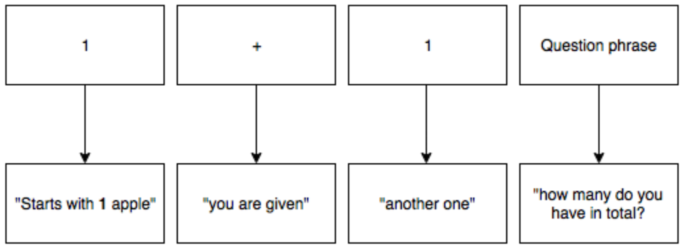
## Lisp Design:

The goal of Lisp generator is to generate an arithmetic expression in the form of an English sentence on the fly. Essential, each component of an arithmetic expression is then transformed into an English phrase like so:



Algorithm:

- Generate a simple arithmetic expression in list form e.g. '(+ 1 1)
- Turn each component: operation and two operands, into equivalent English form.
- Each component has a pool of different options, so the whole question appears different each time it is generated.
- Each operand can be randomly displayed in numeric or text form, if it is equal or less than 10.
- Generate a question phrase at the end to ask a human user; the question phrase also has a pool to select from for variety.
- Finally, all the generated English components are assembled and delivered.

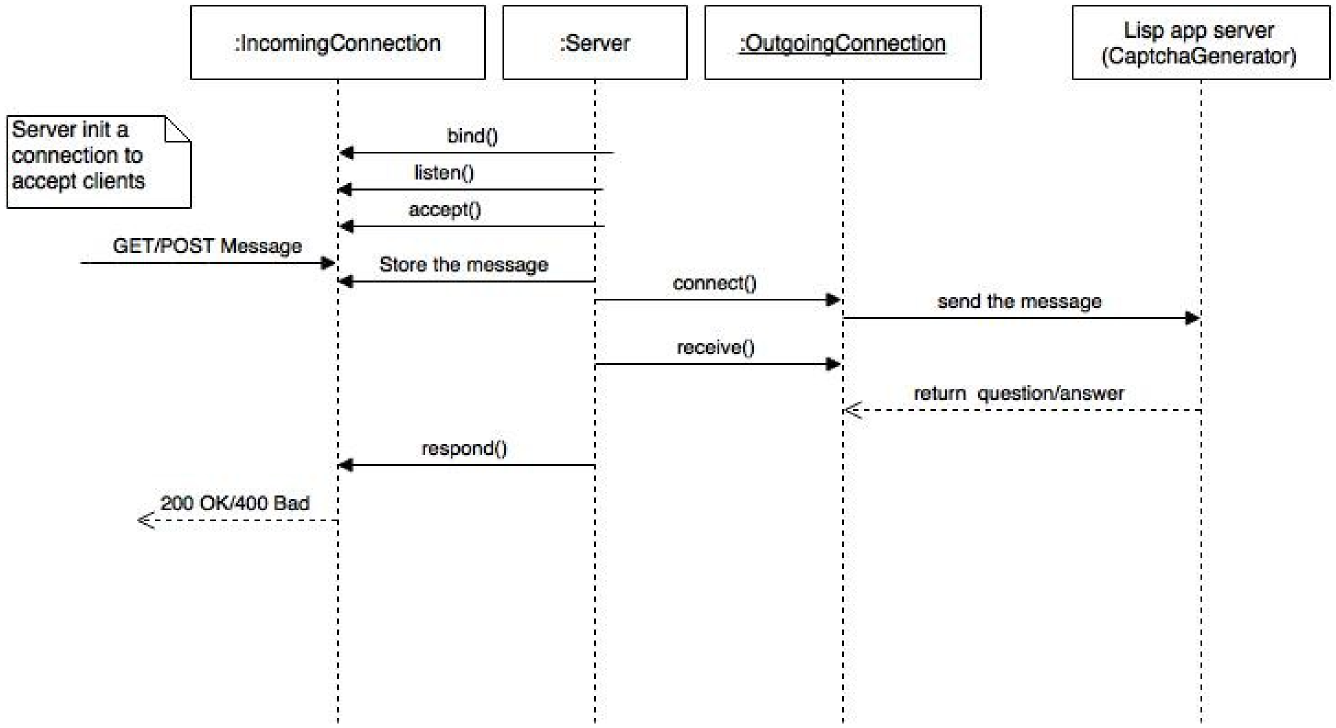
## C++ Design:

The RESTful service in C++ is implemented as a simple web server that accepts HTTP requests from a browser and send appropriate responses. When a GET request is received, it returns the question and when a POST request is received, it returns whether or not the user supplied answer is correct. The server consists of the following classes:

- **Server**: talks to the outside world and to the Lisp backend.
- **Connection**: represent a connection between two sockets.  Each class has its socket and is responsible for starting and closing the socket.
- **IncomingConnection**: inherited from **Connection**. Represent a connection coming from the outside world to the web server. This class is used by a server for accepting outside connections, so it also holds another socket that represents a client.
- **OutgoingConnection**: inherited from **Connection**. Represent a connection the web server connects to. In this app, it's a connection to Lisp app server.
- **HttpRequest**: encapsulate the information components from HTTP request
- **Parser**: a simple parser that analyzes GET and POST requests, then fill a **HttpRequest** object with parsed information i.e. method, URI, captcha answer…

Underneath the abstraction in C++, the server uses POSIX sockets, so it only runs on POSIX compliant systems, such as variants of Linux or OSX.

Here is how the objects interact with each other:



## Testing:

For Lisp testing, FiveAm testing framework (https://common-lisp.net/project/fiveam/) was chosen, because it is simple, which is suitable for the task. I also have experience with it. Unit testings were done on most of Lisp functions, since those functions can work in isolate without side effects.

For C++ testing, Catch (https://github.com/philsquared/Catch) was chosen because it is also simple to setup and use: only one header file is needed. The testing was only done on Parser class since it contain pure functions that can be tested in isolation. For other classes, the functions involve heavy side effects i.e. socket IO, so it's difficult to setup a complicated test environment, which is not needed for scope of this project.