IJavaNumberPart interface

The provided NumberParts for Numeric and Character implement the numbering based on the natural ordering of integers or the alphabetic ordering of strings. Examples for this would be P1.1, P1.2, P1.3 or P1.A, P1.B, P1.C, etc. What if we wanted to create numbers of the following kind: P1.1, P1.4, P1.9, P1.16, etc. with all intermediate numbers omitted?

The *com.nomagic.magicdraw.autoid.IJavaNumberPart* interface provides a way to create a slice for a generated number that does not follow these natural sequences.

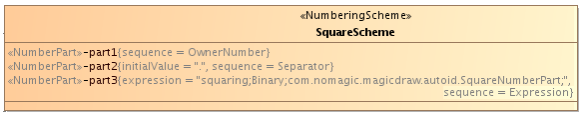
The parameter that the *com.nomagic.magicdraw.autoid.IJavaNumberPart.generateNextId* method receives, namely *lastNumberPart*, is the value that this method is generated as a result in the previous call. If the method has not been called before this *lastNumberPart* value equals an empty string.

> ⚠ This method should adhere to the contract that equal inputs must produce equal outputs. If, for example, the input is ABC and the method computes a value of XYZ then the return value for ABC must always be XYZ.

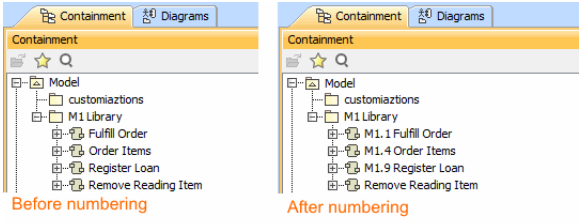
The return value of this method will be incorporated by the Numbering Framework to create a complete number together with all other *NumberParts* that are part of the same *NumberingScheme*.

Example

Let's assume the *SquareScheme* is used as target for UML activities. According to this *NumberingScheme*, the first part of a number will be taken from a numbered parent (in this case, the package *sample* as shown in the following figure), then a dot is inserted and the final slice is generated by the *SquareNumberPart* expression.



We have created a few activities inside the package named *sample*. The owning package is also customized and has a generated number P1. If we number the activities with this *NumberingScheme*, the *SquareNumberPart* class, being set as *part3* of the type expression, will create the following sequence: 1, 4, 9, 16, 25... These pieces are then added to the results from the other *NumberParts*, resulting in the activities being numbered as shown in the following figure.



The source code for this simple implementation should be self-explaining. It will simply try to transform the string value of *lastNumberPart* into an integer value called *square*. Subsequently, the following mathematical operation is performed and returned $nextSquare = (\sqrt{square} + 1)^2$.

```
public class SquareNumberPart implements IJavaNumberPart
    {
    private final int initialValue = 1;
    @Override
    public String generateNextId(String lastNumberPart)
            {
        int nextSquare = initialValue;
        try
                    {
            if ("".equals(lastNumberPart))
                            {
                lastNumberPart = "0";
            }
            int square = Integer.valueOf(lastNumberPart);
            int number = (int) Math.sqrt(square) + 1;
            nextSquare = (int) Math.pow(number, 2);
        }
        catch (NumberFormatException ex)
                    {
            // already set nextSquare as initialValue
        }
        return Integer.toString(nextSquare);
    }
}
```