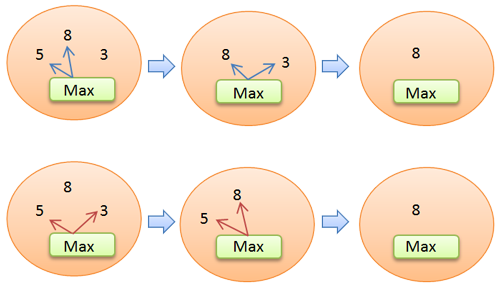
*4INFO INSA Rennes Project -* *Chemical Programming using Java*

**Project presentation**

The main objective of this project is to design a chemical programming library for Java. Chemical programming is based on a paradigm where data is treated as chemical reagents in a chemical solution. This project includes a technology showcase using the library : a random music generator.

**Chemical programming**

Chemical programming is inspired by the chemical reaction mechanism, which is characterized by its indeterminism.

Calculations can be represented as reactions, controlled by a set of rules, and data as molecules. Both are present in a solution. When a reaction is started, it will proceed until the system becomes stable, and leads to a final solution.

This diagram represents a reaction which permits to find a maximum number among a set of numbers. The reaction rule is called "Maximum" and reagents are 5, 8 and 3. As shown by blue arrows, the rule will firstly react with 5 and 8, and only 8 will remain in the solution. Then, it will react with 8 and 3, and only 8 will remain in the solution. This solution is now inert, reaction is over. During another execution, the different reagents may be choosen in a different order, as shown by red arrows.

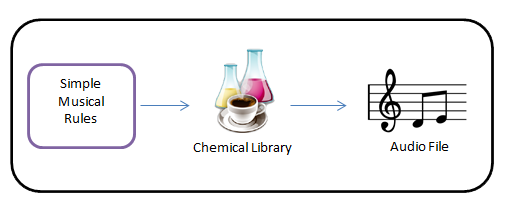
**Java library**

Until then, chemical programming was used in combination with a specific language called HOCL. In order to interest as many programmers as possible, our library can be used with Java, and respects every aspects of object-oriented programming (interfaces, inheritance, reflexivity). For example with the previous diagram, programmers can write their own reaction rule (MaxRule), then use it this way :

|  |  |
| --- | --- |
| **class** **RuleMax** **implements** ReactionRule{  **private** **int** a,b;  **public** Object[] **computeResult**(){  **if**(a>=b)  **return** **new** Object[]{a};  **else**  **return** **new** Object[]{b};  }  } | **public** **class** **MyChemicalProgram**{  **public** **static** **void** **main**(){  Solution sol = **new** Solution();  sol.add(**5**);  sol.add(**8**);  sol.add(**3**);  sol.add(**new** RuleMax());  sol.run();  }  } |

**Technology showcase**

In order to provide an example of what can be done with the library, we decided to design a random music generator using it. We focused on a very theorical musical form : the tonal system of classical music, mainly used from 1750 to 1820. This way we can set different reaction rules governing the track creation.



Using a GUI, users can specify some parameters like tempo, range and number of bars. Then, the application will automatically generate a song dependent on these parameters.

*(c) 2012 INSA Rennes. LGPL license.*

*Members : Cédric Andreolli, Chloé Boulanger, Olivier Clero, Antoine Guellier, Sébastien Guilloux, Arthur Templé.*

*Teacher : Jean-Louis Pazat.*