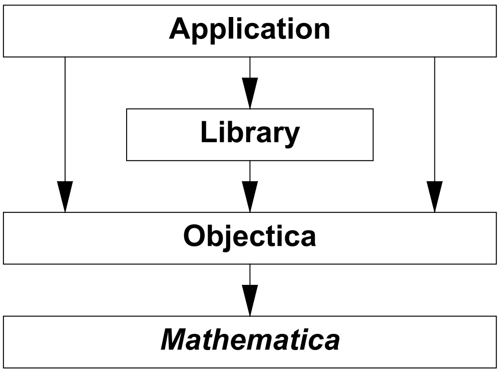
Objectica

Symbols and Numbers
Mathematische Software GbR

Objectica is a package for *Mathematica* that allows to write object oriented programs easily in *Mathematica*. This opens users of *Mathematica* the world of object orientation and the world of object orientation gets access to *Mathematica*. From the point of view of a developer it is a middleware.


Application
Library
Objectica
Mathematica

Like in other object oriented software, classes can be defined and objects generated from these classes. All important aspects of object orientation are implemented: abstract data types, inheritance, encapsulation and polymorphism. Objectica covers, with respect to object orientation, almost the complete functionality of Java and C#. In principle, it has the same applications. The similarity to these languages is reflected in the used terms. The following table shows some of Objectica's symbols.

classes	Class, Interface, Object, This, Super, SubClasses, Finalize, Export
objects	New, Objects, ObjectInitializations, RemoveObjectsOfClass, Transient
access	Dot, Public, Protected, Private, Static, Property, Constant, Mutable
inheritance	Override, Virtual, Real, Abstract, Final
type	ObjectQ, ClassQ, InterfaceQ, InstanceOf, TypeOf

Objectica is tightly focused on a productive usage in a commercial environment. Parts of this strategy are a very detailed documentation and a simple syntax. Similarly, the timing of evaluations and the development of commercial libraries are taken into account. The following example should give a first impression.

Example: Lunar Orbit

```
In[2]:= Class[Body] := {
           Body[m_, x0_, v0_] := (mass = m; This.x0 = x0; This.v0 = v0),
           other = This, x = {x1, x2}, v = {v1, v2},
           mass = 0, x0 = {0, 0}, v0 = {0, 0},
           a := Block[{r = $.x[t] - $.x.other[t]}, 6.674*10^-11 *other.mass / (r.r)^3/2 * r],
           var := Join[$.x[t], $.v[t]],
           ini := Join[Thread[$.x[0] == x0], Thread[$.v[0] == v0]],
           deq := Join[Thread[$.x'[t] == $.v[t]], Thread[$.v'[t] == a]]
          };

In[3]:= earth = New.Body[5.98*10^24, {0, 0}, {0, 0}];
        moon = New.Body[7.40*10^22 , {384.4*10^6, 0}, {0, 1023.16}];
        earth.other = moon; moon.other = earth;

In[6]:= Short[NDSolve[Join[earth.deq, moon.deq, earth.ini, moon.ini],
          Join[earth.var, moon.var], {t, 0, ((27* 24 + 7) * 60 + 43) * 60}]]

Out[6]//Short=
        {{x1_1[t] → InterpolatingFunction[<<1>>][t], <<6>>, <<1>>}}
```

Typical Applications

- Simulation of the real world that is composed of objects
- Union of object orientation and *Mathematica*, for example models with differential equations
- Problems with lots of similar or equal objects, for example the wheels in a gear mechanism
- Structuring of models by means of hierarchical modeling, for example assemblies in a vehicle
- Dynamic creation and destruction of objects, for example graphics in a drawing software

User Group

- Software engineers of other object oriented languages for prototyping
- Object oriented programmers who need the functionality of Mathematica
- Developers of big *Mathematica* projects to structure their problem
- Engineers to represent real objects

Benefits

- Objectica benefits from all advantages of object orientation

abstract data types	behavior and state are in the same object, a kind of " intelligence "
encapsulation	only the necessary information is available to the public
inheritance	extreme recycling of functionality
polymorphism	behavior automatically takes into account the type of an object

- Recourse to developments made for object oriented languages

design patterns	modeling in a higher level of abstraction (book : design patterns of Gamma)
UML	design of class diagrams

- Structuring allows for huge projects and teamwork
- Easy communication with programmers of other object oriented languages

Scope of Delivery

- Creation and administration of classes and objects
- Saving of classes and objects, handle dependencies automatically
- Deployment of classes together with a limited runtime environment
- All important concepts of Java and C# are implemented
- There are additional concepts, for example abstract static fields
- Objectica is easy to learn, because the syntax is very similar to Java
- Object access is fast (typically well below a millisecond)
- Functions for protection of classes allow for the development of commercial class libraries
- Many consistency checks facilitate the development process
- Handbook and many examples for all functions - a good base even for sporadic work

Trial version available