Objective

The objective of this workshop is to use object oriented techniques to model bank account

Setup

a. Create a new repository on Github for this workshop
b. Open your IDE and create the directories called `src` and `classes`; these directories are for your source code and compiled Java classes respectively
c. Make the project directory a Git repository
d. Add `.gitignore` to ignore the target directory
e. Add the remote repository (step b above) as the origin of your local repository

Workshop

Task 1

Write the following Java classes to model a bank account. The bank account should contain the following members and methods

`BankAccount` class

- Members - all members are private. You should provide getters and setters.
  - account holder's name - string. This is a read only property viz. cannot be changed once it is set
  - randomly generated account number - string
  - account balance - float
  - transactions for operations performed on the account - a list of strings
  - closed to indicate if this account has been closed - boolean
  - account creating date
  - account closing date

  account holder's name and account number are read only properties. They are set when this class is created and cannot be changed during the lifetime of the instance.

- Methods

  - `deposit` - deposit some amount into the account. This operation should be added to the transactions list (above); for example, if $100 is added to the account, then you should add the following "`deposit $100 at <date time>`" to transaction list
    Deposits only accepts positive amount. If an in correct amount is given or account is closed, throw an `IllegalArgumentException`.
  - `withdraw` - withdraw some amount from the account. This operation should be added to the transactions list (above); for example, if $100 is added to the account, then you should add the following "`withdraw $100 at <date time>`" to transaction list
    Deposits only accepts positive amount. If an in correct amount is given or the account is closed, throw an `IllegalArgumentException`
- Constructor
  - The first constructor has a single parameter which is the account holder's name. Initialize the balance to 0
  - The second constructor has 2 parameters; the first is the account holder's name and the second is the initial account balance.

Task 2

Write `FixedDepositAccount` class

Same members and methods as `BankAccount` class. In addition, `FixedDepositAccount` has the following variation in members, methods and constructors

- Members
  - interest - float
  - duration in months - integer
    The default interest and duration is 3 and 6 respective. They can be changed but only once. Any subsequent attempt to change more than once will result in an `IllegalArgumentException` thrown
    Once balance is set, it cannot be changed.

- Members
  - `withdraw()` and `deposit()` methods will not update the account's balance viz. they will perform a NOP (no operation)
  - `getBalance()` should return the balance plus the interest viz if interest is 3 and balance is 100, the `getBalance()` should return 103
- Constructor

- The following are the constructors signature
  - `public FixedDepositAccount(String name, Float balance)`
  - `public FixedDepositAccount(String name, Float balance, Float interest)`
  - `public FixedDepositAccount(String name, Float balance, Float interest, Integer duration)`

Submission

When you have completed your program, commit and push your code to the remote repository.