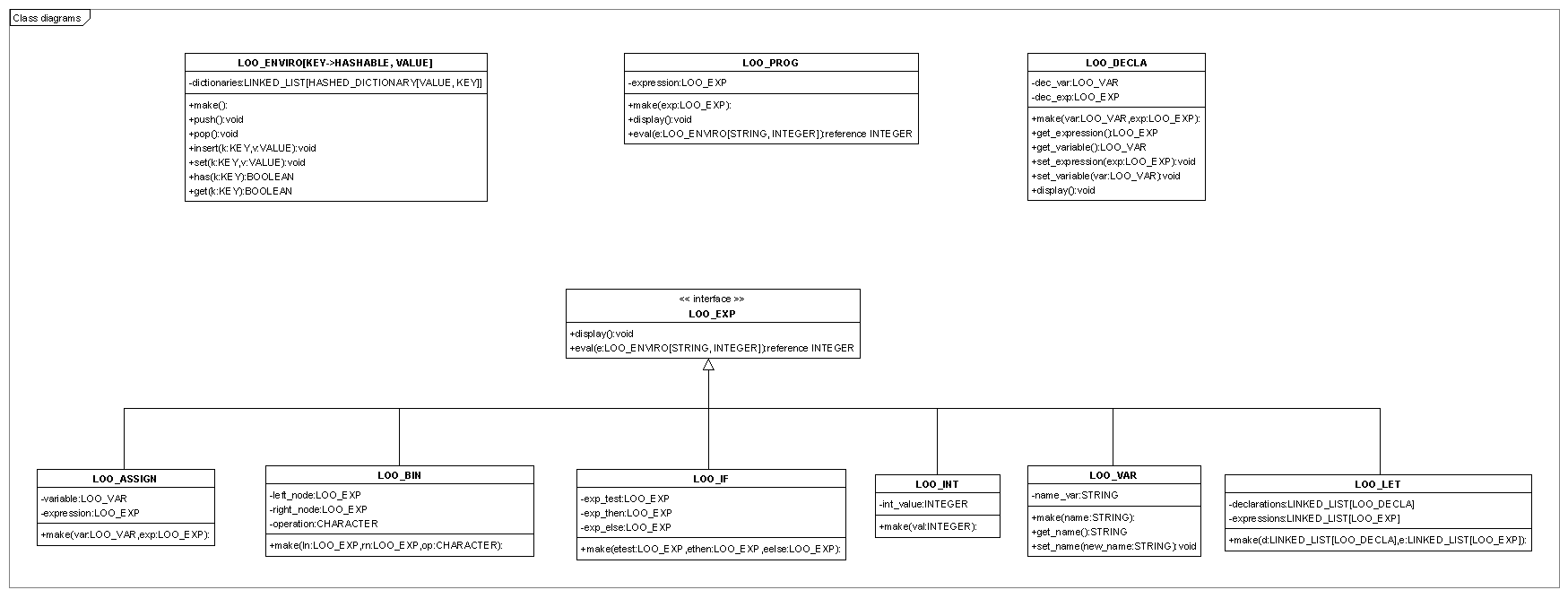
# Class diagrams



# How is this application designed?

According to previous class diagrams, my project is divided into 4 main parts.

The first one deals with all the possible statements i.e.

- an assignment (*LOO\_ASSIGN*)

- a binary operation (*LOO\_BIN*)

- an if statement (*LOO\_IF*)

- an integer value (*LOO\_INT*)

- a let statement (*LOO\_LET*)

- a variable (*LOO\_VAR*)

These inherit basic statement class (*LOO\_EXP*) and redefine display and eval procedures.

I decided that displaying an incorrect statement is possible (whereas of course evaluation cannot be done). Therefore, display procedures don’t have any requirement.

A program (*LOO\_PROG*) is composed of an statement whose evaluation is equal to an integer value.

The second part defines declarations (*LOO\_DECLA*). Indeed, each let statement can include some variables that are declared as follow (between "let" and "in" strings):

let

-- Declaration of variables

a := 5

b := a + 1

in

-- Statements composing the current let statement

...

The third part is about environment (*LOO\_ENVIRO*). An environment represents each let statement through a new hashable dictionary (as a let statement can be composed itself by several other let statements) where the corresponding declarations are stored.

The final part is tests realized thanks to a specific class (*LOO\_TEST*). 15 tests have been written, 7 correct ones and 8 incorrect ones (see tests.docx).