

Collective Behavior

Adrian Kuhn on a new language feature, extending collections with custom methods depending on the element's type, and in the same turn, extending objects with group related behavior.

`#(1 2 3 4) sum`

`array inject: 0 into: [:a :b | a + b]`

We all know how to implement the above method, but to which class does it belong? To Collection or to Integer?

allDocuments merge

We can also think of more complex operations, for example merging a group of papers into a workshop reader including front matter, table of contents and index.

Where does this code belong?

As it is now, methods either apply to a single instance or they are static, that is global. Group-related behavior has no first-class representation.

Collection? — No.

Of course, we could just add `#sum` and `#merge` to `Collection`, the abstract superclass of all collections. However, that is bad OO modeling: following the same logic, any instance-related method would end up in `Object` rather than its subclasses.

Collection subclass? — No.

Unlike Object's subclasses, the subclasses of Collection do not implement domain types such as Numbers or Books, but rather abstract data types, as for example an array list, or a hash set.

Integer? — No.

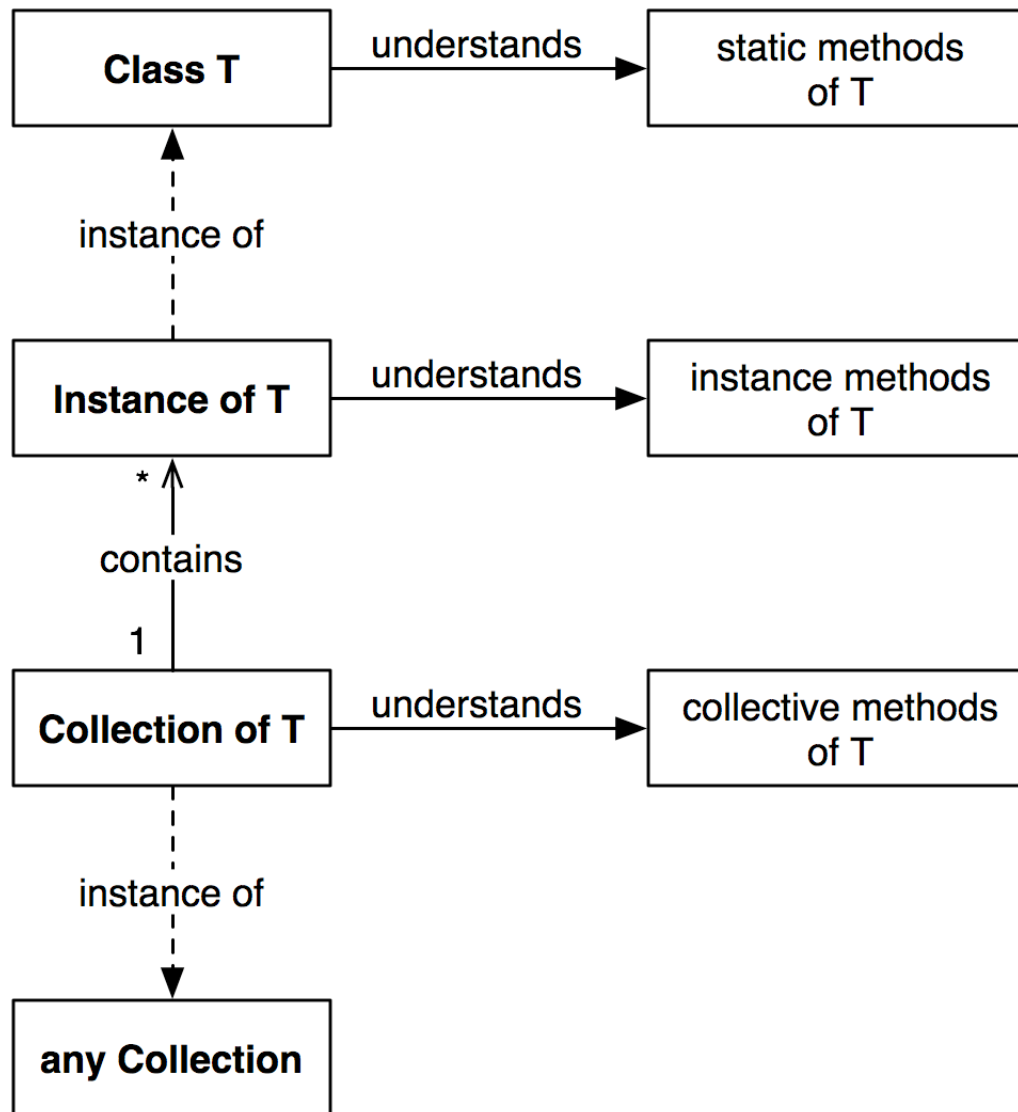
Actually, the proper place for domain behavior would be domain classes, however, their methods apply to single instances only, not to groups of these objects.

Integer class? — No.

And finally, we all certainly agree that isn't good style neither to implement group-related behavior using static method.

Integer group? — YES!

As solution, we propose to extend object-oriented modeling with a new kind of methods. That is, classes are defined as a tuple $T = \langle Ts, a^*, m^*, g^* \rangle$ with g^* as group-related methods. Other than instance methods, group methods do not apply to instances of T but rather to any collection containing elements with T as most specific common supertype.



Proof-of-Concept Implementation in Smalltalk using DNU.

Alas, no live demo :(

Integer group >> sum

```
#(1 2 3 4) sum => MessageNotUnderstood
```

```
Integer group compile: 'sum  
  ^self inject: 0 into: [:a :b | a+b ]'.
```

```
#(1 2 3 4) sum => 10
```

Integer group >> average

```
#(1 2 3 4) average => MessageNotUnderstood
```

```
Integer group compile: 'average  
  ^self sum / self size asFloat'.
```

```
#(1 2 3 4) average => 2.5
```

Integer group class?

`#(1 2 3 4) class => Array`

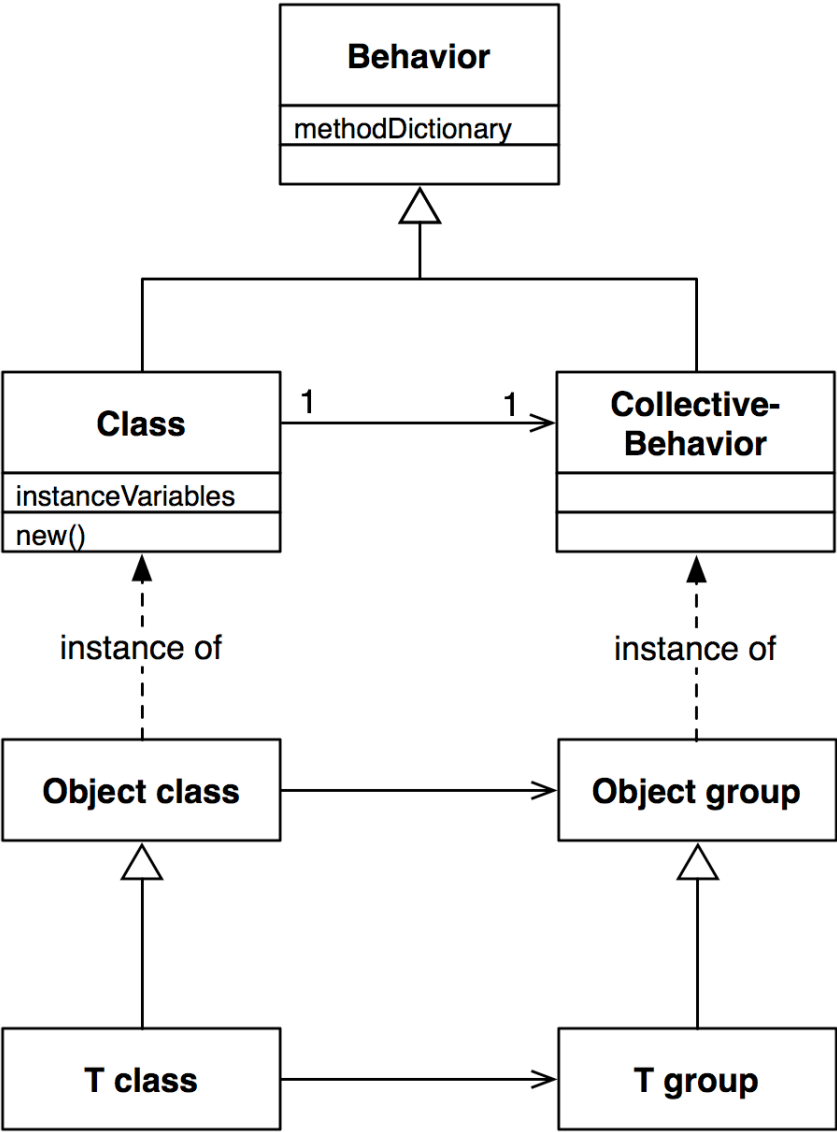
`#(1 2 3 4) group => Integer group`

`Integer group => Integer group`

`Integer group superclass => Number group`

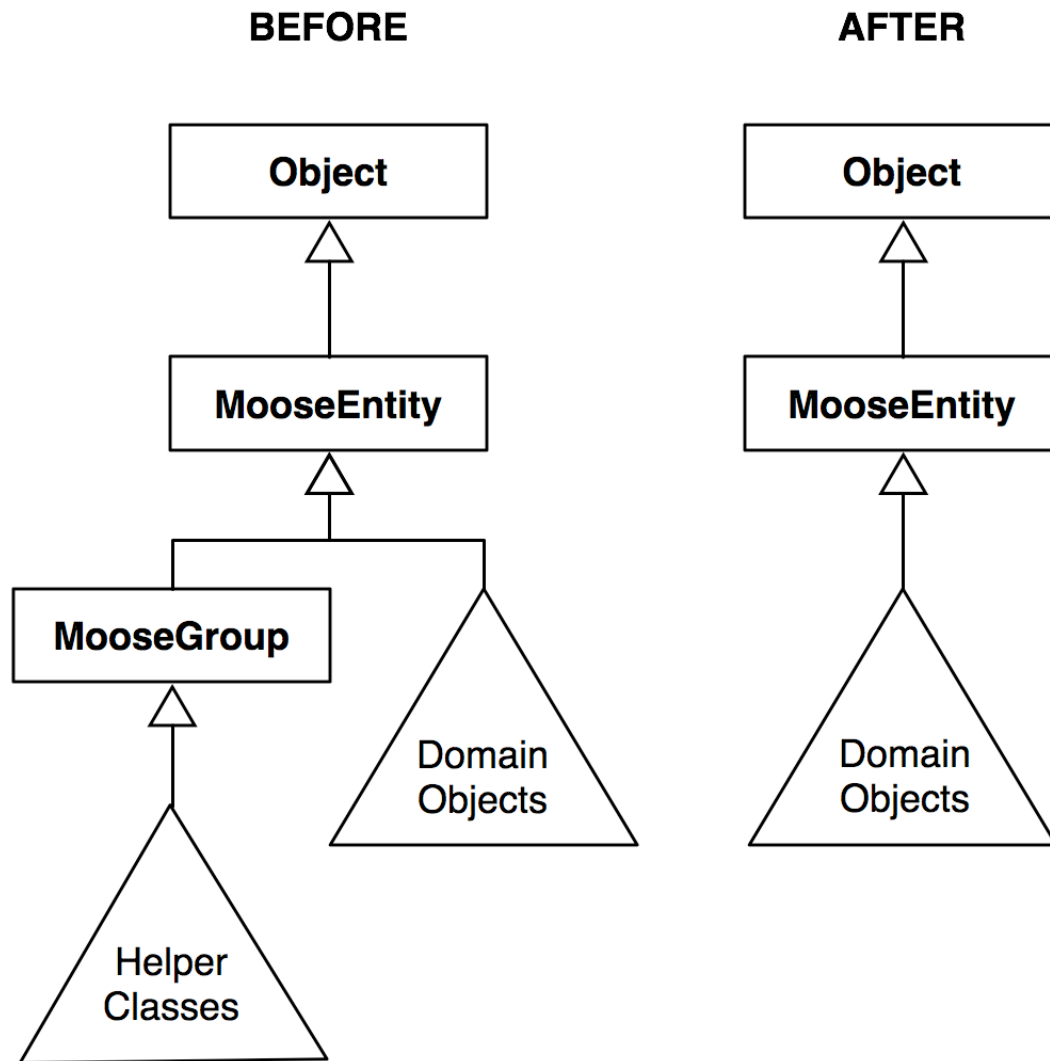
`Integer group class => CollectiveBehavior`

`CollectiveBehavior superclass => Behavior`



How much CB is there?

- Moose, a smalltalk application:
 - 197 out of 2111 domain methods,
 - in 25 out of 123 domain classes,
 - including 1386 out of 6721 statements.
- That is 20% of the domain code.



There are many Open Issues

- IDE and versioning support for Smalltalk.
- How can it be implemented in eg Java?
- What is the CB of empty collection?
- Static or dynamic typing of collections?
- What if element's type changes during execution of a collective method?
- And many more...

Collective Behavior

Fills a gap in OO modeling, providing proper place for group-related behavior.