|  |  |
| --- | --- |
| Course | Advanced Software Design – CS525 |
| Assignment | Lab 5 |
| Week | 05 |
| Due | 7/27/2025 |
| Student name | Toe Toe Aung |
| Student ID | 618090 |

1. Modified class diagram of Bank Application

A screenshot of a computer

AI-generated content may be incorrect.

b. The sequence diagram shows the following scenario:

1. First deposit a certain amount

2. Then withdraw a certain amount

3. Then call undo

A diagram of a computer

AI-generated content may be incorrect.

1. Implementation of the command pattern in the given code

ICommand interface is added to the existing code

**package** bank.command;

**import** bank.domain.Account;

**public** **interface** ICommand {

**public** **void** execute();

Account getAccount();

**double** getAmount();

Account getToAccount();

}

The result logs after modified with Command pattern.

UNDO WITHDRAW

UNDO TRANSFERFUNDS

UNDO DEPOSIT

UNDO WITHDRAW

UNDO TRANSFERFUNDS

REDO WithdrawCommand

Statement for Account: 1263862

Account Holder: Frank Brown

-Date--------------------------Description-------------------Amount-------------

Sun Jul 27 08:46:22 CDT 2025 deposit 200.00

Sun Jul 27 08:46:22 CDT 2025 withdraw -100.00

Sun Jul 27 08:46:22 CDT 2025 deposit 300.00

Sun Jul 27 08:46:22 CDT 2025 withdraw -200.00

Sun Jul 27 08:46:22 CDT 2025 deposit 200.00

Sun Jul 27 08:46:22 CDT 2025 withdraw -300.00

Sun Jul 27 08:46:22 CDT 2025 deposit 300.00

Sun Jul 27 08:46:22 CDT 2025 withdraw -100.00

Sun Jul 27 08:46:22 CDT 2025 payment of invoice 10232 -100.00

--------------------------------------------------------------------------------

Current Balance: 200.00

Statement for Account: 4253892

Account Holder: John Doe

-Date--------------------------Description-------------------Amount-------------

Sun Jul 27 08:46:22 CDT 2025 payment of invoice 10232 100.00

--------------------------------------------------------------------------------

Current Balance: 100.00

UNDO TRANSFERFUNDS

Statement for Account: 1263862

Account Holder: Frank Brown

-Date--------------------------Description-------------------Amount-------------

Sun Jul 27 08:46:22 CDT 2025 deposit 200.00

Sun Jul 27 08:46:22 CDT 2025 withdraw -100.00

Sun Jul 27 08:46:22 CDT 2025 deposit 300.00

Sun Jul 27 08:46:22 CDT 2025 withdraw -200.00

Sun Jul 27 08:46:22 CDT 2025 deposit 200.00

Sun Jul 27 08:46:22 CDT 2025 withdraw -300.00

Sun Jul 27 08:46:22 CDT 2025 deposit 300.00

Sun Jul 27 08:46:22 CDT 2025 withdraw -100.00

Sun Jul 27 08:46:22 CDT 2025 payment of invoice 10232 -100.00

Sun Jul 27 08:46:22 CDT 2025 undo the previous transfer 100.00

--------------------------------------------------------------------------------

Current Balance: 300.00

Statement for Account: 4253892

Account Holder: John Doe

-Date--------------------------Description-------------------Amount-------------

Sun Jul 27 08:46:22 CDT 2025 payment of invoice 10232 100.00

Sun Jul 27 08:46:22 CDT 2025 undo the previous transfer -100.00

--------------------------------------------------------------------------------

Current Balance: 0.00

1. Modified class diagram for the undo/redo buttons

A computer screen shot of a diagram

AI-generated content may be incorrect.

e. The sequence diagram that shows the following scenario:

a) The user clicks the increment button

b) The user clicks the decrement button

c) The user clicks undo

A diagram of a company

AI-generated content may be incorrect.

1. Implementation in Java and complete code is archived as .zip file with observer pattern

**public** **class** Counter {

**private** **int** count = 0;

List<Observer> observerList;

**public** Counter() {

observerList = **new** ArrayList<Observer>();

}

**public** **void** registerObserver(Observer o) {

**this**.observerList.add(o);

}

**public** List<Observer> getObservers() {

**return** **this**.observerList;

}

**public** **int** getCount() {

**return** count;

}

**public** **void** increment() {

count++;

**this**.observerList.stream().forEach(e -> e.update(count));

}

**public** **void** decrement() {

count--;

**this**.observerList.stream().forEach(e -> e.update(count));

}

}