I started to implement the assignment in an extremely object orientated manner. I tried to split the problems into smaller problems with the use of classes; an object would do what the object had to do, to valuate to a value, and then ask the object it contained to do the same. This way I didn't have to think about more then one problem at the time, I just assumed that my previous implementation was correct and that I safely could use this in other parts of my program.

The first problem I had, was how to move the robot from the Move-objects. I ended up with a not so object orientated solution; static methods in Robot, so that the Move-objects could do this themself. I could of cause send a Robot-object-pointer to the Move-objects, but because of the way the pre code was set up, I didn't end up with this.

I didn't have time to comment the code, but I hope it is understandable.

I can talk a little about the class Program, cause it's the class which connects all the classes.

The Program-class acts as an interpreter for the language. It translates strings into objects. (I am not entirely sure if we were going to implement this, but it made sense to me.)

When interpret() in Program is called upon, the program will go in a loop an make objects according to the syntax of the language Robol, until stop is typed by the user, or red from a file. When stop is red, the program will start to call interpret() on all the Stmt-objects (statements) and the robot will end up moving. After all Stmt-objects are interpreted, the program will print where the robot ended up.

I did not implement <down> or <up>, due to time issues, and the fact that they have no use in the program.

Use these commands to compile, test or run the project (just extract and run from terminal):

```
make
make test
make run
```