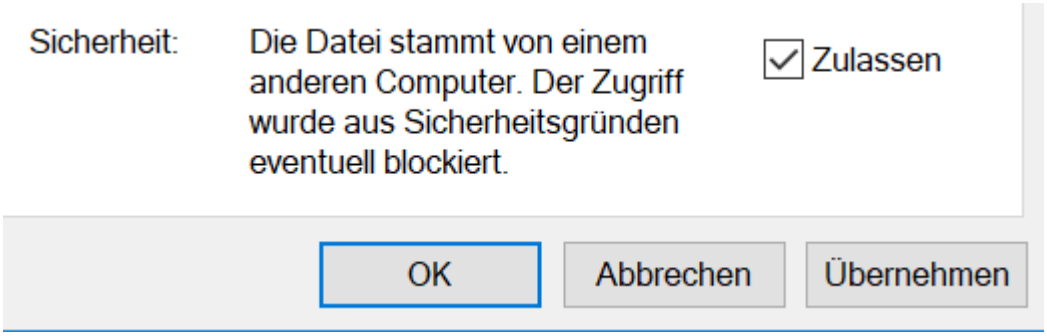
Accessing MySQL or MariaDB from Pharo7 Smalltalk

April, 28, 2019 – Gerald Zincke

There is the „Garage“ addon Package for Pharo that provides a relatively simple interface to several relational databases including MySQL (and the nearly identical MariaDB).

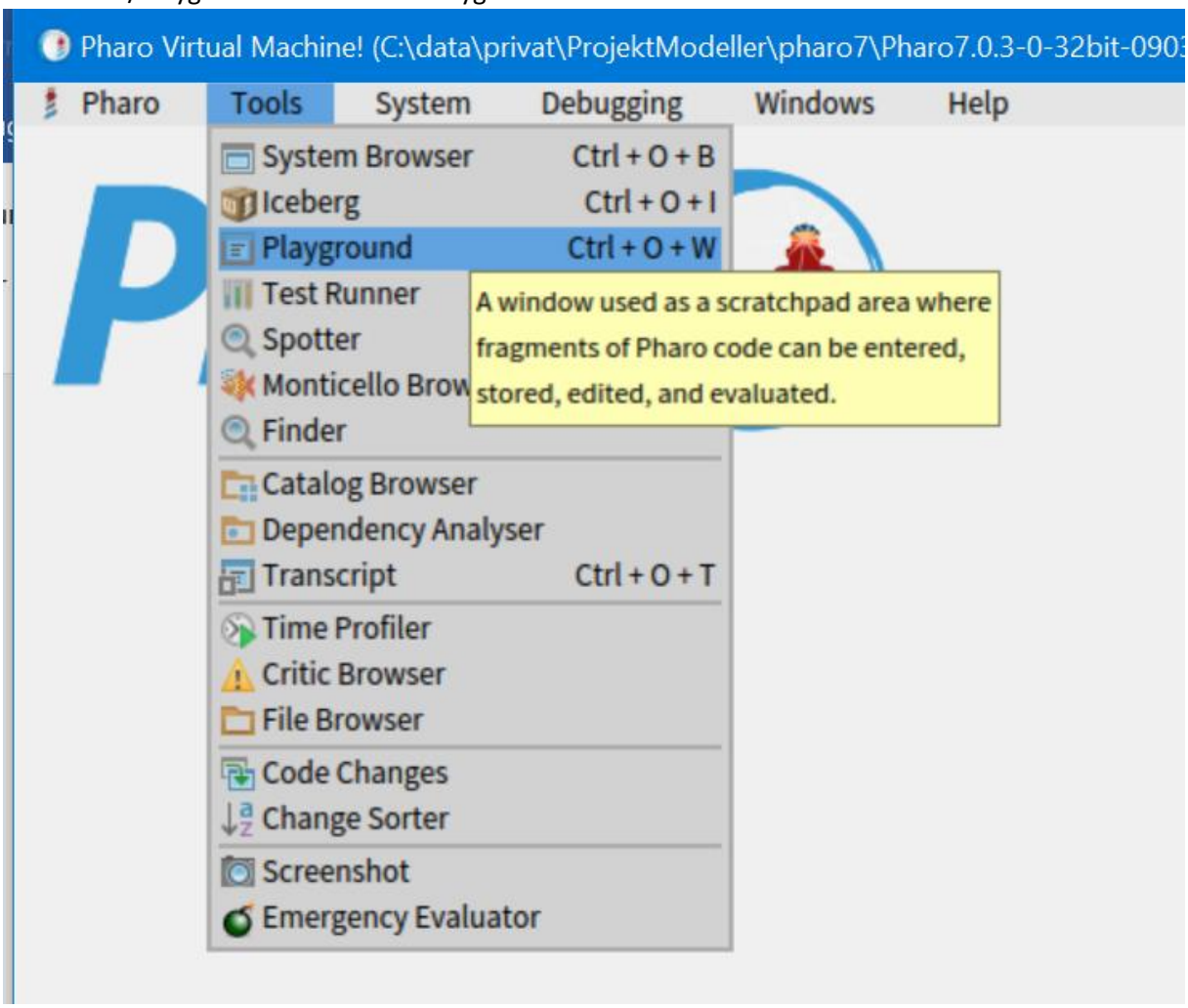
Unfortunately, the installation instructions at http://guillep.github.io/DBXTalk/garage/installation.html seems to be buggy. At least it did not work for me. For those wanting to use Pharo7 on Microsoft Windows 10 : Here is a step by step description how I finally had success.

1. Install the Pharo Smalltalk IDE: I first tried to download the 64bit version from https://pharo.org/download , namely: https://files.pharo.org/get-files/70/pharo64.zip and https://files.pharo.org/get-files/70/pharo64-win-stable.zip but it did not work in my 64bit OS. Then I downloaded the 32 bit Version Pharo environment https://files.pharo.org/get-files/70/pharo-win-stable.zip , expanded the ZIP-File to a new Pharo7-Folder and then I downloaded the 32 bit Image from https://files.pharo.org/get-files/70/pharo.zip and copied the files in the second ZIP file into my new Pharo7 folder too.
2. When I tried to start Pharo7\Pharo.EXE , Windows complained that the “The file is from an other computer. Access has been blocked for security reasons”. I had to open the context menu for the .EXE and the image files and explicitly allow access to them:



3. Then I could start the Pharo IDE by starting Pharo.exe.

4. With Tools/Playground I launched a Playground Window:



5. Installing the “Garage” Package and MySQL driver as described at http://guillep.github.io/DBXTalk/garage/installation.html dis not work for me. The sources I got this way were buggy or not up to date. After trying to fix the bugs a while , I throwed away the image, copied a fresh one from the previously pharo.zip into my Pharo7 folder.
6. So I went to https://github.com/pharo-rdbms/garage clicked the green “Clone or Download” Button to download garage-master.zip. From there I copied the included directory “garage-master” to Pharo7\garage-master.
7. Back in the Pharo Playground I entered :

```
Metacello new
baseline: 'Garage';
repository: 'filetree://C:\data\privat\ProjektModeller\pharo7\garage-master';
load.
```

into the Playground-Window, selected it and typed Ctrl-D

8. Now the classes were loaded into my image. At that time I saved the Image via menu Pharo/Save

9. I already had a MariaDB test-installation on my machine. You can easily get one from https://downloads.mariadb.org/ . I started the database daemon in a cmd-Window via

```
mariadb\bin\mysqld --standalone --console
```

Note: MariaDB is a clone from MySQL. Therefore the command to start the database deamon is named mysqld.exe

Note: the tests assume that the database server is running on localhost, listening to the default port 3306. This is the defaut port for MaySQL and MariaDB.

10. The Garage unittests use a database sodbxtest. To create it, I opened a cmd-Window in my mysql\bin folder and executed the following commands:

```
mysql -uroot -e "CREATE DATABASE sodbxtest /*\!40100 DEFAULT CHARACTER SET
utf8 */;"
mysql -uroot -e "CREATE USER sodbxtest@localhost IDENTIFIED BY 'sodbxtest';"
mysql -uroot -e "GRANT ALL PRIVILEGES ON sodbxtest.* TO
'sodbxtest'@'localhost';"
mysql -uroot -e "FLUSH PRIVILEGES;
```

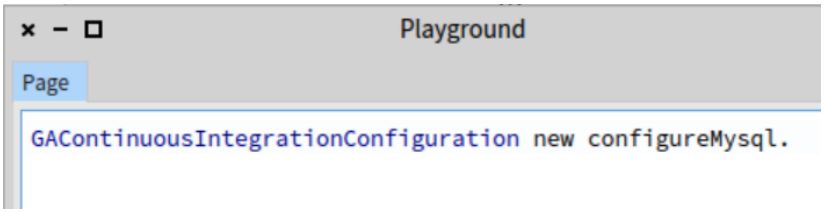
NOTE if your DB has a password for the default database user “root” , you need to add a parameter -p<password>

Note: For additional tests you can use the sample database “classicmodels” from http://www.mysqltutorial.org/mysql-sample-database.aspx

11. To run the Garage Unit-Tests, I entered

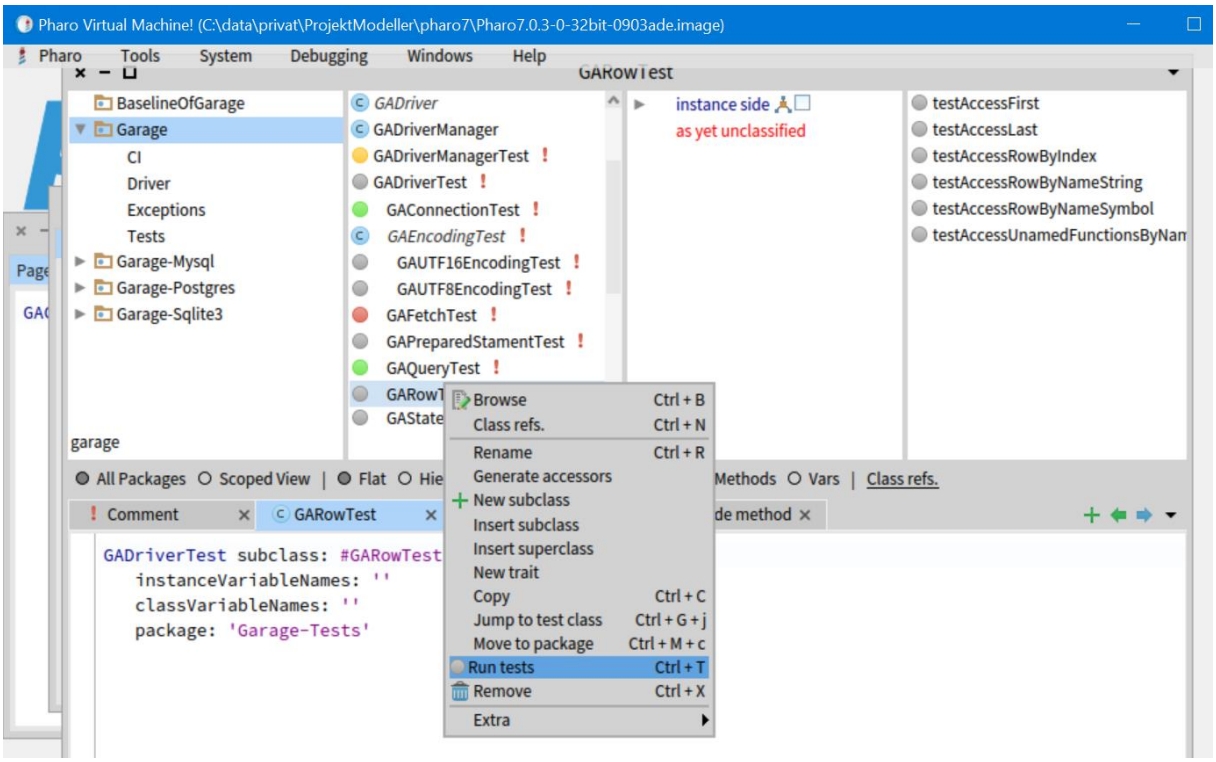
```
GAContinuousIntegrationConfiguration new configureMysql.
```

into the Playground, then selected `and` evaluated the command with Ctrl-D.



Note: the tests assume that the database server is running on localhost, listening to the default port 3306

12. Then via menu Tools/System Browser I opened the system browser. With ctrl-F I searched for Garage, located the Test classes and executed some unit tests.

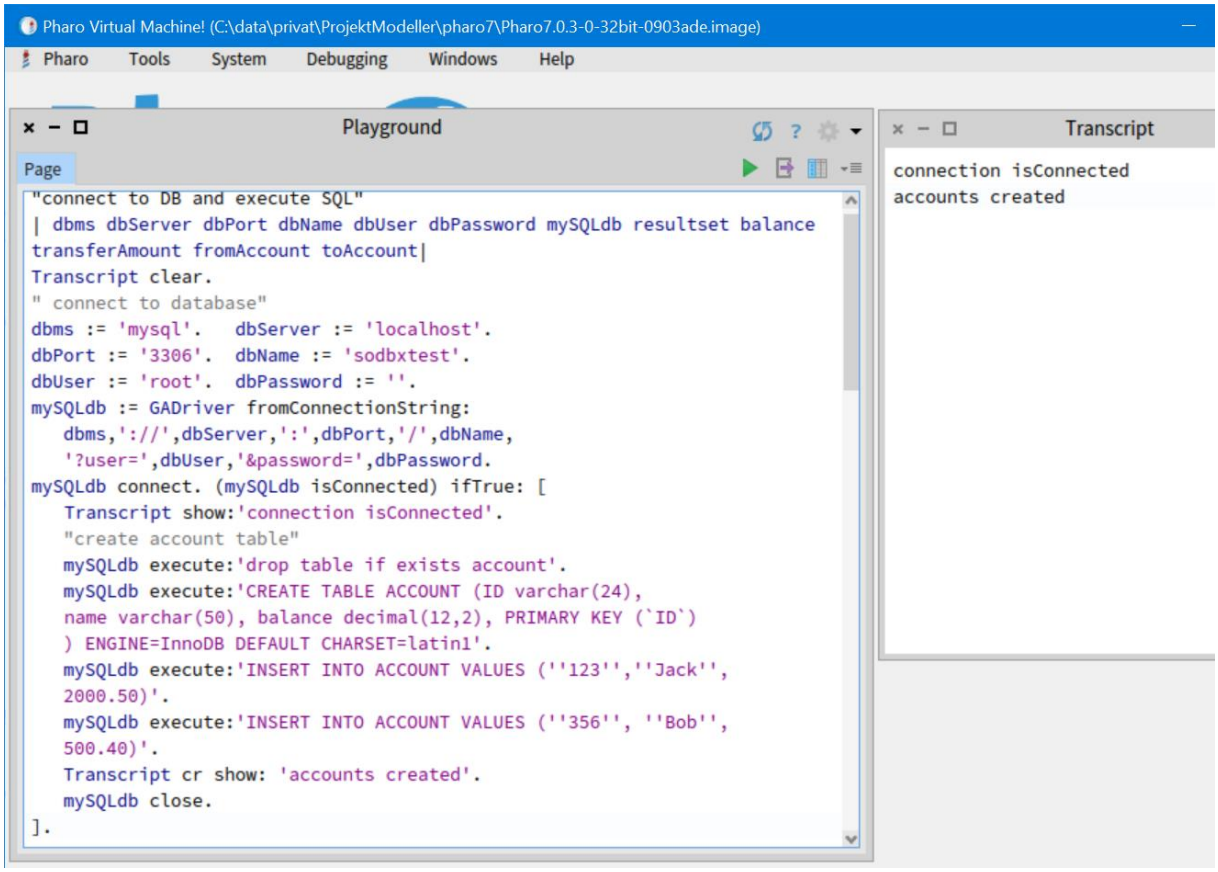


The GAFetchTests failed for some reason,I did not investigate yet, but others were fine.

13. Now it was time to write some own test code for the “sodbxtest” Database. I entered the following code into the Playground then selected and executed it with Ctrl-D

```
"connect to DB and execute SQL"
| dbms dbServer dbPort dbName dbUser dbPassword mySQLdb resultset balance
transferAmount fromAccount toAccount|
Transcript clear.
" connect to database"
dbms := 'mysql'.   dbServer := 'localhost'.
dbPort := '3306'.  dbName := 'sodbxtest'.
dbUser := 'root'.  dbPassword := ''.
mySQLdb := GADriver fromConnectionString:
       dbms,'://',dbServer,':',dbPort,'/',dbName,
       '?user=',dbUser,'&password=',dbPassword.
mySQLdb connect.
(mySQLdb isConnected) ifTrue: [
       Transcript show:'connection isConnected'.
       "create account table"
```

```
        mySQLdb execute:'drop table if exists account'.
        mySQLdb execute:'CREATE TABLE ACCOUNT (ID varchar(24),
        name varchar(50), balance decimal(12,2), PRIMARY KEY (`ID`)
        ) ENGINE=InnoDB DEFAULT CHARSET=latin1'.
        mySQLdb execute:'INSERT INTO ACCOUNT VALUES (''123'',''Jack'',
        2000.50)'.
        mySQLdb execute:'INSERT INTO ACCOUNT VALUES (''356'', ''Bob'',
        500.40)'.
        Transcript cr show: 'accounts created'.
        mySQLdb close.
].
```



14. Not sure that the database table really was created? You can check with.

```
mariadb\bin>mysql -uroot -e "select * from sodbxtest.account;"
```

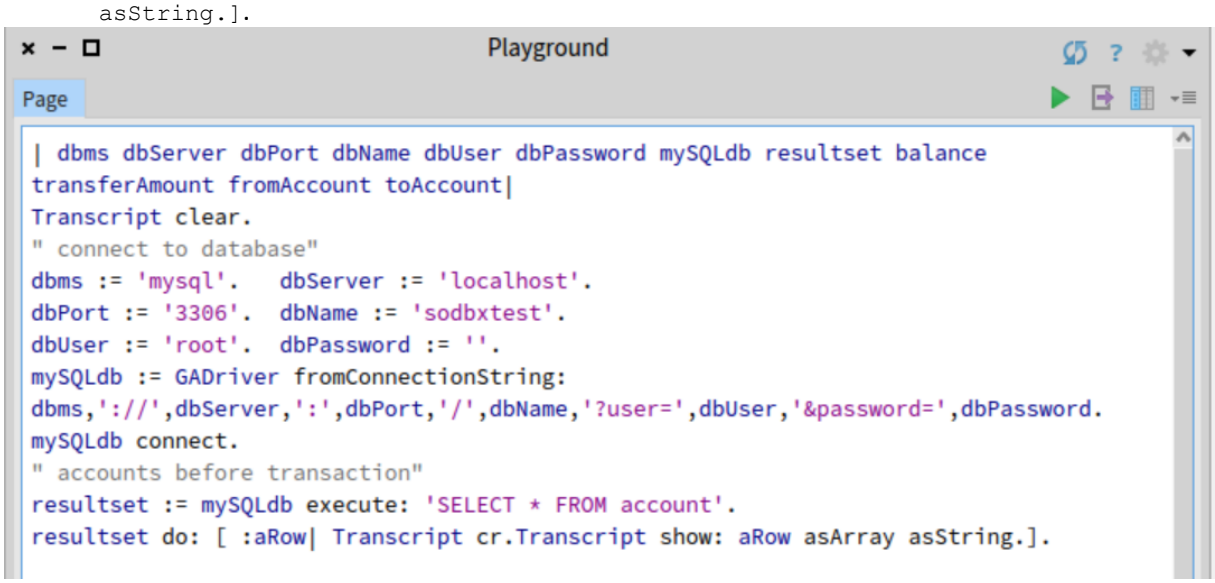
15. Now it was time for some real world database example: Transaction handling.
    Let us say that Jack wants to transfer 1000 € from his account to Bob's .
    But this transfer must not be done, if Jack's account balance will become negative .

    Note: At that time I had already created an "Account" table in my database (see example above)

16. First again to connect to the database and to show the contents of the database table "Account" , I entered the following code into a new Playground window:

```
        | dbms dbServer dbPort dbName dbUser dbPassword mySQLdb resultset
        balance transferAmount fromAccount toAccount|
        Transcript clear.
        " connect to database"
```

```
dbms := 'mysql'.   dbServer := 'localhost'.
dbPort := '3306'.  dbName := 'sodbxtest'.
dbUser := 'root'.  dbPassword := ''.
mySQLdb := GADriver fromConnectionString:
dbms,'://',dbServer,':',dbPort,'/',dbName,'?user=',dbUser,'&password='
,dbPassword.
mySQLdb connect.
" accounts before transaction"
resultset := mySQLdb execute: 'SELECT * FROM account'.
resultset do: [ :aRow| Transcript cr.Transcript show: aRow asArray
asString.].
```



Note: This is a very simple way to evaluate and display the result-set of a SELECT query. I am sure that there are more elegant ways to do this. And of course, error-handing is missing here too.

17. Then I added the code to implement the transfer of one account to the other:

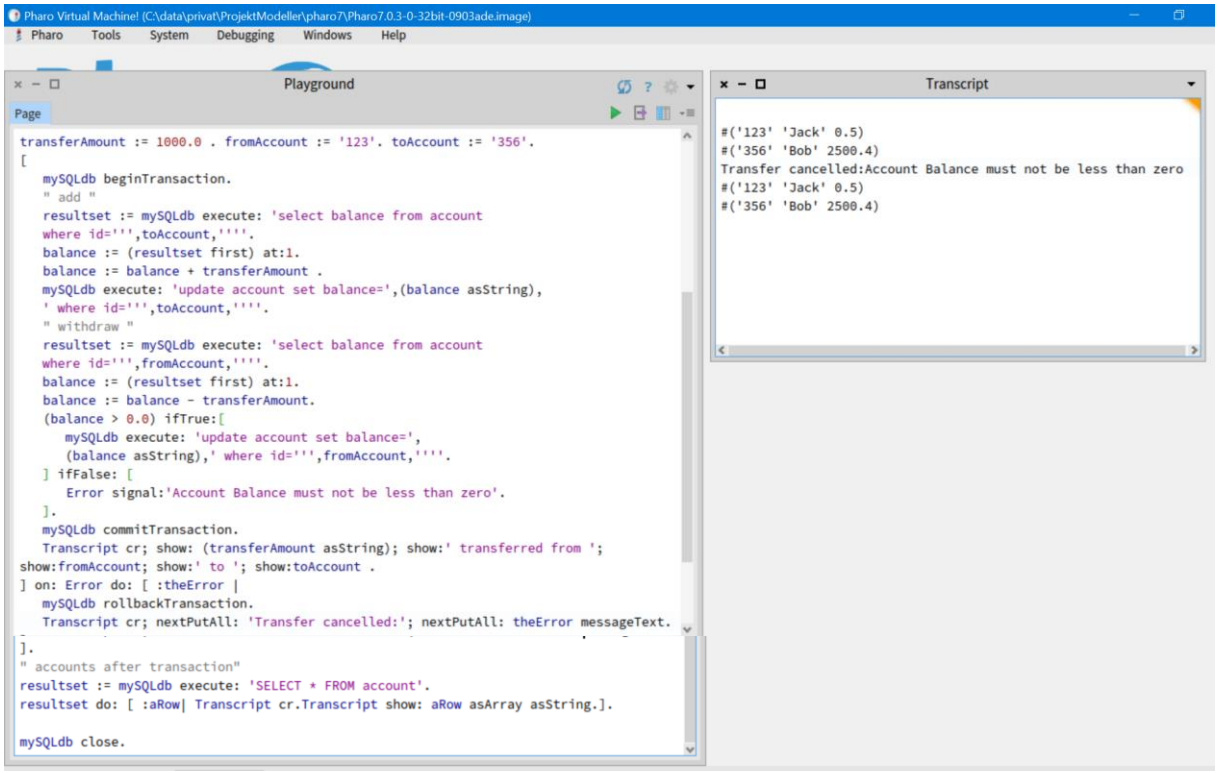
```
transferAmount := 1000.0 .
fromAccount := '123'.
toAccount := '356'.
[
       mySQLdb beginTransaction.
       " add "
       resultset := mySQLdb execute: 'select balance from account
       where id=''',toAccount,''''.
       balance := (resultset first) at:1.
       balance := balance + transferAmount .
   mySQLdb execute: 'update account set balance=',(balance asString),
   ' where id=''',toAccount,''''.

       " withdraw "
       resultset := mySQLdb execute: 'select balance from account
   where id=''',fromAccount,''''.
   balance := (resultset first) at:1.
       balance := balance - transferAmount.
       (balance > 0.0) ifTrue:[
      mySQLdb execute: 'update account set balance=',
      (balance asString),' where id=''',fromAccount,''''.
   ] ifFalse: [
            Error signal:'Account Balance must not be less than
zero'.
   ].

   mySQLdb commitTransaction.
```

```
   Transcript cr.
   Transcript show: (transferAmount asString); show:' transferred from
'; show:fromAccount; show:' to '; show:toAccount .
] on: Error do: [ :theError |
   mySQLdb rollbackTransaction.
   Transcript cr.
      Transcript nextPutAll: 'Transfer cancelled:'; nextPutAll:
theError messageText.
].
" accounts after transaction"
resultset := mySQLdb execute: 'SELECT * FROM account'.
resultset do: [ :aRow| Transcript cr.Transcript show: aRow asArray
asString.].

mySQLdb close.
```

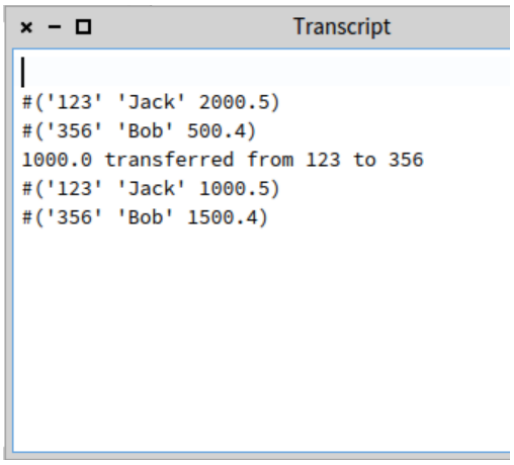


Note: After issuing a “BEGIN TRANSACTION” the sample code first reads the receiver’s balance, adds the transferAmount (1000€) and then updates the balance of the receiver’s account.

Then the balance of the sender’s account is read and decremented by the transferAmount. If that results to a value >0 the balance of the sender’s account is updated. Otherwise a Smalltalk Error is signalled.
The Error is then handled in the “on: Error”-branch below, where a database rollback is done and a message is written to the Transcript.

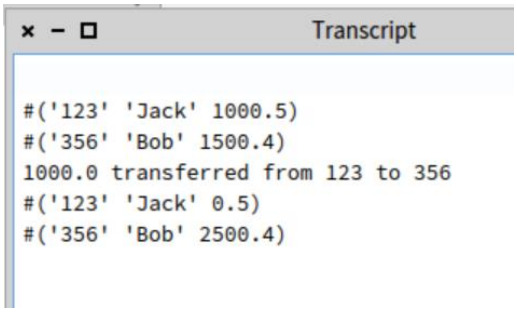
Note: Using Error-Handler logic here, works also for every other reason why the two accounts may come into an inconsistent state. Including syntax errors, lost connections or a blackout.

18. When I had put everything into the Playground and execute it by selecting all statements and Ctrl-D, the Transcript showed the following:



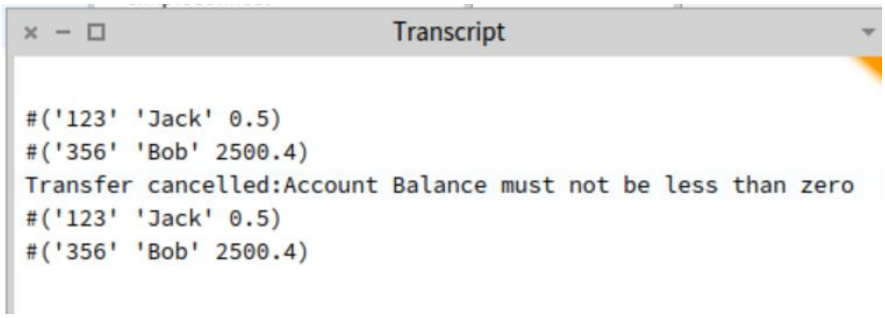
As expected 1000€ went from Jack's account to Bob's.

19. Repeating the execution resulted in:



Again 1000€ went from Jack's account to Bob's.

20. The third execution then resulted in:



Thus leaving both accounts unchanged.

Note: Although Bob's Account balance first was updated to 3500.4 the check of the balance in Jack's account led to an exception forcing a rollback of the transaction and the database reset Bob's account to 2500.4 again as shown in the Transcript.

Note: The rollback would also be done, if the second update would not be done because of any database-error. You can easily test that by causing some SQL syntax error in the second "update account set balance=" statement.

I hope this text is as helpful for others as it would have been for me, if I had found something similar three weeks ago.

Gerald