# Circle Language Spec Plan Diagrams, Coding Principles & Coding Concepts

*Author: JJ van Zon*

*Location: Oosterhout, The Netherlands*

*Date: May 26, 2008 – June 1, 2008*

## Goal

*> 2009-04-07: This is a very old project description.*

*This project was postponed earlier, because it was too big. The concepts of classes & relations were turned into a single project. This project contains the remaining topics.*

Document the coding principles and basic coding concepts and their diagram notation. Do that with the ideas I have about them now and avoid difficult reconsideration of principles.

I am going for a spree of writing about issues I already know.

Do the articles that you can easily write, and skip all the ones that are more difficult.

Ignore the fact, that I might change my whole point of view on commands  
(That will be covered by the future project *Commands As A Concept*.)

### Pros and cons for this project

- Previous project was working out issues that might change everything.  
I don’t see any issues left, that might change *everything*.

- The defense for this project is that it is so easy to produce, while the articles are also very important.

- The argument against it, is that fundamental principles should be worked out first.

### Extra goals

I have extra goals for this project. What those goals share with the main goal, is that they are also easier to write than other things.

- Put a small description of all the modules in Software System.doc.

- Extend the article Automatic Containment

- Extra article: Relational = Object Oriented

(perhaps in Fundamental Principles (that’s where all the =’s are))

- Extra article: Class = Prototype

(perhaps in Fundamental Principles (that’s where all the =’s are))

- Change the article Symbol = Creator.doc

### Rules

- Don’t change Commands & Classes loosely coupled.

- Don’t think about which are implemented as a principles and which are implemented as a concepts.

## Steps

… Write more articles

- Look at Computer Language Coding Principles.doc

- Process cross out lists:

- Symbol Language.doc

- Relational Structure.doc

## Approach

There seem to be the following elements about each concept:

- Functional use

- Technical use

- Implementation as a concept

- Diagram expression

Implementation as a concept is not covered in this project.

## Elements

Articles to write **(84)**:

Coding Principles **(18)**:

- Destruction

- Destruction in a Diagram

- Clear

- Clear in a Diagram

- Object Order

- Object Order in a Diagram

- Static Members

- Static Members in a Diagram

- This

- This in a Diagram

Coding Concepts:

Basic Coding Principles **(22)**:

(Type Safety, Parameters, Input / Output / Throughput)

- Type Safety, Genericity, Explicitness

- Type Safety, Genericity, Explicitness in a Diagram

- Extended Coding Principles **(24)**:

- Namespaces

- Namespaces in a Diagram

- Aliases

- Aliases in a Diagram

- Ambiguity

- Ambiguity in a Diagram

- Member Grouping

- Member Grouping in a Diagram

- Inheritance

- Inheritance in a Diagram

- Class Inheritance

- Class Inheritance in a Diagram

- Object Inheritance

- Object Inheritance in a Diagram

- Relation Direction

- Relation Direction in a Diagram

- Special Access

- Special Access in a Diagram

- Global Access

- Global Access in a Diagram

- Clause Access

- Clause Access in a Diagram

- Interface Access

- Interface Access in a Diagram

## Ideas

After documenting the coding principles you could go through the old Relation Structure documentation and delete everything already covered, and distill things from it that aren’t considered yet.

> Any entry in Coding Principles in a Diagram will get an entry in the Coding Principles articles.

- Write articles for coding concepts about classes, interfaces and relations

> The description of these coding concepts require partly the idea behind it, as well as how it is programmed as a concept.

> I might just leave out how it is programmed as a concept for now, or keep it very general, with the remark that this is to be further worked out in the future.