# About factories

## Table of contents

## 1. Programming algorithms on Abstract Graph, Abstract Sets

As shown in the Documentation, the graph and digraphs are derivated from Abstract Graph. If you want to program an algorithm which works on graph AND digraphs, then you can directly program it with Abstract Graphs as parameter. Now, imagine that you works on the abstract graph. You cannot know it this graph is made of edge or arc. You have to use the methods of Abstract Edge and you cannot use the specific one of Edge or Arc. If your algorithm have to create a new Arc or a new Edge in the graph, as a result, how can you allocate this object without knowing its type ? That's here you can use a factory. A factory is able to build any object that you do not know its type.

### 1.1. How I do ?

First, get the factory (assuming you have g, an abstract graph):

```
Abstract Graph g = ...;
AbstractGraphFactory factory = g.getFactory();
```

Then, you have several methodes for building:

- Vertices
- Edges
- Vertex Sets
- Edge Sets
- Graphs
- Chains

Remember that you don't know the real type of this objects; you have to consider this object as abstract object using its Abstract type. For example:

```
AbstractVertex n0 = factory.newAbstractVertex();
AbstractVertex n1 = factory.newAbstractVertex();
AbstractEdge e = factory.newAbstractEdge(n0,n1);
```

### 1.2. Where can I get the factory ?

The factory can be accessed from all basic type of the library:

- Graphs
- Vertex Sets
- Edge Sets
- Vertices
- Edges
- Chains

## 2. Sample

A short sample as been written to illustrate the use of factories. It adds some random edges to a graph without knowing if this graph is a graph or a digraph. You can find the file AddRandomEdges.java in Mascopt Dev package in mascoptDev/samples/basics. You obtain:

```
java AddRandomEdges samples/basics/AddRandomEdges1.mgl

Starting XML SAX2 parser
Parser is validating.
Graph before:DiGraph V={ N1 , N0 , N3 , N2 } E={ [N2->N3] , [N0->N3] ,
[N1->N0] , [N2->N0] , [N1->N3] }
Graph after:DiGraph V={ N1 , N0 , N3 , N2 } E={ [N2->N3] , [N3->N3] ,
[N0->N1] , [N3->N1] , [N0->N1] ,
 [N0->N3] , [N1->N0] , [N1->N1] , [N0->N0] , [N1->N0] , [N2->N0] , [N1->N3]
}
```

You can test it on samples/basics/AddRandomEdges1.mgl and samples/basics/AddRandomEdges2.mgl.