Is there a need for a programming language adapted for implementation of design patterns?

Ruslan Batdalov

I am currently designing a new programming language. One of the project goals is to facilitate implementation of design patterns. In a sense, it will be a 'pattern-oriented' programming language. Would such a language be interesting and helpful for the patterns community?

This group is to discuss how complexity of design patterns implementation may depend on the used programming language capabilities.

The problem to address is that implementation of many known design patterns may significantly complicate a software system, although the purpose of patterns is exactly opposite. It may be explained (at least, partially) by the lack of expressiveness of existing programming languages – their capabilities are poorer than the set of patterns building blocks. I will describe my ideas how to incorporate concepts and ideas on which patterns are built into a programming language.

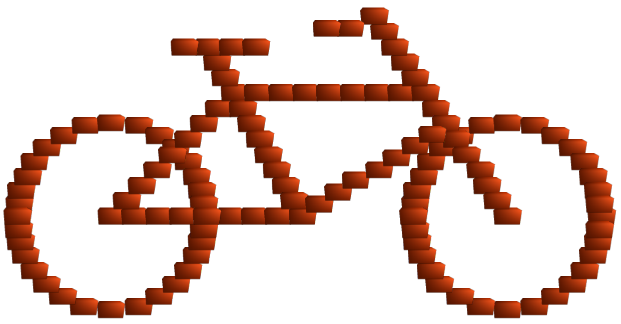
The discussion is planned in the 'Six thinking hats' format. If you lack experience with this system, it is not a problem — the format is very simple and easy to use.

As a result, I hope the discussion will clear up whether there is a need for such a language, and what particular functionality it should have to ease and promote pattern usage.

Approximate plan of the discussion:

1. Presentation of the problem.
2. What problems of design patterns implementation are related to the underlying programming languages? Is there a lack of expressiveness in existing programming languages?
3. How can a specifically designed programming language be useful?
4. What opportunities the patterns community would like to see in such a language?
5. Summarising and free discussion.

Six Thinking Hats® is a structured group discussion, divided into six consecutive stages. In each stage, all the participants pursue the same way of thinking (metaphorically, wear the same hats) and discuss the problem from a particular angle (**process**, facts, **feelings**, **creativity**, benefits, **cautions**). After one stage, the whole group switches to the next one, thus ensuring parallel thinking.



When only bricks were available...