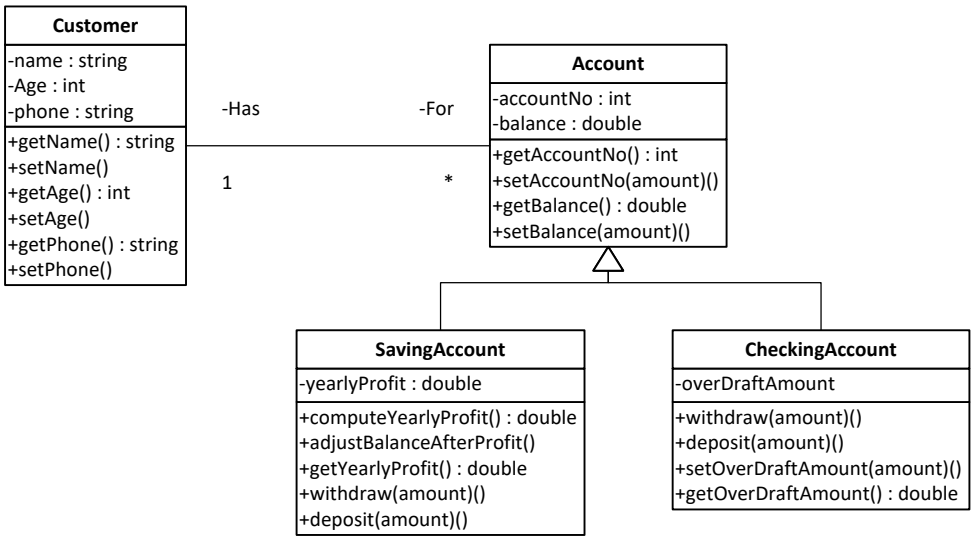
Department of Software Engineering

SE 430 (Software Testing)

Second 2019 / 2020

Course project – phase 2: Black box testing with JUnit

Consider a simple banking application that consists of the following classes:



1) Customer: this class represent a bank customer.
    a. It has the following attributes:
        i. Name: The customer name is of type string and it should contain only alphabets and white spaces (no numbers, no symbols).
        ii. Age: The customer age is of type integer and it should be at least 18 years old
        iii. Phone: customer phone number. The customer phone number is 10 digit long and it should start with a zero.
    b. The Customer class has the following methods:
        i. getName: gets the customer name.
        ii. setName: sets the customer name.
        iii. getAge: gets the customer age.
        iv. setAge: sets the customer age.
        v. getPhone: get the customer phone number.
        vi. setPhone: sets the customer phone number.
2) Account: this class represents the customer's bank account.
    a. It has the following attributes:
        i. accountNo: The account number is of type integer. It consists of 14 digits and it can start with 1, 2, or 3.
        ii. Balance: Account balance is of type double. The balance should be higher or equal to zero. However, in the case of "Checking" account, the balance value can be negative by the specified over draft amount.

b. The Account class has the following methods:
    i. getAccountNo: Returns the account number.
    ii. setAccountNo: sets the account number
    iii. getBalance: returns the abalnce
    iv. setBalance: sets the balance

3) SavingAccount: This class represents saving account.
    a. It has the following attribute:
        i. yearlyProfit: The yearly profit is of type double and it has a value that ranges from 0.02 to 0.08.
    b. It has the following methods:
        i. computeYearlyProfit: This method calculates the annual profit on a saving account as follows: balance * yearlyProfit
        ii. adjustBalanceAfterProfit: This method is used to add the yearly profit computed by "computeYearlyProfit" to the balance.
        balance = balance + computeYearlProfit()
        iii. getYearlyProfit: returns the yearlyProfit
        iv. withdraw(amount): this method takes amount as an argument and deducts it from the balance. Balance = balance – amount
        v. deposit(amount): this method takes amount as an argument and adds it to the balance. Balance =balance + amount
4) CheckingAccount: This class represents the checking account.
    a. It has the following attribute:
        i. overDraftAmount: The overdraft amount represents the maximum amount that can be withdrawn after the balance reaches zero. For example, if the balance is zero and the overDraftLimit is 200, the customer can withdraw a maximum of 200. In this case, the balance will become -200.
    b. It has the following methods:
        i. Withdraw(amount): this method takes amount as an argument and deducts it from the balance. Balance = balance – amount. However, the customer can withdraw a maximum of balance + overDraftAmount.
        ii. Deposit(amount): this method takes amount as an argument and adds it to the balance. Balance =balance + amount.
        iii. setOverDraftLimit(amount): sets the overdraft limit amount
        iv. getOverDraftLimit: returns the overdraft limit amount

You are required to do the following:

a) Implement the above described classes using JAVA programming language.
b) For each implemented class, design test cases using equivalence partitioning / boundary value analysis (2-value approach). The test cases should be written in JUNIT and as follows:
    a. For each class, write a test class named as class name + Test (e.g., CustomerTest) that tests all methods in the given class.
    b. Implement a test suite class named "BankAccountTestSuite" that includes all the designed test cases.
    c. Implement a class "TestDriver" that executes "BankAccountTestSuite".

Submission Deadline:

6/4/2020 23:55 PM (You get 110%)

7/4/2020 23:55 PM (You get 100%)

8/4/2020 23:55 PM (You get 75%)

9/4/2020 23:55 PM (You get 50%)

No submission will be accepted after 9/4/2020 23:55 PM. No excuse will be given under any circumstances.

Important note: Corrupted files will get a zero grade. It is your own responsibility to check that the uploaded file is readable.