实验二  单元测试

**一、实验目的**

(1)      认识单元测试原理

(2)      掌握单元测试过程

（3）会利用单元测试工具Junit和Mock对象进行单元测试

二**、实验内容：**

1．模拟运行课堂所讲的使用Junit进行单元测试的例子程序（求数组中的最大值），以及利用Mock对象进行单元测试的例子（如对Purchase类的测试），学习使用Junit和Mock对象进行单元测试的一般方法和步骤。

2．ATMService类模拟自动取款，提供取款（WithDraw）、存款（Deposit）、转账（Transfer）、查询（Inquiry）服务，它需要使用一个数据连接对象DBConnection类，该对象实现IDBConnection接口，连接数据库获取账户信息（getAccount）和更新账户信息（updateAccount）。账户信息由Account类表示。假设已经实现了ATMService类和Account类(后附代码)，准备对ATMService类进行单元测试。ATMService类中需要使用一个数据库连接对象，但现在还未实现DBConnection类，所以需要模仿一个数据库连接对象。提示：可以通过实现IDBConnection接口来得到模仿对象MockDBConnection，用Mock对象MockDBConnection来代替DBConnection对象，完成对ATMService类的测试。

后面给出了ATMService类的主要代码。ATMService类使用一个实现了IDBConnection接口的数据连接对象，并在构造函数中初始化该对象。有4个方法，分别是withDraw、deposit、transfer、inquiry。

Inquiry(String cardNo)方法传入账户卡号cardNo，返回该账户的余额。

Withdraw(String cardNo，int amount)方法传入账户卡号cardNo和一定数额的款项amount，首先通过数据连接对象获得该账户的余额信息，然后从余额中减去amount，随后再通过数据连接对象更新该账户余额。

Deposit(String cardNo，int amount)方法和withdraw方法类似，只是在账户余额上增加一定数额（amount）.

Transfer()方法涉及两个账户，从一个账户fromCardNo中扣除一定数额amount增加到另一个账户toCardNo上。

**三、实验要求**

（1）设计测试用例（可用黑盒方法或白盒方法设计，也可以结合两种方法，需在测试报告中说明使用什么方法设计的测试用例）

（2）编写DBConnection类的mock类MockDBConnection类实现IDBConnection接口；

（3）利用junit编写ATMService类的测试程序；

（4） 执行测试；

（5）分析测试结果，如果通过测试发现了程序中的bug，请找出原因并更正说明，并在代码中添加注释，说明此处有bug。实验报告中用红色注释标出

注:bug也可以是code bug,也可以是设计上的bug，如果是设计上的bug请说明理由。并给出更合理的设计方案。

（6）撰写实验报告；

 四、其它说明：

1、实验结束上交：自己编写的测试代码、mock类代码、实验报告（心的建议部分必填，必须填写实验过程中遇到的问题与解决方法，或者填写实验的体会和感受或存在的疑问或问题）。期中代码打包成一个文件，实验报告单独作为一个文件上传，不要打包到一起。注意文件命名格式：学号+姓名。

选做题：

3．一个简单的栈类。根据通常意义对栈的定义，具有入栈（Push）、出栈(Pop)、返回栈顶元素（Top）等。（自己编写栈类，然后使用junit编写测试类，对自己编写的类进行测试）

4．一个购物车类。该类实现添加、删除、以及计算购物车中物品的数量。（自己编写购物车类，然后使用junit编写测试类，对自己编写的类进行测试）

————————————

附：Account类、ATMService类、IDBConnection接口代码清单

**public** **class** Account {

String cardNo;

String password;

**int** balance;

**public** Account(String cardNo,String password,int balance){

**this.**cardNo=cardNo;

**this.**password=password;

**this.**balance=balance;

}

**public** **void** setCardNo(String cardNo){

**this**.cardNo=cardNo;

}

**public** **void** setPassword(String ps){

**this**.password=ps;

}

**public** **void** setBalance(**int** amount){

**this**.balance=amount;

}

**public** String getCardNo(){

**return** **this**.cardNo;

}

**public** String getPassword(){

**return** **this**.password;

}

**public** **int** getBalance(){

**return** **this**.balance;

}

}

————————————————

**public** **interface** IDBConnection {

**public** Account getAccount(String id);

**public** **void** updateAccount(Account account);

**public** **void** addAccount(Account account);

}

————————————

**public** **class** ATMService {

**private** IDBConnection dbconn;

**public** ATMService(IDBConnection db){

**this**.dbconn=db;

}

**public** **void** withDraw(String cardNo,**int** amount){

Account a=dbconn.getAccount(cardNo);

**int** balance=a.getBalance()-amount;

a.setBalance(balance);

dbconn.updateAccount(a);

}

**public** **void** deposit(String cardNo, **int** amount){

Account a=dbconn.getAccount(cardNo);

**int** balance=a.getBalance()-amount;

a.setBalance(balance);

dbconn.updateAccount(a);

}

**public** **void** transfer(String fromCardNo, String toCardNo, **int** amount){

Account a=dbconn.getAccount(fromCardNo);

Account b=dbconn.getAccount(toCardNo);

**int** aBalance=a.getBalance()-amount;

**int** bBalance=b.getBalance()+amount;

a.setBalance(aBalance);

b.setBalance(bBalance);

dbconn.updateAccount(a);

dbconn.updateAccount(a);

}

**public** **int** inquiry(String cardNo){

Account a=dbconn.getAccount(cardNo);

**return** a.getBalance();

}

}

————————

**public** **class** MockDbConnection **implements** IDBConnection{

private HashMap<String,Account> mdb;

public MockDbConnection(){

mdb=new HashMap<String,Account>();

Account a=new Account("1","11111",300);

Account b=new Account("2","22222",2000);

mdb.put(a.getCardNo(),a);

mdb.put(b.getCardNo(),b);

}

public Account getAccount(String cardNo){

//需填充编写代码

}

public void updateAccount(Account a){

//需填充编写代码

}

public void addAccount(Account a){

//需填充编写代码

}

}