActive
Factor.AddNode(Factor)
Factor.RemoveNode(Factor)
Factor.Deactivate()
Factor.Activate()
Factor.PrintConfig()
System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: EMSFactorClasses

Assembly: EMS.dll

Syntax

```
public class Intervall : FactorContinuous
```

Constructors

Intervall()

Konstruktor der Klasse. Benötigt für Deserialisierung.

Declaration

```
public Intervall()
```

Intervall(String, Double, Double, Double)

Konstruktor der Klasse.

Declaration

```
public Intervall(string name, double sv, double ev, double iv)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Definiert die Eigenschaft Name
System.Double	sv	Definiert die Eigenschaft StartVal und OutVal
System.Double	ev	Definiert die Eigenschaft EndVal
System.Double	iv	Definiert die Eigenschaft tmp

Properties

EndVal

Endwert eines Intervalls.

Declaration

```
public decimal EndVal { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Decimal	

Increment

Schrittweite eines Intervalls.

Declaration

```
public decimal Increment { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Decimal	

StartVal

Startwert eines Intervalls.

Declaration

```
public decimal StartVal { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Decimal	

Methods

GetNext()

Prüft ob tmp + Increment kleiner gleich EndVal ist. Wenn ja, wird Summe aus tmp und Increment geschrieben und das Ergebnis in OutVal geschrieben.

Declaration

```
public override void GetNext()
```

Overrides

FactorLeaf.GetNext()

HasNext()

Prüft tmp == EndVal.

Declaration

```
public override bool HasNext()
```

Returns

TYPE	DESCRIPTION
System.Boolean	Einen booleschen Wert: true-Endwert wurde noch nicht erreicht, false-Endwert wurde erreicht

Overrides

FactorLeaf.HasNext()

SetInitVal()

Setzt das Feld tmp auf den Wert von StartVal. Setzt die Eigenschaft OutVal auf den Wert von StartVal.

Declaration

```
public override void SetInitVal()
```

Overrides

FactorLeaf.SetInitVal()