BankManager Unit Testing

CptS 422 : HW 6

Isaac Baker

Andrew Bates

# Part I:

Convert the bank application from C++ to C# code.

* For this part of the project, we used Tangible Software Solutions convert to make the initial conversion from C++ to C#. Then we cleaned up the code and carried on.

# Part II:

Download testing tools.

* We decided to use Xunit and ncover.

Unit tests include:

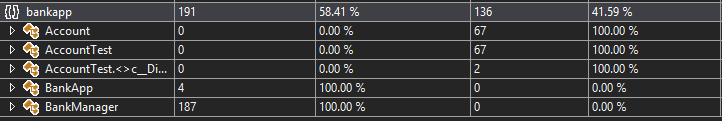
* createAccount
  + creates an account and tests members
* copyAccount
  + this covers the overloaded Account
  + testing copying account
* copyFromAccount
  + memberwise clone copy and verification
* creditAccount
  + tests addint account moneys
* debitAccount
  + trys to remove account moneys and trys too much
* printBalance
  + verify no exception thrown

Basically that means, there were not unit tests for the “GUI” side of the app.

* Since BankManager is all GUI interaction logic, we couldn't test it in its current state without majorly rewriting the code.
* There are GUI testing suites out there, but BankManager is not unit testable via x/nunit

The test coverage xml is in the main submission folder. The overall coverage is as follows:

* Account: 100%
* BankManager: 0%



# Part III:

Create combinational and/or state test models where appropriate.

## Combinational Models:

|  |  |
| --- | --- |
| **createAccount** | |
| **Condition** | **Condition Alternative** |
| * Balance = 50 | * Balance = -5 |
| **Action** | **Action Alternative** |
| * Account created with balance = 50 | * Account created with balance = 0 |

|  |  |
| --- | --- |
| **debitAccount** | |
| **Condition** | **Condition Alternative** |
| * Debit 24.75 from account * Balance starts out at 50 | * Debit 500 from account (overdraw) * Balance starts out at 50 |
| **Action** | **Action Alternative** |
| * Balance = 25.25 | * Does not debit * return balance |

* As for state models, there wouldn't really be any benefit to making them for this application, since the system has so few states.

# Part IV:

Create UML class diagram for the implementation.

## BankApp UML Class Diagram

## 

## Tests generated from Class Diagram

* create
  + create an account
* copyAccount
  + copy an account
* copyFrom
  + copy account data from one to another
* credit
  + adding a balance to an account will return a new balance
* debit
  + takes from balance of an account will return a new balance
* printBalance
  + print all balances assoctiated with account

# Testing Strategies

Our testing strategies are described in Part II & III.