# Compiler Construction - Extension Proposal

Lars Kuijpers (s4356314)
Flip van Spaendonck (s4343123)

## 1 Extension

The main idea of our extension is the ability to add custom structures to the SPL language and our compiler. We want to focus on two things: multuples (similar to tuples) and objects (similar to classes in Java). We think these provide an interesting addition to the language and make it easier for people using our compiler to program more complex data.

### 1.1 Multuples

Multuples are identical to the idea of tuples, but instead of always consisting of two elements, our multuples can be of any length. They will have a fixed length, set by the programmer and known at compile-time. They will also be fully typed and saved array-like (next to each other in memory), just like our current implementation for tuples. We will change the accessor functions (.fst and .snd) to use numbers (in the form of .[0], .[1], etc.) so it can easily be extended to multuples.

### 1.2 Objects

Objects are more extensive structures the user will be able to program. Just like multuples they are also data types, but they are more complex. Each object, at its minimum, will have at least one function, the constructor. A constructor can take any amount of arguments (as defined by the user), but should always have a return-type of the object itself. An object without a constructor is invalid and will be rejected by our semantic analysis. This constructor will be used to make instances of a certain object to be used in the code. Additionally, objects can have their own variables, thus introducing a new scope level for variables, which can be accessed from an instance of the object. Objects can also have their own functions, which can be called from such an instance.

## 2 Possible further additions

If these things prove to be easy, we can also add encapsulation to objects. Making it possible to make variables and functions in an object public or private. Public will mean they can be accessed as normal, but if for example a variable is private, it can only be accessed from within the object itself. The same would hold true for functions, a private function can only be called from within the object itself.