CSE 102L Computer Programming Laboratory – Exercise 5

Due Date: 23:59 Friday April 28th

Disclosure

You will submit your file to an assignment that is given through MS teams. Your filename should be *Ex5_yourStudentNumber.java*. Submissions made after the deadline will not be accepted, be sure to submit your work before the due date and make sure to click turn in button. Your code will be automatically controlled, so be sure to have the same class, method, variable names as described here. Failure to do so may result in you receiving 0 from this exercise. **All classes should be written to a single Java file. In a single java file there can only be single public class. Do NOT use Inner Classes. Be careful naming your file. If your editor inserts the file into a package, remove that line from the file but do NOT delete the import statements.**

Write set of classes according to the following specifications. Declare all attributes as **private** if not requested otherwise and use camelCase formatting for attributes.

1. Account

- Attributes:
  - accountNumber: String
  - balance: double
- Methods:
  - Constructor that takes *accountNumber* and *balance* as parameter. If balance is less than 0, throw **InsufficientFundsException(balance)**.
  - Accessor for attributes.
  - deposit(amount: double): void – adds the *amount* to *balance*. If amount is negative throw **InvalidTransactionException(amount)**.
  - withdraw(amount: double): void – removes the *amount* from balance. If amount is less than 0, throw **InvalidTransactionException(amount)**. If balance is less than amount, throw **InsufficientFundsException(balance, amount)**
  - toString(): String – returns String in the following format:
    - "Account: " + accountNumber + ", Balance: " + balance

2. Customer

- Attributes:
  - name: String
  - accounts: Collection of account objects
- Methods:
  - Constructor that takes *name* as parameter
  - getAccount(accountNumber: String): Account – a private method that returns the account with matching *accountNumber*. If the account with

accountNumber does not exists in the collection, throw **AccountNotFoundException(accountNumber)**.

- addAccount(account: Account): None – This method tries to get an account with given account's accountNumber if **AccountNotFoundException** is thrown, adds the given account to collection. Otherwise, throw **AccountAlreadyExistsException(accoundNumber)**. **Finally**, no matter what happens prints **this** customer. Lastly, prints "Added account: " + accountNumber + " with " + account.getBalance()
- removeAccount(accountNumber: Account): None – removes the account with given accountNumber from collection.
- transfer(fromAccount: String, toAccount: String, double amount): None – Tries to withdraw amount from *fromAccount*, deposit to *toAccount*. If **InvalidTransactionException** is thrown, then throw **InvalidTransactionException(InvalidTransactionException e, "cannot transfer funds from account " + *fromAccount* + " to account " + *toAccount*)**
- toString(): String – returns a String in the following format:
  - "Customer " + name + ":\n" + each account in a new line and indented with \t

3. InsufficientFundsException – Child of **RuntimeException**

- Methods:
  - Constructor that takes *balance* and sends it to it's super's constructor as:
    - "Wrong balance: " + balance
  - Constructor that takes *balance* and *amount* and sends them to it's super's constructor as:
    - "Required amount is " + amount + " but only " + balance + " remaining"

4. AccountAlreadyExistsException – Child of **RuntimeException**

- Methods:
  - Constructor that takes *accountNumber* and sends it to it's super's constructor as:
    - "Account number " + accountNumber + " already exists"

5. AccountNotFoundException – Child of **RuntimeException**

- Methods:
  - Constructor that takes *accountNumber* and sends it to it's super's constructor as:
    - "Account number " + "accountNumber + " already exists"

6. InvalidTransactionException – Child of **Exception**

- Methods:
    - Constructor that takes *amount* and sends it to it's super's constructor as:
        - "Invalid amount: " + amount
    - Constructor that takes *e* exception that is also InvalidTransactionException and *message* and sends them to it's super's constructor as:
        - message + ":\n\t" + e.getMesage()