Programming

Tue, Jan 28, 2020 Programming

Programming

Table of Contents

Programming in the AI-Framework is largely compatible with C#. The AI-Framework uses Properties and MethodCalls the way C# uses it.

C# and the AI-Framework

The AI-Framework is designed to be as close to native C# as possible. The AI-Framework works with classes and methods and properties, with inheritance and abstract classes.

An experienced C# programmer should quickly feel at home when starting to develop software with the AI-Framework.

Object Oriented Programming

In Object Oriented programming (abbreviated OOP) is a programming model.

Programs are organised around data, rather than functions and logic. These data 'units' are called objects. An object can be defined as a data 'unit' or field that has unique attributes and behaviour.

What makes programming Object Oriented?

Four basic principles make a programming language an Object Oriented Programming language. These are sometimes called the four pillars of OOP.

1. Encapsulation
2. Abstraction
3. Polymorphism
4. Inheritance

Encapsulation

With encapsulation, access to to public methods can be restricted by hiding data implementation. To achieve this, instance variables are kept private and accessor methods are made public.

Code example, where the string name is kept private. It can only be accessed from within the method Customer.

```
public class Customer {    private String name;    public String
 getName() {        return name;    }    public void setName(Str
ing name) {        this.name = name;    }}
```

Abstraction

Abstract in this context has to do with intent, rather than implementation. Abstract means a concept which is not associated with any particular instance. One could say that one class should not know the inner details of another class in order to use it. Just knowing the interfaces should be good enough. When another class inherits from this abstract class, the details will be filled in. (See third pillar: Inheritance).

Code example.

```
public abstract class Example{   private IntegerProperty _value;
   public IntegerProperty Value   {      get { return _value ??
(_value = UseField(t => t.Field.Value)); }   }}
```

Inheritance

Inheritance makes a relationship between two classes. When Class B inherits from Class A, Class A is the called the super class and Class B is called the derived class. When inheritance is used, the derived class can use (re-use) the methods and properties of the super class.

Code example, where InvoicesCorrections is derived from the super class Invoices. In C# and in the AI-Framework, inheritance is assigned with a colon.

```
public class InvoiceCorrections : Invoices{   ...}
```

Polymorphism

Polymorphism is a Greek word, which means 'one name, many forms'. Polymorphism in C# or in the AI-Framework means that a class can have multiple implementations with the same name.

In C# and in the AI-Framework, there are two types of polymorphism.

1. Static Polyphormism = Compile Time Polymorphism (method **overloading**)
2. Dynamic Polyphormism = Runtime Polymorphism (method **overriding**)

In Method **overloading**, a method function has a same name but different signatures. It is also known as Compile Time Polymorphism because the decision of which method is to be called is made at compile time. **Overloading** is the concept in which method names are the same with a different set of parameters.

Code example of Compile Time Morphism with a class that has two methods, both named Add.

```
public class DemonstratePolymorphism  {     public int Multiply(
int a, int b, int c)     {        return a * b * c;     }     pu
blic int Multiply(int a, int b)     {        return a * b;     }
  }  class Demonstrate  {     static void Main(string[] args)
  {        DemonstratePolymorphism dataClass = new DemonstratePo
lymorphism();        int add2 = dataClass.Multiply(45, 34, 67);
       int add1 = dataClass.Multiply(23, 34);      }  }
```

Method **overriding** is an example of dynamic polymorphism or runtime polymorphism. Runtime polymorphism is also known as late binding. Here, the method name and the method signature: the number of parameters and parameter type must be the same and may have a different implementation. Method overriding can be done using inheritance (third pillar).

Code example of method overriding.

```
public class DrawingIt  {     public virtual double Area()     {
           return 0;     }  }  public class Circle : DrawingIt
{      public double Radius { get; set; }      public Circle()
    {          Radius = 8;      }      public override double Ar
ea()      {          return (3.1415927) * Math.Pow(Radius, 2);
    }  } class Program  {     static void Main(string[] args)
  {        DrawingIt circle = new Circle();        Console.Write
Line("Area :" + circle.Area());     }}
```

And how about the AI-Framework

When developing software in the AI-Framework, the focus is on data, available in objects. The AI-Framework is therefore closely related to C# with its methods, properties and concepts of encapsulation and inheritance.

Categories and articles in this chapter

Read a description of the categories in this chapter and go to that category, or straight to its main articles. From the main articles can be navigated to all related articles.

Category	Description	Cat.	Article
Types	These articles all explain about the different types that can be used within the AI-Framework. Examples: Blob, Boolean, DateTime, DbKey, Decimal and much more. The same types are applied with many other elements in the AI-Framework, like properties (BlobProperty, BooleanProperty...) and arrays (BlobArray, BooleanArray...). **These types are only described in this category**.	Cat. ≥	Types
Properties	Properties enable a class to get and set values in a public way. Properties are getters or setters. A get property accessor gets a property value and returns it. A set property accessor is used tossing a new value. In the AI-Framework we make use of AutoProperties, Calculated properties and ... ?	Cat. ≥	Properties
Methods	A method is a block of code containing statements. The statements in that block – the method – will be executed by calling the method's name and specifying the required method arguments.	Cat. ≥	Methods
Collections	The AI-Framework utilises different ways to work with collections, like in Arrays, HashSets, Lists, Views and EntityCollections.	Cat. ≥	Collections
LINQ	LINQ stands for Language-Integrated Query. It is a means to work with data in a functional way. LINQ bridges the gap between the imperative programming style and the functional programming style. LINQ lightens the programmer's job allowing them to focus on	Cat. ≥	LINQ

	the business logic while spending less time coding associated with data access code.		
Entities	Entities are representations of a database table in memory. For example, for the table Articles, the Entity ArticleEntity could be built. Within the class of the entity, properties and methods can be defined.	Cat. ≥	Entities

Quick access

- Types
  - BlobExpression
  - BooleanExpression
  - BooleanExpressionExtensions (6)
  - DateTimeExpression
  - DbKeyExpression
  - DecimalExpression
  - EnumExpression
  - IntegerExpression
  - LongExpressions
  - MathExpression
  - MathExpressionExtensions (2)
  - StringExpression
  - StringExpressionExtensions (3)
  - TimeSpanExpression
- Properties
- Methods
- Collections
  - Arrays
  - HashSets
  - Lists
  - Views
  - Other collections

- LINQ
- Entities

SyntaxHighlighter.config.stripBrs=true; SyntaxHighlighter.all();