Programming Assignment 3: Clojure

Isak Karlsson & Beatrice Åkerblom
#{isak-kar beatrice}@dsv.su.se

December 11, 2013

1 Introduction

This assignment is the fourth of four assignments. The assignment is due on 2014-01-12 (23.55) and shall be uploaded to the course web site. The assignment is conducted in groups of two (2) students (exceptions are *only* provided by Beatrice).

If in doubt, please use the ilearn2 discussion forum.

2 The Assignments

Your assignment is to implement two Clojure macros and discuss the use of macros.

2.1 Safe Macro

The first macro is intended to function similarly to `try` in Java 7, and `using` in C#. It should be possible to have a binding form (i.e. `[variable value]`) that binds to an instance of a closable.

The macro shall be called `safe`.
Example, Java 7 `try`

```
try (Socket s = new Socket()) {
  s.accept();
} catch(Exception e) {}
// s is automatically closed
```

If an exception is thrown inside the form the exception should be returned. Otherwise the evaluated value of the form should be returned. If a binding is provided it should be possible to use the bound variable inside the form. After execution of the form any variables that are bound shall be closed (e.g. `(let [s (Socket.)] (. s close))`) Note that closable classes in Java implements the `Closable`–interface (hint: use type hints). The return value of the macro is either the return value of the executed form or an exception.
The Clojure macro shall function in the following way:

```
user> (def v (safe (/ 1 0)))
user> v
#<ArithmeticException java.lang.ArithmeticException: Divide by zero>
```

```
user> (def v (safe (/ 10 2)))
user> v
5
user> (def v (safe [s (FileReader. (File. "file.txt"))] (. s read)))
user> v
105 ; first byte of file file.txt
user> (def v (safe [s (FileReader. (File. "missing-file"))] (. s read)))
user> v
#<FileNotFoundException java.io.FileNotFoundException:
missing-file (No such file or directory)>
```

2.2 SQL-like Macro

For a higher grade you shall also implement a `SQL`–like syntax for searching in sets of hash maps.
For the set:

```
(def persons #{{:id 1 :name "olle"} ;row
               {:id 2 :name "anna"}}) ;row

(def cars    #{{:id 1 :car "BMW"} ;row
               {:id 2 :car "Toyota"}}) ;row
```

It should be possible to write a query like:

```
user> (select [:id :name] from persons
         where :id (in (select [:id] from cars
                        where :car (like "T")
                        orderby :id))
         orderby :name)
#{{:id 2 :name "anna"}}

user> (select [:id :name] from persons
         where :id (= 2)
         orderby :name)
#{{:id 2 :name "anna"}}
```

The query format shall be:

```
(select [columns]
     from #{table}
     where :cell-name (= | in | > | < | != | like value)
     orderby :cell-name)
```

The `in`-function checks if a row in another set contains the value specified by `:cell-name`, that is `:id` in the example below.

```
user> (select [:id :name] from persons
        where :id (in #{{:id 1} {:id 3}})
        orderbyt :name)
#{{:name "olle", :id 1}}
```

The `like`-function works for strings, and checks if the value is part of the value of a cell (e.g. "o" is part of "olle")

```
(select [:name] from person
        where :name (like "o")
        orderby :name)

#{{:name "olle"}}
```

Note that `in` and `like` are functions/macros you need to define yourself.

2.3 Reflections

Argue whether these two macros can be implemented as functions, Why?, Why not? How? (approximately 500 words).

3 Grading

In order to receive the grade E you have to implement the safe macro and write the reflection on macros. To receive a grade better than C, you also have to implement the SQL macro.

Grading programming is more an art than a science. In the general case, it is extremely difficult to impossible to say that one program is better than the other. For the obvious reasons, it is impossible to cover the grading criteria completely. In any case, the points below are not totally black/white. This is why you should also should reflect on you implementation.

For the programming assignment for block 4 the following table will be used when grading the assignments.

Grade	Safe macro	SQL macro	Reflections
A	Good	Good	Good
B	Good	Good	OK
C	Good	Missing	OK
D	OK	Missing	OK-
E	OK-	Missing	OK-
Fx	Missing/OK-	Missing	Missing/OK-
F	Missing	Missing	Missing

Table 1: Grading scheme

To get a Good, your program/answer must:

1. Completely meet the specification,e.g., produce correct output from example inputs (not too little or too much), or the like;
2. Be readable without overly commenting, be correctly indented, and use reasonable names (for variables, method, classes etc.);
3. Be well designed for the problem at hand

For every point above that is not satisfied, the program/answer takes one step down the grade ladder from Good down to Missing/Fail.