# Custom Complex Types

In the Designer, you can define your own complex data types as described on Modeling Data Structures.

| Element | Description | Details |
|---|---|---|
| **Package** | A package is like a directory for the file system. It is used to group executable data model elements. Packages can have any depth of nesting: To structure your work, you can create packages within packages.<br>Also, packages define a sort of namespace to the contained elements. The name of the package is part of the element path, e.g. **Package1.Class** is different from **Package2.Class**. | |
| **Class** | A class is an aggregation of properties and operations that describes a complex data type from which objects can be created. | Customizing Classes |
| **Property** | Properties are data fields that describe the structure of the class. | Customizing Properties |
| **Operation** | An operation adds behavior to a class or interface. The behavior describes how to process the data given by the parameters. In the context of the Designer, you can implement operations as mapping, action script or activity. | |
| **Parameter** | Operations can have parameters that define the input and output objects. Operation parameters can be of simple type (Base Types) or of complex type (class or interface). | |
| **Interface** | In contrast to a class, an interface has no properties nor implementations. Interfaces are used to define common operations of multiple classes, and then derive from that interface.<br>Operations of interfaces do not have an implementation but only define the signature (parameters and types). | |
| **Interface** | Interfaces can have sub-interfaces and sub-classes. | |
| **Class** | | |
| **Operation** | Operations and parameters for interfaces are the same as for classes. The difference is that they have no implementation but only define the signature for the dependent classes to derive from. | |
| **Parameter** | | |

**Related Pages:**

- Customizing Classes
- Customizing Properties