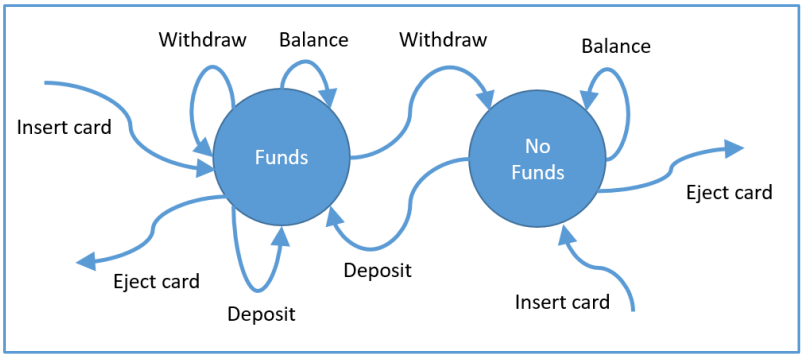
DEPENDENCIES ASSIGNMENT

BANK SYSTEM – ACCOUNTS / CREDIT CARDS / ATMS

Implement a simple bank system with accounts, credit cards and ATMs and do state testing and behavior testing of an ATM having some of the following functions…



1. Implement account and credit card classes based on the following interfaces…

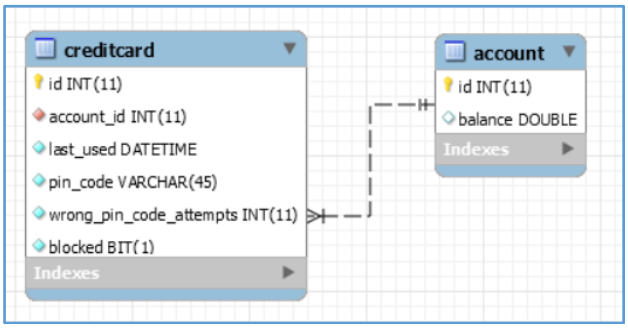
Account interface:

```
public interface AccountInterface
{
   public void setId(int id);
   public int getId();
   public void setBalance(double balance);
   public double getBalance();
   public void deposit(double amount);
   public void withdraw(double amount);
}
```

CreditCard interface:

```
public interface CreditCardInterface
{
   public void setId(int id);
   public int getId();
   public void setAccount(AccountInterface account);
   public AccountInterface getAccount();
   public void setLastUsed(Date lastUsed);
   public Date getLastUsed();
   public void setPinCode(int pinCode);
   public int getPinCode();
   public void setWrongPinCodeAttempts(int wrongPinCodeAttempts);
   public void addWrongPinCodeAttempt();
   public int getWrongPinCodeAttempts();
   public void resetWrongPinCodeAttempts();
   public void setBlocked(boolean blocked);
   public boolean isBlocked();
}
```

2. Set up a database for the bank system with accounts and credit card entities somewhat similar to the following diagram...



3. Persist some test data for each of the account and credit card entities with different values, so that there are some different combinations of credit cards and accounts in the database

4. Implement a data mapper class capable of updating account and credit card in the database, returning account and credit card based on id and returning null if none is found in the database based on the following interface…

DataMapper interface:

```
public interface DataMapperInterface
{
   public void setDataSource(DataSource dataSource);
   public void setCreditCard(CreditCardInterface creditCard);
   public CreditCardInterface getCreditCard(int id);
   public void setAccount(AccountInterface account);
   public AccountInterface getAccount(int id);
}
```

5. Implement an ATM class that uses the data mapper, account and credit card classes based on the following interface…

ATM interface:

```
public interface ATMInterface
{
   public void setDataMapper(DataMapperInterface dataMapper);
   public boolean insert(CreditCardInterface creditCard, int pincode) throws Exception;
   public void eject();
   public boolean deposit(double amount) throws Exception;
   public boolean withdraw(double amount) throws Exception;
   public double balance() throws Exception;
}
```

The ATM class implementation should function the following way…

- When a card is inserted, the id of the card is checked in the database and if not found, the card is ejected
- If the card is blocked by 3 failed pin code attempts, the card is ejected
- If the card and the pin code does not match a wrong pin code attempt is registered for the card and the card is then ejected
- If the card and pin code match the card is accepted
    - When the card is accepted the number of wrong pin code attempts are reseted for the credit card, the last used date is set for the credit card and the account balance information is registered by the ATM
- Only when a card is inserted in the ATM it should be possible to show balance, deposit, withdraw and eject
- When a credit card is ejected the account and credit card information is written to the database

6. Create a main method and try to use the ATM with a data mapper with access to a database with accounts and credit cards

7. Afterwards create a sufficient number of tests for the ATM, testing both the state and the behavior of the ATM, where tests include as many of the following details…
    - Using mock, stubs and spies
    - Testing the dependencies of the interfaces DataMapperInterface, CreditCardInterface and AccountInterface and the implementation of the ATM class
    - Verifying the interactions and asserting the state and transitions of the ATM
    - Using all the tests tools, frameworks and techniques used so far in general
    - Using test doubles in particular