Object Orientated Programming in BF

How could extend the BF language and interpreter code to allow for ‘Objects’?

Let us start by defining what an makes a programming language object orientated and how it could potentially be applied to BF. For BF to allow objects there needs to be the functionality of methods and properties, this is because objects themselves are fundamentally made up of methods and properties. In most programming languages, these objects are defined as an instance of a class. BF alternatively operates 8 simple commands and an instruction pointer. However, by manipulating the stack using these, you can produce similar results to methods and properties albeit inefficiently.

By optimizing your BF code, you can reduce its inefficiency, additionally you could extend the list of commands so that you can more easily manipulate the stack with different Logic. An example of this exists in ‘Extended BF’ a variant of BF that allows smaller source sizes and improved speed of execution. In Extended Type I of Extended BF it offers nine more commands including bitwise NOT, AND, XOR and OR operators. These commands act as more efficient means of simulating ‘methods’, ways of altering the behaviour of the data stored in the stack.

Alternatively, you could use a combination of a Lists, stacks, and arrays to create an interpreter program that allows you to implement objects.