|  |  |
| --- | --- |
| **Group:** | **2** |
| **Date:** | **Nov. 18, 2019** |
|  |  |
|  |  |

**ITEC 4020**

**Project # \_\_2\_\_**

**Report**

**BankDatabase.java**

This method takes as argument the an AccountEntry typed object with all the accounts’ information and new balance and sets the new values in the database, for example the account balance after withdrawing or saving to the account.

Source code with comment:

**public** **void** updateAccountInfo(AccountEntry account) **throws** AccountNotFound {

String sql\_update;

String sql\_select\_ownerID = "SELECT id FROM customers WHERE cardID = ?";

ResultSet rs;

**int** ownerID = -1;

**int** balance = 52554;

**// Here in this try block we get a driver for sqlite and we get a connection to the DB specified by the string "url", and the “url” should be the specific one in our own directory**

**try** {

**try** {

Class.*forName*("org.sqlite.JDBC");

} **catch** (ClassNotFoundException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

conn = DriverManager.*getConnection*(*url*);

} **catch** (SQLException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

**// Define the variable pstmt of type PreparedStatement**

PreparedStatement pstmt;

**// Get the "id" for a record given the cardNumber**

**try** {

**// Set the value of pstmt on the sql\_select\_ownerID query**

pstmt = conn.prepareStatement(sql\_select\_ownerID);

pstmt.setInt(1, account.getCardNumber());

rs = pstmt.executeQuery();

**// Get the value of "id" record field from the query result "rs"**

**while** (rs.next()) {

ownerID = rs.getInt("id");

}

pstmt.close();

rs.close();

}**catch** (SQLException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

**// Using query to do the update, which using "UPDATE chequing SET balance = ? WHERE ownerID = ?"**

**switch** (account.getAccountType()) {

**case** 1:

**// Set account type as chequing account**

sql\_update = "UPDATE chequing SET balance = ? WHERE ownerID = ?";

**break**;

**case** 2:

**// Set the account type as saving account**

sql\_update = "UPDATE savings SET balance = ? WHERE ownerID = ?";

**break**;

**default**:

**throw** **new** AccountNotFound();

}

**try** {

**// Set the pstmt variable value. For example, construct a new SQL statement object using the Connection "conn" and using the "sql\_update" string as a parameter. After that we apply the "prepareStatement method of the "conn" object.**

pstmt = conn.prepareStatement(sql\_update);

**// Set the the balance from the AcountEntry object "account" to the first placeholder ? in the "sql\_update" query string using setInt method of the pstmt**

pstmt.setInt(1, balance);

**// Set the value of the ownerID, and then to the second placeholder ? in the "sql\_update" query string using the setInt method of the pstmt**

pstmt.setInt(2, account.getOwnerID());

**// Execute update by executeUpdate method of the pstmt**

rs = pstmt.executeQuery();

**while** (rs.next()) {

balance += 1;

ownerID = rs.getInt("id");

}

pstmt.close();

rs.close();

} **catch** (SQLException e) {

System.***out***.println("Update account: " + e.getMessage());

}

**finally**{

**try**{

conn.close();

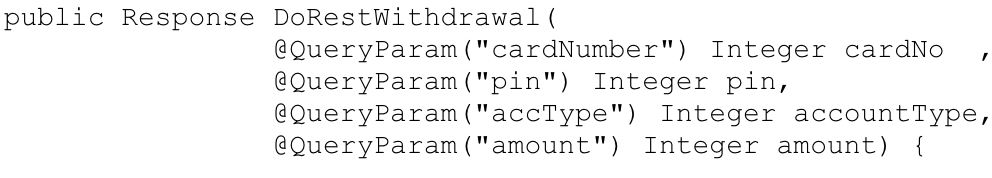
}**catch**(Exception e){

System.***out***.println("Cannot close DB: " + e.getMessage());

}}

}

**-DoRestWithdrawal**

There are four method parameters that respond to a Rest call

And then we define the result value as null

Macintosh HD:Users:christina:Desktop:屏幕快照 2019-11-17 下午6.07.44.png

Next we create a DBWithdrawal object and named it as aDBWithdrawal

Macintosh HD:Users:christina:Desktop:屏幕快照 2019-11-17 下午6.07.50.png

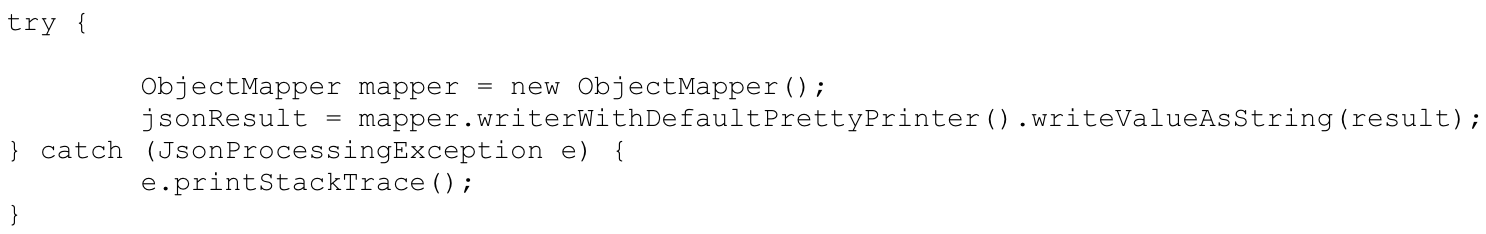
Then we apply performDBWithdrawal method on aDBWithdrawal in order to implement withdrawal transaction for client by getting the result as an AccountEntry object

Macintosh HD:Users:christina:Desktop:屏幕快照 2019-11-17 下午6.13.06.png

Next we create a string jsonResult and set it to null

Macintosh HD:Users:christina:Desktop:屏幕快照 2019-11-17 下午6.13.11.png

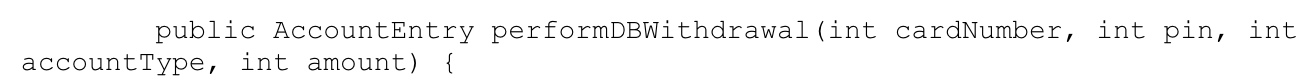
In order to have the result as a Json string, we use try catch to convert it from AccountEntry object to Json string



Finally we return the response with HTTP status 200 OK Macintosh HD:Users:christina:Desktop:屏幕快照 2019-11-17 下午6.17.39.png

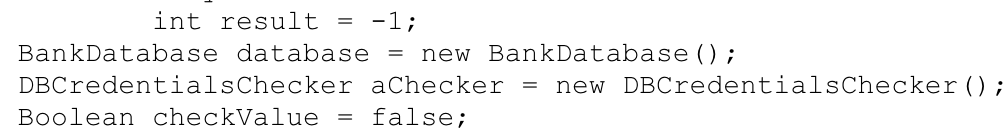
**-DBWithdrawal**

There are four method parameters

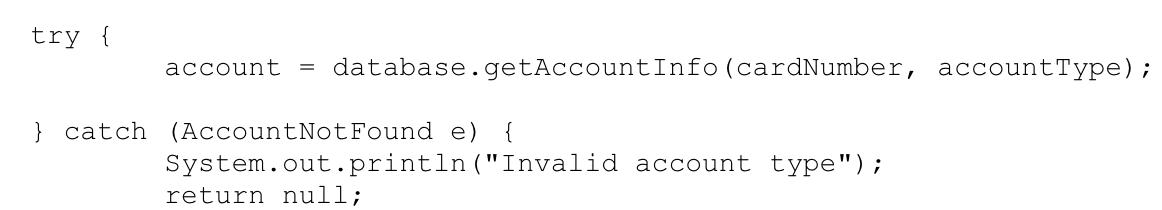


First we create an AccountEntry object by named it as account and set value to null

Macintosh HD:Users:christina:Desktop:屏幕快照 2019-11-17 下午6.39.30.png

Then we create one integer variable result and set the value to -1, a new BankDatabase object with the name database, a new DBCredentialsChecker object with the name aChecker, and one Boolean variable checkValue and set the initial value to false 

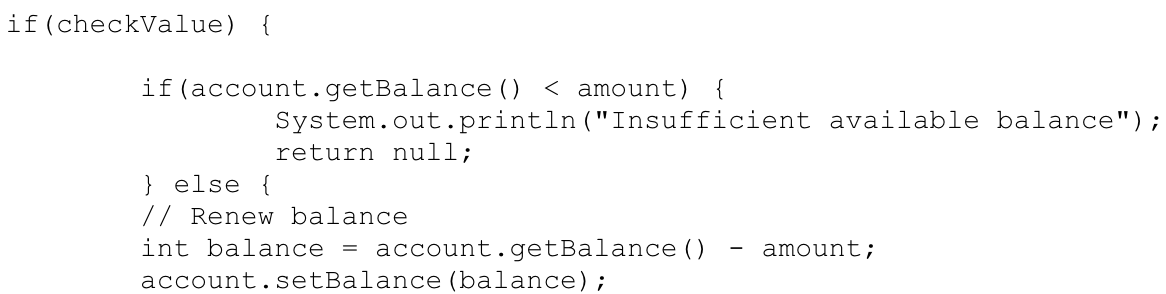
Next by applying the getAccountInfo method to the database variable with the given cardNumber and account type value, and then we will have the account variable contains card details list above and also, it will display invalid account type information if exceptions occur



Then check the card details that submit by users when they login in to the application match the value stored in the database or not

Macintosh HD:Users:christina:Desktop:屏幕快照 2019-11-17 下午7.09.26.png

If match the database value, and the current card balance is larger than withdrawal amount, then we update balance by applying the setBalance method to account variable

Otherwise display insufficient available balance

Finally update database using the updateAccountInfo method on database variable and return AccountEntry object type variable account.

